Vector

Jonas Eschle, Jim Pivarski, Eduardo Rodrigues, and Henry Schre

Apr 23, 2024

EXAMPLE GALLERY

Vector is a Python 3.8+ library (Python 3.6 and 3.7 supported till `v0.9.0` and `v1.0.0`, respectively) for 2D, 3D, and Lorentz vectors, especially *arrays of vectors*, to solve common physics problems in a NumPy-like way.

Main features of Vector:

- Pure Python with NumPy as its only dependency. This makes it easier to install.
- Vectors may be represented in a variety of coordinate systems: Cartesian, cylindrical, pseudorapidity, and any combination of these with time or proper time for Lorentz vectors. In all, there are 12 coordinate systems: {x - y vs - in the azimuthal plane} × {z vs vs longitudinally} × {t vs temporally}.
- Uses names and conventions set by ROOT's TLorentzVector and Math::LorentzVector, as well as scikit-hep/math, uproot-methods TLorentzVector, henryiii/hepvector, and coffea.nanoevents.methods.vector.
- **Implemented on a variety of backends:**
  - pure Python objects
  - NumPy arrays of vectors (as a structured array subclass)
  - Awkward Arrays of vectors
  - potential for more: CuPy, TensorFlow, Torch, JAX…
- NumPy/Awkward backends also implemented in Numba for JIT-compiled calculations on vectors.
- Distinction between geometrical vectors, which have a minimum of attribute and method names, and vectors representing momentum, which have synonyms like `pt` = `rho`, `energy` = `t`, `mass` = `tau`.

CHAPTER

ONE

INSTALLATION

Vector is available on PyPI as well as on conda. The library can be installed using `pip` -

```
pip install vector
```

or using `conda` -

```
conda install -c conda-forge vector
```

CHAPTER

TWO

EXAMPLE GALLERY

`Vector` has several examples covering the basics as well as some advanced usage of the library. The example gallery covers almost all the features offered by `vector` and any new additions to the gallery are welcomed.

Note: Adding more examples and improving the existing examples for newcomers is still in progress.

2.1 Introduction to Vector

Vector is a Python 3.8+ library (Python 3.6 and 3.7 supported till `v0.9.0` and `v1.0.0`, respectively) for 2D, 3D, and Lorentz vectors, especially *arrays of vectors*, to solve common physics problems in a NumPy-like way.

Main features of Vector:

- Pure Python with NumPy as its only dependency. This makes it easier to install.
- Vectors may be represented in a variety of coordinate systems: Cartesian, cylindrical, spherical, and any combination of these with time or proper time for Lorentz vectors. In all, there are 12 coordinate systems: {x-y vs ρ-ϕ in the azimuthal plane} × {z vs θ vs η longitudinally} × {t vs τ temporally}.
- Uses names and conventions set by ROOT's TLorentzVector and Math::LorentzVector, as well as scikit-hep/math, uproot-methods TLorentzVector, henryiii/hepvector, and coffea.nanoevents.methods.vector.
- Implemented on a variety of backends:
  - pure Python objects
  - SymPy vectors
  - NumPy arrays of vectors (as a structured array subclass)
  - Awkward Arrays of vectors
  - potential for more: CuPy, TensorFlow, Torch…
- Awkward backend also implemented in Numba for JIT-compiled calculations on vectors.
- JAX and Dask support through Awkward Arrays.
- Distinction between geometrical vectors, which have a minimum of attribute and method names, and vectors representing momentum, which have synonyms like `pt` = `rho`, `energy` = `t`, `mass` = `tau`.

This notebook requires Vector, NumPy, Awkward Array, SymPy, and Numba to run.

[1]:
```
import awkward as ak  # at least version 1.2.0rc5
import numba as nb
import numpy as np
import sympy
```

(continues on next page)

(continued from previous page)

```
import vector
```

2.1.1 Constructing a vector or an array of vectors

The easiest way to create one or many vectors is with a helper function:

- `vector.obj` to make a pure Python vector object,
- `vector.arr` to make a NumPy array of vectors (lowercase, like `np.array`),
- `vector.awk` to make an Awkward Array of vectors (uppercase, like `ak.Array`).
- `vector.zip` to make an Awkward Array of vectors (similar to `ak.zip`)

Pure Python vectors

You can directly use the `VectorObject` classes to construct object type vectors:

```
[2]: vector.VectorObject2D(x=1.1, y=2.2)
[2]: VectorObject2D(x=1.1, y=2.2)

[3]: vector.MomentumObject3D(px=1.1, py=2.2, pz=3.3)
[3]: MomentumObject3D(px=1.1, py=2.2, pz=3.3)

[4]: vector.VectorObject4D(x=1.1, y=2.2, eta=3.3, tau=4.4)
[4]: VectorObject4D(x=1.1, y=2.2, eta=3.3, tau=4.4)
```

and so on for every class.

Or, you can use a single wrapper function to construct all possible combinations of object type vectors:

```
[5]: vector.obj(x=3, y=4)  # Cartesian 2D vector
[5]: VectorObject2D(x=3, y=4)

[6]: vector.obj(rho=5, phi=0.9273)  # same in polar coordinates
[6]: VectorObject2D(rho=5, phi=0.9273)

[7]: vector.obj(x=3, y=4).isclose(
         vector.obj(rho=5, phi=0.9273)
     )  # use "isclose" unless they are exactly equal
[7]: True

[8]: vector.obj(x=3, y=4, z=-2)  # Cartesian 3D vector
[8]: VectorObject3D(x=3, y=4, z=-2)

[9]: vector.obj(x=3, y=4, z=-2, t=10)  # Cartesian 4D vector
```

[9]:
```
VectorObject4D(x=3, y=4, z=-2, t=10)
```

[10]:
```
vector.obj(
    rho=5, phi=0.9273, eta=-0.39, t=10
)  # in rho-phi-eta-t cylindrical coordinates
```

[10]:
```
VectorObject4D(rho=5, phi=0.9273, eta=-0.39, t=10)
```

[11]:
```
vector.obj(
    pt=5, phi=0.9273, eta=-0.39, E=10
)  # use momentum-synonyms to get a momentum vector
```

[11]:
```
MomentumObject4D(pt=5, phi=0.9273, eta=-0.39, E=10)
```

[12]:
```
vector.obj(rho=5, phi=0.9273, eta=-0.39, t=10) == vector.obj(
    pt=5, phi=0.9273, eta=-0.390035, E=10
)
```

[12]:
```
False
```

[13]:
```
vector.obj(
    rho=5, phi=0.9273, eta=-0.39, t=10
).tau  # geometrical vectors have to use geometrical names ("tau", not "mass")
```

[13]:
```
8.426194916448265
```

[14]:
```
vector.obj(
    pt=5, phi=0.9273, eta=-0.39, E=10
).mass  # momentum vectors can use momentum names (as well as geometrical ones)
```

[14]:
```
8.426194916448265
```

[15]:
```
vector.obj(
    pt=5, phi=0.9273, theta=1.9513, mass=8.4262
)  # any combination of azimuthal, longitudinal, and temporal coordinates is allowed
```

[15]:
```
MomentumObject4D(pt=5, phi=0.9273, theta=1.9513, mass=8.4262)
```

[16]:
```
vector.obj(x=3, y=4, z=-2, t=10).isclose(
    vector.obj(pt=5, phi=0.9273, theta=1.9513, mass=8.4262)
)
```

[16]:
```
True
```

[17]:
```
# Test instance type for any level of granularity.
(
    isinstance(
        vector.obj(x=1.1, y=2.2), vector.Vector
    ),  # is a vector or array of vectors
    isinstance(vector.obj(x=1.1, y=2.2), vector.Vector2D),  # is 2D (not 3D or 4D)
    isinstance(
        vector.obj(x=1.1, y=2.2), vector.VectorObject
    ),  # is a vector object (not an array)
    isinstance(vector.obj(px=1.1, py=2.2), vector.Momentum),  # has momentum synonyms
```

(continues on next page)

(continued from previous page)

```
    isinstance(
        vector.obj(x=1.1, y=2.2, z=3.3, t=4.4), vector.Planar
    ),  # has transverse plane (2D, 3D, or 4D)
    isinstance(
        vector.obj(x=1.1, y=2.2, z=3.3, t=4.4), vector.Spatial
    ),  # has all spatial coordinates (3D or 4D)
    isinstance(
        vector.obj(x=1.1, y=2.2, z=3.3, t=4.4), vector.Lorentz
    ),  # has temporal coordinates (4D)
    isinstance(
        vector.obj(x=1.1, y=2.2, z=3.3, t=4.4).azimuthal, vector.AzimuthalXY
    ),  # azimuthal coordinate type
    isinstance(
        vector.obj(x=1.1, y=2.2, z=3.3, t=4.4).longitudinal, vector.LongitudinalZ
    ),  # longitudinal coordinate type
    isinstance(
        vector.obj(x=1.1, y=2.2, z=3.3, t=4.4).temporal, vector.TemporalT
    ),  # temporal coordinate type
)
```

[17]:
```
(True, True, True, True, True, True, True, True, True, True)
```

The allowed keyword arguments for 2D vectors are:

- `x` and `y` for Cartesian azimuthal coordinates,
- `px` and `py` for momentum,
- `rho` and `phi` for polar azimuthal coordinates,
- `pt` and `phi` for momentum.

For 3D vectors, you need the above and:

- `z` for the Cartesian longitudinal coordinate,
- `pz` for momentum,
- `theta` for the spherical polar angle (from 0 to π, inclusive),
- `eta` for pseudorapidity, which is a kind of spherical polar angle.

For 4D vectors, you need the above and:

- `t` for the Cartesian temporal coordinate,
- `E` or `energy` to get four-momentum,
- `tau` for the "proper time" (temporal coordinate in the vector's rest coordinate system),
- `M` or `mass` to get four-momentum.

Since momentum vectors have momentum-synonyms *in addition* to the geometrical names, any momentum-synonym will make the whole vector a momentum vector.

If you want to bypass the dimension and coordinate system inference through keyword arguments (e.g. for static typing), you can use specialized constructors:

[18]:
```
vector.VectorObject2D.from_xy(1.1, 2.2)
```

[18]:
```
VectorObject2D(x=1.1, y=2.2)
```

[19]:
```
vector.MomentumObject3D.from_rhophiz(1.1, 2.2, 3.3)
```

[19]:
```
MomentumObject3D(pt=1.1, phi=2.2, pz=3.3)
```

[20]:
```
vector.VectorObject4D.from_xyetatau(1.1, 2.2, 3.3, 4.4)
```

[20]:
```
VectorObject4D(x=1.1, y=2.2, eta=3.3, tau=4.4)
```

and so on, for all combinations of azimuthal, longitudinal, and temporal coordinates, geometric and momentum-flavored.

SymPy vectors

> **Note:** Operations on SymPy vectors are only 100% compatible with numeric vectors (Python, NumPy, and Awkward backends) if the vectors are positive time-like, that is, if `t**2 > x**2 + y**2 + z**2`. The space-like and negative time-like cases have different sign conventions.

You can directly use the `VectorSympy` and `MomentumSympy` classes to construct object type vectors:

[21]:
```
x, y, z, t, px, py, pz, eta, tau = sympy.symbols(
    "x y z t px py pz eta tau",
    real=True,  # see sympy assumptions to add more restrictions on the symbols
)
```

[22]:
```
vector.VectorSympy2D(x=x, y=y)
```

[22]:
```
VectorSympy2D(x=x, y=y)
```

[23]:
```
vector.MomentumSympy3D(px=px, py=py, pz=pz)
```

[23]:
```
MomentumSympy3D(px=px, py=py, pz=pz)
```

[24]:
```
vector.VectorSympy4D(x=x, y=y, eta=eta, tau=tau)
```

[24]:
```
VectorSympy4D(x=x, y=y, eta=eta, tau=tau)
```

and so on for every class.

[25]:
```
# Test instance type for any level of granularity.
(
    # is a vector or array of vectors
    isinstance(vector.VectorSympy2D(x=x, y=y), vector.Vector),
    # is 2D (not 3D or 4D)
    isinstance(vector.VectorSympy2D(x=x, y=y), vector.Vector2D),
    # is a sympy vector (not an array)
    isinstance(vector.VectorSympy2D(x=x, y=y), vector.VectorSympy),
    # has momentum synonyms
    isinstance(vector.MomentumSympy2D(px=px, py=py), vector.Momentum),
    # has transverse plane (2D, 3D, or 4D)
    isinstance(vector.VectorSympy4D(x=x, y=y, z=z, t=t), vector.Planar),
    # has all spatial coordinates (3D or 4D)
    isinstance(vector.VectorSympy4D(x=x, y=y, z=z, t=t), vector.Spatial),
    # has temporal coordinates (4D)
    isinstance(vector.VectorSympy4D(x=x, y=y, z=z, t=t), vector.Lorentz),
```

(continues on next page)

(continued from previous page)

```
    # azimuthal coordinate type
    isinstance(vector.VectorSympy4D(x=x, y=y, z=z, t=t).azimuthal, vector.AzimuthalXY),
    # longitudinal coordinate type
    isinstance(
        vector.VectorSympy4D(x=x, y=y, z=z, t=t).longitudinal, vector.LongitudinalZ
    ),
    # temporal coordinate type
    isinstance(vector.VectorSympy4D(x=x, y=y, z=z, t=t).temporal, vector.TemporalT),
)
```

[25]:
```
(True, True, True, True, True, True, True, True, True, True)
```

Since `VectorSympy2D`, `VectorSympy3D`, `VectorSympy4D`, and their momentum equivalents operate on SymPy expressions, all of the normal SymPy methods and functions work on the results, coordinates, and the vectors.

[26]:
```
sympy.init_session()  # latex printing
```

```
IPython console for SymPy 1.12 (Python 3.11.5-64-bit) (ground types: python)

These commands were executed:
>>> from sympy import *
>>> x, y, z, t = symbols('x y z t')
>>> k, m, n = symbols('k m n', integer=True)
>>> f, g, h = symbols('f g h', cls=Function)
>>> init_printing()

Documentation can be found at https://docs.sympy.org/1.12/
```

[27]:
```
v1 = vector.VectorSympy2D(x=x, y=y)
sympy.Eq(v1.rho, sympy.sqrt(x**2 + y**2))
```

[27]:
True

[28]:
```
v2 = vector.VectorSympy4D(x=x, y=y, z=z, t=t)
```

[29]:
```
v2.to_rhophithetatau().tau
```

[29]:
$$\sqrt{|-t^2 + x^2 + y^2 + z^2|}$$

[30]:
```
values = {x: 3, y: 2, z: 1, t: 10}  # t**2 > x**2 + y**2 + z**2
```

[31]:
```
v2.is_timelike()
```

[31]:
$$t^2 - x^2 - y^2 - z^2 > 0$$

[32]:
```
v2.is_timelike().subs(values)
```

[32]:
True

[33]:
```
v2.to_rhophithetatau().tau.subs(values).evalf()
```

[33]:
$$9.2736184954957$$

[34]:
```
v2.boost(v2.to_beta3())
```

[34]:
```
VectorSympy4D(x=x*(1 + x**2/(t**2*(1 + 1/sqrt(1 - x**2/t**2 - y**2/t**2 - z**2/t**2))*(1
↪- x**2/t**2 - y**2/t**2 - z**2/t**2))) + x/sqrt(1 - x**2/t**2 - y**2/t**2 - z**2/t**2)
↪+ x*y**2/(t**2*(1 + 1/sqrt(1 - x**2/t**2 - y**2/t**2 - z**2/t**2))*(1 - x**2/t**2 -
↪y**2/t**2 - z**2/t**2)) + x*z**2/(t**2*(1 + 1/sqrt(1 - x**2/t**2 - y**2/t**2 - z**2/
↪t**2))*(1 - x**2/t**2 - y**2/t**2 - z**2/t**2)), y=y*(1 + y**2/(t**2*(1 + 1/sqrt(1 -
↪x**2/t**2 - y**2/t**2 - z**2/t**2))*(1 - x**2/t**2 - y**2/t**2 - z**2/t**2))) + y/
↪sqrt(1 - x**2/t**2 - y**2/t**2 - z**2/t**2) + x**2*y/(t**2*(1 + 1/sqrt(1 - x**2/t**2 -
↪y**2/t**2 - z**2/t**2))*(1 - x**2/t**2 - y**2/t**2 - z**2/t**2)) + y*z**2/(t**2*(1 + 1/
↪sqrt(1 - x**2/t**2 - y**2/t**2 - z**2/t**2))*(1 - x**2/t**2 - y**2/t**2 - z**2/t**2)),
↪z=z*(1 + z**2/(t**2*(1 + 1/sqrt(1 - x**2/t**2 - y**2/t**2 - z**2/t**2))*(1 - x**2/t**2
↪- y**2/t**2 - z**2/t**2))) + z/sqrt(1 - x**2/t**2 - y**2/t**2 - z**2/t**2) + x**2*z/
↪(t**2*(1 + 1/sqrt(1 - x**2/t**2 - y**2/t**2 - z**2/t**2))*(1 - x**2/t**2 - y**2/t**2 -
↪z**2/t**2)) + y**2*z/(t**2*(1 + 1/sqrt(1 - x**2/t**2 - y**2/t**2 - z**2/t**2))*(1 -
↪x**2/t**2 - y**2/t**2 - z**2/t**2)), t=t/sqrt(1 - x**2/t**2 - y**2/t**2 - z**2/t**2) +
↪x**2/(t*sqrt(1 - x**2/t**2 - y**2/t**2 - z**2/t**2)) + y**2/(t*sqrt(1 - x**2/t**2 -
↪y**2/t**2 - z**2/t**2)) + z**2/(t*sqrt(1 - x**2/t**2 - y**2/t**2 - z**2/t**2)))
```

[35]:
```
v2.boost(v2.to_beta3()).t
```

[35]:
$$\frac{t}{\sqrt{1 - \frac{x^2}{t^2} - \frac{y^2}{t^2} - \frac{z^2}{t^2}}} + \frac{x^2}{t\sqrt{1 - \frac{x^2}{t^2} - \frac{y^2}{t^2} - \frac{z^2}{t^2}}} + \frac{y^2}{t\sqrt{1 - \frac{x^2}{t^2} - \frac{y^2}{t^2} - \frac{z^2}{t^2}}} + \frac{z^2}{t\sqrt{1 - \frac{x^2}{t^2} - \frac{y^2}{t^2} - \frac{z^2}{t^2}}}$$

[36]:
```
v2.boost(v2.to_beta3()).t.simplify()
```

[36]:
$$\frac{t\sqrt{\frac{t^2 - x^2 - y^2 - z^2}{t^2}}\left(t^2 + x^2 + y^2 + z^2\right)}{t^2 - x^2 - y^2 - z^2}$$

[37]:
```
v2.boost(v2.to_beta3()).t.subs(values)
```

[37]:
$$\frac{57\sqrt{86}}{43}$$

[38]:
```
v2.boost(v2.to_beta3()).t.subs(values).evalf()
```

[38]:
```
12.292936145192
```

All of the keyword arguments and rules that apply to `vector.obj` construction apply to `vector.VectorSympyND` and `vector.MomentumObjectND` objects.

NumPy arrays of vectors

You can directly use the `VectorNumpy` classes to construct object type vectors:

[39]:
```
# NumPy-like arguments (literally passed through to NumPy)
vector.VectorNumpy2D(
    [(1.1, 2.1), (1.2, 2.2), (1.3, 2.3), (1.4, 2.4), (1.5, 2.5)],
    dtype=[("x", float), ("y", float)],
)
```

[39]:
```
VectorNumpy2D([(1.1, 2.1), (1.2, 2.2), (1.3, 2.3), (1.4, 2.4), (1.5, 2.5)],
              dtype=[('x', '<f8'), ('y', '<f8')])
```

[40]:
```
# Pandas-like arguments (dict from names to column arrays)
vector.VectorNumpy2D({"x": [1.1, 1.2, 1.3, 1.4, 1.5], "y": [2.1, 2.2, 2.3, 2.4, 2.5]})
```

[40]:
```
VectorNumpy2D([(1.1, 2.1), (1.2, 2.2), (1.3, 2.3), (1.4, 2.4), (1.5, 2.5)],
              dtype=[('x', '<f8'), ('y', '<f8')])
```

[41]:
```
# As with objects, the coordinate system and dimension is taken from the names of the␣
↪fields.
vector.VectorNumpy4D(
    {
        "x": [1.1, 1.2, 1.3, 1.4, 1.5],
        "y": [2.1, 2.2, 2.3, 2.4, 2.5],
        "z": [3.1, 3.2, 3.3, 3.4, 3.5],
        "t": [4.1, 4.2, 4.3, 4.4, 4.5],
    }
)
```

[41]:
```
VectorNumpy4D([(1.1, 2.1, 3.1, 4.1), (1.2, 2.2, 3.2, 4.2), (1.3, 2.3, 3.3, 4.3),
               (1.4, 2.4, 3.4, 4.4), (1.5, 2.5, 3.5, 4.5)],
              dtype=[('x', '<f8'), ('y', '<f8'), ('z', '<f8'), ('t', '<f8')])
```

and so on for every class.

Or, you can use a single wrapper function to construct all possible combinations of NumPy type vectors:

[42]:
```
# NumPy-like arguments (literally passed through to NumPy)
vector.array(
    [(1.1, 2.1), (1.2, 2.2), (1.3, 2.3), (1.4, 2.4), (1.5, 2.5)],
    dtype=[("x", float), ("y", float)],
)
```

[42]:
```
VectorNumpy2D([(1.1, 2.1), (1.2, 2.2), (1.3, 2.3), (1.4, 2.4), (1.5, 2.5)],
              dtype=[('x', '<f8'), ('y', '<f8')])
```

[43]:
```
# Pandas-like arguments (dict from names to column arrays)
vector.array({"x": [1.1, 1.2, 1.3, 1.4, 1.5], "y": [2.1, 2.2, 2.3, 2.4, 2.5]})
```

[43]:
```
VectorNumpy2D([(1.1, 2.1), (1.2, 2.2), (1.3, 2.3), (1.4, 2.4), (1.5, 2.5)],
              dtype=[('x', '<f8'), ('y', '<f8')])
```

[44]:
```
# As with objects, the coordinate system and dimension is taken from the names of the␣
↪fields.
vector.array(
    {
        "x": [1.1, 1.2, 1.3, 1.4, 1.5],
        "y": [2.1, 2.2, 2.3, 2.4, 2.5],
        "z": [3.1, 3.2, 3.3, 3.4, 3.5],
        "t": [4.1, 4.2, 4.3, 4.4, 4.5],
    }
)
```

[44]:
```
VectorNumpy4D([(1.1, 2.1, 3.1, 4.1), (1.2, 2.2, 3.2, 4.2), (1.3, 2.3, 3.3, 4.3),
               (1.4, 2.4, 3.4, 4.4), (1.5, 2.5, 3.5, 4.5)],
              dtype=[('x', '<f8'), ('y', '<f8'), ('z', '<f8'), ('t', '<f8')])
```

[45]:
```
vector.array(
    {
        "pt": [1.1, 1.2, 1.3, 1.4, 1.5],
        "phi": [2.1, 2.2, 2.3, 2.4, 2.5],
        "eta": [3.1, 3.2, 3.3, 3.4, 3.5],
        "M": [4.1, 4.2, 4.3, 4.4, 4.5],
    }
)
```

[45]:
```
MomentumNumpy4D([(1.1, 2.1, 3.1, 4.1), (1.2, 2.2, 3.2, 4.2), (1.3, 2.3, 3.3, 4.3),
                 (1.4, 2.4, 3.4, 4.4), (1.5, 2.5, 3.5, 4.5)],
                dtype=[('rho', '<f8'), ('phi', '<f8'), ('eta', '<f8'), ('tau', '<f8')])
```

Existing NumPy arrays can be viewed as arrays of vectors, but it needs to be a structured array with recognized field names.

[46]:
```
# NumPy array         # interpret groups of four values as named fields              #␣
↪give it vector properties and methods
np.arange(0, 24, 0.1).view(
    [("x", float), ("y", float), ("z", float), ("t", float)]
).view(vector.VectorNumpy4D)
```

[46]:
```
VectorNumpy4D([( 0. ,  0.1,  0.2,  0.3), ( 0.4,  0.5,  0.6,  0.7),
               ( 0.8,  0.9,  1. ,  1.1), ( 1.2,  1.3,  1.4,  1.5),
               ( 1.6,  1.7,  1.8,  1.9), ( 2. ,  2.1,  2.2,  2.3),
               ( 2.4,  2.5,  2.6,  2.7), ( 2.8,  2.9,  3. ,  3.1),
               ( 3.2,  3.3,  3.4,  3.5), ( 3.6,  3.7,  3.8,  3.9),
               ( 4. ,  4.1,  4.2,  4.3), ( 4.4,  4.5,  4.6,  4.7),
               ( 4.8,  4.9,  5. ,  5.1), ( 5.2,  5.3,  5.4,  5.5),
               ( 5.6,  5.7,  5.8,  5.9), ( 6. ,  6.1,  6.2,  6.3),
               ( 6.4,  6.5,  6.6,  6.7), ( 6.8,  6.9,  7. ,  7.1),
               ( 7.2,  7.3,  7.4,  7.5), ( 7.6,  7.7,  7.8,  7.9),
               ( 8. ,  8.1,  8.2,  8.3), ( 8.4,  8.5,  8.6,  8.7),
               ( 8.8,  8.9,  9. ,  9.1), ( 9.2,  9.3,  9.4,  9.5),
               ( 9.6,  9.7,  9.8,  9.9), (10. , 10.1, 10.2, 10.3),
               (10.4, 10.5, 10.6, 10.7), (10.8, 10.9, 11. , 11.1),
               (11.2, 11.3, 11.4, 11.5), (11.6, 11.7, 11.8, 11.9),
               (12. , 12.1, 12.2, 12.3), (12.4, 12.5, 12.6, 12.7),
               (12.8, 12.9, 13. , 13.1), (13.2, 13.3, 13.4, 13.5),
               (13.6, 13.7, 13.8, 13.9), (14. , 14.1, 14.2, 14.3),
               (14.4, 14.5, 14.6, 14.7), (14.8, 14.9, 15. , 15.1),
               (15.2, 15.3, 15.4, 15.5), (15.6, 15.7, 15.8, 15.9),
               (16. , 16.1, 16.2, 16.3), (16.4, 16.5, 16.6, 16.7),
               (16.8, 16.9, 17. , 17.1), (17.2, 17.3, 17.4, 17.5),
               (17.6, 17.7, 17.8, 17.9), (18. , 18.1, 18.2, 18.3),
               (18.4, 18.5, 18.6, 18.7), (18.8, 18.9, 19. , 19.1),
               (19.2, 19.3, 19.4, 19.5), (19.6, 19.7, 19.8, 19.9),
               (20. , 20.1, 20.2, 20.3), (20.4, 20.5, 20.6, 20.7),
               (20.8, 20.9, 21. , 21.1), (21.2, 21.3, 21.4, 21.5),
               (21.6, 21.7, 21.8, 21.9), (22. , 22.1, 22.2, 22.3),
               (22.4, 22.5, 22.6, 22.7), (22.8, 22.9, 23. , 23.1),
               (23.2, 23.3, 23.4, 23.5), (23.6, 23.7, 23.8, 23.9)],
              dtype=[('x', '<f8'), ('y', '<f8'), ('z', '<f8'), ('t', '<f8')])
```

Since `VectorNumpy2D`, `VectorNumpy3D`, `VectorNumpy4D`, and their momentum equivalents are NumPy array sub-

classes, all of the normal NumPy methods and functions work on them.

[47]:
```
np.arange(0, 24, 0.1).view(
    [("x", float), ("y", float), ("z", float), ("t", float)]
).view(vector.VectorNumpy4D).reshape(6, 5, 2)
```

[47]:
```
VectorNumpy4D([[[( 0. ,  0.1,  0.2,  0.3), ( 0.4,  0.5,  0.6,  0.7)],
                [( 0.8,  0.9,  1. ,  1.1), ( 1.2,  1.3,  1.4,  1.5)],
                [( 1.6,  1.7,  1.8,  1.9), ( 2. ,  2.1,  2.2,  2.3)],
                [( 2.4,  2.5,  2.6,  2.7), ( 2.8,  2.9,  3. ,  3.1)],
                [( 3.2,  3.3,  3.4,  3.5), ( 3.6,  3.7,  3.8,  3.9)]],

               [[( 4. ,  4.1,  4.2,  4.3), ( 4.4,  4.5,  4.6,  4.7)],
                [( 4.8,  4.9,  5. ,  5.1), ( 5.2,  5.3,  5.4,  5.5)],
                [( 5.6,  5.7,  5.8,  5.9), ( 6. ,  6.1,  6.2,  6.3)],
                [( 6.4,  6.5,  6.6,  6.7), ( 6.8,  6.9,  7. ,  7.1)],
                [( 7.2,  7.3,  7.4,  7.5), ( 7.6,  7.7,  7.8,  7.9)]],

               [[( 8. ,  8.1,  8.2,  8.3), ( 8.4,  8.5,  8.6,  8.7)],
                [( 8.8,  8.9,  9. ,  9.1), ( 9.2,  9.3,  9.4,  9.5)],
                [( 9.6,  9.7,  9.8,  9.9), (10. , 10.1, 10.2, 10.3)],
                [(10.4, 10.5, 10.6, 10.7), (10.8, 10.9, 11. , 11.1)],
                [(11.2, 11.3, 11.4, 11.5), (11.6, 11.7, 11.8, 11.9)]],

               [[(12. , 12.1, 12.2, 12.3), (12.4, 12.5, 12.6, 12.7)],
                [(12.8, 12.9, 13. , 13.1), (13.2, 13.3, 13.4, 13.5)],
                [(13.6, 13.7, 13.8, 13.9), (14. , 14.1, 14.2, 14.3)],
                [(14.4, 14.5, 14.6, 14.7), (14.8, 14.9, 15. , 15.1)],
                [(15.2, 15.3, 15.4, 15.5), (15.6, 15.7, 15.8, 15.9)]],

               [[(16. , 16.1, 16.2, 16.3), (16.4, 16.5, 16.6, 16.7)],
                [(16.8, 16.9, 17. , 17.1), (17.2, 17.3, 17.4, 17.5)],
                [(17.6, 17.7, 17.8, 17.9), (18. , 18.1, 18.2, 18.3)],
                [(18.4, 18.5, 18.6, 18.7), (18.8, 18.9, 19. , 19.1)],
                [(19.2, 19.3, 19.4, 19.5), (19.6, 19.7, 19.8, 19.9)]],

               [[(20. , 20.1, 20.2, 20.3), (20.4, 20.5, 20.6, 20.7)],
                [(20.8, 20.9, 21. , 21.1), (21.2, 21.3, 21.4, 21.5)],
                [(21.6, 21.7, 21.8, 21.9), (22. , 22.1, 22.2, 22.3)],
                [(22.4, 22.5, 22.6, 22.7), (22.8, 22.9, 23. , 23.1)],
                [(23.2, 23.3, 23.4, 23.5), (23.6, 23.7, 23.8, 23.9)]]],
              dtype=[('x', '<f8'), ('y', '<f8'), ('z', '<f8'), ('t', '<f8')])
```

All of the keyword arguments and rules that apply to `vector.obj` construction apply to `vector.array` dtypes.

Geometrical names are used in the dtype, even if momentum-synonyms are used in construction.

[48]:
```
vector.array(
    {"px": [1, 2, 3, 4], "py": [1.1, 2.2, 3.3, 4.4], "pz": [0.1, 0.2, 0.3, 0.4]}
)
```

[48]:
```
MomentumNumpy3D([(1., 1.1, 0.1), (2., 2.2, 0.2), (3., 3.3, 0.3), (4., 4.4, 0.4)],
                dtype=[('x', '<f8'), ('y', '<f8'), ('z', '<f8')])
```

Awkward Arrays of vectors

Awkward Arrays are arrays with more complex data structures than NumPy allows, such as variable-length lists, nested records, missing and even heterogeneous data (multiple data types: use sparingly).

The `vector.Array` function behaves exactly like the ak.Array constructor, except that it makes arrays of vectors.

[49]:
```
vector.Array(
    [
        [{"x": 1, "y": 1.1, "z": 0.1}, {"x": 2, "y": 2.2, "z": 0.2}],
        [],
        [{"x": 3, "y": 3.3, "z": 0.3}],
        [
            {"x": 4, "y": 4.4, "z": 0.4},
            {"x": 5, "y": 5.5, "z": 0.5},
            {"x": 6, "y": 6.6, "z": 0.6},
        ],
    ]
)
```

[49]:
```
<VectorArray3D [[{x: 1, y: 1.1, ...}, {...}], ...] type='4 * var * Vector3D...'>
```

If you want *any* records named "`Vector2D`", "`Vector3D`", "`Vector4D`", "`Momentum2D`", "`Momentum3D`", or "`Momentum4D`" to be interpreted as vectors, register the behaviors globally.

[50]:
```
vector.register_awkward()
```

[51]:
```
ak.Array(
    [
        [{"x": 1, "y": 1.1, "z": 0.1}, {"x": 2, "y": 2.2, "z": 0.2}],
        [],
        [{"x": 3, "y": 3.3, "z": 0.3}],
        [
            {"x": 4, "y": 4.4, "z": 0.4},
            {"x": 5, "y": 5.5, "z": 0.5},
            {"x": 6, "y": 6.6, "z": 0.6},
        ],
    ],
    with_name="Vector3D",
)
```

[51]:
```
<VectorArray3D [[{x: 1, y: 1.1, ...}, {...}], ...] type='4 * var * Vector3D...'>
```

All of the keyword arguments and rules that apply to `vector.obj` construction apply to `vector.Array` field names.

Finally, the `VectorAwkward` mixins can be subclassed to create custom vector classes. The awkward behavior classes and projections must be named as `*Array`. For example, `coffea` uses the following names - `TwoVectorArray`, `ThreeVectorArray`, `PolarTwoVectorArray`, `SphericalThreeVectorArray`, ...

2.1.2 Vector properties

Any geometrical coordinate can be computed from vectors in any coordinate system; they'll be provided or computed as needed.

[52]:
```
vector.obj(x=3, y=4).rho
```
[52]:
```
5.0
```

[53]:
```
vector.obj(rho=5, phi=0.9273).x
```
[53]:
```
2.99998087197215
```

[54]:
```
vector.obj(rho=5, phi=0.9273).y
```
[54]:
```
4.00001434594943
```

[55]:
```
vector.obj(x=1, y=2, z=3).theta
```
[55]:
```
0.640522312679424
```

[56]:
```
vector.obj(x=1, y=2, z=3).eta
```
[56]:
```
1.10358684156015
```

Some properties are not coordinates, but derived from them.

[57]:
```
vector.obj(x=1, y=2, z=3).costheta
```
[57]:
```
0.801783725737273
```

[58]:
```
vector.obj(x=1, y=2, z=3).mag  # spatial magnitude
```
[58]:
```
3.74165738677394
```

[59]:
```
vector.obj(x=1, y=2, z=3).mag2  # spatial magnitude squared
```
[59]:
```
14
```

These properties are provided because they can be computed faster or with more numerical stability in different coordinate systems. For instance, the magnitude ignores `phi` in polar coordinates.

[60]:
```
vector.obj(rho=3, phi=0.123456789, z=4).mag2
```
[60]:
```
25
```

Momentum vectors have geometrical properties as well as their momentum-synonyms.

[61]:
```
vector.obj(px=3, py=4).rho
```
[61]:
```
5.0
```

[62]:
```
vector.obj(px=3, py=4).pt
```
[62]:
```
5.0
```

[63]:
```
vector.obj(x=1, y=2, z=3, E=4).tau
```
[63]:
```
1.4142135623731
```

[64]:
```
vector.obj(x=1, y=2, z=3, E=4).mass
```

[64]:
```
1.4142135623731
```

Here's the key thing: *arrays of vectors return arrays of coordinates*.

[65]:
```
vector.array(
    {
        "x": [1.0, 2.0, 3.0, 4.0, 5.0],
        "y": [1.1, 2.2, 3.3, 4.4, 5.5],
        "z": [0.1, 0.2, 0.3, 0.4, 0.5],
    }
).theta
```

[65]:
```
array([1.50363023, 1.50363023, 1.50363023, 1.50363023, 1.50363023])
```

[66]:
```
vector.Array(
    [
        [{"x": 1, "y": 1.1, "z": 0.1}, {"x": 2, "y": 2.2, "z": 0.2}],
        [],
        [{"x": 3, "y": 3.3, "z": 0.3}],
        [{"x": 4, "y": 4.4, "z": 0.4}, {"x": 5, "y": 5.5, "z": 0.5}],
    ]
).theta
```

[66]:
```
<Array [[1.5, 1.5], [], [1.5], [1.5, 1.5]] type='4 * var * float64'>
```

[67]:
```
# Make a large, random NumPy array of 3D momentum vectors.
array = (
    np.random.normal(0, 1, 150)
    .view([(x, float) for x in ("x", "y", "z")])
    .view(vector.MomentumNumpy3D)
    .reshape(5, 5, 2)
)
array
```

[67]:
```
MomentumNumpy3D([[[( 0.55677765, -2.39134767, -1.14009875e+00),
                   (-0.4099286 , -1.35111044,  1.22805909e+00)],
                  [( 2.17202778, -0.68150151,  1.67793905e-01),
                   (-0.41973476, -1.62000792, -2.62404832e-04)],
                  [(-1.51641767, -1.40147683,  2.47076578e+00),
                   ( 0.60257785, -1.30590029, -6.77753691e-01)],
                  [( 0.19640056,  0.20738716,  2.04561027e-02),
                   (-0.53479171,  0.94471872,  7.09228434e-02)],
                  [( 0.01832378,  1.12336773, -9.97689716e-01),
                   (-1.67101189,  0.62237498, -6.39758959e-02)]],

                 [[( 0.87762728, -1.98374752, -2.01469599e+00),
                   ( 0.17471933,  0.12161887, -9.42606081e-02)],
                  [(-0.5445472 , -0.8928596 , -6.90405216e-01),
                   ( 0.29222871, -0.14303945, -7.06930314e-01)],
                  [( 0.11103332, -0.81876775,  2.21759321e-01),
                   ( 0.6930197 ,  0.44511978,  3.74892308e-01)],
                  [( 0.47797932, -0.54189062, -1.06199035e+00),
                   ( 0.70067255, -0.01567734,  4.51905779e-01)],
                  [(-0.98069469, -0.67226787,  1.63755227e+00),
```

(continues on next page)

(continued from previous page)

```
                  ( 1.33964903, -1.01568176,  9.33791315e-01)]],

                [[(-0.79781737,  0.8604048 ,  1.75923001e-01),
                  (-1.27055185,  0.28886057,  8.56541197e-01)],
                 [(-1.05366515, -0.15041122,  7.55757703e-01),
                  ( 1.65815748,  1.32132905, -1.46950305e+00)],
                 [(-0.67879973, -1.95059887, -1.15938843e-01),
                  (-1.7877064 , -0.69352448,  3.20265686e-01)],
                 [( 0.28149666,  0.54879607, -7.82537808e-01),
                  ( 0.31366026,  0.36907726,  3.36517183e-01)],
                 [(-1.01885702, -0.66434627,  6.41372208e-01),
                  (-0.77216596,  0.08913472, -2.49368534e-01)]],

                [[(-0.75470986, -0.15824692, -2.04355139e-01),
                  ( 0.24682491, -0.44316793, -2.55907965e-01)],
                 [(-1.14906251,  1.08234259, -4.37065854e-01),
                  ( 0.39882584, -0.80339686,  8.25692598e-01)],
                 [(-0.2832258 ,  1.04610965,  4.32860052e-01),
                  (-0.27051201, -1.54016307, -6.61530929e-01)],
                 [(-0.10674002,  0.14105683,  1.27337139e+00),
                  (-0.4728045 ,  0.22134932, -1.07465967e+00)],
                 [(-0.10652267,  0.00870547,  1.32620623e+00),
                  (-0.96161573,  0.4993631 ,  5.57284082e-01)]],

                [[( 0.51468134,  0.48772909, -2.59407633e-01),
                  ( 0.61031019,  1.25211695, -5.04208824e-01)],
                 [(-0.31302188, -0.07605632, -1.93811954e+00),
                  ( 1.02092718,  0.26294964, -9.27338521e-01)],
                 [( 1.88041867, -0.91344964, -6.85620574e-01),
                  (-0.07825839, -0.14948375, -1.52505471e-01)],
                 [(-1.71248754,  0.25225344, -1.28889867e+00),
                  (-0.96428705,  0.87685202, -4.98042758e-01)],
                 [(-0.45582323,  0.40835731,  1.82438535e-01),
                  ( 0.58168782,  1.04561222, -7.12156826e-01)]]],
                dtype=[('x', '<f8'), ('y', '<f8'), ('z', '<f8')])
```

[68]:
```
# Get the transverse momentum of each one.
array.pt
```

[68]:
```
array([[[2.45530956, 1.41192807],
        [2.27643339, 1.67350021],
        [2.06486316, 1.43821961],
        [0.2856267 , 1.08558539],
        [1.12351716, 1.78315208]],

       [[2.16921272, 0.21288023],
        [1.04581543, 0.32535812],
        [0.82626208, 0.82365522],
        [0.72257157, 0.70084792],
        [1.18899376, 1.68115108]],

       [[1.17337504, 1.30297446],
```

(continues on next page)

(continued from previous page)

```
        [1.06434664, 2.12023506],
        [2.06533412, 1.91751672],
        [0.61677994, 0.48435605],
        [1.21631641, 0.77729356]],

       [[0.77112195, 0.50726753],
        [1.57854684, 0.89694401],
        [1.08377223, 1.56373879],
        [0.1768911 , 0.52205327],
        [0.10687781, 1.08354433]],

       [[0.70906738, 1.39293768],
        [0.32212926, 1.05424609],
        [2.09054171, 0.16872986],
        [1.7309666 , 1.30334914],
        [0.61198898, 1.19652231]]])
```

[69]:
```
# The array and its components have the same shape.
array.shape
```

[69]: $(5, 5, 2)$

[70]:
```
array.pt.shape
```

[70]: $(5, 5, 2)$

[71]:
```
# Make a large, random Awkward Array of 3D momentum vectors.
array = vector.Array(
    [
        [
            {x: np.random.normal(0, 1) for x in ("px", "py", "pz")}
            for inner in range(np.random.poisson(1.5))
        ]
        for outer in range(50)
    ]
)
array
```

[71]:
```
<MomentumArray3D [[], ..., [{x: 0.463, ...}, {...}]] type='50 * var * Momen...'>
```

[72]:
```
# Get the transverse momentum of each one, in the same nested structure.
array.pt
```

[72]:
```
<Array [[], [2.32], ..., [0.283], [0.472, 2.39]] type='50 * var * float64'>
```

[73]:
```
# The array and its components have the same list lengths (and can therefore be used␣
↪together in subsequent calculations).
ak.num(array)
```

[73]:
```
<Array [0, 1, 2, 2, 0, 2, 1, 0, ..., 0, 1, 1, 4, 4, 2, 1, 2] type='50 * int64'>
```

[74]:
```
ak.num(array.pt)
```

[74]:
```
<Array [0, 1, 2, 2, 0, 2, 1, 0, ..., 0, 1, 1, 4, 4, 2, 1, 2] type='50 * int64'>
```

2.1.3 Vector methods

Vector methods require arguments (in parentheses), which may be scalars or other vectors, depending on the calculation.

[75]:
```
vector.obj(x=3, y=4).rotateZ(0.1)
```

[75]:
```
VectorObject2D(x=2.585678829246765, y=4.279516911052588)
```

[76]:
```
vector.obj(rho=5, phi=0.4).rotateZ(0.1)
```

[76]:
```
VectorObject2D(rho=5, phi=0.5)
```

[77]:
```
# Broadcasts a scalar rotation angle of 0.5 to all elements of the NumPy array.
print(
    vector.array({"rho": [1, 2, 3, 4, 5], "phi": [0.1, 0.2, 0.3, 0.4, 0.5]}).rotateZ(
        0.5
    )
)
```

```
[(1., 0.6) (2., 0.7) (3., 0.8) (4., 0.9) (5., 1. )]
```

[78]:
```
# Matches each rotation angle to an element of the NumPy array.
print(
    vector.array({"rho": [1, 2, 3, 4, 5], "phi": [0.1, 0.2, 0.3, 0.4, 0.5]}).rotateZ(
        np.array([0.1, 0.2, 0.3, 0.4, 0.5])
    )
)
```

```
[(1., 0.2) (2., 0.4) (3., 0.6) (4., 0.8) (5., 1. )]
```

[79]:
```
# Broadcasts a scalar rotation angle of 0.5 to all elements of the Awkward Array.
print(
    vector.Array(
        [[{"rho": 1, "phi": 0.1}, {"rho": 2, "phi": 0.2}], [], [{"rho": 3, "phi": 0.3}]]
    ).rotateZ(0.5)
)
```

```
[[{rho: 1, phi: 0.6}, {rho: 2, phi: 0.7}], [], [{rho: 3, phi: 0.8}]]
```

[80]:
```
# Broadcasts a rotation angle of 0.1 to both elements of the first list, 0.2 to the␣
↪empty list, and 0.3 to the only element of the last list.
print(
    vector.Array(
        [[{"rho": 1, "phi": 0.1}, {"rho": 2, "phi": 0.2}], [], [{"rho": 3, "phi": 0.3}]]
    ).rotateZ([0.1, 0.2, 0.3])
)
```

```
[[{rho: 1, phi: 0.2}, {rho: 2, phi: 0.3}], [], [{rho: 3, phi: 0.6}]]
```

[81]:
```
# Matches each rotation angle to an element of the Awkward Array.
print(
```

(continues on next page)

(continued from previous page)

```
    vector.Array(
        [[{"rho": 1, "phi": 0.1}, {"rho": 2, "phi": 0.2}], [], [{"rho": 3, "phi": 0.3}]]
    ).rotateZ([[0.1, 0.2], [], [0.3]])
)
```

```
[[{rho: 1, phi: 0.2}, {rho: 2, phi: 0.4}], [], [{rho: 3, phi: 0.6}]]
```

Some methods are equivalent to binary operators.

[82]:
```
vector.obj(x=3, y=4).scale(10)
```

[82]:
```
VectorObject2D(x=30, y=40)
```

[83]:
```
vector.obj(x=3, y=4) * 10
```

[83]:
```
VectorObject2D(x=30, y=40)
```

[84]:
```
10 * vector.obj(x=3, y=4)
```

[84]:
```
VectorObject2D(x=30, y=40)
```

[85]:
```
vector.obj(rho=5, phi=0.5) * 10
```

[85]:
```
VectorObject2D(rho=50, phi=0.5)
```

Some methods involve more than one vector.

[86]:
```
vector.obj(x=1, y=2).add(vector.obj(x=5, y=5))
```

[86]:
```
VectorObject2D(x=6, y=7)
```

[87]:
```
vector.obj(x=1, y=2) + vector.obj(x=5, y=5)
```

[87]:
```
VectorObject2D(x=6, y=7)
```

[88]:
```
vector.obj(x=1, y=2).dot(vector.obj(x=5, y=5))
```

[88]:
```
15
```

[89]:
```
vector.obj(x=1, y=2) @ vector.obj(x=5, y=5)
```

[89]:
```
15
```

The vectors can use different coordinate systems. Conversions are necessary, but minimized for speed and numeric stability.

[90]:
```
vector.obj(x=3, y=4) @ vector.obj(x=6, y=8)  # both are Cartesian, dot product is exact
```

[90]:
```
50
```

[91]:
```
vector.obj(rho=5, phi=0.9273) @ vector.obj(
    x=6, y=8
)  # one is polar, dot product is approximate
```

[91]:
```
49.9999999994283
```

[92]:
```
vector.obj(x=3, y=4) @ vector.obj(
    rho=10, phi=0.9273
)  # one is polar, dot product is approximate
```

[92]:
```
49.9999999994283
```

[93]:
```
vector.obj(rho=5, phi=0.9273) @ vector.obj(
    rho=10, phi=0.9273
)  # both are polar, a formula that depends on phi differences is used
```

[93]:
```
50.0
```

In Python, some "operators" are actually built-in functions, such as `abs`.

[94]:
```
abs(vector.obj(x=3, y=4))
```

[94]:
```
5.0
```

Note that `abs` returns

- `rho` for 2D vectors
- `mag` for 3D vectors
- `tau` (`mass`) for 4D vectors

Use the named properties when you want magnitude in a specific number of dimensions; use `abs` when you want the magnitude for any number of dimensions.

The vectors can be from different backends. Normal rules for broadcasting Python numbers, NumPy arrays, and Awkward Arrays apply.

[95]:
```
vector.array({"x": [1, 2, 3, 4, 5], "y": [0.1, 0.2, 0.3, 0.4, 0.5]}) + vector.obj(
    x=10, y=5
)
```

[95]:
```
VectorNumpy2D([(11., 5.1), (12., 5.2), (13., 5.3), (14., 5.4), (15., 5.5)],
              dtype=[('x', '<f8'), ('y', '<f8')])
```

[96]:
```
(
    vector.Array(
        [  # an Awkward Array of vectors
            [{"x": 1, "y": 1.1}, {"x": 2, "y": 2.2}],
            [],
            [{"x": 3, "y": 3.3}],
            [{"x": 4, "y": 4.4}, {"x": 5, "y": 5.5}],
        ]
    )
    + vector.obj(x=10, y=5)  # and a single vector object
)
```

[96]:
```
<VectorArray2D [[{x: 11, y: 6.1}, {...}], ..., [...]] type='4 * var * Vecto...'>
```

[97]:
```
(
    vector.Array(
        [  # an Awkward Array of vectors
            [{"x": 1, "y": 1.1}, {"x": 2, "y": 2.2}],
            [],
```

(continues on next page)

(continued from previous page)

```
            [{"x": 3, "y": 3.3}],
            [{"x": 4, "y": 4.4}, {"x": 5, "y": 5.5}],
        ]
    )
    + vector.array(
        {"x": [4, 3, 2, 1], "y": [0.1, 0.1, 0.1, 0.1]}
    )  # and a NumPy array of vectors
)
```

[97]:
```
<VectorArray2D [[{x: 5, y: 1.2}, {...}], ..., [...]] type='4 * var * Vector...'>
```

Some operations are defined for 2D or 3D vectors, but are usable on higher-dimensional vectors because the additional components can be ignored or are passed through unaffected.

[98]:
```
vector.obj(rho=1, phi=0.5).deltaphi(
    vector.obj(rho=2, phi=0.3)
)  # deltaphi is a planar operation (defined on the transverse plane)
```

[98]:
```
0.2
```

[99]:
```
vector.obj(rho=1, phi=0.5, z=10).deltaphi(
    vector.obj(rho=2, phi=0.3, theta=1.4)
)  # but we can use it on 3D vectors
```

[99]:
```
0.2
```

[100]:
```
vector.obj(rho=1, phi=0.5, z=10, t=100).deltaphi(
    vector.obj(rho=2, phi=0.3, theta=1.4, tau=1000)
)  # and 4D vectors
```

[100]:
```
0.2
```

[101]:
```
vector.obj(rho=1, phi=0.5).deltaphi(
    vector.obj(rho=2, phi=0.3, theta=1.4, tau=1000)
)  # and mixed dimensionality
```

[101]:
```
0.2
```

This is especially useful for giving 4D vectors all the capabilities of 3D vectors.

[102]:
```
vector.obj(x=1, y=2, z=3).rotateX(np.pi / 4)
```

[102]:
```
VectorObject3D(x=1, y=-0.7071067811865472, z=3.5355339059327378)
```

[103]:
```
vector.obj(x=1, y=2, z=3, tau=10).rotateX(np.pi / 4)
```

[103]:
```
VectorObject4D(x=1, y=-0.7071067811865472, z=3.5355339059327378, tau=10)
```

[104]:
```
vector.obj(pt=1, phi=1.3, eta=2).deltaR(vector.obj(pt=2, phi=0.3, eta=1))
```

[104]:
```
1.4142135623731
```

[105]:
```
vector.obj(pt=1, phi=1.3, eta=2, mass=5).deltaR(
    vector.obj(pt=2, phi=0.3, eta=1, mass=10)
)
```

[105]:
```
1.4142135623731
```

For a few operations - +, -, ==, !=, ... - the dimension of the vectors should be equal. This can be achieved by using the `like` method, `to_{coordinate_name}` methods, `to_Vector*D` methods. The `to_Vector*D` methods provide more flexibility to the users, that is, new coordinate values can be passed into the methods as named arguments.

[106]:
```
v1 = vector.obj(x=1, y=2, z=3)
v2 = vector.obj(x=1, y=2)

print(v1 - v2.like(v1))           # transforms v2 to v1's coordinate system (imputes z=0)
print(v1.like(v2) - v2)           # transforms v1 to v2's coordinate system (removes z)
print(v1 - v2.to_xyz())           # transforms v2 to xyz coordinates (imputes z=0)
print(v1.to_xy() - v2)            # transforms v1 to xy coordinates (removes z)
print(v1 - v2.to_Vector3D(z=3))   # transforms v2 to 3D (imputes z=3)
print(v1.to_Vector2D() - v2)      # transforms v1 to 2D (removes z)
```

```
VectorObject3D(x=0, y=0, z=3.0)
VectorObject2D(x=0, y=0)
VectorObject3D(x=0, y=0, z=3.0)
VectorObject2D(x=0, y=0)
VectorObject3D(x=0, y=0, z=0)
VectorObject2D(x=0, y=0)
```

Similarly, for a few vector methods, the dimension of the input vectors are type checked strictly.

For instance, a cross-product is only defined for 3D and 7D vectors; hence, running the method on a 4D vector will error out.

[107]:
```
vector.obj(x=0.1, y=0.2, z=0.3).cross(vector.obj(x=0.4, y=0.5, z=0.6))
```

[107]:
```
VectorObject3D(x=-0.03, y=0.06, z=-0.030000000000000013)
```

The (current) list of properties and methods is:

Planar (2D, 3D, 4D):

- `x` (`px`)
- `y` (`py`)
- `rho` (`pt`): two-dimensional magnitude
- `rho2` (`pt2`): two-dimensional magnitude squared
- `phi`
- `deltaphi(vector)`: difference in `phi` (signed and rectified to $-\pi$ through π)
- `rotateZ(angle)`
- `transform2D(obj)`: the `obj` must supply components through `obj["xx"]`, `obj["xy"]`, `obj["yx"]`, `obj["yy"]`
- `is_parallel(vector, tolerance=1e-5)`: only true *if they're pointing in the same direction*
- `is_antiparallel(vector, tolerance=1e-5)`: only true *if they're pointing in opposite directions*
- `is_perpendicular(vector, tolerance=1e-5)`

Spatial (3D, 4D):

- `z` (`pz`)
- `theta`
- `eta`

- `costheta`
- `cottheta`
- `mag` (`p`): three-dimensional magnitude, does not include temporal component
- `mag2` (`p2`): three-dimensional magnitude squared
- `cross`: cross-product (strictly 3D)
- `deltaangle(vector)`: difference in angle (always non-negative)
- `deltaeta(vector)`: difference in `eta` (signed)
- `deltaR(vector)`: $\Delta R = \sqrt{\Delta\phi^2 + \Delta\eta^2}$
- `deltaR2(vector)`: the above, squared
- `rotateX(angle)`
- `rotateY(angle)`
- `rotate_axis(axis, angle)`: the magnitude of `axis` is ignored, but it must be at least 3D
- `rotate_euler(phi, theta, psi, order="zxz")`: the arguments are in the same order as ROOT::Math::EulerAngles, and `order="zxz"` agrees with ROOT's choice of conventions
- `rotate_nautical(yaw, pitch, roll)`
- `rotate_quaternion(u, i, j, k)`: again, the conventions match ROOT::Math::Quaternion.
- `transform3D(obj)`: the `obj` must supply components through `obj["xx"]`, `obj["xy"]`, etc.
- `is_parallel(vector, tolerance=1e-5)`: only true *if they're pointing in the same direction*
- `is_antiparallel(vector, tolerance=1e-5)`: only true *if they're pointing in opposite directions*
- `is_perpendicular(vector, tolerance=1e-5)`

Lorentz (4D only):

- `t` (`E`, `energy`): follows the ROOT::Math::LorentzVector behavior of treating spacelike vectors as negative `t` and negative `tau` and truncating wrong-direction timelike vectors
- `t2` (`E2`, `energy2`)
- `tau` (`M`, `mass`): see note above
- `tau2` (`M2`, `mass2`)
- `beta`: scalar(s) between 0 (inclusive) and 1 (exclusive, unless the vector components are infinite)
- `deltaRapidityPhi`: $\Delta R_{\text{rapidity}} = \Delta\phi^2 + \Delta\text{rapidity}^2$
- `deltaRapidityPhi2`: the above, squared
- `gamma`: scalar(s) between 1 (inclusive) and ∞
- `rapidity`: scalar(s) between 0 (inclusive) and ∞
- `boost_p4(four_vector)`: change coordinate system using another 4D vector as the difference
- `boost_beta(three_vector)`: change coordinate system using a 3D beta vector (all components between -1 and $+1$)
- `boost(vector)`: uses the dimension of the given `vector` to determine behavior
- `boostX(beta=None, gamma=None)`: supply `beta` xor `gamma`, but not both
- `boostY(beta=None, gamma=None)`: supply `beta` xor `gamma`, but not both

- `boostZ(beta=None, gamma=None)`: supply `beta` xor `gamma`, but not both
- `transform4D(obj)`: the `obj` must supply components through `obj["xx"]`, `obj["xy"]`, etc.
- `to_beta3()`: turns a `four_vector` (for `boost_p4`) into a `three_vector` (for `boost_beta3`)
- `is_timelike(tolerance=0)`
- `is_spacelike(tolerance=0)`
- `is_lightlike(tolerance=1e-5)`: note the different tolerance

All numbers of dimensions:

- `unit()`: note the parentheses
- `dot(vector)`: can also use the `@` operator
- `add(vector)`: can also use the + operator
- `subtract(vector)`: can also use the - operator
- `scale(factor)`: can also use the * operator
- `equal(vector)`: can also use the == operator, but consider `isclose` instead
- `not_equal(vector)`: can also use the `!=` operator, but consider `isclose` instead
- `sum()`: can also use the `numpy.sum` or `awkward.sum`, only for NumPy and Awkward vectors
- `count_nonzero()`: can also use `numpy.count_nonzero` or `awkward.count_nonzero`, only for NumPy and Awkward vectors
- `count()`: can also use `awkward.count`, only for Awkward vectors
- `isclose(vector, rtol=1e-5, atol=1e-8, equal_nan=False)`: works like np.isclose; arrays also have an allclose method
- `to_VectorND(coordinates)/to_ND(coordinates)`: replace `N` with the required vector dimension
- `to_{coordinate-names}`: for example - `to_rhophietatau`
- `like(other)`: projects the vector into the dimensions of `other`, for example - `two_d_vector.like(three_d_vector)`

2.1.4 Compiling your Python with Numba

Numba is a just-in-time (JIT) compiler for a mathematically relevant subset of NumPy and Python. It allows you to write fast code without leaving the Python environment. The drawback of Numba is that it can only compile code blocks involving objects and functions that it recognizes.

The Vector library includes extensions to inform Numba about vector objects, vector NumPy arrays, and vector Awkward Arrays. At the time of writing, the implementation of vector NumPy arrays is incomplete due to numba/numba#6148.

For instance, consider the following function:

[108]:
```
@nb.njit
def compute_mass(v1, v2):
    return (v1 + v2).mass
```

[109]:
```
compute_mass(vector.obj(px=1, py=2, pz=3, E=4), vector.obj(px=-1, py=-2, pz=-3, E=4))
```

[109]:
```
8.0
```

When the two `MomentumObject4D` objects are passed as arguments, Numba recognizes them and replaces the Python objects with low-level structs. When it compiles the function, it recognizes + as the 4D `add` function and recognizes `.mass` as the `tau` component of the result.

Although this demonstrates that Numba can manipulate vector objects, there is no performance advantage (and a likely disadvantage) to compiling a calculation on just a few vectors. The advantage comes when many vectors are involved, in arrays.

[110]:
```
# This is still not a large number. You want millions.
array = vector.Array(
    [
        [
            dict(
                {x: np.random.normal(0, 1) for x in ("px", "py", "pz")},
                E=np.random.normal(10, 1),
            )
            for inner in range(np.random.poisson(1.5))
        ]
        for outer in range(50)
    ]
)
array
```

[110]:
```
<MomentumArray4D [[{x: 0.315, y: 0.693, ...}], ...] type='50 * var * Moment...'>
```

[111]:
```
@nb.njit
def compute_masses(array):
    out = np.empty(len(array), np.float64)
    for i, event in enumerate(array):
        total = vector.obj(px=0.0, py=0.0, pz=0.0, E=0.0)
        for vec in event:
            total = total + vec
        out[i] = total.mass
    return out
```

[112]:
```
compute_masses(array)
```

[112]:
```
array([ 9.1537046 , 16.76309287,  0.        ,  0.        , 19.91509641,
       10.54846375,  9.50567043, 50.02632217, 29.51323214, 15.98664711,
        9.45752217, 20.20656999, 38.98606898,  0.        ,  9.32749399,
        9.80117359, 10.70486904, 10.92982222, 35.85520701,  0.        ,
       29.04849791, 19.86279222, 10.31575156, 41.42782629, 11.75731099,
        9.81553407, 39.64420462, 20.78386006, 19.88365545,  8.76680915,
        9.54468769, 21.83630629,  8.90974542, 21.75954326, 19.27688599,
       10.06840737,  9.11771604, 10.78960379,  0.        , 10.55577844,
       19.37816397, 48.01216378,  0.        , 31.96752978, 28.30692309,
       28.78329415, 23.09265714,  0.        , 31.48155952,  7.83341747])
```

2.1.5 Talks about vector

- 9th October 2023 - What's new with Vector? First major release is out! - PyHEP 2023 (virtual)
- 13th September 2022 - Constructing HEP vectors and analyzing HEP data using Vector - PyHEP 2022 (virtual)
- 20th July 2022 - Analysis Grand Challenge / HEP Scientific Python Ecosystem - DANCE/CoDaS@Snowmass 2022 computational and data science software training
- 25th April 2022 - Foundation libraries (uproot, awkward, hist, mplhep) - IRIS-HEP AGC Tools 2022 Workshop
- 3rd November 2021 - Data handling: uproot, awkward & vector - IRIS-HEP AGC Tools 2021 Workshop

Status as of November 17, 2023

First major release of vector is out and the package has reached a stable position. The work is spearheaded by bug reports and feature requests created on GitHub. It can only be improved by your feedback!

[ ]:

2.2 Structure of Vector

CHAPTER

THREE

TALKS ABOUT VECTOR

- 13th September 2022 - Constructing HEP vectors and analyzing HEP data using Vector - PyHEP 2022 (virtual)
- 20th July 2022 - Analysis Grand Challenge / HEP Scientific Python Ecosystem - DANCE/CoDaS@Snowmass 2022 computational and data science software training
- 25th April 2022 - Foundation libraries (uproot, awkward, hist, mplhep) - IRIS-HEP AGC Tools 2022 Workshop
- 3rd November 2021 - Data handling: uproot, awkward & vector - IRIS-HEP AGC Tools 2021 Workshop

CHAPTER

FOUR

CHANGES IN VECTOR'S API

The `changelog` file describes the changes in `vector`'s API and usage introduced in every new version. These changes can be breaking changes or minor adjustments, hence one should go through this file and their existing codebase while updating `vector`'s version.

4.1 Changelog

4.1.1 Version 1.4

Version 1.4.0

Features

- feat: allow coord values in to_<coord_names> methods #446
- feat: a sympy backend #442

Bug fixes

- fix: call the square implementation for power 2 on object vectors #444
- fix: use negfactor in negfactor scale test #456

Maintenance

- chore: test on numpy 2.0 #451

4.1.2 Version 1.3

Version 1.3.1

Features

- feat: make momentum-ness infectious #437

Bug fixes

- fix: support dask-awkward 2024.3.0 #436
- fix: momentum coords should not be repeated with generic coords in subclasses #438

Version 1.3.0

Features

- feat: coordinate transformation functions with momentum names #424
- feat: allow momentum coords in to_Vector*D methods + cleanup #423
- feat: add a lite nox session + add numba as optional dependency #431
- feat: `like` method for projecting vector into the coordinate space of another vector + better type errors and hints #426
- feat: add support for dask-awkward arrays in vector constructors #429
- feat: short names for to_VectorND methods #432

4.1.3 Version 1.2

Version 1.2.0

Bug fixes

- fix: result of an infix operation should be demoted to the lowest possible dimension #413
- fix: all infix operations should not depend on the order of arguments #413
- fix: return the correct awkward record when performing an infix operation #413
- fix: respect user defined awkward mixin subclasses and projection classes #413

Documentation

- Update `README` and `intro.ipynb` to include the latest developments #399
- docs: add docs for vector.zip #390
- Fix Vector* mixin's docstring #404

Maintenance

- chore: repo review updates #408
- black -> ruff format #414
- chore: migrate to pytest-doctestplus #416

4.1.4 Version 1.1

Version 1.1.1.post1

Maintenance

- chore: support Python 3.12 #388
- Fix CI badge in README and docs #386

Version 1.1.1

Bug fixes

- fix: `keepdims` in `numpy.sum` should not be `None` #376

Maintenance

- chore: remove license string (not standard) #371
- chore: blackend-docs moved #370
- chore: use 2x faster black mirror #367
- chore: clean up VCS versioning #363
- chore: target-version no longer needed by Black or Ruff #359
- chore: ruff moved to astral-sh #358

Version 1.1.0

Features

- feat: implement `sum`, `count`, and `count_nonzero` reductions #347

Maintenance

- chore: remove Python `3.7` support #355
- chore: use trusted publisher deployment #354
- chore: replace custom definition of np.isclose with numba's np.isclose #348

4.1.5 Version 1.0

Version 1.0.0

Features

- feat: add constructors for `VectorObject3D` and `MomentumObject3D` #231
- feat: add constructors for `VectorObject4D` and `MomentumObject4D` #232
- feat: update `to_Vector3D` to pass new coordinate values #278
- feat: allow passing coordinates to to_Vector-D #319

Bug fixes

- fix: better elif conditions for obj __init__ methods #316

Documentation

- docs: a readable changelog #320

Maintenance

- ci: use numpy~=1.24.0 in pre-commit #308
- fix: update discheck #305
- ci: update number of builds for codecov bot #314
- chore: move to using Ruff #315
- chore: update copyright and license for 2022 and 2023 #321

4.1.6 Version 0.11

Version 0.11.0

Features

- Add constructors for `VectorObject2D` and `MomentumObject2D`#89
- Add support for awkward v2 (and keep supporting v1) #284

Bug fixes

- `vector.arr` should construct `NumPy` vectors #254
- Development dependency missing #280

Documentation

- docs: add a section for talks #264
- docs: fix missing backslash in latex for readme #285
- docs: update changelog.md, PR template, and CONTRIBUTING.md #275
- docs: add a developer guide #233

Maintenance

- chore: add PyLint and additional pre-commit hooks #260
- chore: pull request template-Priyadarshi #271
- chore: add issue templates #267
- chore: better and long term fix for flake8-bugbear #298
- chore: bump mypy and revert python-version #263
- chore: fix the failing mypy hook by pinning python-version #261
- chore: ignore flake8 B905 + improve bug report template #297
- chore: minor cleanups #266
- chore: test on `awkward v1.10.0` and add cov to `noxfile` #256
- chore: use Python 3.11! #282
- chore: zenodo badge sync #269
- ci: test notebooks on PRs #272

4.1.7 Version 0.10

Version 0.10.0

Maintenance

- Remove Python `3.6` support #251

4.1.8 Version 0.9

Version 0.9.0

Features

- Implements deltaRapidityPhi and deltaRapidityPhi2. #175
- Remove underscores #192
- feat: add git archive support #244

Bug fixes

- fix bad values for high (abs) eta #172
- Add custom reprs to awkward coordinate classes #212
- Explicitly set posinf and neginf in nan_to_num so they stay infinite. #173
- Add type checks in constructors #210

Documentation

- Add Conda and Zenodo badges to the README-rodrigues [#183][]
- Tests and docs for deltaRapidityPhi #187
- docs: fix intro notebook, and submodule and subpackage index #191
- docs: add codecov badge to README #203
- docs: fix warnings #193
- docs: add docstrings in the `backends.numpy` module #195
- docs: add docstrings in the `backends.object` module #201
- docs: improve the landing page and API docs structure #204
- docs: add docstrings in the `backends.awkward` module #207
- Implement doctests in CI #211
- docs: Add CITATION.cff Citation File Format file #243
- docs: update changelog [#248][]

Maintenance

- chore: wheel not required for setuptools PEP 517 (all-repos) #176
- fix: bump black to 22.3.0 due to click 8.1 release #181
- ci: fix a test and update CI to catch errors regularly #199
- chore: fix conda badge and update dependabot #213
- docs: render module level docstrings in documentation #218
- chore: pass repo review #219

- chore: migrate to hatchling #223
- chore: add `codecov.yml` #229
- chore: add `pyproject-fmt` pre-commit hook #230
- chore: remove redundant `tool.check-manifest` from `pyproject.toml` #235
- chore: support `awkward` `v1` and `v2` together #226
- chore: build and test on Python `3.10` and `3.11-dev` #252

4.1.9 Version 0.8

Version 0.8.5

- Added boostCM_of to clarify #134, supported by scaleD and negD #135
- Let 'eta' be NaN if 'z' is NaN #139
- Defined dot product without absolute value #148
- Fixed numpy array code examples in documentation #151
- Vector components may be NumpyArrayType or IndexedArrayType in Numba #162
- VectorNumpy pickle support to enable multiprocessing #163
- pre-commit and style cleanup #164

Version 0.8.4

- Allow VectorObject, VectorNumpy, and VectorAwkward to be subclassed by other projects #128

Version 0.8.3

- Fixed Awkward Arrays of momentum vectors in Numba #112

Version 0.8.2

- Fixed missing momentum synonyms in CoordinatesAwkward #84
- Added vector.zip #94
- Allowed lowercase e and m for energy and mass #95
- Fixed _wrap_result for methods called on an ak.Record #100
- Fixed error in calculation of deltaangle #105
- Fixed Awkward version check #82
- Pinned Python version for dis (in tests) #90
- Using myst-parser (in docs) #91

Version 0.8.1

- Fix issue importing without Awkward installed #76

Version 0.8.0

First release to PyPI. Initial implementation. Initial features:

- 2D, 3D, and Lorentz vectors
- Single, Array, and Awkward forms
- Supports Numba / Awkward + Numba
- Multiple coordinate systems
- Geometric / momentum versions
- Statically typed

You can currently construct vectors using `obj/arr/awk` (or `obj/array/Array`) for single, NumPy, and Awkward vectors, respectively. The next version is likely to improve the vector construction process.

CHAPTER

FIVE

GETTING HELP

- `Vector`'s code is hosted on GitHub.
- If something is not working the way it should, or if you want to request a new feature, create a new issue on GitHub.
- To discuss something related to `vector`, use the discussions tab on GitHub or `vector`'s gitter (Scikit-HEP/vector) chat room.

CHAPTER

SIX

CONTRIBUTING TO VECTOR

If you are planning to develop `vector` (thank you!), or if you want to use the latest commit of `vector` on your local machine, you might want to install it from the source. Refer to our Contributing Guidelines for more information.

CHAPTER

SEVEN

API REFERENCE

The API reference details the functionality of each `class` and `function` present in `vector`'s codebase.

7.1 vector

7.1.1 vector package

Subpackages

vector.backends package

Vector comes loaded with 3 + 2 backends; a pure Python object backend, a NumPy backend, an Awkward Array backend, an Object-Numba, and an Awkward-Numba backend to leverage JIT (Just In Time) compiled calculations on vectors. Other potential future vanilla backends include Tensorflow and JAX, and other possible future Numba-backends include Numba-NumPy.

Submodules

vector.backends.object module

Defines behaviors for Object vectors. New vectors created with the respective classes

```
vector.VectorObject2D(...)
vector.VectorObject3D(...)
```

will have these behaviors built in (and will pass them to any derived objects).

The class methods can also be used to construct object type vectors -

```
vec = vector.VectorObject2D.from_xy(1, 2)
```

Additionally, object type vectors can also be constructed using -

```
vector.obj(...)
```

function.

class vector.backends.object.**AzimuthalObject**

Bases: *CoordinatesObject*, *Azimuthal*

Azimuthal class for the Object backend.

class vector.backends.object.**AzimuthalObjectRhoPhi**(*rho: float*, *phi: float*)

Bases: *AzimuthalObject*, *AzimuthalRhoPhi*, *TupleRhoPhi*

Class for the `rho` and `phi` (azimuthal) coordinates of Object backend. Use the `elements` property to retrieve the coordinates.

property elements: tuple[float, float]

Azimuthal coordinates (`rho` and `phi`) as a tuple. Each coordinate is a scalar and not a vector.

Examples

```
>>> import vector
>>> v = vector.VectorObject2D(rho=1, phi=2)
>>> az = v.azimuthal
>>> az.elements
(1, 2)
```

class vector.backends.object.**AzimuthalObjectXY**(*x: float*, *y: float*)

Bases: *AzimuthalObject*, *AzimuthalXY*, *TupleXY*

Class for the `x` and `y` (azimuthal) coordinates of Object backend. Use the `elements` property to retrieve the coordinates.

property elements: tuple[float, float]

Azimuthal coordinates (`x` and `y`) as a tuple. Each coordinate is a scalar and not a vector.

Examples

```
>>> import vector
>>> v = vector.VectorObject2D(x=1, y=2)
>>> az = v.azimuthal
>>> az.elements
(1, 2)
```

class vector.backends.object.**CoordinatesObject**

Bases: `object`

Coordinates class for the Object backend.

class vector.backends.object.**LongitudinalObject**

Bases: *CoordinatesObject*, *Longitudinal*

Longitudinal class for the Object backend.

class vector.backends.object.**LongitudinalObjectEta**(*eta: float*)

Bases: *LongitudinalObject*, *LongitudinalEta*, *TupleEta*

Class for the `eta` (longitudinal) coordinate of Object backend. Use the `elements` property to retrieve the coordinates.

property elements: tuple[float]

Longitudinal coordinates (`theta`) as a tuple. Each coordinate is a scalar and not a vector.

Examples

```
>>> import vector
>>> v = vector.obj(rho=1, phi=2, eta=3)
>>> lc = v.longitudinal
>>> lc.elements
(3,)
```

class vector.backends.object.**LongitudinalObjectTheta**(*theta: float*)

Bases: *LongitudinalObject*, *LongitudinalTheta*, *TupleTheta*

Class for the `theta` (longitudinal) coordinate of Object backend. Use the `elements` property to retrieve the coordinates.

property elements: tuple[float]

Longitudinal coordinates (`theta`) as a tuple. Each coordinate is a scalar and not a vector.

Examples

```
>>> import vector
>>> v = vector.obj(rho=1, phi=2, theta=3)
>>> lc = v.longitudinal
>>> lc.elements
(3,)
```

class vector.backends.object.**LongitudinalObjectZ**(*z: float*)

Bases: *LongitudinalObject*, *LongitudinalZ*, *TupleZ*

Class for the `z` (longitudinal) coordinate of Object backend. Use the `elements` property to retrieve the coordinates.

property elements: tuple[float]

Longitudinal coordinates (`z`) as a tuple. Each coordinate is a scalar and not a vector.

Examples

```
>>> import vector
>>> v = vector.obj(rho=1, phi=2, z=3)
>>> lc = v.longitudinal
>>> lc.elements
(3,)
```

class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *z: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *eta: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *theta: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *z: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*)

class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*)
class vector.backends.object.**MomentumObject2D**(*, *pt: float*, *phi: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *pz: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *z: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.object.**MomentumObject2D**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.object.**MomentumObject2D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *theta: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.object.**MomentumObject2D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.object.**MomentumObject2D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.object.**MomentumObject2D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.object.**MomentumObject2D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.object.**MomentumObject2D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)

class vector.backends.object.**MomentumObject2D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.object.**MomentumObject2D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.object.**MomentumObject2D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.object.**MomentumObject2D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.object.**MomentumObject2D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.object.**MomentumObject2D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.object.**MomentumObject2D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.object.**MomentumObject2D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)

class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.object.**MomentumObject2D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.object.**MomentumObject2D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.object.**MomentumObject2D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.object.**MomentumObject2D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.object.**MomentumObject2D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.object.**MomentumObject2D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.object.**MomentumObject2D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)

class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.object.**MomentumObject2D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.object.**MomentumObject2D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.object.**MomentumObject2D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.object.**MomentumObject2D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.object.**MomentumObject2D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.object.**MomentumObject2D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.object.**MomentumObject2D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)

class vector.backends.object.**MomentumObject2D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.object.**MomentumObject2D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.object.**MomentumObject2D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.object.**MomentumObject2D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.object.**MomentumObject2D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.object.**MomentumObject2D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.object.**MomentumObject2D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.object.**MomentumObject2D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.object.**MomentumObject2D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.object.**MomentumObject2D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.object.**MomentumObject2D**(*__azumthal:* Azimuthal)
class vector.backends.object.**MomentumObject2D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector.backends.object.**MomentumObject2D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *PlanarMomentum*, *VectorObject2D*

Two dimensional momentum vector class for the object backend.

Examples

```
>>> import vector
>>> vec = vector.MomentumObject2D(px=1, py=2)
>>> vec.px, vec.py
(1, 2)
>>> vec = vector.MomentumObject2D(pt=1, phi=2)
>>> vec.pt, vec.phi
(1, 2)
>>> vec = vector.MomentumObject2D(azimuthal=vector.backends.object.
↪AzimuthalObjectXY(1, 2))
>>> vec.px, vec.py
(1, 2)
```

The `vector.obj()` function can also be used to construct 2D momentum object type vectors.

For two dimensional vector objects, see *vector.backends.object.VectorObject2D*.

GenericClass

alias of *VectorObject2D*

MomentumClass

alias of *MomentumObject2D*

ProjectionClass2D

alias of *MomentumObject2D*

ProjectionClass3D

alias of *MomentumObject3D*

ProjectionClass4D

alias of *MomentumObject4D*

azimuthal: *AzimuthalObject*

property pt: float

Momentum-synonym for *vector._methods.VectorProtocolPlanar.rho*.

property px: float

Momentum-synonym for *vector._methods.VectorProtocolPlanar.x*.

property py: float

Momentum-synonym for *vector._methods.VectorProtocolPlanar.y*.

class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *z: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *eta: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *theta: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *z: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*)

class vector.backends.object.**MomentumObject3D**(*, *pt: float*, *phi: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *pz: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *z: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.object.**MomentumObject3D**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.object.**MomentumObject3D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *theta: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.object.**MomentumObject3D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.object.**MomentumObject3D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.object.**MomentumObject3D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.object.**MomentumObject3D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.object.**MomentumObject3D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.object.**MomentumObject3D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)

class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.object.**MomentumObject3D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.object.**MomentumObject3D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.object.**MomentumObject3D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.object.**MomentumObject3D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.object.**MomentumObject3D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.object.**MomentumObject3D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.object.**MomentumObject3D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)

class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.object.**MomentumObject3D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.object.**MomentumObject3D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.object.**MomentumObject3D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.object.**MomentumObject3D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.object.**MomentumObject3D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.object.**MomentumObject3D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.object.**MomentumObject3D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)

class vector.backends.object.**MomentumObject3D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.object.**MomentumObject3D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.object.**MomentumObject3D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.object.**MomentumObject3D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.object.**MomentumObject3D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.object.**MomentumObject3D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.object.**MomentumObject3D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.object.**MomentumObject3D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)

class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.object.**MomentumObject3D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.object.**MomentumObject3D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.object.**MomentumObject3D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.object.**MomentumObject3D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.object.**MomentumObject3D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.object.**MomentumObject3D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.object.**MomentumObject3D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.object.**MomentumObject3D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.object.**MomentumObject3D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.object.**MomentumObject3D**(*__azumthal:* Azimuthal)
class vector.backends.object.**MomentumObject3D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector.backends.object.**MomentumObject3D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *SpatialMomentum*, *VectorObject3D*

Three dimensional momentum vector class for the object backend.

Examples

```
>>> import vector
>>> vec = vector.MomentumObject3D(px=1, py=2, pz=3)
>>> vec.px, vec.py, vec.pz
(1, 2, 3)
>>> vec = vector.MomentumObject3D(pt=1, phi=2, pz=3)
>>> vec.pt, vec.phi, vec.pz
(1, 2, 3)
>>> vec = vector.MomentumObject3D(
...     azimuthal=vector.backends.object.AzimuthalObjectXY(1, 2),
...     longitudinal=vector.backends.object.LongitudinalObjectTheta(3)
... )
>>> vec.x, vec.y, vec.theta
(1, 2, 3)
```

The `vector.obj()` function can also be used to construct 3D momentum object type vectors.

For three dimensional vector objects, see *vector.backends.object.VectorObject3D*.

GenericClass
alias of *VectorObject3D*

MomentumClass
alias of *MomentumObject3D*

ProjectionClass2D
alias of *MomentumObject2D*

ProjectionClass3D
alias of *MomentumObject3D*

ProjectionClass4D
alias of *MomentumObject4D*

azimuthal: *AzimuthalObject*

longitudinal: *LongitudinalObject*

property pt: float
Momentum-synonym for *vector._methods.VectorProtocolPlanar.rho*.

property px: float
Momentum-synonym for *vector._methods.VectorProtocolPlanar.x*.

property py: float
Momentum-synonym for *vector._methods.VectorProtocolPlanar.y*.

property pz: float
Momentum-synonym for *vector._methods.VectorProtocolSpatial.z*.

class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *z: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *eta: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *theta: float*)

class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *z: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*)
class vector.backends.object.**MomentumObject4D**(*, *pt: float*, *phi: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *pz: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *z: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.object.**MomentumObject4D**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.object.**MomentumObject4D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *theta: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.object.**MomentumObject4D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.object.**MomentumObject4D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.object.**MomentumObject4D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.object.**MomentumObject4D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.object.**MomentumObject4D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)

class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.object.**MomentumObject4D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.object.**MomentumObject4D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.object.**MomentumObject4D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.object.**MomentumObject4D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.object.**MomentumObject4D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.object.**MomentumObject4D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.object.**MomentumObject4D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)

class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.object.**MomentumObject4D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.object.**MomentumObject4D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.object.**MomentumObject4D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.object.**MomentumObject4D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.object.**MomentumObject4D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.object.**MomentumObject4D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.object.**MomentumObject4D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)

class vector.backends.object.**MomentumObject4D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.object.**MomentumObject4D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.object.**MomentumObject4D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.object.**MomentumObject4D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.object.**MomentumObject4D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.object.**MomentumObject4D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.object.**MomentumObject4D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *z: float*, *m: float*)

class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.object.**MomentumObject4D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.object.**MomentumObject4D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.object.**MomentumObject4D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.object.**MomentumObject4D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.object.**MomentumObject4D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.object.**MomentumObject4D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.object.**MomentumObject4D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.object.**MomentumObject4D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.object.**MomentumObject4D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.object.**MomentumObject4D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.object.**MomentumObject4D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.object.**MomentumObject4D**(*__azumthal:* Azimuthal)
class vector.backends.object.**MomentumObject4D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)

class vector.backends.object.**MomentumObject4D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *LorentzMomentum*, *VectorObject4D*

Four dimensional momentum vector class for the object backend.

Examples

```
>>> import vector
>>> vec = vector.MomentumObject4D(px=1, py=2, pz=3, t=4)
>>> vec.px, vec.py, vec.pz, vec.t
(1, 2, 3, 4)
>>> vec = vector.MomentumObject4D(pt=1, phi=2, pz=3, M=4)
>>> vec.pt, vec.phi, vec.pz, vec.M
(1, 2, 3, 4)
>>> vec = vector.MomentumObject4D(
...     azimuthal=vector.backends.object.AzimuthalObjectXY(1, 2),
...     longitudinal=vector.backends.object.LongitudinalObjectTheta(3),
...     temporal=vector.backends.object.TemporalObjectTau(4)
... )
>>> vec.x, vec.y, vec.theta, vec.tau
(1, 2, 3, 4)
```

The `vector.obj()` function can also be used to construct 4D momentum object type vectors.

For four dimensional vector objects, see *vector.backends.object.VectorObject4D*.

property E: float

Momentum-synonyor *vector._methods.VectorProtocolLorentz.t*.

GenericClass

alias of *VectorObject4D*

property M: float

Momentum-synonym for *vector._methods.VectorProtocolLorentz.tau*.

MomentumClass

alias of *MomentumObject4D*

ProjectionClass2D

alias of *MomentumObject2D*

ProjectionClass3D

alias of *MomentumObject3D*

ProjectionClass4D

alias of *MomentumObject4D*

azimuthal: *AzimuthalObject*

property e: float

Momentum-synonym for *vector._methods.VectorProtocolLorentz.t*.

property energy: float

Momentum-synonym for *vector._methods.VectorProtocolLorentz.t*.

longitudinal: *LongitudinalObject*

property m: float

Momentum-synonym for *vector._methods.VectorProtocolLorentz.tau*.

property mass: float

Momentum-synonym for *vector._methods.VectorProtocolLorentz.tau*.

property pt: float

Momentum-synonym for *vector._methods.VectorProtocolPlanar.rho*.

property px: float

Momentum-synonym for *vector._methods.VectorProtocolPlanar.x*.

property py: float

Momentum-synonym for *vector._methods.VectorProtocolPlanar.y*.

property pz: float

Momentum-synonym for *vector._methods.VectorProtocolSpatial.z*.

temporal: *TemporalObject*

class vector.backends.object.**TemporalObject**

Bases: *CoordinatesObject*, *Temporal*

Temporal class for the Object backend.

class vector.backends.object.**TemporalObjectT**(*t: float*)

Bases: *TemporalObject*, *TemporalT*, *TupleT*

Class for the `t` (temporal) coordinate of Object backend. Use the `elements` property to retrieve the coordinates.

property elements: tuple[float]

Temporal coordinates (`t`) as a tuple. Each coordinate is a scalar and not a vector.

Examples

```
>>> import vector
>>> v = vector.obj(rho=1, phi=2, theta=3, t=4)
>>> tc = v.temporal
>>> tc.elements
(4,)
```

class vector.backends.object.**TemporalObjectTau**(*tau: float*)

Bases: *TemporalObject*, *TemporalTau*, *TupleTau*

Class for the `tau` (temporal) coordinate of Object backend. Use the `elements` property to retrieve the coordinates.

property elements: tuple[float]

Temporal coordinates (`tau`) as a tuple. Each coordinate is a scalar and not a vector.

Examples

```
>>> import vector
>>> v = vector.obj(rho=1, phi=2, theta=3, tau=4)
>>> tc = v.temporal
>>> tc.elements
(4,)
```

class vector.backends.object.**TupleEta**(*eta: float*)

Bases: NamedTuple

eta coordinate as a NamedTuple.

_asdict()

Return a new dict which maps field names to their values.

_field_defaults = {}

_fields = ('eta',)

classmethod _make(*iterable*)

Make a new TupleEta object from a sequence or iterable

_replace(***kwds*)

Return a new TupleEta object replacing specified fields with new values

eta: float

Alias for field number 0

class vector.backends.object.**TupleRhoPhi**(*rho: float*, *phi: float*)

Bases: NamedTuple

rho and phi coordinates as a NamedTuple.

_asdict()

Return a new dict which maps field names to their values.

_field_defaults = {}

_fields = ('rho', 'phi')

classmethod _make(*iterable*)

Make a new TupleRhoPhi object from a sequence or iterable

_replace(***kwds*)

Return a new TupleRhoPhi object replacing specified fields with new values

phi: float

Alias for field number 1

rho: float

Alias for field number 0

class vector.backends.object.**TupleT**(*t: float*)

Bases: NamedTuple

t coordinate as a NamedTuple.

_asdict()

Return a new dict which maps field names to their values.

_field_defaults = {}

_fields = ('t',)

classmethod _make(*iterable*)

Make a new TupleT object from a sequence or iterable

_replace(***kwds*)

Return a new TupleT object replacing specified fields with new values

t: float

Alias for field number 0

class vector.backends.object.**TupleTau**(*tau: float*)

Bases: NamedTuple

tau coordinate as a NamedTuple.

_asdict()

Return a new dict which maps field names to their values.

_field_defaults = {}

_fields = ('tau',)

classmethod _make(*iterable*)

Make a new TupleTau object from a sequence or iterable

_replace(***kwds*)

Return a new TupleTau object replacing specified fields with new values

tau: float

Alias for field number 0

class vector.backends.object.**TupleTheta**(*theta: float*)

Bases: NamedTuple

theta coordinates as a NamedTuple.

_asdict()

Return a new dict which maps field names to their values.

_field_defaults = {}

_fields = ('theta',)

classmethod _make(*iterable*)

Make a new TupleTheta object from a sequence or iterable

_replace(***kwds*)

Return a new TupleTheta object replacing specified fields with new values

theta: float

Alias for field number 0

class vector.backends.object.**TupleXY**(*x: float*, *y: float*)

Bases: NamedTuple

x and y coordinates as a NamedTuple.

_asdict()

Return a new dict which maps field names to their values.

_field_defaults = {}

_fields = ('x', 'y')

classmethod _make(*iterable*)

Make a new TupleXY object from a sequence or iterable

_replace(***kwds*)

Return a new TupleXY object replacing specified fields with new values

x: float

Alias for field number 0

y: float

Alias for field number 1

class vector.backends.object.**TupleZ**(*z: float*)

Bases: NamedTuple

z coordinate as a NamedTuple.

_asdict()

Return a new dict which maps field names to their values.

_field_defaults = {}

_fields = ('z',)

classmethod _make(*iterable*)

Make a new TupleZ object from a sequence or iterable

_replace(***kwds*)

Return a new TupleZ object replacing specified fields with new values

z: float

Alias for field number 0

class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *z: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *eta: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *theta: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *z: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*)

class vector.backends.object.**VectorObject**(*, *pt: float*, *phi: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *pz: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *z: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.object.**VectorObject**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.object.**VectorObject**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *theta: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.object.**VectorObject**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.object.**VectorObject**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.object.**VectorObject**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.object.**VectorObject**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.object.**VectorObject**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.object.**VectorObject**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *z: float*, *tau: float*)

class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.object.**VectorObject**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.object.**VectorObject**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.object.**VectorObject**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.object.**VectorObject**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.object.**VectorObject**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.object.**VectorObject**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.object.**VectorObject**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *eta: float*, *E: float*)

class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.object.**VectorObject**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.object.**VectorObject**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.object.**VectorObject**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.object.**VectorObject**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.object.**VectorObject**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.object.**VectorObject**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.object.**VectorObject**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)

class vector.backends.object.**VectorObject**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.object.**VectorObject**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.object.**VectorObject**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.object.**VectorObject**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.object.**VectorObject**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.object.**VectorObject**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.object.**VectorObject**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.object.**VectorObject**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *theta: float*, *m: float*)

class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.object.**VectorObject**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.object.**VectorObject**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.object.**VectorObject**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.object.**VectorObject**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.object.**VectorObject**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.object.**VectorObject**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.object.**VectorObject**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.object.**VectorObject**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.object.**VectorObject**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.object.**VectorObject**(*__azumthal:* Azimuthal)
class vector.backends.object.**VectorObject**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector.backends.object.**VectorObject**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *Vector*

Mixin class for Object vectors.

lib = <module 'numpy' from '/home/docs/checkouts/readthedocs.org/user_builds/vector/envs/stable/lib/python3.12/site-packages/numpy/__init__.py'>

class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *z: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *eta: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *theta: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *z: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*)
class vector.backends.object.**VectorObject2D**(*, *pt: float*, *phi: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *pz: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *z: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.object.**VectorObject2D**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.object.**VectorObject2D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *theta: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.object.**VectorObject2D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.object.**VectorObject2D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.object.**VectorObject2D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)

class vector.backends.object.**VectorObject2D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.object.**VectorObject2D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.object.**VectorObject2D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.object.**VectorObject2D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.object.**VectorObject2D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.object.**VectorObject2D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.object.**VectorObject2D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *z: float*, *E: float*)

class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.object.**VectorObject2D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.object.**VectorObject2D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.object.**VectorObject2D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.object.**VectorObject2D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.object.**VectorObject2D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.object.**VectorObject2D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.object.**VectorObject2D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.object.**VectorObject2D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)

class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.object.**VectorObject2D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.object.**VectorObject2D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.object.**VectorObject2D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.object.**VectorObject2D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.object.**VectorObject2D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.object.**VectorObject2D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.object.**VectorObject2D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)

class vector.backends.object.**VectorObject2D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.object.**VectorObject2D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.object.**VectorObject2D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.object.**VectorObject2D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.object.**VectorObject2D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.object.**VectorObject2D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.object.**VectorObject2D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.object.**VectorObject2D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.object.**VectorObject2D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)

class vector.backends.object.**VectorObject2D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.object.**VectorObject2D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.object.**VectorObject2D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.object.**VectorObject2D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.object.**VectorObject2D**(*__azumthal:* Azimuthal)
class vector.backends.object.**VectorObject2D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector.backends.object.**VectorObject2D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *VectorObject*, *Planar*, *Vector2D*

Two dimensional vector class for the object backend.

Examples

```
>>> import vector
>>> vec = vector.VectorObject2D(x=1, y=2)
>>> vec.x, vec.y
(1, 2)
>>> vec = vector.VectorObject2D(rho=1, phi=2)
>>> vec.rho, vec.phi
(1, 2)
>>> vec = vector.VectorObject2D(azimuthal=vector.backends.object.
↪AzimuthalObjectXY(1, 2))
>>> vec.x, vec.y
(1, 2)
```

The following class methods can also be used to construct 2D object type vectors -

- *VectorObject2D.from_xy()*
- *VectorObject2D.from_rhophi()*

Additionally, the `vector.obj()` function can also be used to construct 2D object type vectors.

For two dimensional momentum vector objects, see *vector.backends.object.MomentumObject2D*.

GenericClass
alias of *VectorObject2D*

MomentumClass
alias of *MomentumObject2D*

ProjectionClass2D
alias of *VectorObject2D*

ProjectionClass3D
alias of *VectorObject3D*

ProjectionClass4D
alias of *VectorObject4D*

_wrap_result(*cls: Any*, *result: Any*, *returns: Any*, *num_vecargs: Any*) → Any
Wraps the raw result of a compute function as a scalar or a vector.

Parameters

- **result** – Value or tuple of values from a compute function.
- **returns** – Signature from a `dispatch_map`.
- **num_vecargs** (*int*) – Number of vector arguments in the function that would be treated on an equal footing (i.e. `add` has two, but `rotate_axis` has only one: the `axis` is secondary).

azimuthal: *AzimuthalObject*

classmethod from_rhophi(*rho: float*, *phi: float*) → *VectorObject2D*

Constructs a `VectorObject2D` from polar coordinates.

Use *vector.backends.object.MomentumObject2D* to construct a vector with momentum properties and methods.

Examples

```
>>> import vector
>>> vec = vector.VectorObject2D.from_rhophi(1, 2)
>>> vec
VectorObject2D(rho=1, phi=2)
```

classmethod from_xy(*x: float*, *y: float*) → *VectorObject2D*

Constructs a `VectorObject2D` from Cartesian coordinates.

Use *vector.backends.object.MomentumObject2D* to construct a vector with momentum properties and methods.

Examples

```
>>> import vector
>>> vec = vector.VectorObject2D.from_xy(1, 2)
>>> vec
VectorObject2D(x=1, y=2)
```

property phi: float

The polar ϕ coordinate of the vector or every vector in the array (in radians, always between $-\pi$ and π).

property rho: float

The polar ρ coordinate of the vector or every vector in the array.

This is also the magnitude of the 2D azimuthal part of the vector (not including any longitudinal or temporal parts).

property x: float

The Cartesian x coordinate of the vector or every vector in the array.

property y: float

The Cartesian y coordinate of the vector or every vector in the array.

class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*)

class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*)

class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *z: float*)

class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *eta: float*)

class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *theta: float*)

class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *z: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*)
class vector.backends.object.**VectorObject3D**(*, *pt: float*, *phi: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *pz: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *z: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.object.**VectorObject3D**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.object.**VectorObject3D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *theta: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.object.**VectorObject3D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.object.**VectorObject3D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.object.**VectorObject3D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.object.**VectorObject3D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.object.**VectorObject3D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)

class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.object.**VectorObject3D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.object.**VectorObject3D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.object.**VectorObject3D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.object.**VectorObject3D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.object.**VectorObject3D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.object.**VectorObject3D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.object.**VectorObject3D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)

class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.object.**VectorObject3D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.object.**VectorObject3D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.object.**VectorObject3D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.object.**VectorObject3D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.object.**VectorObject3D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.object.**VectorObject3D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.object.**VectorObject3D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)

class vector.backends.object.**VectorObject3D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.object.**VectorObject3D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.object.**VectorObject3D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.object.**VectorObject3D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.object.**VectorObject3D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.object.**VectorObject3D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.object.**VectorObject3D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *z: float*, *m: float*)

class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.object.**VectorObject3D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.object.**VectorObject3D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.object.**VectorObject3D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.object.**VectorObject3D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.object.**VectorObject3D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.object.**VectorObject3D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.object.**VectorObject3D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.object.**VectorObject3D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.object.**VectorObject3D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.object.**VectorObject3D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.object.**VectorObject3D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.object.**VectorObject3D**(*__azumthal:* Azimuthal)
class vector.backends.object.**VectorObject3D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)

class vector.backends.object.**VectorObject3D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *VectorObject*, *Spatial*, *Vector3D*

Three dimensional vector class for the object backend. Use the class methods -

Examples

```
>>> import vector
>>> vec = vector.VectorObject3D(x=1, y=2, z=3)
>>> vec.x, vec.y, vec.z
(1, 2, 3)
>>> vec = vector.VectorObject3D(rho=1, phi=2, eta=3)
>>> vec.rho, vec.phi, vec.eta
(1, 2, 3)
>>> vec = vector.VectorObject3D(
...     azimuthal=vector.backends.object.AzimuthalObjectXY(1, 2),
...     longitudinal=vector.backends.object.LongitudinalObjectTheta(3)
... )
>>> vec.x, vec.y, vec.theta
(1, 2, 3)
```

The following class methods can also be used to construct 3D object type vectors -

- *VectorObject3D.from_xyz()*
- *VectorObject3D.from_xytheta()*
- *VectorObject3D.from_xyeta()*
- *VectorObject3D.from_rhophiz()*
- *VectorObject3D.from_rhophitheta()*
- *VectorObject3D.from_rhophieta()*

Additionally, the `vector.obj()` function can also be used to construct 3D object type vectors.

For three dimensional momentum vector objects, see *vector.backends.object.MomentumObject3D*.

GenericClass
alias of *VectorObject3D*

MomentumClass
alias of *MomentumObject3D*

ProjectionClass2D
alias of *VectorObject2D*

ProjectionClass3D
alias of *VectorObject3D*

ProjectionClass4D
alias of *VectorObject4D*

_wrap_result(*cls: Any*, *result: Any*, *returns: Any*, *num_vecargs: Any*) → Any
Wraps the raw result of a compute function as a scalar or a vector.

Parameters

- **result** – Value or tuple of values from a compute function.
- **returns** – Signature from a `dispatch_map`.
- **num_vecargs** (*int*) – Number of vector arguments in the function that would be treated on an equal footing (i.e. `add` has two, but `rotate_axis` has only one: the `axis` is secondary).

azimuthal: *AzimuthalObject*

property eta: float

The pseudorapidity η coordinate of the vector or every vector in the array (in radians, always between 0 ($+z$) and π ($-z$)).

classmethod from_rhophieta(*rho: float*, *phi: float*, *eta: float*) → *VectorObject3D*

Constructs a `VectorObject3D` from polar azimuthal coordinates and a pseudorapidity η.

Use *vector.backends.object.MomentumObject3D* to construct a vector with momentum properties and methods.

Examples

```
>>> import vector
>>> vec = vector.VectorObject3D.from_rhophieta(1, 1, 1)
>>> vec
VectorObject3D(rho=1, phi=1, eta=1)
```

classmethod from_rhophitheta(*rho: float*, *phi: float*, *theta: float*) → *VectorObject3D*

Constructs a `VectorObject3D` from polar azimuthal coordinates and a polar angle θ.

Use *vector.backends.object.MomentumObject3D* to construct a vector with momentum properties and methods.

Examples

```
>>> import vector
>>> vec = vector.VectorObject3D.from_rhophitheta(1, 1, 1)
>>> vec
VectorObject3D(rho=1, phi=1, theta=1)
```

classmethod from_rhophiz(*rho: float*, *phi: float*, *z: float*) → *VectorObject3D*

Constructs a `VectorObject3D` from polar azimuthal coordinates and a Cartesian longitudinal coordinate z.

Use *vector.backends.object.MomentumObject3D* to construct a vector with momentum properties and methods.

Examples

```
>>> import vector
>>> vec = vector.VectorObject3D.from_rhophiz(1, 1, 1)
>>> vec
VectorObject3D(rho=1, phi=1, z=1)
```

classmethod from_xyeta(*x: float*, *y: float*, *eta: float*) → *VectorObject3D*

Constructs a `VectorObject3D` from Cartesian coordinates and a pseudorapidity η.

Use *vector.backends.object.MomentumObject3D* to construct a vector with momentum properties and methods.

Examples

```
>>> import vector
>>> vec = vector.VectorObject3D.from_xyeta(1, 1, 1)
>>> vec
VectorObject3D(x=1, y=1, eta=1)
```

classmethod from_xytheta(*x: float*, *y: float*, *theta: float*) → *VectorObject3D*

Constructs a `VectorObject3D` from Cartesian azimuthal coordinates and a polar angle θ.

Use *vector.backends.object.MomentumObject3D* to construct a vector with momentum properties and methods.

Examples

```
>>> import vector
>>> vec = vector.VectorObject3D.from_xytheta(1, 1, 1)
>>> vec
VectorObject3D(x=1, y=1, theta=1)
```

classmethod from_xyz(*x: float*, *y: float*, *z: float*) → *VectorObject3D*

Constructs a `VectorObject3D` from Cartesian coordinates.

Use *vector.backends.object.MomentumObject3D* to construct a vector with momentum properties and methods.

Examples

```
>>> import vector
>>> vec = vector.VectorObject3D.from_xyz(1, 1, 1)
>>> vec
VectorObject3D(x=1, y=1, z=1)
```

longitudinal: *LongitudinalObject*

property phi: float

The polar ϕ coordinate of the vector or every vector in the array (in radians, always between $-\pi$ and π).

property rho: float

The polar ρ coordinate of the vector or every vector in the array.

This is also the magnitude of the 2D azimuthal part of the vector (not including any longitudinal or temporal parts).

property theta: float

The spherical θ coordinate (polar angle) of the vector or every vector in the array (in radians, always between 0 ($+z$) and π ($-z$)).

property x: float

The Cartesian x coordinate of the vector or every vector in the array.

property y: float

The Cartesian y coordinate of the vector or every vector in the array.

property z: float

The Cartesian z coordinate of the vector or every vector in the array.

class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *z: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *eta: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *theta: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *z: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*)
class vector.backends.object.**VectorObject4D**(*, *pt: float*, *phi: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *pz: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *z: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.object.**VectorObject4D**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.object.**VectorObject4D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *theta: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.object.**VectorObject4D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.object.**VectorObject4D**(*, *pt: float*, *phi: float*, *eta: float*)

class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.object.**VectorObject4D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.object.**VectorObject4D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.object.**VectorObject4D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.object.**VectorObject4D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.object.**VectorObject4D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.object.**VectorObject4D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.object.**VectorObject4D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)

class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.object.**VectorObject4D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.object.**VectorObject4D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.object.**VectorObject4D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.object.**VectorObject4D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.object.**VectorObject4D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.object.**VectorObject4D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.object.**VectorObject4D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)

class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.object.**VectorObject4D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.object.**VectorObject4D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.object.**VectorObject4D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.object.**VectorObject4D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.object.**VectorObject4D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.object.**VectorObject4D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.object.**VectorObject4D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.object.**VectorObject4D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)

class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.object.**VectorObject4D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.object.**VectorObject4D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.object.**VectorObject4D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.object.**VectorObject4D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.object.**VectorObject4D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.object.**VectorObject4D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)

class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.object.**VectorObject4D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.object.**VectorObject4D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.object.**VectorObject4D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.object.**VectorObject4D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.object.**VectorObject4D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.object.**VectorObject4D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.object.**VectorObject4D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.object.**VectorObject4D**(*__azumthal:* Azimuthal)
class vector.backends.object.**VectorObject4D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector.backends.object.**VectorObject4D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *VectorObject*, *Lorentz*, *Vector4D*

Four dimensional vector class for the object backend. Use the class methods -

Examples

```
>>> import vector
>>> vec = vector.VectorObject4D(x=1, y=2, z=3, t=4)
>>> vec.x, vec.y, vec.z, vec.t
(1, 2, 3, 4)
>>> vec = vector.VectorObject4D(rho=1, phi=2, eta=3, tau=4)
>>> vec.rho, vec.phi, vec.eta, vec.tau
(1, 2, 3, 4)
>>> vec = vector.VectorObject4D(
...     azimuthal=vector.backends.object.AzimuthalObjectXY(1, 2),
...     longitudinal=vector.backends.object.LongitudinalObjectTheta(3),
...     temporal=vector.backends.object.TemporalObjectTau(4)
... )
>>> vec.x, vec.y, vec.theta, vec.tau
(1, 2, 3, 4)
```

The following class methods can also be used to construct 4D object type vectors -

- *VectorObject4D.from_xyzt()*
- *VectorObject4D.from_xythetat()*
- *VectorObject4D.from_xyetat()*
- *VectorObject4D.from_rhophizt()*
- *VectorObject4D.from_rhophithetat()*

- *VectorObject4D.from_rhophietat()*
- *VectorObject4D.from_xyztau()*
- *VectorObject4D.from_xythetatau()*
- *VectorObject4D.from_xyetatau()*
- *VectorObject4D.from_rhophiztau()*
- *VectorObject4D.from_rhophithetatau()*
- *VectorObject4D.from_rhophietatau()*

Additionally, the `vector.obj()` function can also be used to construct 4D object type vectors.

For four dimensional momentum vector objects, see *vector.backends.object.MomentumObject4D*.

GenericClass

alias of *VectorObject4D*

MomentumClass

alias of *MomentumObject4D*

ProjectionClass2D

alias of *VectorObject2D*

ProjectionClass3D

alias of *VectorObject3D*

ProjectionClass4D

alias of *VectorObject4D*

_wrap_result(*cls: Any*, *result: Any*, *returns: Any*, *num_vecargs: Any*) → Any

Wraps the raw result of a compute function as a scalar or a vector.

Parameters

- **result** – Value or tuple of values from a compute function.
- **returns** – Signature from a `dispatch_map`.
- **num_vecargs** (*int*) – Number of vector arguments in the function that would be treated on an equal footing (i.e. `add` has two, but `rotate_axis` has only one: the `axis` is secondary).

azimuthal: *AzimuthalObject*

property eta: float

The pseudorapidity η coordinate of the vector or every vector in the array (in radians, always between 0 ($+z$) and π ($-z$)).

classmethod from_rhophietat(*rho: float*, *phi: float*, *eta: float*, *t: float*) → *VectorObject4D*

Constructs a `VectorObject3D` from polar azimuthal coordinates, a pseudorapidity η, and a time coordinate t.

Use *vector.backends.object.MomentumObject3D* to construct a vector with momentum properties and methods.

Examples

```
>>> import vector
>>> vec = vector.VectorObject4D.from_rhophietat(1, 1, 1, 1)
>>> vec
VectorObject4D(rho=1, phi=1, eta=1, t=1)
```

classmethod from_rhophietatau(*rho: float*, *phi: float*, *eta: float*, *tau: float*) → *VectorObject4D*

Constructs a `VectorObject3D` from polar azimuthal coordinates, a pseudorapidity η, and a proper time coordinate τ.

Use *vector.backends.object.MomentumObject3D* to construct a vector with momentum properties and methods.

Examples

```
>>> import vector
>>> vec = vector.VectorObject4D.from_rhophietatau(1, 1, 1, 1)
>>> vec
VectorObject4D(rho=1, phi=1, eta=1, tau=1)
```

classmethod from_rhophithetat(*rho: float*, *phi: float*, *theta: float*, *t: float*) → *VectorObject4D*

Constructs a `VectorObject3D` from polar azimuthal coordinates, a polar angle θ, and a time coordinate t.

Use *vector.backends.object.MomentumObject3D* to construct a vector with momentum properties and methods.

Examples

```
>>> import vector
>>> vec = vector.VectorObject4D.from_rhophithetat(1, 1, 1, 1)
>>> vec
VectorObject4D(rho=1, phi=1, theta=1, t=1)
```

classmethod from_rhophithetatau(*rho: float*, *phi: float*, *theta: float*, *tau: float*) → *VectorObject4D*

Constructs a `VectorObject3D` from polar azimuthal coordinates, a polar angle θ, and a proper time coordinate τ.

Use *vector.backends.object.MomentumObject3D* to construct a vector with momentum properties and methods.

Examples

```
>>> import vector
>>> vec = vector.VectorObject4D.from_rhophithetatau(1, 1, 1, 1)
>>> vec
VectorObject4D(rho=1, phi=1, theta=1, tau=1)
```

classmethod from_rhophizt(*rho: float*, *phi: float*, *z: float*, *t: float*) → *VectorObject4D*

Constructs a `VectorObject3D` from polar azimuthal coordinates, a Cartesian longitudinal coordinate z, and a time coordinate t.

Use *vector.backends.object.MomentumObject3D* to construct a vector with momentum properties and methods.

Examples

```
>>> import vector
>>> vec = vector.VectorObject4D.from_rhophizt(1, 1, 1, 1)
>>> vec
VectorObject4D(rho=1, phi=1, z=1, t=1)
```

classmethod from_rhophiztau(*rho: float*, *phi: float*, *z: float*, *tau: float*) → *VectorObject4D*

Constructs a `VectorObject3D` from polar azimuthal coordinates, a Cartesian longitudinal coordinate z, and a proper time coordinate τ.

Use *vector.backends.object.MomentumObject3D* to construct a vector with momentum properties and methods.

Examples

```
>>> import vector
>>> vec = vector.VectorObject4D.from_rhophiztau(1, 1, 1, 1)
>>> vec
VectorObject4D(rho=1, phi=1, z=1, tau=1)
```

classmethod from_xyetat(*x: float*, *y: float*, *eta: float*, *t: float*) → *VectorObject4D*

Constructs a `VectorObject3D` from Cartesian coordinates, a pseudorapidity η, and a time coordinate t.

Use *vector.backends.object.MomentumObject3D* to construct a vector with momentum properties and methods.

Examples

```
>>> import vector
>>> vec = vector.VectorObject4D.from_xyetat(1, 1, 1, 1)
>>> vec
VectorObject4D(x=1, y=1, eta=1, t=1)
```

classmethod from_xyetatau(*x: float*, *y: float*, *eta: float*, *tau: float*) → *VectorObject4D*

Constructs a `VectorObject3D` from Cartesian coordinates, a pseudorapidity η, and a proper time coordinate τ.

Use *vector.backends.object.MomentumObject3D* to construct a vector with momentum properties and methods.

Examples

```
>>> import vector
>>> vec = vector.VectorObject4D.from_xyetatau(1, 1, 1, 1)
>>> vec
VectorObject4D(x=1, y=1, eta=1, tau=1)
```

classmethod from_xythetat(*x: float*, *y: float*, *theta: float*, *t: float*) → *VectorObject4D*

Constructs a `VectorObject3D` from Cartesian azimuthal coordinates, a polar angle θ, and a time coordinate t.

Use *vector.backends.object.MomentumObject3D* to construct a vector with momentum properties and methods.

Examples

```
>>> import vector
>>> vec = vector.VectorObject4D.from_xythetat(1, 1, 1, 1)
>>> vec
VectorObject4D(x=1, y=1, theta=1, t=1)
```

classmethod from_xythetatau(*x: float*, *y: float*, *theta: float*, *tau: float*) → *VectorObject4D*

Constructs a `VectorObject3D` from Cartesian azimuthal coordinates, a polar angle θ, and a proper time coordinate τ.

Use *vector.backends.object.MomentumObject3D* to construct a vector with momentum properties and methods.

Examples

```
>>> import vector
>>> vec = vector.VectorObject4D.from_xythetatau(1, 1, 1, 1)
>>> vec
VectorObject4D(x=1, y=1, theta=1, tau=1)
```

classmethod from_xyzt(*x: float*, *y: float*, *z: float*, *t: float*) → *VectorObject4D*

Constructs a `VectorObject3D` from Cartesian coordinates and a time coordinate t.

Use *vector.backends.object.MomentumObject3D* to construct a vector with momentum properties and methods.

Examples

```
>>> import vector
>>> vec = vector.VectorObject4D.from_xyzt(1, 1, 1, 1)
>>> vec
VectorObject4D(x=1, y=1, z=1, t=1)
```

classmethod from_xyztau(*x: float*, *y: float*, *z: float*, *tau: float*) → *VectorObject4D*

Constructs a `VectorObject3D` from Cartesian coordinates and a proper time coordinate τ.

Use *vector.backends.object.MomentumObject3D* to construct a vector with momentum properties and methods.

Examples

```
>>> import vector
>>> vec = vector.VectorObject4D.from_xyztau(1, 1, 1, 1)
>>> vec
VectorObject4D(x=1, y=1, z=1, tau=1)
```

longitudinal: *LongitudinalObject*

property phi: float

The polar ϕ coordinate of the vector or every vector in the array (in radians, always between $-\pi$ and π).

property rho: float

The polar ρ coordinate of the vector or every vector in the array.

This is also the magnitude of the 2D azimuthal part of the vector (not including any longitudinal or temporal parts).

property t: float

The Cartesian t (time) coordinate of the vector or every vector in the array.

If t is derived from τ, it is not allowed to be `NaN`.

```
t = sqrt(max(copysign(tau**2, tau) + mag**2, 0))
```

property tau: float

The Lorentz magnitude τ (proper time) of the vector or every vector in the array.

If τ is derived from t, spacelike vectors are represented by negative proper times.

```
tau = copysign(sqrt(abs(t**2 - mag**2)), t**2 - mag**2)
```

temporal: *TemporalObject*

property theta: float

The spherical θ coordinate (polar angle) of the vector or every vector in the array (in radians, always between 0 ($+z$) and π ($-z$)).

property x: float

The Cartesian x coordinate of the vector or every vector in the array.

property y: float

The Cartesian y coordinate of the vector or every vector in the array.

property z: float

The Cartesian z coordinate of the vector or every vector in the array.

vector.backends.object.**_gather_coordinates**(*planar_class: type[*VectorObject2D*]*, *spatial_class: type[*VectorObject3D*]*, *lorentz_class: type[*VectorObject4D*]*, *coordinates: dict[str, Any]*) → Any

Helper function for *vector.backends.object.obj()*.

Constructs and returns a 2D, 3D, or 4D `VectorObject` or `MomentumObject` with the provided coordinates (dictionary), planar (`VectorObject2D` or `MomentumObject2D`), spatial (`VectorObject3D` or `MomentumObject3D`), and lorentz (`VectorObject4D` or `MomentumObject4D`) classes.

vector.backends.object.**_is_type_safe**(*coordinates: dict[str, Any]*) → bool

vector.backends.object.**_replace_data**(*obj: Any*, *result: Any*) → Any

vector.backends.object.**obj**(*, *x: float*, *y: float*) → *VectorObject2D*
vector.backends.object.**obj**(*, *x: float*, *py: float*) → *MomentumObject2D*
vector.backends.object.**obj**(*, *px: float*, *y: float*) → *MomentumObject2D*
vector.backends.object.**obj**(*, *px: float*, *py: float*) → *MomentumObject2D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*) → *VectorObject2D*
vector.backends.object.**obj**(*, *pt: float*, *phi: float*) → *MomentumObject2D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *z: float*) → *VectorObject3D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *pz: float*) → *MomentumObject3D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *z: float*) → *MomentumObject3D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *pz: float*) → *MomentumObject3D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *z: float*) → *MomentumObject3D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *pz: float*) → *MomentumObject3D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *z: float*) → *MomentumObject3D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *pz: float*) → *MomentumObject3D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *z: float*) → *VectorObject3D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *pz: float*) → *MomentumObject3D*
vector.backends.object.**obj**(*, *pt: float*, *phi: float*, *z: float*) → *MomentumObject3D*
vector.backends.object.**obj**(*, *pt: float*, *phi: float*, *pz: float*) → *MomentumObject3D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *theta: float*) → *VectorObject3D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *theta: float*) → *MomentumObject3D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *theta: float*) → *MomentumObject3D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *theta: float*) → *MomentumObject3D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *theta: float*) → *VectorObject3D*
vector.backends.object.**obj**(*, *pt: float*, *phi: float*, *theta: float*) → *MomentumObject3D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *eta: float*) → *VectorObject3D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *eta: float*) → *MomentumObject3D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *eta: float*) → *MomentumObject3D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *eta: float*) → *MomentumObject3D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *eta: float*) → *VectorObject3D*
vector.backends.object.**obj**(*, *pt: float*, *phi: float*, *eta: float*) → *MomentumObject3D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *z: float*, *t: float*) → *VectorObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *pz: float*, *t: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *z: float*, *t: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *pz: float*, *t: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *z: float*, *t: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *pz: float*, *t: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *z: float*, *t: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *pz: float*, *t: float*) → *MomentumObject4D*

vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *z: float*, *t: float*) → *VectorObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pt: float*, *phi: float*, *z: float*, *t: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *theta: float*, *t: float*) → *VectorObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *theta: float*, *t: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *theta: float*, *t: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *theta: float*, *t: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*) → *VectorObject4D*
vector.backends.object.**obj**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *eta: float*, *t: float*) → *VectorObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *eta: float*, *t: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *eta: float*, *t: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *eta: float*, *t: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*) → *VectorObject4D*
vector.backends.object.**obj**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *z: float*, *tau: float*) → *VectorObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *pz: float*, *tau: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *z: float*, *tau: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *pz: float*, *tau: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *z: float*, *tau: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *pz: float*, *tau: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *z: float*, *tau: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *pz: float*, *tau: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*) → *VectorObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *theta: float*, *tau: float*) → *VectorObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *theta: float*, *tau: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *theta: float*, *tau: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *theta: float*, *tau: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*) → *VectorObject4D*
vector.backends.object.**obj**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *eta: float*, *tau: float*) → *VectorObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *eta: float*, *tau: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *eta: float*, *tau: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *eta: float*, *tau: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*) → *VectorObject4D*
vector.backends.object.**obj**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *z: float*, *E: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *pz: float*, *E: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *z: float*, *E: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *pz: float*, *E: float*) → *MomentumObject4D*

vector.backends.object.**obj**(*, *px: float*, *y: float*, *z: float*, *E: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *pz: float*, *E: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *z: float*, *E: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *pz: float*, *E: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *z: float*, *E: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pE: float*, *phi: float*, *z: float*, *E: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *theta: float*, *E: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *theta: float*, *E: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *theta: float*, *E: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *theta: float*, *E: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *eta: float*, *E: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *eta: float*, *E: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *eta: float*, *E: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *eta: float*, *E: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *z: float*, *e: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *pz: float*, *e: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *z: float*, *e: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *pz: float*, *e: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *z: float*, *e: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *pz: float*, *e: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *z: float*, *e: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *pz: float*, *e: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *z: float*, *e: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pe: float*, *phi: float*, *z: float*, *e: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *theta: float*, *e: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *theta: float*, *e: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *theta: float*, *e: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *theta: float*, *e: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *eta: float*, *e: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *eta: float*, *e: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *eta: float*, *e: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *eta: float*, *e: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*) → *MomentumObject4D*

vector.backends.object.**obj**(*, *x: float*, *y: float*, *z: float*, *energy: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *pz: float*, *energy: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *z: float*, *energy: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *pz: float*, *energy: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *z: float*, *energy: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *pz: float*, *energy: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *z: float*, *energy: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *pz: float*, *energy: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *theta: float*, *energy: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *theta: float*, *energy: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *theta: float*, *energy: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *theta: float*, *energy: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *eta: float*, *energy: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *eta: float*, *energy: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *eta: float*, *energy: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *eta: float*, *energy: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *z: float*, *M: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *pz: float*, *M: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *z: float*, *M: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *pz: float*, *M: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *z: float*, *M: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *pz: float*, *M: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *z: float*, *M: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *pz: float*, *M: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *z: float*, *M: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pM: float*, *phi: float*, *z: float*, *M: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *theta: float*, *M: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *theta: float*, *M: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *theta: float*, *M: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *theta: float*, *M: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *eta: float*, *M: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *eta: float*, *M: float*) → *MomentumObject4D*

vector.backends.object.**obj**(*, *px: float*, *y: float*, *eta: float*, *M: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *eta: float*, *M: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *z: float*, *m: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *pz: float*, *m: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *z: float*, *m: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *pz: float*, *m: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *z: float*, *m: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *pz: float*, *m: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *z: float*, *m: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *pz: float*, *m: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *z: float*, *m: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pm: float*, *phi: float*, *z: float*, *m: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *theta: float*, *m: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *theta: float*, *m: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *theta: float*, *m: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *theta: float*, *m: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *eta: float*, *m: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *eta: float*, *m: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *eta: float*, *m: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *eta: float*, *m: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *z: float*, *mass: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *pz: float*, *mass: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *z: float*, *mass: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *pz: float*, *mass: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *z: float*, *mass: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *pz: float*, *mass: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *z: float*, *mass: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *pz: float*, *mass: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *theta: float*, *mass: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *theta: float*, *mass: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *theta: float*, *mass: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *theta: float*, *mass: float*) → *MomentumObject4D*

vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *y: float*, *eta: float*, *mass: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *x: float*, *py: float*, *eta: float*, *mass: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *y: float*, *eta: float*, *mass: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *px: float*, *py: float*, *eta: float*, *mass: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*) → *MomentumObject4D*
vector.backends.object.**obj**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*) → *MomentumObject4D*

Constructs a single `Object` type vector, whose type is determined by the keyword-only arguments to this function.

Allowed combinations are:

- (2D) `x`, `y`
- (2D) `rho`, `phi`
- (3D) `x`, `y`, `z`
- (3D) `x`, `y`, `theta`
- (3D) `x`, `y`, `eta`
- (3D) `rho`, `phi`, `z`
- (3D) `rho`, `phi`, `theta`
- (3D) `rho`, `phi`, `eta`
- (4D) `x`, `y`, `z`, `t`
- (4D) `x`, `y`, `z`, `tau`
- (4D) `x`, `y`, `theta`, `t`
- (4D) `x`, `y`, `theta`, `tau`
- (4D) `x`, `y`, `eta`, `t`
- (4D) `x`, `y`, `eta`, `tau`
- (4D) `rho`, `phi`, `z`, `t`
- (4D) `rho`, `phi`, `z`, `tau`
- (4D) `rho`, `phi`, `theta`, `t`
- (4D) `rho`, `phi`, `theta`, `tau`
- (4D) `rho`, `phi`, `eta`, `t`
- (4D) `rho`, `phi`, `eta`, `tau`

in which

- `px` may be substituted for `x`
- `py` may be substituted for `y`
- `pt` may be substituted for `rho`
- `pz` may be substituted for `z`
- `E` may be substituted for `t`
- `e` may be substituted for `t`
- `energy` may be substituted for `t`

- `M` may be substituted for `tau`
- `m` may be substituted for `tau`
- `mass` may be substituted for `tau`

to make the vector a momentum vector.

Alternatively, the *vector.backends.object.VectorObject2D*, *vector.backends.object.VectorObject3D*, and *vector.backends.object.VectorObject4D* classes (with momentum subclasses) have explicit constructors:

- *vector.backends.object.VectorObject2D.from_xy()*
- *vector.backends.object.VectorObject2D.from_rhophi()*
- *vector.backends.object.VectorObject3D.from_xyz()*
- *vector.backends.object.VectorObject3D.from_xytheta()*
- *vector.backends.object.VectorObject3D.from_xyeta()*
- *vector.backends.object.VectorObject3D.from_rhophiz()*
- *vector.backends.object.VectorObject3D.from_rhophitheta()*
- *vector.backends.object.VectorObject3D.from_rhophieta()*
- *vector.backends.object.VectorObject4D.from_xyzt()*
- *vector.backends.object.VectorObject4D.from_xyztau()*
- *vector.backends.object.VectorObject4D.from_xythetat()*
- *vector.backends.object.VectorObject4D.from_xythetatau()*
- *vector.backends.object.VectorObject4D.from_xyetat()*
- *vector.backends.object.VectorObject4D.from_xyetatau()*
- *vector.backends.object.VectorObject4D.from_rhophizt()*
- *vector.backends.object.VectorObject4D.from_rhophiztau()*
- *vector.backends.object.VectorObject4D.from_rhophithetat()*
- *vector.backends.object.VectorObject4D.from_rhophithetatau()*
- *vector.backends.object.VectorObject4D.from_rhophietat()*
- *vector.backends.object.VectorObject4D.from_rhophietatau()*

vector.backends.numpy module

Defines behaviors for NumPy Array. New arrays created with the

```
vector.array(...)
```

function will have these behaviors built in (and will pass them to any derived arrays).

class vector.backends.numpy.**AzimuthalNumpy**

Bases: *CoordinatesNumpy*, *Azimuthal*

Azimuthal class for the NumPy backend.

ObjectClass: type[*AzimuthalObject*]

class vector.backends.numpy.**AzimuthalNumpyRhoPhi**(**args: Any*, ***kwargs: Any*)

Bases: *AzimuthalNumpy*, *AzimuthalRhoPhi*, *GetItem*, ndarray

Class for the rho and phi (azimuthal) coordinates of NumPy backend. Creates a structured NumPy array and returns it as an AzimuthalNumpyXY object.

Examples

```
>>> import vector
>>> vector.backends.numpy.AzimuthalNumpyRhoPhi([(1, 1), (2.1, 3.1)], dtype=[("rho",␣
↪float), ("phi", float)])
AzimuthalNumpyRhoPhi([(1. , 1. ), (2.1, 3.1)],
                     dtype=[('rho', '<f8'), ('phi', '<f8')])
```

ObjectClass

alias of *AzimuthalObjectRhoPhi*

_IS_MOMENTUM: ClassVar[bool] = False

property elements: tuple[ndarray, ndarray]

Azimuthal coordinates (rho and phi) as a tuple.

Each coordinate is a NumPy array of values and not a vector.

Examples

```
>>> import vector
>>> vec = vector.backends.numpy.AzimuthalNumpyRhoPhi([(1, 1), (2.1, 3.1)],␣
↪dtype=[("rho", float), ("phi", float)])
>>> vec.elements
(array([1. , 2.1]), array([1. , 3.1]))
```

property phi: ndarray

The phi coordinates.

property rho: ndarray

The rho coordinates.

class vector.backends.numpy.**AzimuthalNumpyXY**(**args: Any*, ***kwargs: Any*)

Bases: *AzimuthalNumpy*, *AzimuthalXY*, *GetItem*, ndarray

Class for the x and y (azimuthal) coordinates of NumPy backend. Creates a structured NumPy array and returns it as an AzimuthalNumpyXY object.

Examples

```
>>> import vector
>>> vector.backends.numpy.AzimuthalNumpyXY([(1, 1), (2.1, 3.1)], dtype=[("x",␣
↪float), ("y", float)])
AzimuthalNumpyXY([(1. , 1. ), (2.1, 3.1)],
                 dtype=[('x', '<f8'), ('y', '<f8')])
```

ObjectClass

alias of *AzimuthalObjectXY*

_IS_MOMENTUM: ClassVar[bool] = False

property elements: tuple[ndarray, ndarray]

Azimuthal coordinates (`x` and `y`) as a tuple.

Each coordinate is a NumPy array of values and not a vector.

Examples

```
>>> import vector
>>> vec = vector.backends.numpy.AzimuthalNumpyXY([(1, 1), (2.1, 3.1)], dtype=[(
↪"x", float), ("y", float)])
>>> vec.elements
(array([1. , 2.1]), array([1. , 3.1]))
```

property x: ndarray

The `x` coordinates.

property y: ndarray

The `y` coordinates.

class vector.backends.numpy.**CoordinatesNumpy**

Bases: `object`

Coordinates class for the Numpy backend.

dtype: dtype[Any]

lib = <module 'numpy' from '/home/docs/checkouts/readthedocs.org/user_builds/vector/envs/stable/lib/python3.12/site-packages/numpy/__init__.py'>

class vector.backends.numpy.**GetItem**

Bases: `object`

_IS_MOMENTUM: ClassVar[bool]

class vector.backends.numpy.**LongitudinalNumpy**

Bases: *CoordinatesNumpy*, *Longitudinal*

Longitudinal class for the NumPy backend.

ObjectClass: type[*LongitudinalObject*]

class vector.backends.numpy.**LongitudinalNumpyEta**(**args: Any*, ***kwargs: Any*)

Bases: *LongitudinalNumpy*, *LongitudinalEta*, *GetItem*, ndarray

Class for the `eta` (longitudinal) coordinate of NumPy backend. Creates a structured NumPy array and returns it as a LongitudinalNumpyEta object.

Examples

```
>>> import vector
>>> vector.backends.numpy.LongitudinalNumpyEta([(1), (2.1)], dtype=[("eta", float)])
LongitudinalNumpyEta([(1. ,), (2.1,)], dtype=[('eta', '<f8')])
```

ObjectClass

alias of *LongitudinalObjectEta*

_IS_MOMENTUM: ClassVar[bool] = False

property elements: tuple[ndarray]

Longitudinal coordinates (`eta`) as a tuple.

Each coordinate is a NumPy array of values and not a vector.

Examples

```
>>> import vector
>>> vec = vector.backends.numpy.LongitudinalNumpyTheta([(1), (2.1)], dtype=[(
↪"theta", float)])
>>> vec.elements
(array([1. , 2.1]),)
```

property eta: ndarray

The `eta` coordinates.

class vector.backends.numpy.**LongitudinalNumpyTheta**(**args: Any*, ***kwargs: Any*)

Bases: *LongitudinalNumpy*, *LongitudinalTheta*, *GetItem*, ndarray

Class for the `theta` (longitudinal) coordinate of NumPy backend. Creates a structured NumPy array and returns it as a LongitudinalNumpyTheta object.

Examples

```
>>> import vector
>>> vector.backends.numpy.LongitudinalNumpyTheta([(1), (2.1)], dtype=[("theta",␣
↪float)])
LongitudinalNumpyTheta([(1. ,), (2.1,)], dtype=[('theta', '<f8')])
```

ObjectClass

alias of *LongitudinalObjectTheta*

_IS_MOMENTUM: ClassVar[bool] = False

property elements: tuple[ndarray]

Longitudinal coordinates (theta) as a tuple.

Each coordinate is a NumPy array of values and not a vector.

Examples

```
>>> import vector
>>> vec = vector.backends.numpy.LongitudinalNumpyTheta([(1), (2.1)], dtype=[(
↪"theta", float)])
>>> vec.elements
(array([1. , 2.1]),)
```

property theta: ndarray

The theta coordinates.

class vector.backends.numpy.**LongitudinalNumpyZ**(**args: Any*, ***kwargs: Any*)

Bases: *LongitudinalNumpy*, *LongitudinalZ*, *GetItem*, ndarray

Class for the z (longitudinal) coordinate of NumPy backend. Creates a structured NumPy array and returns it as a LongitudinalNumpyZ object.

Examples

```
>>> import vector
>>> vector.backends.numpy.LongitudinalNumpyZ([(1), (2.1)], dtype=[("z", float)])
LongitudinalNumpyZ([(1. ,), (2.1,)], dtype=[('z', '<f8')])
```

ObjectClass

alias of *LongitudinalObjectZ*

_IS_MOMENTUM: ClassVar[bool] = False

property elements: tuple[ndarray]

Longitudinal coordinates (z) as a tuple.

Each coordinate is a NumPy array of values and not a vector.

Examples

```
>>> import vector
>>> vec = vector.backends.numpy.LongitudinalNumpyZ([(1), (2.1)], dtype=[("z",␣
↪float)])
>>> vec.elements
(array([1. , 2.1]),)
```

property z: ndarray

The z coordinates.

class vector.backends.numpy.**MomentumNumpy2D**(**args: Any*, ***kwargs: Any*)

Bases: *PlanarMomentum*, *VectorNumpy2D*

Two dimensional momentum vector class for the NumPy backend. This class can be directly used to construct two dimensional NumPy momentum vectors. For two dimensional NumPy vectors see *vector.backends.numpy.VectorNumpy2D*.

Examples

```
>>> import vector
>>> vec = vector.MomentumNumpy2D([(1.1, 2.1), (1.2, 2.2), (1.3, 2.3), (1.4, 2.4),␣
↪(1.5, 2.5)],
...                dtype=[('px', float), ('py', float)])
>>> vec
MomentumNumpy2D([(1.1, 2.1), (1.2, 2.2), (1.3, 2.3), (1.4, 2.4), (1.5, 2.5)],
                dtype=[('x', '<f8'), ('y', '<f8')])
```

GenericClass

alias of *VectorNumpy2D*

MomentumClass

alias of *MomentumNumpy2D*

ObjectClass

alias of *MomentumObject2D*

ProjectionClass2D

alias of *MomentumNumpy2D*

ProjectionClass3D

alias of *MomentumNumpy3D*

ProjectionClass4D

alias of *MomentumNumpy4D*

_IS_MOMENTUM: ClassVar[bool] = True

class vector.backends.numpy.**MomentumNumpy3D**(**args: Any*, ***kwargs: Any*)

Bases: *SpatialMomentum*, *VectorNumpy3D*

Three dimensional momentum vector class for the NumPy backend. This class can be directly used to construct three dimensional NumPy momentum vectors. For three dimensional NumPy vectors see *vector.backends.numpy.VectorNumpy3D*.

Examples

```
>>> import vector
>>> vec = vector.MomentumNumpy3D([(1.1, 2.1, 3.1), (1.2, 2.2, 3.2), (1.3, 2.3, 3.3),
↪ (1.4, 2.4, 3.4), (1.5, 2.5, 3.5)],
...                dtype=[('px', float), ('py', float), ('pz', float)])
>>> vec
MomentumNumpy3D([(1.1, 2.1, 3.1), (1.2, 2.2, 3.2), (1.3, 2.3, 3.3), (1.4, 2.4, 3.4),
                 (1.5, 2.5, 3.5)], dtype=[('x', '<f8'), ('y', '<f8'), ('z', '<f8')])
```

GenericClass

alias of *VectorNumpy3D*

MomentumClass

alias of *MomentumNumpy3D*

ObjectClass

alias of *MomentumObject3D*

ProjectionClass2D

alias of *MomentumNumpy2D*

ProjectionClass3D

alias of *MomentumNumpy3D*

ProjectionClass4D

alias of *MomentumNumpy4D*

_IS_MOMENTUM: ClassVar[bool] = True

class vector.backends.numpy.**MomentumNumpy4D**(**args: Any*, ***kwargs: Any*)

Bases: *LorentzMomentum*, *VectorNumpy4D*

Four dimensional momentum vector class for the NumPy backend. This class can be directly used to construct four dimensional NumPy momentum vectors. For three dimensional NumPy vectors see *vector.backends.numpy.VectorNumpy4D*.

Examples

```
>>> import vector
>>> vec = vector.MomentumNumpy4D([(1.1, 2.1, 3.1, 4.1), (1.2, 2.2, 3.2, 4.2), (1.3,␣
↪2.3, 3.3, 4.3), (1.4, 2.4, 3.4, 4.4), (1.5, 2.5, 3.5, 4.5)],
...                 dtype=[('px', float), ('py', float), ('pz', float), ('t', float)])
>>> vec
MomentumNumpy4D([(1.1, 2.1, 3.1, 4.1), (1.2, 2.2, 3.2, 4.2), (1.3, 2.3, 3.3, 4.3),
                 (1.4, 2.4, 3.4, 4.4), (1.5, 2.5, 3.5, 4.5)],
                dtype=[('x', '<f8'), ('y', '<f8'), ('z', '<f8'), ('t', '<f8')])
```

GenericClass

alias of *VectorNumpy4D*

MomentumClass

alias of *MomentumNumpy4D*

ObjectClass

alias of *MomentumObject4D*

ProjectionClass2D

alias of *MomentumNumpy2D*

ProjectionClass3D

alias of *MomentumNumpy3D*

ProjectionClass4D

alias of *MomentumNumpy4D*

_IS_MOMENTUM: ClassVar[bool] = True

class vector.backends.numpy.**TemporalNumpy**

Bases: *CoordinatesNumpy*, *Temporal*

Temporal class for the NumPy backend.

ObjectClass: type[*TemporalObject*]

class vector.backends.numpy.**TemporalNumpyT**(**args: Any*, ***kwargs: Any*)

Bases: *TemporalNumpy*, *TemporalT*, *GetItem*, ndarray

Class for the t (temporal) coordinate of NumPy backend. Creates a structured NumPy array and returns it as a TemporalNumpyT object.

Examples

```
>>> import vector
>>> vector.backends.numpy.TemporalNumpyT([(1), (2.1)], dtype=[("t", float)])
TemporalNumpyT([(1. ,), (2.1,)], dtype=[('t', '<f8')])
```

ObjectClass

alias of *TemporalObjectT*

_IS_MOMENTUM: ClassVar[bool] = False

property elements: tuple[ndarray]

Temporal coordinates (t) as a tuple.

Each coordinate is a NumPy array of values and not a vector.

Examples

```
>>> import vector
>>> vec = vector.backends.numpy.TemporalNumpyT([(1), (2.1)], dtype=[("t",␣
↪float)])
>>> vec.elements
(array([1. , 2.1]),)
```

property t: ndarray

The t coordinates.

class vector.backends.numpy.**TemporalNumpyTau**(**args: Any*, ***kwargs: Any*)

Bases: *TemporalNumpy*, *TemporalTau*, *GetItem*, ndarray

Class for the tau (temporal) coordinate of NumPy backend.

ObjectClass

alias of *TemporalObjectTau*

_IS_MOMENTUM: ClassVar[bool] = False

property elements: tuple[ndarray]

Temporal coordinates (tau) as a tuple.

Each coordinate is a NumPy array of values and not a vector.

Examples

```
>>> import vector
>>> vec = vector.backends.numpy.TemporalNumpyTau([(1), (2.1)], dtype=[("tau",␣
↪float)])
>>> vec.elements
(array([1. , 2.1]),)
```

property tau: ndarray

The tau coordinates.

class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *z: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *eta: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *theta: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *z: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pt: float*, *phi: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *pz: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *z: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *theta: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *z: float*, *t: float*)

class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.numpy.**VectorNumpy**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.numpy.**VectorNumpy**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.numpy.**VectorNumpy**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.numpy.**VectorNumpy**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *z: float*, *E: float*)

class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *eta: float*, *e: float*)

class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)

class vector.backends.numpy.**VectorNumpy**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)

class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.numpy.**VectorNumpy**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.numpy.**VectorNumpy**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.numpy.**VectorNumpy**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.numpy.**VectorNumpy**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.numpy.**VectorNumpy**(*__azumthal:* Azimuthal)
class vector.backends.numpy.**VectorNumpy**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector.backends.numpy.**VectorNumpy**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *Vector*, *GetItem*

Mixin class for NumPy vectors.

allclose(*other:* VectorProtocol, *rtol: float | ndarray = 1e-05*, *atol: float | ndarray = 1e-08*, *equal_nan: bool | ndarray = False*) → Any

Like `np.ndarray.allclose`, but for VectorNumpy.

dtype: numpy.dtype[Any]

lib = <module 'numpy' from '/home/docs/checkouts/readthedocs.org/user_builds/vector/envs/stable/lib/python3.12/site-packages/numpy/__init__.py'>

sum(*axis: int | None = None*, *dtype: dtype[Any] | str | None = None*, *out: Any | None = None*, *keepdims: bool = False*, *initial: Any = None*, *where: Any = None*) → SameVectorNumpyType

class vector.backends.numpy.**VectorNumpy2D**(**args: Any*, ***kwargs: Any*)

Bases: *VectorNumpy*, *Planar*, *Vector2D*, `ndarray`

Two dimensional vector class for the NumPy backend. This class can be directly used to construct two dimensional NumPy vectors. For two dimensional Momentum NumPy vectors see *vector.backends.numpy.MomentumNumpy2D*.

Examples

```
>>> import vector
>>> vec = vector.VectorNumpy2D([(1.1, 2.1), (1.2, 2.2), (1.3, 2.3), (1.4, 2.4), (1.
↪5, 2.5)],
...                 dtype=[('x', float), ('y', float)])
>>> vec
VectorNumpy2D([(1.1, 2.1), (1.2, 2.2), (1.3, 2.3), (1.4, 2.4), (1.5, 2.5)],
              dtype=[('x', '<f8'), ('y', '<f8')])
```

GenericClass

alias of *VectorNumpy2D*

MomentumClass

alias of *MomentumNumpy2D*

ObjectClass

alias of *VectorObject2D*

ProjectionClass2D

alias of *VectorNumpy2D*

ProjectionClass3D

alias of *VectorNumpy3D*

ProjectionClass4D

alias of *VectorNumpy4D*

_IS_MOMENTUM: ClassVar[bool] = False

_azimuthal_type: type[*AzimuthalNumpyXY*] | type[*AzimuthalNumpyRhoPhi*]

_wrap_result(*cls: Any*, *result: Any*, *returns: Any*, *num_vecargs: Any*) → Any

Wraps the raw result of a compute function as an array of scalars or an array of vectors.

Parameters

- **result** – Value or tuple of values from a compute function.
- **returns** – Signature from a `dispatch_map`.
- **num_vecargs** (*int*) – Number of vector arguments in the function that would be treated on an equal footing (i.e. `add` has two, but `rotate_axis` has only one: the `axis` is secondary).

property azimuthal: *AzimuthalNumpy*

Returns the azimuthal type class for the given `VectorNumpy2D` object.

Examples

```
>>> import vector
>>> vec = vector.array([
...     (1.1, 2.1), (1.2, 2.2), (1.3, 2.3), (1.4, 2.4), (1.5, 2.5)
... ], dtype=[("x", float), ("y", float)])
>>> vec.azimuthal
AzimuthalNumpyXY([(1.1, 2.1), (1.2, 2.2), (1.3, 2.3), (1.4, 2.4),
                  (1.5, 2.5)], dtype=[('x', '<f8'), ('y', '<f8')])
```

class vector.backends.numpy.**VectorNumpy3D**(**args: Any*, ***kwargs: Any*)

Bases: *VectorNumpy*, *Spatial*, *Vector3D*, ndarray

Three dimensional vector class for the NumPy backend. This class can be directly used to construct three dimensional NumPy vectors. For three dimensional Momentum NumPy vectors see *vector.backends.numpy.MomentumNumpy3D*.

Examples

```
>>> import vector
>>> vec = vector.VectorNumpy3D([(1.1, 2.1, 3.1), (1.2, 2.2, 3.2), (1.3, 2.3, 3.3),␣
↪(1.4, 2.4, 3.4), (1.5, 2.5, 3.5)],
...                 dtype=[('x', float), ('y', float), ('z', float)])
>>> vec
VectorNumpy3D([(1.1, 2.1, 3.1), (1.2, 2.2, 3.2), (1.3, 2.3, 3.3), (1.4, 2.4, 3.4),
               (1.5, 2.5, 3.5)], dtype=[('x', '<f8'), ('y', '<f8'), ('z', '<f8')])
```

GenericClass

alias of *VectorNumpy3D*

MomentumClass

alias of *MomentumNumpy3D*

ObjectClass

alias of *VectorObject3D*

ProjectionClass2D

alias of *VectorNumpy2D*

ProjectionClass3D

alias of *VectorNumpy3D*

ProjectionClass4D

alias of *VectorNumpy4D*

_IS_MOMENTUM: ClassVar[bool] = False

_azimuthal_type: type[*AzimuthalNumpyXY*] | type[*AzimuthalNumpyRhoPhi*]

_longitudinal_type: type[*LongitudinalNumpyZ*] | type[*LongitudinalNumpyTheta*] | type[*LongitudinalNumpyEta*]

_wrap_result(*cls: Any*, *result: Any*, *returns: Any*, *num_vecargs: Any*) → Any

Wraps the raw result of a compute function as an array of scalars or an array of vectors.

Parameters

- **result** – Value or tuple of values from a compute function.
- **returns** – Signature from a `dispatch_map`.
- **num_vecargs** (*int*) – Number of vector arguments in the function that would be treated on an equal footing (i.e. `add` has two, but `rotate_axis` has only one: the `axis` is secondary).

property azimuthal: *AzimuthalNumpy*

Returns the azimuthal type class for the given `VectorNumpy3D` object.

property longitudinal: *LongitudinalNumpy*

Returns the longitudinal type class for the given `VectorNumpy3D` object.

class vector.backends.numpy.**VectorNumpy4D**(**args: Any*, ***kwargs: Any*)

Bases: *VectorNumpy*, *Lorentz*, *Vector4D*, ndarray

Four dimensional vector class for the NumPy backend. This class can be directly used to construct four dimensional NumPy vectors. For four dimensional Momentum NumPy vectors see *vector.backends.numpy.MomentumNumpy4D*.

Examples

```
>>> import vector
>>> vec = vector.VectorNumpy4D([(1.1, 2.1, 3.1, 4.1), (1.2, 2.2, 3.2, 4.2), (1.3, 2.
↪3, 3.3, 4.3), (1.4, 2.4, 3.4, 4.4), (1.5, 2.5, 3.5, 4.5)],
...                 dtype=[('x', float), ('y', float), ('z', float), ('t', float)])
>>> vec
VectorNumpy4D([(1.1, 2.1, 3.1, 4.1), (1.2, 2.2, 3.2, 4.2), (1.3, 2.3, 3.3, 4.3),
               (1.4, 2.4, 3.4, 4.4), (1.5, 2.5, 3.5, 4.5)],
              dtype=[('x', '<f8'), ('y', '<f8'), ('z', '<f8'), ('t', '<f8')])
```

GenericClass

alias of *VectorNumpy4D*

MomentumClass

alias of *MomentumNumpy4D*

ObjectClass

alias of *VectorObject4D*

ProjectionClass2D

alias of *VectorNumpy2D*

ProjectionClass3D

alias of *VectorNumpy3D*

ProjectionClass4D

alias of *VectorNumpy4D*

_IS_MOMENTUM: ClassVar[bool] = False

_azimuthal_type: type[*AzimuthalNumpyXY*] | type[*AzimuthalNumpyRhoPhi*]

_longitudinal_type: type[*LongitudinalNumpyZ*] | type[*LongitudinalNumpyTheta*] | type[*LongitudinalNumpyEta*]

_temporal_type: type[*TemporalNumpyT*] | type[*TemporalNumpyTau*]

_wrap_result(*cls: Any*, *result: Any*, *returns: Any*, *num_vecargs: Any*) → Any

Wraps the raw result of a compute function as an array of scalars or an array of vectors.

Parameters

- **result** – Value or tuple of values from a compute function.
- **returns** – Signature from a `dispatch_map`.
- **num_vecargs** (*int*) – Number of vector arguments in the function that would be treated on an equal footing (i.e. `add` has two, but `rotate_axis` has only one: the `axis` is secondary).

property azimuthal: *AzimuthalNumpy*

Returns the azimuthal type class for the given `VectorNumpy4D` object.

property longitudinal: *LongitudinalNumpy*

Returns the longitudinal type class for the given `Vectornumpy4D` object.

property temporal: *TemporalNumpy*

Returns the azimuthal type class for the given `VectorNumpy4D` object.

vector.backends.numpy._array_from_columns(*columns: dict[str, Any]*) → Any

Converts a dictionary (or columns) of coordinates to an array.

Parameters
columns (*dict*) – The dictionary of coordinates to be converted.

Returns
A structured array of coordinates.

Return type
np.ndarray

Examples

```
>>> import vector
>>> vector.backends.numpy._array_from_columns({"x": [1, 2, 3], "y": [1, 2, 4]})
array([(1., 1.), (2., 2.), (3., 4.)], dtype=[('x', '<f8'), ('y', '<f8')])
```

vector.backends.numpy._array_repr(*array:* VectorNumpy2D | VectorNumpy3D | VectorNumpy4D, *is_momentum: bool*) → str

Constructs the value for __repr__ function of the provided VectorNumpy class.

vector.backends.numpy._getitem(*array:* VectorNumpy2D | VectorNumpy3D | VectorNumpy4D, *where: Any*, *is_momentum: bool*) → float | ndarray

Implementation for the __getitem__ method. See *GetItem* for more details.

vector.backends.numpy._has(*array:* VectorNumpy2D | VectorNumpy3D | VectorNumpy4D | MomentumNumpy2D | MomentumNumpy3D | MomentumNumpy4D | CoordinatesNumpy, *names: tuple[str, ...]*) → bool

Checks if a NumPy vector has the provided coordinate attributes.

Parameters

- **array** (*NumPy vector*) – A NumPy Vector whose coordinate attributes have to be checked.
- **names** (*tuple*) – Names of the attributes.

Returns
If the attribute exists or not.

Return type
bool

Examples

```
>>> from vector.backends.numpy import _has
>>> from vector._methods import _coordinate_class_to_names
>>> vec = vector.array([
...     (1.1, 2.1), (1.2, 2.2), (1.3, 2.3), (1.4, 2.4), (1.5, 2.5)
... ], dtype=[("x", float), ("y", float)])
>>> _has(vec, ("x", "y"))
True
>>> _has(vec, _coordinate_class_to_names[vector._methods.AzimuthalXY])
True
>>> _has(vec, _coordinate_class_to_names[vector._methods.AzimuthalRhoPhi])
False
```

vector.backends.numpy.**_is_type_safe**(*array:* VectorNumpy2D | VectorNumpy3D | VectorNumpy4D | MomentumNumpy2D | MomentumNumpy3D | MomentumNumpy4D | CoordinatesNumpy) → bool

vector.backends.numpy.**_reduce_count_nonzero**(*a: T*, *axis: int | None = None*, ***, *keepdims: bool = False*) → Any

vector.backends.numpy.**_reduce_sum**(*a: T*, *axis: int | None = None*, *dtype: Any = None*, *out: Any = None*, *keepdims: bool = False*, *initial: Any = None*, *where: Any = None*) → T

vector.backends.numpy.**_setitem**(*array:* VectorNumpy2D | VectorNumpy3D | VectorNumpy4D, *where: Any*, *what: Any*, *is_momentum: bool*) → None

vector.backends.numpy.**_shape_of**(*result: tuple[ndarray, ...] | Any*) → tuple[int, ...]

Calculates the shape of a tuple of ``numpy.array``s. The shape returned is the highest (numerical) value of the shapes present in the tuple.

Parameters
result (*tuple*) – A tuple of ``numpy.array``s.

Returns
The calculated shape.

Return type
tuple

Examples

```
>>> from vector.backends.numpy import _shape_of
>>> import numpy as np
>>> _shape_of((np.array([1]), np.array([2])))
(1,)
>>> _shape_of((np.array([1]), np.array([2, 8]), np.array([0])))
(2,)
>>> _shape_of((np.array([1]), np.array([2, 8])))
(2,)
```

vector.backends.numpy.**_toarrays**(*result: tuple[Any, ...] | Any*) → tuple[ndarray, ...]

Converts a tuple of values to a tuple of ``numpy.array``s.

Parameters
result (*tuple*) – A tuple of values to be converted.

Returns
A tuple of `numpy.array`.

Return type
tuple

Examples

```
>>> from vector.backends.numpy import _toarrays
>>> _toarrays((1, 2, 3, 4))
(array([1.]), array([2.]), array([3.]), array([4.]))
>>> _toarrays((1, 2, (1, 2, 3)))
(array([1.]), array([2.]), array([[1., 2., 3.]]))
```

(continues on next page)

(continued from previous page)

```
>>> _toarrays((1, 2, (1, 2, False)))
(array([1.]), array([2.]), array([[1., 2., 0.]]))
```

vector.backends.numpy.**array**(**args: Any*, ***kwargs: Any*) → *VectorNumpy*

Constructs a NumPy array of vectors, whose type is determined by the dtype of the structured array or Pandas-style "columns" argument.

All allowed signatures for `np.array` can be used in this function, plus one more:

```
vector.array({"x": x_column, "y": y_column})
```

to make an array with `dtype=[("x", x_column.dtype), ("y", y_column.dtype)]`.

The array must have structured `dtype` (i.e. `dtype.names is not None`) and the following combinations of names are allowed:

- (2D) `x`, `y`
- (2D) `rho`, `phi`
- (3D) `x`, `y`, `z`
- (3D) `x`, `y`, `theta`
- (3D) `x`, `y`, `eta`
- (3D) `rho`, `phi`, `z`
- (3D) `rho`, `phi`, `theta`
- (3D) `rho`, `phi`, `eta`
- (4D) `x`, `y`, `z`, `t`
- (4D) `x`, `y`, `z`, `tau`
- (4D) `x`, `y`, `theta`, `t`
- (4D) `x`, `y`, `theta`, `tau`
- (4D) `x`, `y`, `eta`, `t`
- (4D) `x`, `y`, `eta`, `tau`
- (4D) `rho`, `phi`, `z`, `t`
- (4D) `rho`, `phi`, `z`, `tau`
- (4D) `rho`, `phi`, `theta`, `t`
- (4D) `rho`, `phi`, `theta`, `tau`
- (4D) `rho`, `phi`, `eta`, `t`
- (4D) `rho`, `phi`, `eta`, `tau`

in which

- `px` may be substituted for `x`
- `py` may be substituted for `y`
- `pt` may be substituted for `rho`
- `pz` may be substituted for `z`

- `E` may be substituted for `t`
- `e` may be substituted for `t`
- `energy` may be substituted for `t`
- `M` may be substituted for `tau`
- `m` may be substituted for `tau`
- `mass` may be substituted for `tau`

to make the vector a momentum vector.

vector.backends.awkward module

Defines behaviors for Awkward Array. New arrays created with the

```
vector.Array(...)
vector.awk(...)
vector.zip(...)
```

function will have these behaviors built in (and will pass them to any derived arrays).

Alternatively, you can

```
vector.register_awkward()
```

to install the behaviors globally, so that any record named `Vector2D`, `Vector3D`, `Vector4D`, `Momentum2D`, `Momentum3D`, or `Momentum4D` will have these properties and methods.

The Awkward-Vectors-in-Numba extension is also implemented here, since it requires two non-strict dependencies of Vector: Awkward and Numba. Awkward's `ak.behavior` manages this non-strictness well.

class vector.backends.awkward.**AwkwardProtocol**(**args*, ***kwargs*)

Bases: *Protocol*

_abc_impl = <_abc._abc_data object>

_is_protocol = True

class vector.backends.awkward.**AzimuthalAwkward**

Bases: *CoordinatesAwkward*, *Azimuthal*

Azimuthal class for the Awkward backend. See -

- *AzimuthalAwkward.from_fields()*
- *AzimuthalAwkward.from_momentum_fields()*

to construct azimuthal type objects.

classmethod from_fields(*array: Array*) → *AzimuthalAwkward*

Create a *vector.backends.awkward.AzimuthalAwkwardXY* or a *vector.backends.awkward.AzimuthalAwkwardRhoPhi*, depending on the fields in `array`.

Examples

```
>>> import vector
>>> import awkward as ak
>>> a = ak.Array([{"x": [1, 2]}, {"y": [1]}])
>>> az = vector.backends.awkward.AzimuthalAwkward.from_fields(a)
>>> az
AzimuthalAwkwardXY(<Array [[1, 2], None] type='2 * option[var * int64]'>,
↪<Array [None, [1]] type='2 * option[var * int64]'>)
>>> az.elements
(<Array [[1, 2], None] type='2 * option[var * int64]'>, <Array [None, [1]] type=
↪'2 * option[var * int64]'>)
```

classmethod from_momentum_fields(*array: Array*) → *AzimuthalAwkward*

Create a *vector.backends.awkward.AzimuthalAwkwardXY* or a *vector.backends.awkward.AzimuthalAwkwardRhoPhi*, depending on the fields in `array`, allowing momentum synonyms.

Examples

```
>>> import vector
>>> import awkward as ak
>>> a = ak.Array([{"px": [1, 2]}, {"py": [1]}])
>>> az = vector.backends.awkward.AzimuthalAwkward.from_momentum_fields(a)
>>> az
AzimuthalAwkwardXY(<Array [[1, 2], None] type='2 * option[var * int64]'>,
↪<Array [None, [1]] type='2 * option[var * int64]'>)
>>> az.elements
(<Array [[1, 2], None] type='2 * option[var * int64]'>, <Array [None, [1]] type=
↪'2 * option[var * int64]'>)
```

class vector.backends.awkward.**AzimuthalAwkwardRhoPhi**(*rho: Any*, *phi: Any*)

Bases: *AzimuthalAwkward*, *AzimuthalRhoPhi*

Class for the `rho` and `phi` (azimuthal) coordinates of Awkward backend.

Examples

```
>>> import vector
>>> import awkward as ak
>>> a = ak.Array([{"rho": [1, 2]}, {"phi": [1]}])
>>> az = vector.backends.awkward.AzimuthalAwkwardRhoPhi(a["rho"], a["phi"])
>>> az
AzimuthalAwkwardRhoPhi(<Array [[1, 2], None] type='2 * option[var * int64]'>,
↪<Array [None, [1]] type='2 * option[var * int64]'>)
>>> az.elements
(<Array [[1, 2], None] type='2 * option[var * int64]'>, <Array [None, [1]] type='2␣
↪* option[var * int64]'>)
```

property elements: tuple[ArrayOrRecord, ArrayOrRecord]

Azimuthal coordinates (`rho` and `phi`) as a tuple.

Examples

```
>>> import vector
>>> az = vector.backends.awkward.AzimuthalAwkwardRhoPhi([1, 2, 3], [1, 2])
>>> az.elements
([1, 2, 3], [1, 2])
```

phi: Any

rho: Any

class vector.backends.awkward.**AzimuthalAwkwardXY**(*x: Any*, *y: Any*)

Bases: *AzimuthalAwkward*, *AzimuthalXY*

Class for the x and y (azimuthal) coordinates of Awkward backend.

Examples

```
>>> import vector
>>> import awkward as ak
>>> a = ak.Array([{"x": [1, 2]}, {"y": [1]}])
>>> az = vector.backends.awkward.AzimuthalAwkwardXY(a["x"], a["y"])
>>> az
AzimuthalAwkwardXY(<Array [[1, 2], None] type='2 * option[var * int64]'>, <Array␣
↪[None, [1]] type='2 * option[var * int64]'>)
>>> az.elements
(<Array [[1, 2], None] type='2 * option[var * int64]'>, <Array [None, [1]] type='2␣
↪* option[var * int64]'>)
```

property elements: tuple[ArrayOrRecord, ArrayOrRecord]

Azimuthal coordinates (x and y) as a tuple.

Examples

```
>>> import vector
>>> az = vector.backends.awkward.AzimuthalAwkwardXY([1, 2, 3], [1, 2])
>>> az.elements
([1, 2, 3], [1, 2])
```

x: Any

y: Any

class vector.backends.awkward.**CoordinatesAwkward**

Bases: object

lib: ModuleType = <module 'numpy' from '/home/docs/checkouts/readthedocs.org/user_builds/vector/envs/stable/lib/python3.12/site-packages/numpy/__init__.py'>

class vector.backends.awkward.**LongitudinalAwkward**

Bases: *CoordinatesAwkward*, *Longitudinal*

Longitudinal class for the Awkward backend. See -

- *LongitudinalAwkward.from_fields()*
- *LongitudinalAwkward.from_momentum_fields()*

to construct longitudinal type objects.

classmethod from_fields(*array: Array*) → *LongitudinalAwkward*

Create a *vector.backends.awkward.LongitudinalAwkwardZ*, a *vector.backends.awkward.LongitudinalAwkwardTheta*, or a *vector.backends.awkward.LongitudinalAwkwardEta*, depending on the fields in `array`.

Examples

```
>>> import vector
>>> import awkward as ak
>>> a = ak.Array([{"theta": [1, 0]}])
>>> l = vector.backends.awkward.LongitudinalAwkward.from_fields(a)
>>> l
LongitudinalAwkwardTheta(<Array [[1, 0]] type='1 * var * int64'>,)
>>> l.elements
(<Array [[1, 0]] type='1 * var * int64'>,)
```

classmethod from_momentum_fields(*array: Array*) → *LongitudinalAwkward*

Create a *vector.backends.awkward.LongitudinalAwkwardZ*, a *vector.backends.awkward.LongitudinalAwkwardTheta*, or a *vector.backends.awkward.LongitudinalAwkwardEta*, depending on the fields in `array`, allowing momentum synonyms.

Examples

```
>>> import vector
>>> import awkward as ak
>>> a = ak.Array([{"theta": [1, 0]}])
>>> l = vector.backends.awkward.LongitudinalAwkward.from_momentum_fields(a)
>>> l
LongitudinalAwkwardTheta(<Array [[1, 0]] type='1 * var * int64'>,)
>>> l.elements
(<Array [[1, 0]] type='1 * var * int64'>,)
```

class vector.backends.awkward.**LongitudinalAwkwardEta**(*eta: Any*)

Bases: *LongitudinalAwkward*, *LongitudinalEta*

Class for the `eta` (longitudinal) coordinate of Awkward backend.

Examples

```
>>> import vector
>>> import awkward as ak
>>> a = ak.Array([{"eta": [1, 2]}])
>>> l = vector.backends.awkward.LongitudinalAwkwardEta(a["eta"])
>>> l
LongitudinalAwkwardEta(<Array [[1, 2]] type='1 * var * int64'>,)
```

(continues on next page)

(continued from previous page)

```
>>> l.elements
(<Array [[1, 2]] type='1 * var * int64'>,)
```

property elements: tuple[ArrayOrRecord]

Longitudinal coordinates (`eta`) as a tuple.

Examples

```
>>> import vector
>>> l = vector.backends.awkward.LongitudinalAwkwardEta(5)
>>> l.elements
(5,)
```

eta: Any

class vector.backends.awkward.**LongitudinalAwkwardTheta**(*theta: Any*)

Bases: *LongitudinalAwkward*, *LongitudinalTheta*

Class for the `theta` (longitudinal) coordinate of Awkward backend.

Examples

```
>>> import vector
>>> import awkward as ak
>>> a = ak.Array([{"theta": [1, 2]}])
>>> l = vector.backends.awkward.LongitudinalAwkwardTheta(a["theta"])
>>> l
LongitudinalAwkwardTheta(<Array [[1, 2]] type='1 * var * int64'>,)
>>> l.elements
(<Array [[1, 2]] type='1 * var * int64'>,)
```

property elements: tuple[ArrayOrRecord]

Longitudinal coordinates (`theta`) as a tuple.

Examples

```
>>> import vector
>>> l = vector.backends.awkward.LongitudinalAwkwardTheta(5)
>>> l.elements
(5,)
```

theta: Any

class vector.backends.awkward.**LongitudinalAwkwardZ**(*z: Any*)

Bases: *LongitudinalAwkward*, *LongitudinalZ*

Class for the `z` (longitudinal) coordinate of Awkward backend.

Examples

```
>>> import vector
>>> import awkward as ak
>>> a = ak.Array([{"z": [1, 2]}])
>>> l = vector.backends.awkward.LongitudinalAwkwardZ(a["z"])
>>> l
LongitudinalAwkwardZ(<Array [[1, 2]] type='1 * var * int64'>,)
>>> l.elements
(<Array [[1, 2]] type='1 * var * int64'>,)
```

property elements: tuple[ArrayOrRecord]

Longitudinal coordinates (z) as a tuple.

Examples

```
>>> import vector
>>> l = vector.backends.awkward.LongitudinalAwkwardZ(5)
>>> l.elements
(5,)
```

z: Any

class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *z: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pt: float*, *phi: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *z: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *theta: float*)

class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)

class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)

class vector.backends.awkward.**MomentumArray2D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)

class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)

class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray2D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray2D**(*__azumthal:* Azimuthal)
class vector.backends.awkward.**MomentumArray2D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector.backends.awkward.**MomentumArray2D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *MomentumAwkward2D*, Array

Defines awkward behavior for a 2 dimensional momentum vector.

See *MomentumRecord2D* and *VectorArray2D* for the corresponding Record and Vector classes.

GenericClass
alias of *VectorArray2D*

MomentumClass
alias of *MomentumArray2D*

ProjectionClass2D
alias of *MomentumArray2D*

ProjectionClass3D
alias of *MomentumArray3D*

ProjectionClass4D
alias of *MomentumArray4D*

_abc_impl = <_abc._abc_data object>

allclose(*other:* VectorProtocol, *rtol: Any = 1e-05*, *atol: Any = 1e-08*, *equal_nan: Any = False*) → Any

Like `np.ndarray.allclose`, but for MomentumArray4D.

class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *z: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pt: float*, *phi: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *z: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)

class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)

class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)

class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)

class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)

class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray3D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray3D**(*__azumthal:* Azimuthal)
class vector.backends.awkward.**MomentumArray3D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector.backends.awkward.**MomentumArray3D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *MomentumAwkward3D*, Array

Defines awkward behavior for a 3 dimensional momentum vector.

See *MomentumRecord3D* and *VectorArray3D* for the corresponding Record and Vector classes.

GenericClass

alias of *VectorArray3D*

MomentumClass

alias of *MomentumArray3D*

ProjectionClass2D

alias of *MomentumArray2D*

ProjectionClass3D

alias of *MomentumArray3D*

ProjectionClass4D

alias of *MomentumArray4D*

_abc_impl = <_abc._abc_data object>

allclose(*other:* VectorProtocol, *rtol: Any = 1e-05*, *atol: Any = 1e-08*, *equal_nan: Any = False*) → Any

class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *z: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pt: float*, *phi: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *z: float*)

class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)

class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)

class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)

class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)

class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray4D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumArray4D**(*__azumthal:* Azimuthal)
class vector.backends.awkward.**MomentumArray4D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector.backends.awkward.**MomentumArray4D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *MomentumAwkward4D*, Array

Defines awkward behavior for a 4 dimensional momentum vector.

See *MomentumRecord4D* and *VectorArray4D* for the corresponding Record and Vector classes.

GenericClass

alias of *VectorArray4D*

MomentumClass

alias of *MomentumArray4D*

ProjectionClass2D

alias of *MomentumArray2D*

ProjectionClass3D

alias of *MomentumArray3D*

ProjectionClass4D

alias of *MomentumArray4D*

_abc_impl = <_abc._abc_data object>

allclose(*other:* VectorProtocol, *rtol: Any = 1e-05*, *atol: Any = 1e-08*, *equal_nan: Any = False*) → Any

class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *z: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pt: float*, *phi: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *z: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)

class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *z: float*, *E: float*)

class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)

class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)

class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)

class vector.backends.awkward.**MomentumAwkward2D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward2D**(*__azumthal:* Azimuthal)
class vector.backends.awkward.**MomentumAwkward2D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector.backends.awkward.**MomentumAwkward2D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *PlanarMomentum*, *VectorAwkward2D*

Two dimensional momentum vector class for the Awkward backend. Two dimensional momentum vectors for the users are defined using the *MomentumArray2D* class.

See *VectorAwkward2D* for momentum vectors.

property azimuthal: *AzimuthalAwkward*

Returns an Azimuthal type object containing the azimuthal coordinates.

Use the `elements` property of the created object to access the coordinates.

Examples

```
>>> import vector
>>> a = vector.Array(
...     [{"px": 1, "py": 1.1}, {"px": 2, "py": 2.2}],
... )
>>> a.azimuthal
AzimuthalAwkwardXY(<Array [1, 2] type='2 * int64'>, <Array [1.1, 2.2] type='2 *␣
↪float64'>)
>>> a.azimuthal.elements
(<Array [1, 2] type='2 * int64'>, <Array [1.1, 2.2] type='2 * float64'>)
```

class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *z: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pt: float*, *phi: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *pz: float*)

class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *z: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)

class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)

class vector.backends.awkward.**MomentumAwkward3D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)

class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)

class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward3D**(*__azumthal:* Azimuthal)
class vector.backends.awkward.**MomentumAwkward3D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector.backends.awkward.**MomentumAwkward3D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *SpatialMomentum*, *VectorAwkward3D*

Three dimensional momentum vector class for the Awkward backend. Three dimensional momentum vectors for the users are defined using the *MomentumArray3D* class.

See *VectorAwkward3D* for momentum vectors.

property azimuthal: ***AzimuthalAwkward***

Returns an Azimuthal type object containing the azimuthal coordinates.

Use the `elements` property of the created object to access the coordinates.

Examples

```
>>> import vector
>>> a = vector.Array(
...       [{"px": 1, "py": 1.1, "z": 0.1}, {"px": 2, "py": 2.2, "z": 0.2}],
... )
>>> a.azimuthal
AzimuthalAwkwardXY(<Array [1, 2] type='2 * int64'>, <Array [1.1, 2.2] type='2 *␣
↪float64'>)
>>> a.azimuthal.elements
(<Array [1, 2] type='2 * int64'>, <Array [1.1, 2.2] type='2 * float64'>)
```

property longitudinal: *LongitudinalAwkward*

Returns a Longitudinal type object containing the longitudinal coordinates.

Use the `elements` property of the created object to access the coordinates.

Examples

```
>>> import vector
>>> a = vector.Array(
...       [{"px": 1, "py": 1.1, "z": 0.1}, {"px": 2, "py": 2.2, "z": 0.2}],
... )
>>> a.longitudinal
LongitudinalAwkwardZ(<Array [0.1, 0.2] type='2 * float64'>,)
>>> a.longitudinal.elements
(<Array [0.1, 0.2] type='2 * float64'>,)
```

class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *z: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pt: float*, *phi: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *z: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *pz: float*)

class vector.backends.awkward.**MomentumAwkward4D**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)

class vector.backends.awkward.**MomentumAwkward4D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)

class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)

class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)

class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumAwkward4D**(*__azumthal:* Azimuthal)
class vector.backends.awkward.**MomentumAwkward4D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector.backends.awkward.**MomentumAwkward4D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *LorentzMomentum*, *VectorAwkward4D*

Four dimensional momentum vector class for the Awkward backend. Four dimensional momentum vectors for the users are defined using the *MomentumArray4D* class.

See *VectorAwkward4D* for momentum vectors.

property azimuthal: *AzimuthalAwkward*

Returns an Azimuthal type object containing the azimuthal coordinates.

Use the `elements` property of the created object to access the coordinates.

Examples

```
>>> import vector
>>> a = vector.Array(
...       [{"px": 1, "py": 1.1, "z": 0.1, "m": 1}, {"px": 2, "py": 2.2, "z": 0.
↪2, "m": 3}],
... )
>>> a.azimuthal
AzimuthalAwkwardXY(<Array [1, 2] type='2 * int64'>, <Array [1.1, 2.2] type='2 *␣
↪float64'>)
>>> a.azimuthal.elements
(<Array [1, 2] type='2 * int64'>, <Array [1.1, 2.2] type='2 * float64'>)
```

property longitudinal: *LongitudinalAwkward*

Returns a Longitudinal type object containing the longitudinal coordinates.

Use the `elements` property of the created object to access the coordinates.

Examples

```
>>> import vector
>>> a = vector.Array(
...       [{"px": 1, "py": 1.1, "z": 0.1, "m": 1}, {"px": 2, "py": 2.2, "z": 0.
↪2, "m": 3}],
... )
>>> a.longitudinal
LongitudinalAwkwardZ(<Array [0.1, 0.2] type='2 * float64'>,)
>>> a.longitudinal.elements
(<Array [0.1, 0.2] type='2 * float64'>,)
```

property temporal: *TemporalAwkward*

Returns a Temporal type object containing the temporal coordinates.

Use the `elements` property of the created object to access the coordinates. .. rubric:: Examples

```
>>> import vector
>>> a = vector.Array(
...       [{"px": 1, "py": 1.1, "z": 0.1, "m": 1}, {"px": 2, "py": 2.2, "z": 0.
↪2, "m": 3}],
... )
>>> a.temporal
TemporalAwkwardTau(<Array [1, 3] type='2 * int64'>,)
>>> a.temporal.elements
(<Array [1, 3] type='2 * int64'>,)
```

class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*)

class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*)

class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *z: float*)

class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *eta: float*)

class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *theta: float*)

class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *z: float*)

class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *eta: float*)

class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pt: float*, *phi: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *z: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)

class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)

class vector.backends.awkward.**MomentumRecord2D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)

class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)

class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord2D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord2D**(*__azumthal:* Azimuthal)
class vector.backends.awkward.**MomentumRecord2D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)

class vector.backends.awkward.**MomentumRecord2D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *MomentumAwkward2D*, Record

Defines awkward behavior for a 2 dimensional momentum record.

See *MomentumArray2D* and *VectorArray2D* for the corresponding Momentum and Vector classes.

GenericClass

alias of *VectorRecord2D*

MomentumClass

alias of *MomentumRecord2D*

ProjectionClass2D

alias of *MomentumRecord2D*

ProjectionClass3D

alias of *MomentumRecord3D*

ProjectionClass4D

alias of *MomentumRecord4D*

class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *z: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pt: float*, *phi: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *z: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *eta: float*)

class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)

class vector.backends.awkward.**MomentumRecord3D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)

class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)

class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)

class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord3D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord3D**(*__azumthal:* Azimuthal)
class vector.backends.awkward.**MomentumRecord3D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector.backends.awkward.**MomentumRecord3D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *MomentumAwkward3D*, Record

Defines awkward behavior for a 3 dimensional momentum record.

See *MomentumArray3D* and *VectorArray3D* for the corresponding Momentum and Vector classes.

GenericClass

alias of *VectorRecord3D*

MomentumClass

alias of *MomentumRecord3D*

ProjectionClass2D

alias of *MomentumRecord2D*

ProjectionClass3D

alias of *MomentumRecord3D*

ProjectionClass4D

alias of *MomentumRecord4D*

class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *z: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *z: float*)

class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pt: float*, *phi: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *z: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)

class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)

class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)

class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)

class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord4D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**MomentumRecord4D**(*__azumthal:* Azimuthal)
class vector.backends.awkward.**MomentumRecord4D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)

class vector.backends.awkward.**MomentumRecord4D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *MomentumAwkward4D*, Record

Defines awkward behavior for a 4 dimensional momentum record.

See *MomentumArray4D* and *VectorArray4D* for the corresponding Momentum and Vector classes.

GenericClass

alias of *VectorRecord4D*

MomentumClass

alias of *MomentumRecord4D*

ProjectionClass2D

alias of *MomentumRecord2D*

ProjectionClass3D

alias of *MomentumRecord3D*

ProjectionClass4D

alias of *MomentumRecord4D*

class vector.backends.awkward.**TemporalAwkward**

Bases: *CoordinatesAwkward*, *Temporal*

Temporal class for the Awkward backend. See -

- *TemporalAwkward.from_fields()*
- *TemporalAwkward.from_momentum_fields()*

to construct longitudinal type objects.

classmethod from_fields(*array: Array*) → *TemporalAwkward*

Create a vector.backends.awkward.TemporalT or a vector.backends.awkward.TemporalTau, depending on the fields in array.

Examples

```
>>> import vector
>>> import awkward as ak
>>> a = ak.Array([{"tau": [1, 0]}])
>>> t = vector.backends.awkward.TemporalAwkward.from_fields(a)
>>> t
TemporalAwkwardTau(<Array [[1, 0]] type='1 * var * int64'>,)
>>> t.elements
(<Array [[1, 0]] type='1 * var * int64'>,)
```

classmethod from_momentum_fields(*array: Array*) → *TemporalAwkward*

Create a vector.backends.awkward.TemporalT or a vector.backends.awkward.TemporalTau, depending on the fields in array, allowing momentum synonyms.

Examples

```
>>> import vector
>>> import awkward as ak
>>> a = ak.Array([{"mass": [1, 0]}])
>>> t = vector.backends.awkward.TemporalAwkward.from_momentum_fields(a)
>>> t
TemporalAwkwardTau(<Array [[1, 0]] type='1 * var * int64'>,)
>>> t.elements
(<Array [[1, 0]] type='1 * var * int64'>,)
```

class vector.backends.awkward.**TemporalAwkwardT**(*t: Any*)

Bases: *TemporalAwkward*, *TemporalT*

Class for the `t` (temporal) coordinate of Awkward backend.

Examples

```
>>> import vector
>>> import awkward as ak
>>> a = ak.Array([{"t": [1, 2]}])
>>> t = vector.backends.awkward.TemporalAwkwardT(a["t"])
>>> t
TemporalAwkwardT(<Array [[1, 2]] type='1 * var * int64'>,)
>>> t.elements
(<Array [[1, 2]] type='1 * var * int64'>,)
```

property elements: tuple[ArrayOrRecord]

Temporal coordinates (`t`) as a tuple.

Examples

```
>>> import vector
>>> t = vector.backends.awkward.TemporalAwkwardT(5)
>>> t.elements
(5,)
```

t: Any

class vector.backends.awkward.**TemporalAwkwardTau**(*tau: Any*)

Bases: *TemporalAwkward*, *TemporalTau*

Class for the `tau` (temporal) coordinate of Awkward backend.

Examples

```
>>> import vector
>>> import awkward as ak
>>> a = ak.Array([{"tau": [1, 2]}])
>>> t = vector.backends.awkward.TemporalAwkwardTau(a["tau"])
>>> t
TemporalAwkwardTau(<Array [[1, 2]] type='1 * var * int64'>,)
>>> t.elements
(<Array [[1, 2]] type='1 * var * int64'>,)
```

property elements: tuple[ArrayOrRecord]

Temporal coordinates (`tau`) as a tuple.

Examples

```
>>> import vector
>>> t = vector.backends.awkward.TemporalAwkwardTau(5)
>>> t.elements
(5,)
```

tau: Any

class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *z: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pt: float*, *phi: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *z: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *theta: float*)

class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorArray2D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorArray2D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)

class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray2D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray2D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)

class vector.backends.awkward.**VectorArray2D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)

class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)

class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray2D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray2D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray2D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray2D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray2D**(*__azumthal:* Azimuthal)
class vector.backends.awkward.**VectorArray2D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector.backends.awkward.**VectorArray2D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *VectorAwkward2D*, Array

Defines awkward behavior for a 2 dimensional vector.

See *VectorRecord2D* and *MomentumArray2D* for the corresponding Record and Momentum classes.

GenericClass
alias of *VectorArray2D*

MomentumClass
alias of *MomentumArray2D*

ProjectionClass2D
alias of *VectorArray2D*

ProjectionClass3D
alias of *VectorArray3D*

ProjectionClass4D
alias of *VectorArray4D*

_abc_impl = <_abc._abc_data object>

allclose(*other:* VectorProtocol, *rtol: Any = 1e-05*, *atol: Any = 1e-08*, *equal_nan: Any = False*) → Any

Like `np.ndarray.allclose`, but for VectorArray2D.

class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *z: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pt: float*, *phi: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *z: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)

class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorArray3D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorArray3D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray3D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray3D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)

class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)

class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)

class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray3D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)

class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray3D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray3D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray3D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray3D**(*__azumthal:* Azimuthal)
class vector.backends.awkward.**VectorArray3D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector.backends.awkward.**VectorArray3D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *VectorAwkward3D*, Array

Defines awkward behavior for a 3 dimensional vector.

See *VectorRecord3D* and *MomentumArray3D* for the corresponding Record and Momentum classes.

GenericClass
alias of *VectorArray3D*

MomentumClass
alias of *MomentumArray3D*

ProjectionClass2D
alias of *VectorArray2D*

ProjectionClass3D
alias of *VectorArray3D*

ProjectionClass4D
alias of *VectorArray4D*

_abc_impl = <_abc._abc_data object>

allclose(*other:* VectorProtocol, *rtol: Any = 1e-05*, *atol: Any = 1e-08*, *equal_nan: Any = False*) → Any
Like np.ndarray.allclose, but for VectorArray3D.

class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *z: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pt: float*, *phi: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *z: float*)

class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)

class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorArray4D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorArray4D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray4D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray4D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)

class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)

class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)

class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray4D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray4D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray4D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray4D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorArray4D**(*__azumthal:* Azimuthal)
class vector.backends.awkward.**VectorArray4D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector.backends.awkward.**VectorArray4D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *VectorAwkward4D*, Array

Defines awkward behavior for a 4 dimensional vector.

See *VectorRecord4D* and *MomentumArray4D* for the corresponding Record and Momentum classes.

GenericClass

alias of *VectorArray4D*

MomentumClass

alias of *MomentumArray4D*

ProjectionClass2D

alias of *VectorArray2D*

ProjectionClass3D

alias of *VectorArray3D*

ProjectionClass4D

alias of *VectorArray4D*

_abc_impl = <_abc._abc_data object>

allclose(*other:* VectorProtocol, *rtol: Any = 1e-05*, *atol: Any = 1e-08*, *equal_nan: Any = False*) → Any

Like `np.ndarray.allclose`, but for VectorArray4D.

class vector.backends.awkward.**VectorAwkward**

Bases: `object`

Mixin class for Awkward vectors.

_wrap_result(*cls: Any*, *result: Any*, *returns: Any*, *num_vecargs: Any*) → Any

Wraps the raw result of a compute function as an array of scalars or an array of vectors.

Parameters

- **result** – Value or tuple of values from a compute function.
- **returns** – Signature from a `dispatch_map`.
- **num_vecargs** (*int*) – Number of vector arguments in the function that would be treated on an equal footing (i.e. `add` has two, but `rotate_axis` has only one: the `axis` is secondary).

lib: ModuleType = <module 'numpy' from '/home/docs/checkouts/readthedocs.org/user_builds/vector/envs/stable/lib/python3.12/site-packages/numpy/__init__.py'>

class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *z: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pt: float*, *phi: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *z: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *theta: float*)

class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)

class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)

class vector.backends.awkward.**VectorAwkward2D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)

class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)

class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward2D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward2D**(*__azumthal:* Azimuthal)
class vector.backends.awkward.**VectorAwkward2D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector.backends.awkward.**VectorAwkward2D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *VectorAwkward*, *Planar*, *Vector2D*

Two dimensional vector class for the Awkward backend. Two dimensional awkward vectors for the users are defined using the *VectorArray2D* class.

See *MomentumAwkward2D* for momentum vectors.

property azimuthal: *AzimuthalAwkward*

Returns an Azimuthal type object.

Use the `elements` property of the created object to access the coordinates.

Examples

```
>>> import vector
>>> a = vector.Array(
...     [{"x": 1, "y": 1.1}, {"x": 2, "y": 2.2}],
... )
>>> a.azimuthal
AzimuthalAwkwardXY(<Array [1, 2] type='2 * int64'>, <Array [1.1, 2.2] type='2 *␣
↪float64'>)
>>> a.azimuthal.elements
(<Array [1, 2] type='2 * int64'>, <Array [1.1, 2.2] type='2 * float64'>)
```

class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *z: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pt: float*, *phi: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *z: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)

class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)

class vector.backends.awkward.**VectorAwkward3D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)

class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)

class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)

class vector.backends.awkward.**VectorAwkward3D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward3D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward3D**(*__azumthal:* Azimuthal)
class vector.backends.awkward.**VectorAwkward3D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector.backends.awkward.**VectorAwkward3D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *VectorAwkward*, *Spatial*, *Vector3D*

Three dimensional vector class for the Awkward backend. Three dimensional awkward vectors for the users are defined using the *VectorArray3D* class.

See *MomentumAwkward3D* for momentum vectors.

property azimuthal: *AzimuthalAwkward*

Returns an Azimuthal type object containing the azimuthal coordinates.

Use the `elements` property of the created object to access the coordinates.

Examples

```
>>> import vector
>>> a = vector.Array(
...       [{"x": 1, "y": 1.1, "z": 0.1}, {"x": 2, "y": 2.2, "z": 0.2}],
... )
>>> a.azimuthal
AzimuthalAwkwardXY(<Array [1, 2] type='2 * int64'>, <Array [1.1, 2.2] type='2 *␣
↪float64'>)
>>> a.azimuthal.elements
(<Array [1, 2] type='2 * int64'>, <Array [1.1, 2.2] type='2 * float64'>)
```

property longitudinal: *LongitudinalAwkward*

Returns a Longitudinal type object containing the longitudinal coordinates.

Use the `elements` property of the created object to access the coordinates.

Examples

```
>>> import vector
>>> a = vector.Array(
...       [{"x": 1, "y": 1.1, "z": 0.1}, {"x": 2, "y": 2.2, "z": 0.2}],
... )
>>> a.longitudinal
LongitudinalAwkwardZ(<Array [0.1, 0.2] type='2 * float64'>,)
>>> a.longitudinal.elements
(<Array [0.1, 0.2] type='2 * float64'>,)
```

class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *z: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *z: float*)

class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pt: float*, *phi: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *z: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)

class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)

class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)

class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)

class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward4D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorAwkward4D**(*__azumthal:* Azimuthal)
class vector.backends.awkward.**VectorAwkward4D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)

class vector.backends.awkward.**VectorAwkward4D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *VectorAwkward*, *Lorentz*, *Vector4D*

Four dimensional momentum vector class for the Awkward backend. Four dimensional momentum vectors for the users are defined using the *MomentumArray4D* class.

See *VectorAwkward4D* for momentum vectors.

property azimuthal: *AzimuthalAwkward*

Returns an Azimuthal type object containing the azimuthal coordinates.

Use the `elements` property of the created object to access the coordinates.

Examples

```
>>> import vector
>>> a = vector.Array(
...       [{"x": 1, "y": 1.1, "z": 0.1, "tau": 1}, {"x": 2, "y": 2.2, "z": 0.2,
↪"tau": 3}],
... )
>>> a.azimuthal
AzimuthalAwkwardXY(<Array [1, 2] type='2 * int64'>, <Array [1.1, 2.2] type='2 *␣
↪float64'>)
>>> a.azimuthal.elements
(<Array [1, 2] type='2 * int64'>, <Array [1.1, 2.2] type='2 * float64'>)
```

property longitudinal: *LongitudinalAwkward*

Returns a Longitudinal type object containing the longitudinal coordinates.

Use the `elements` property of the created object to access the coordinates.

Examples

```
>>> import vector
>>> a = vector.Array(
...       [{"x": 1, "y": 1.1, "z": 0.1, "tau": 1}, {"x": 2, "y": 2.2, "z": 0.2,
↪"tau": 3}],
... )
>>> a.longitudinal
LongitudinalAwkwardZ(<Array [0.1, 0.2] type='2 * float64'>,)
>>> a.longitudinal.elements
(<Array [0.1, 0.2] type='2 * float64'>,)
```

property temporal: *TemporalAwkward*

Returns a Temporal type object containing the temporal coordinates.

Use the `elements` property of the created object to access the coordinates.

Examples

```
>>> import vector
>>> a = vector.Array(
...       [{"x": 1, "y": 1.1, "z": 0.1, "tau": 1}, {"x": 2, "y": 2.2, "z": 0.2,
↪"tau": 3}],
... )
>>> a.temporal
TemporalAwkwardTau(<Array [1, 3] type='2 * int64'>,)
>>> a.temporal.elements
(<Array [1, 3] type='2 * int64'>,)
```

class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *z: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pt: float*, *phi: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *z: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)

class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)

class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)

class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)

class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)

class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord2D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord2D**(*__azumthal:* Azimuthal)
class vector.backends.awkward.**VectorRecord2D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector.backends.awkward.**VectorRecord2D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *VectorAwkward2D*, Record

Defines awkward behavior for a 2 dimensional vector record.

See *VectorArray2D* and *MomentumArray2D* for the corresponding Vector and Momentum classes.

GenericClass

alias of *VectorRecord2D*

MomentumClass

alias of *MomentumRecord2D*

ProjectionClass2D

alias of *VectorRecord2D*

ProjectionClass3D

alias of *VectorRecord3D*

ProjectionClass4D

alias of *VectorRecord4D*

class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *z: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pt: float*, *phi: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *pz: float*)

class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *z: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)

class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)

class vector.backends.awkward.**VectorRecord3D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)

class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)

class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord3D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord3D**(*__azumthal:* Azimuthal)
class vector.backends.awkward.**VectorRecord3D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector.backends.awkward.**VectorRecord3D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *VectorAwkward3D*, Record

Defines awkward behavior for a 3 dimensional vector record.

See *VectorArray3D* and *MomentumArray3D* for the corresponding Vector and Momentum classes.

GenericClass

alias of *VectorRecord3D*

MomentumClass

alias of *MomentumRecord3D*

ProjectionClass2D

alias of *VectorRecord2D*

ProjectionClass3D

alias of *VectorRecord3D*

ProjectionClass4D

alias of *VectorRecord4D*

class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *z: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pt: float*, *phi: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *z: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *z: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *pz: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *z: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *pz: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pt: float*, *phi: float*, *z: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *theta: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *theta: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *eta: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *eta: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *z: float*, *t: float*)

class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *z: float*, *E: float*)

class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)

class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)

class vector.backends.awkward.**VectorRecord4D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)

class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord4D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector.backends.awkward.**VectorRecord4D**(*__azumthal:* Azimuthal)
class vector.backends.awkward.**VectorRecord4D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector.backends.awkward.**VectorRecord4D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *VectorAwkward4D*, Record

Defines awkward behavior for a 4 dimensional vector record.

See *VectorArray4D* and *MomentumArray4D* for the corresponding Vector and Momentum classes.

GenericClass
alias of *VectorRecord4D*

MomentumClass
alias of *MomentumRecord4D*

ProjectionClass2D
alias of *VectorRecord2D*

ProjectionClass3D
alias of *VectorRecord3D*

ProjectionClass4D
alias of *VectorRecord4D*

vector.backends.awkward.**_arraytype_of**(*awkwardtype: Any*, *component: str*) → Any

vector.backends.awkward.**_aztype_of**(*recordarraytype: Any*, *is_momentum: bool*) → Any

vector.backends.awkward.**_class_to_name**(*cls: type[*VectorProtocol*]*) → str

vector.backends.awkward.**_lookup_field**(*record_type: Any*, *name: str*) → int

vector.backends.awkward.**_ltype_of**(*recordarraytype: Any*, *is_momentum: bool*) → Any

vector.backends.awkward.**_no_record**(*x: Array*) → Array | None

vector.backends.awkward.**_numba_lower**(*context: Any*, *builder: Any*, *sig: Any*, *args: Any*) → Any

vector.backends.awkward.**_numba_typer_Momentum2D**(*viewtype: Any*) → Any

vector.backends.awkward.**_numba_typer_Momentum3D**(*viewtype: Any*) → Any

vector.backends.awkward.**_numba_typer_Momentum4D**(*viewtype: Any*) → Any

vector.backends.awkward.**_numba_typer_Vector2D**(*viewtype: Any*) → Any

vector.backends.awkward.**_numba_typer_Vector3D**(*viewtype: Any*) → Any

vector.backends.awkward.**_numba_typer_Vector4D**(*viewtype: Any*) → Any

vector.backends.awkward.**_reduce_count**(*array:* VectorArray2D | VectorArray3D | VectorArray4D | MomentumArray2D | MomentumArray3D | MomentumArray4D, *mask_identity: bool*) → *VectorProtocol*

vector.backends.awkward.**_reduce_count_nonzero**(*array:* VectorArray2D | VectorArray3D | VectorArray4D | MomentumArray2D | MomentumArray3D | MomentumArray4D, *mask_identity: bool*) → Any

vector.backends.awkward.**_reduce_sum**(*array:* VectorArray2D | VectorArray3D | VectorArray4D | MomentumArray2D | MomentumArray3D | MomentumArray4D, *mask_identity: bool*) → *VectorProtocol*

vector.backends.awkward.**_ttype_of**(*recordarraytype: Any*, *is_momentum: bool*) → Any

vector.backends.awkward.**_yes_record**(*x: Array*) → float | Record | None

vector.backends.awkward.**impl**(*array:* VectorArray2D | VectorArray3D | VectorArray4D | MomentumArray2D | MomentumArray3D | MomentumArray4D, *mask_identity: bool*) → Any

vector.backends.awkward_constructors module

vector.backends.awkward_constructors.**Array**(**args: Any*, ***kwargs: Any*) → Any

Constructs an Awkward Array of vectors, whose type is determined by the fields of the record array (which may be nested within lists or other non-record structures).

All allowed signatures for `ak.Array` can be used in this function.

The array must contain records with the following combinations of field names:

- (2D) `x`, `y`
- (2D) `rho`, `phi`
- (3D) `x`, `y`, `z`
- (3D) `x`, `y`, `theta`
- (3D) `x`, `y`, `eta`
- (3D) `rho`, `phi`, `z`
- (3D) `rho`, `phi`, `theta`
- (3D) `rho`, `phi`, `eta`
- (4D) `x`, `y`, `z`, `t`
- (4D) `x`, `y`, `z`, `tau`
- (4D) `x`, `y`, `theta`, `t`
- (4D) `x`, `y`, `theta`, `tau`
- (4D) `x`, `y`, `eta`, `t`
- (4D) `x`, `y`, `eta`, `tau`

- (4D) `rho`, `phi`, `z`, `t`
- (4D) `rho`, `phi`, `z`, `tau`
- (4D) `rho`, `phi`, `theta`, `t`
- (4D) `rho`, `phi`, `theta`, `tau`
- (4D) `rho`, `phi`, `eta`, `t`
- (4D) `rho`, `phi`, `eta`, `tau`

in which

- `px` may be substituted for `x`
- `py` may be substituted for `y`
- `pt` may be substituted for `rho`
- `pz` may be substituted for `z`
- `E` may be substituted for `t`
- `e` may be substituted for `t`
- `energy` may be substituted for `t`
- `M` may be substituted for `tau`
- `m` may be substituted for `tau`
- `mass` may be substituted for `tau`

to make the vector a momentum vector.

No constraints are placed on the types of the vector fields, though if they are not numbers, mathematical operations will fail. Usually, you want them to be integers or floating-point numbers.

vector.backends.awkward_constructors.**_check_names**(*projectable: Any*, *fieldnames: list[str]*) → tuple[bool, int, list[str], Any]

vector.backends.awkward_constructors.**_is_type_safe**(*array_type: Any*) → bool

vector.backends.awkward_constructors.**_recname**(*is_momentum: bool*, *dimension: int*) → str

vector.backends.awkward_constructors.**zip**(*arrays: dict[str, Any]*, *depth_limit: int | None = None*) → Any

Constructs an Awkward Array of vectors, whose type is determined by the fields of the record array (which may be nested within lists or other non-record structures).

This function accepts a subset of `ak.zip`'s arguments.

Parameters

- **arrays** (*dict of str to array-like*) – Arrays, lists, etc. to zip together. Unlike `ak.zip`, this must be a dict with string keys to determine the coordinate system of the arrays; it may not be a tuple.
- **depth_limit** (*None or int*) – If None, attempt to fully broadcast the `array` to all levels. If an int, limit the number of dimensions that get broadcasted. The minimum value is `1`, for no broadcasting.

The array must contain records with the following combinations of field names:

- (2D) `x`, `y`
- (2D) `rho`, `phi`

- (3D) `x`, `y`, `z`
- (3D) `x`, `y`, `theta`
- (3D) `x`, `y`, `eta`
- (3D) `rho`, `phi`, `z`
- (3D) `rho`, `phi`, `theta`
- (3D) `rho`, `phi`, `eta`
- (4D) `x`, `y`, `z`, `t`
- (4D) `x`, `y`, `z`, `tau`
- (4D) `x`, `y`, `theta`, `t`
- (4D) `x`, `y`, `theta`, `tau`
- (4D) `x`, `y`, `eta`, `t`
- (4D) `x`, `y`, `eta`, `tau`
- (4D) `rho`, `phi`, `z`, `t`
- (4D) `rho`, `phi`, `z`, `tau`
- (4D) `rho`, `phi`, `theta`, `t`
- (4D) `rho`, `phi`, `theta`, `tau`
- (4D) `rho`, `phi`, `eta`, `t`
- (4D) `rho`, `phi`, `eta`, `tau`

in which

- `px` may be substituted for `x`
- `py` may be substituted for `y`
- `pt` may be substituted for `rho`
- `pz` may be substituted for `z`
- `E` may be substituted for `t`
- `e` may be substituted for `t`
- `energy` may be substituted for `t`
- `M` may be substituted for `tau`
- `m` may be substituted for `tau`
- `mass` may be substituted for `tau`

to make the vector a momentum vector.

vector.backends._numba module

vector.backends.numba_numpy module

Implements VectorNumpys in Numba. Only __getitem__ should be overloaded to return lowered VectorObjects, and maybe the `vector.array` constructor should also be handled.

As mentioned in scikit-hep/vector#43, this capability is blocked by numba/numba#6148.

vector.backends._numba_object module

Implements VectorObjects in Numba. Every function should be made usable in Numba.

class vector.backends._numba_object.**MomentumObject2DType**(**args: Any*, ***kwargs: Any*)

Bases: *VectorObject2DType*

instance_class

alias of *MomentumObject2D*

vector.backends._numba_object.**MomentumObject2D_constructor_typer**(*context*)

class vector.backends._numba_object.**MomentumObject3DType**(**args: Any*, ***kwargs: Any*)

Bases: *VectorObject3DType*

instance_class

alias of *MomentumObject3D*

vector.backends._numba_object.**MomentumObject3D_constructor_typer**(*context*)

class vector.backends._numba_object.**MomentumObject4DType**(**args: Any*, ***kwargs: Any*)

Bases: *VectorObject4DType*

instance_class

alias of *MomentumObject4D*

vector.backends._numba_object.**MomentumObject4DType_E**(*v*)

vector.backends._numba_object.**MomentumObject4DType_E2**(*v*)

vector.backends._numba_object.**MomentumObject4DType_M**(*v*)

vector.backends._numba_object.**MomentumObject4DType_M2**(*v*)

vector.backends._numba_object.**MomentumObject4DType_energy**(*v*)

vector.backends._numba_object.**MomentumObject4DType_energy2**(*v*)

vector.backends._numba_object.**MomentumObject4DType_mass**(*v*)

vector.backends._numba_object.**MomentumObject4DType_mass2**(*v*)

vector.backends._numba_object.**MomentumObject4DType_transverse_energy**(*v*)

vector.backends._numba_object.**MomentumObject4DType_transverse_energy2**(*v*)

vector.backends._numba_object.**MomentumObject4DType_transverse_mass**(*v*)

vector.backends._numba_object.**MomentumObject4DType_transverse_mass2**(*v*)

vector.backends._numba_object.**MomentumObject4D_constructor_typer**(*context*)

class vector.backends._numba_object.**VectorObject2DType**(**args: Any*, ***kwargs: Any*)

Bases: Type

instance_class

alias of *VectorObject2D*

vector.backends._numba_object.**VectorObject2DType_unit**(*v*)

vector.backends._numba_object.**VectorObject2D_box**(*typ*, *val*, *c*)

vector.backends._numba_object.**VectorObject2D_constructor_typer**(*context*)

vector.backends._numba_object.**VectorObject2D_to_Vector2D**(*v*)

vector.backends._numba_object.**VectorObject2D_to_Vector3D**(*v*)

vector.backends._numba_object.**VectorObject2D_to_Vector4D**(*v*)

vector.backends._numba_object.**VectorObject2D_typeof**(*val*, *c*)

vector.backends._numba_object.**VectorObject2D_unbox**(*typ*, *obj*, *c*)

class vector.backends._numba_object.**VectorObject3DType**(**args: Any*, ***kwargs: Any*)

Bases: Type

instance_class

alias of *VectorObject3D*

vector.backends._numba_object.**VectorObject3DType_unit**(*v*)

vector.backends._numba_object.**VectorObject3D_box**(*typ*, *val*, *c*)

vector.backends._numba_object.**VectorObject3D_constructor_typer**(*context*)

vector.backends._numba_object.**VectorObject3D_to_Vector2D**(*v*)

vector.backends._numba_object.**VectorObject3D_to_Vector3D**(*v*)

vector.backends._numba_object.**VectorObject3D_to_Vector4D**(*v*)

vector.backends._numba_object.**VectorObject3D_typeof**(*val*, *c*)

vector.backends._numba_object.**VectorObject3D_unbox**(*typ*, *obj*, *c*)

class vector.backends._numba_object.**VectorObject4DType**(**args: Any*, ***kwargs: Any*)

Bases: Type

instance_class

alias of *VectorObject4D*

vector.backends._numba_object.**VectorObject4DType_boost**(*v*, *booster*)

vector.backends._numba_object.**VectorObject4DType_boostCM_of**(*v*, *booster*)

vector.backends._numba_object.**VectorObject4DType_boostCM_of_beta3**(*v*, *beta3*)

vector.backends._numba_object.**VectorObject4DType_boostCM_of_p4**(*v*, *p4*)

vector.backends._numba_object.**VectorObject4DType_boost_beta3**(*v*, *beta3*)

vector.backends._numba_object.**VectorObject4DType_boost_p4**(*v*, *p4*)

vector.backends._numba_object.**VectorObject4DType_is_lightlike**(*v*, *tolerance=1e-05*)

vector.backends._numba_object.**VectorObject4DType_is_spacelike**(*v*, *tolerance=0*)

vector.backends._numba_object.**VectorObject4DType_is_timelike**(*v*, *tolerance=0*)

vector.backends._numba_object.**VectorObject4DType_neg4D**(*v*)

vector.backends._numba_object.**VectorObject4DType_scale3D**(*v*, *factor*)

vector.backends._numba_object.**VectorObject4DType_to_beta3**(*v*)

vector.backends._numba_object.**VectorObject4DType_transform4D**(*v*, *obj*)

vector.backends._numba_object.**VectorObject4DType_unit**(*v*)

vector.backends._numba_object.**VectorObject4D_box**(*typ*, *val*, *c*)

vector.backends._numba_object.**VectorObject4D_constructor_typer**(*context*)

vector.backends._numba_object.**VectorObject4D_to_Vector2D**(*v*)

vector.backends._numba_object.**VectorObject4D_to_Vector3D**(*v*)

vector.backends._numba_object.**VectorObject4D_to_Vector4D**(*v*)

vector.backends._numba_object.**VectorObject4D_typeof**(*val*, *c*)

vector.backends._numba_object.**VectorObject4D_unbox**(*typ*, *obj*, *c*)

vector.backends._numba_object.**_awkward_numba_E**(*record*)

vector.backends._numba_object.**_awkward_numba_M**(*record*)

vector.backends._numba_object.**_awkward_numba_e**(*record*)

vector.backends._numba_object.**_awkward_numba_energy**(*record*)

vector.backends._numba_object.**_awkward_numba_eta**(*record*)

vector.backends._numba_object.**_awkward_numba_m**(*record*)

vector.backends._numba_object.**_awkward_numba_mass**(*record*)

vector.backends._numba_object.**_awkward_numba_ptphi**(*record*)

vector.backends._numba_object.**_awkward_numba_pxpy**(*record*)

vector.backends._numba_object.**_awkward_numba_pxy**(*record*)

vector.backends._numba_object.**_awkward_numba_pz**(*record*)

vector.backends._numba_object.**_awkward_numba_rhophi**(*record*)

vector.backends._numba_object.**_awkward_numba_t**(*record*)

vector.backends._numba_object.**_awkward_numba_tau**(*record*)

vector.backends._numba_object.**_awkward_numba_theta**(*record*)

vector.backends._numba_object.**_awkward_numba_xpy**(*record*)

vector.backends._numba_object.**_awkward_numba_xy**(*record*)

vector.backends._numba_object.**_awkward_numba_z**(*record*)

vector.backends._numba_object.**add_binary_method**(*vectortype*, *gn*, *methodname*)

vector.backends._numba_object.**add_boostXYZ**(*methodname*)

vector.backends._numba_object.**add_coordinate_change**(*vectortype*, *azcoordtype*, *lcoordtype*, *tcoordtype*)

vector.backends._numba_object.**add_isclose_method**(*vectortype*)

vector.backends._numba_object.**add_lorentz_property**(*vectortype*, *propertyname*)

vector.backends._numba_object.**add_planar_property**(*vectortype*, *propertyname*)

vector.backends._numba_object.**add_rotateZ**(*vectortype*)

vector.backends._numba_object.**add_spatial_property**(*vectortype*, *propertyname*)

vector.backends._numba_object.**add_tolerance_method**(*vectortype*, *methodname*)

vector.backends._numba_object.**add_transform2D**(*vectortype*)

vector.backends._numba_object.**azimuthalrhophi_coord1**(*v*)

vector.backends._numba_object.**azimuthalrhophi_coord2**(*v*)

vector.backends._numba_object.**azimuthalxy_coord1**(*v*)

vector.backends._numba_object.**azimuthalxy_coord2**(*v*)

vector.backends._numba_object.**dimension_of**(*v*)

vector.backends._numba_object.**flavor_of**(*v1*, *v2*)

vector.backends._numba_object.**is_azimuthaltype**(*typ*)

vector.backends._numba_object.**is_longitudinaltype**(*typ*)

vector.backends._numba_object.**is_temporaltype**(*typ*)

vector.backends._numba_object.**longitudinaleta_coord1**(*v*)

vector.backends._numba_object.**longitudinaltheta_coord1**(*v*)

vector.backends._numba_object.**longitudinalz_coord1**(*v*)

vector.backends._numba_object.**make_AzimuthalObjectRhoPhi**(*v*)

vector.backends._numba_object.**make_AzimuthalObjectXY**(*v*)

vector.backends._numba_object.**make_LongitudinalObjectEta**(*v*)

vector.backends._numba_object.**make_LongitudinalObjectEta_zero**(*v*)

vector.backends._numba_object.**make_LongitudinalObjectTheta**(*v*)

vector.backends._numba_object.**make_LongitudinalObjectTheta_zero**(*v*)

vector.backends._numba_object.**make_LongitudinalObjectZ**(*v*)

vector.backends._numba_object.**make_LongitudinalObjectZ_zero**(*v*)

vector.backends._numba_object.**make_TemporalObjectT**(*v*)

vector.backends._numba_object.**make_TemporalObjectT_zero**(*v*)

vector.backends._numba_object.**make_TemporalObjectTau**(*v*)

vector.backends._numba_object.**make_TemporalObjectTau_zero**(*v*)

vector.backends._numba_object.**nan_to_num**(*x*, *copy=True*, *nan=0.0*, *posinf=None*, *neginf=None*)

vector.backends._numba_object.**numba_aztype**(*typ*)

vector.backends._numba_object.**numba_ltype**(*typ*)

vector.backends._numba_object.**numba_ttype**(*typ*)

vector.backends._numba_object.**numpy_absolute**(*v*)

vector.backends._numba_object.**numpy_add**(*v1*, *v2*)

vector.backends._numba_object.**numpy_cbrt**(*v*)

vector.backends._numba_object.**numpy_matmul**(*v1*, *v2*)

vector.backends._numba_object.**numpy_multiply**(*a*, *b*)

vector.backends._numba_object.**numpy_negative**(*v*)

vector.backends._numba_object.**numpy_positive**(*v*)

vector.backends._numba_object.**numpy_power**(*v*, *expo*)

vector.backends._numba_object.**numpy_sqrt**(*v*)

vector.backends._numba_object.**numpy_square**(*v*)

vector.backends._numba_object.**numpy_subtract**(*v1*, *v2*)

vector.backends._numba_object.**numpy_true_divide**(*a*, *b*)

vector.backends._numba_object.**operator_abs**(*v*)

vector.backends._numba_object.**operator_add**(*v1*, *v2*)

vector.backends._numba_object.**operator_eq**(*v1*, *v2*)

vector.backends._numba_object.**operator_matmul**(*v1*, *v2*)

vector.backends._numba_object.**operator_mul**(*a*, *b*)

vector.backends._numba_object.**operator_ne**(*v1*, *v2*)

vector.backends._numba_object.**operator_neg**(*v*)

vector.backends._numba_object.**operator_pos**(*v*)

vector.backends._numba_object.**operator_pow**(*a*, *b*)

vector.backends._numba_object.**operator_sub**(*v1*, *v2*)

vector.backends._numba_object.**operator_truediv**(*a*, *b*)

vector.backends._numba_object.**operator_truth**(*v*)

vector.backends._numba_object.**temporalt_coord1**(*v*)

vector.backends._numba_object.**temporaltau_coord1**(*v*)

vector.backends._numba_object.**vector_obj**(*unrecognized_argument=None*, *x=None*, *px=None*, *y=None*, *py=None*, *rho=None*, *pt=None*, *phi=None*, *z=None*, *pz=None*, *theta=None*, *eta=None*, *t=None*, *E=None*, *e=None*, *energy=None*, *tau=None*, *M=None*, *m=None*, *mass=None*)

vector.backends._numba_object.**vector_obj_Azimuthal_ptphi**(*x*, *px*, *y*, *py*, *rho*, *pt*, *phi*)

vector.backends._numba_object.**vector_obj_Azimuthal_pxpy**(*x*, *px*, *y*, *py*, *rho*, *pt*, *phi*)

vector.backends._numba_object.**vector_obj_Azimuthal_pxy**(*x*, *px*, *y*, *py*, *rho*, *pt*, *phi*)

vector.backends._numba_object.**vector_obj_Azimuthal_rhophi**(*x*, *px*, *y*, *py*, *rho*, *pt*, *phi*)

vector.backends._numba_object.**vector_obj_Azimuthal_xpy**(*x*, *px*, *y*, *py*, *rho*, *pt*, *phi*)

vector.backends._numba_object.**vector_obj_Azimuthal_xy**(*x*, *px*, *y*, *py*, *rho*, *pt*, *phi*)

vector.backends._numba_object.**vector_obj_Longitudinal_eta**(*z*, *pz*, *theta*, *eta*)

vector.backends._numba_object.**vector_obj_Longitudinal_pz**(*z*, *pz*, *theta*, *eta*)

vector.backends._numba_object.**vector_obj_Longitudinal_theta**(*z*, *pz*, *theta*, *eta*)

vector.backends._numba_object.**vector_obj_Longitudinal_z**(*z*, *pz*, *theta*, *eta*)

vector.backends._numba_object.**vector_obj_Temporal_E**(*t*, *E*, *e*, *energy*, *tau*, *M*, *m*, *mass*)

vector.backends._numba_object.**vector_obj_Temporal_M**(*t*, *E*, *e*, *energy*, *tau*, *M*, *m*, *mass*)

vector.backends._numba_object.**vector_obj_Temporal_e**(*t*, *E*, *e*, *energy*, *tau*, *M*, *m*, *mass*)

vector.backends._numba_object.**vector_obj_Temporal_energy**(*t*, *E*, *e*, *energy*, *tau*, *M*, *m*, *mass*)

vector.backends._numba_object.**vector_obj_Temporal_m**(*t*, *E*, *e*, *energy*, *tau*, *M*, *m*, *mass*)

vector.backends._numba_object.**vector_obj_Temporal_mass**(*t*, *E*, *e*, *energy*, *tau*, *M*, *m*, *mass*)

vector.backends._numba_object.**vector_obj_Temporal_t**(*t*, *E*, *e*, *energy*, *tau*, *M*, *m*, *mass*)

vector.backends._numba_object.**vector_obj_Temporal_tau**(*t*, *E*, *e*, *energy*, *tau*, *M*, *m*, *mass*)

vector.backends._numba_object.**vectortype**

alias of *VectorObject4DType*

vector._compute package

Subpackages

vector._compute.lorentz package

Compute functions for lorentz vectors, which is to say 4D.

Each function is a module with variants for each coordinate system (or combination of coordinate systems) as functions within the module.

Each module has a `dispatch_map` (dict) that maps coordinate types to the appropriate function and its return type(s), and a `dispatch` (function) uses this information to call the right function and return the right type.

The compute functions themselves are restricted to a minimum of Python features: no statements other than assignments and one return, no assumptions about data types. In particular, if statements and loops are not allowed. The tests/test_compute_features.py suite ensures that these rules are followed (though that set of allowed features can be expanded if it doesn't prevent the addition of new backends).

Submodules

vector._compute.lorentz.add module

```
Lorentz.add(self, other)
```

vector._compute.lorentz.add.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.lorentz.add.**make_conversion**(*azimuthal1*, *longitudinal1*, *temporal1*, *azimuthal2*, *longitudinal2*, *temporal2*)

vector._compute.lorentz.beta module

```
@property
Lorentz.beta(self)
```

vector._compute.lorentz.beta.**dispatch**(*v: Any*) → Any

vector._compute.lorentz.beta.**rhophi_eta_t**(*lib*, *rho*, *phi*, *eta*, *t*)

vector._compute.lorentz.beta.**rhophi_eta_tau**(*lib*, *rho*, *phi*, *eta*, *tau*)

vector._compute.lorentz.beta.**rhophi_theta_t**(*lib*, *rho*, *phi*, *theta*, *t*)

vector._compute.lorentz.beta.**rhophi_theta_tau**(*lib*, *rho*, *phi*, *theta*, *tau*)

vector._compute.lorentz.beta.**rhophi_z_t**(*lib*, *rho*, *phi*, *z*, *t*)

vector._compute.lorentz.beta.**rhophi_z_tau**(*lib*, *rho*, *phi*, *z*, *tau*)

vector._compute.lorentz.beta.**xy_eta_t**(*lib*, *x*, *y*, *eta*, *t*)

vector._compute.lorentz.beta.**xy_eta_tau**(*lib*, *x*, *y*, *eta*, *tau*)

vector._compute.lorentz.beta.**xy_theta_t**(*lib*, *x*, *y*, *theta*, *t*)

vector._compute.lorentz.beta.**xy_theta_tau**(*lib*, *x*, *y*, *theta*, *tau*)

vector._compute.lorentz.beta.**xy_z_t**(*lib*, *x*, *y*, *z*, *t*)

vector._compute.lorentz.beta.**xy_z_tau**(*lib*, *x*, *y*, *z*, *tau*)

vector._compute.lorentz.boost_beta3 module

```
Lorentz.boost_beta3(self, beta3)
```

or

```
Lorentz.boost(self, beta3=...)
```

vector._compute.lorentz.boost_beta3.**cartesian_t**(*lib*, *x1*, *y1*, *z1*, *t1*, *betax*, *betay*, *betaz*)

vector._compute.lorentz.boost_beta3.**cartesian_t_rhophi_eta**(*lib*, *x1*, *y1*, *z1*, *t1*, *rho2*, *phi2*, *eta2*)

vector._compute.lorentz.boost_beta3.**cartesian_t_rhophi_theta**(*lib*, *x1*, *y1*, *z1*, *t1*, *rho2*, *phi2*, *theta2*)

vector._compute.lorentz.boost_beta3.**cartesian_t_rhophi_z**(*lib*, *x1*, *y1*, *z1*, *t1*, *rho2*, *phi2*, *z2*)

vector._compute.lorentz.boost_beta3.**cartesian_t_xy_eta**(*lib*, *x1*, *y1*, *z1*, *t1*, *x2*, *y2*, *eta2*)

vector._compute.lorentz.boost_beta3.**cartesian_t_xy_theta**(*lib*, *x1*, *y1*, *z1*, *t1*, *x2*, *y2*, *theta2*)

vector._compute.lorentz.boost_beta3.**cartesian_t_xy_z**(*lib*, *x1*, *y1*, *z1*, *t1*, *x2*, *y2*, *z2*)

vector._compute.lorentz.boost_beta3.**cartesian_tau**(*lib*, *x1*, *y1*, *z1*, *tau1*, *betax*, *betay*, *betaz*)

vector._compute.lorentz.boost_beta3.**cartesian_tau_rhophi_eta**(*lib*, *x1*, *y1*, *z1*, *tau1*, *rho2*, *phi2*, *eta2*)

vector._compute.lorentz.boost_beta3.**cartesian_tau_rhophi_theta**(*lib*, *x1*, *y1*, *z1*, *tau1*, *rho2*, *phi2*, *theta2*)

vector._compute.lorentz.boost_beta3.**cartesian_tau_rhophi_z**(*lib*, *x1*, *y1*, *z1*, *tau1*, *rho2*, *phi2*, *z2*)

vector._compute.lorentz.boost_beta3.**cartesian_tau_xy_eta**(*lib*, *x1*, *y1*, *z1*, *tau1*, *x2*, *y2*, *eta2*)

vector._compute.lorentz.boost_beta3.**cartesian_tau_xy_theta**(*lib*, *x1*, *y1*, *z1*, *tau1*, *x2*, *y2*, *theta2*)

vector._compute.lorentz.boost_beta3.**cartesian_tau_xy_z**(*lib*, *x1*, *y1*, *z1*, *tau1*, *x2*, *y2*, *z2*)

vector._compute.lorentz.boost_beta3.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.lorentz.boost_beta3.**make_conversion**(*azimuthal1*, *longitudinal1*, *temporal1*, *azimuthal2*, *longitudinal2*)

vector._compute.lorentz.boost_p4 module

```
Lorentz.boost_p4(self, p4)
```

or

```
Lorentz.boost(self, p4=...)
```

vector._compute.lorentz.boost_p4.**cartesian_t**(*lib*, *x1*, *y1*, *z1*, *t1*, *energy*, *mass*, *mass2*, *x2*, *y2*, *z2*)

vector._compute.lorentz.boost_p4.**cartesian_t_rhophi_eta_t**(*lib*, *x1*, *y1*, *z1*, *t1*, *rho2*, *phi2*, *eta2*, *t2*)

vector._compute.lorentz.boost_p4.**cartesian_t_rhophi_eta_tau**(*lib*, *x1*, *y1*, *z1*, *t1*, *rho2*, *phi2*, *eta2*, *tau2*)

vector._compute.lorentz.boost_p4.**cartesian_t_rhophi_theta_t**(*lib*, *x1*, *y1*, *z1*, *t1*, *rho2*, *phi2*, *theta2*, *t2*)

vector._compute.lorentz.boost_p4.**cartesian_t_rhophi_theta_tau**(*lib*, *x1*, *y1*, *z1*, *t1*, *rho2*, *phi2*, *theta2*, *tau2*)

vector._compute.lorentz.boost_p4.**cartesian_t_rhophi_z_t**(*lib*, *x1*, *y1*, *z1*, *t1*, *rho2*, *phi2*, *z2*, *t2*)

vector._compute.lorentz.boost_p4.**cartesian_t_rhophi_z_tau**(*lib*, *x1*, *y1*, *z1*, *t1*, *rho2*, *phi2*, *z2*, *tau2*)

vector._compute.lorentz.boost_p4.**cartesian_t_xy_eta_t**(*lib*, *x1*, *y1*, *z1*, *t1*, *x2*, *y2*, *eta2*, *t2*)

vector._compute.lorentz.boost_p4.**cartesian_t_xy_eta_tau**(*lib*, *x1*, *y1*, *z1*, *t1*, *x2*, *y2*, *eta2*, *tau2*)

vector._compute.lorentz.boost_p4.**cartesian_t_xy_theta_t**(*lib*, *x1*, *y1*, *z1*, *t1*, *x2*, *y2*, *theta2*, *t2*)

vector._compute.lorentz.boost_p4.**cartesian_t_xy_theta_tau**(*lib*, *x1*, *y1*, *z1*, *t1*, *x2*, *y2*, *theta2*, *tau2*)

vector._compute.lorentz.boost_p4.**cartesian_t_xy_z_t**(*lib*, *x1*, *y1*, *z1*, *t1*, *x2*, *y2*, *z2*, *t2*)

vector._compute.lorentz.boost_p4.**cartesian_t_xy_z_tau**(*lib*, *x1*, *y1*, *z1*, *t1*, *x2*, *y2*, *z2*, *tau2*)

vector._compute.lorentz.boost_p4.**cartesian_tau**(*lib*, *x1*, *y1*, *z1*, *tau1*, *energy*, *mass*, *mass2*, *x2*, *y2*, *z2*)

vector._compute.lorentz.boost_p4.**cartesian_tau_rhophi_eta_t**(*lib*, *x1*, *y1*, *z1*, *tau1*, *rho2*, *phi2*, *eta2*, *t2*)

vector._compute.lorentz.boost_p4.**cartesian_tau_rhophi_eta_tau**(*lib*, *x1*, *y1*, *z1*, *tau1*, *rho2*, *phi2*, *eta2*, *tau2*)

vector._compute.lorentz.boost_p4.**cartesian_tau_rhophi_theta_t**(*lib*, *x1*, *y1*, *z1*, *tau1*, *rho2*, *phi2*, *theta2*, *t2*)

vector._compute.lorentz.boost_p4.**cartesian_tau_rhophi_theta_tau**(*lib*, *x1*, *y1*, *z1*, *tau1*, *rho2*, *phi2*, *theta2*, *tau2*)

vector._compute.lorentz.boost_p4.**cartesian_tau_rhophi_z_t**(*lib*, *x1*, *y1*, *z1*, *tau1*, *rho2*, *phi2*, *z2*, *t2*)

vector._compute.lorentz.boost_p4.**cartesian_tau_rhophi_z_tau**(*lib*, *x1*, *y1*, *z1*, *tau1*, *rho2*, *phi2*, *z2*, *tau2*)

vector._compute.lorentz.boost_p4.**cartesian_tau_xy_eta_t**(*lib*, *x1*, *y1*, *z1*, *tau1*, *x2*, *y2*, *eta2*, *t2*)

vector._compute.lorentz.boost_p4.**cartesian_tau_xy_eta_tau**(*lib*, *x1*, *y1*, *z1*, *tau1*, *x2*, *y2*, *eta2*, *tau2*)

vector._compute.lorentz.boost_p4.**cartesian_tau_xy_theta_t**(*lib*, *x1*, *y1*, *z1*, *tau1*, *x2*, *y2*, *theta2*, *t2*)

vector._compute.lorentz.boost_p4.**cartesian_tau_xy_theta_tau**(*lib*, *x1*, *y1*, *z1*, *tau1*, *x2*, *y2*, *theta2*, *tau2*)

vector._compute.lorentz.boost_p4.**cartesian_tau_xy_z_t**(*lib*, *x1*, *y1*, *z1*, *tau1*, *x2*, *y2*, *z2*, *t2*)

vector._compute.lorentz.boost_p4.**cartesian_tau_xy_z_tau**(*lib*, *x1*, *y1*, *z1*, *tau1*, *x2*, *y2*, *z2*, *tau2*)

vector._compute.lorentz.boost_p4.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.lorentz.boost_p4.**make_conversion**(*azimuthal1*, *longitudinal1*, *temporal1*, *azimuthal2*, *longitudinal2*, *temporal2*)

vector._compute.lorentz.boostX_beta module

```
Lorentz.boostX(self, beta=...)
```

vector._compute.lorentz.boostX_beta.**dispatch**(*beta: Any*, *v: Any*) → Any

vector._compute.lorentz.boostX_beta.**rhophi_eta_t**(*lib*, *beta*, *rho*, *phi*, *eta*, *t*)

vector._compute.lorentz.boostX_beta.**rhophi_eta_tau**(*lib*, *beta*, *rho*, *phi*, *eta*, *tau*)

vector._compute.lorentz.boostX_beta.**rhophi_theta_t**(*lib*, *beta*, *rho*, *phi*, *theta*, *t*)

vector._compute.lorentz.boostX_beta.**rhophi_theta_tau**(*lib*, *beta*, *rho*, *phi*, *theta*, *tau*)

vector._compute.lorentz.boostX_beta.**rhophi_z_t**(*lib*, *beta*, *rho*, *phi*, *z*, *t*)

vector._compute.lorentz.boostX_beta.**rhophi_z_tau**(*lib*, *beta*, *rho*, *phi*, *z*, *tau*)

vector._compute.lorentz.boostX_beta.**xy_eta_t**(*lib*, *beta*, *x*, *y*, *eta*, *t*)

vector._compute.lorentz.boostX_beta.**xy_eta_tau**(*lib*, *beta*, *x*, *y*, *eta*, *tau*)

vector._compute.lorentz.boostX_beta.**xy_theta_t**(*lib*, *beta*, *x*, *y*, *theta*, *t*)

vector._compute.lorentz.boostX_beta.**xy_theta_tau**(*lib*, *beta*, *x*, *y*, *theta*, *tau*)

vector._compute.lorentz.boostX_beta.**xy_z_t**(*lib*, *beta*, *x*, *y*, *z*, *t*)

vector._compute.lorentz.boostX_beta.**xy_z_tau**(*lib*, *beta*, *x*, *y*, *z*, *tau*)

vector._compute.lorentz.boostX_gamma module

```
Lorentz.boostX(self, gamma=)
```

vector._compute.lorentz.boostX_gamma.**dispatch**(*gamma: Any*, *v: Any*) → Any

vector._compute.lorentz.boostX_gamma.**rhophi_eta_t**(*lib*, *gamma*, *rho*, *phi*, *eta*, *t*)

vector._compute.lorentz.boostX_gamma.**rhophi_eta_tau**(*lib*, *gamma*, *rho*, *phi*, *eta*, *tau*)

vector._compute.lorentz.boostX_gamma.**rhophi_theta_t**(*lib*, *gamma*, *rho*, *phi*, *theta*, *t*)

vector._compute.lorentz.boostX_gamma.**rhophi_theta_tau**(*lib*, *gamma*, *rho*, *phi*, *theta*, *tau*)

vector._compute.lorentz.boostX_gamma.**rhophi_z_t**(*lib*, *gamma*, *rho*, *phi*, *z*, *t*)

vector._compute.lorentz.boostX_gamma.**rhophi_z_tau**(*lib*, *gamma*, *rho*, *phi*, *z*, *tau*)

vector._compute.lorentz.boostX_gamma.**xy_eta_t**(*lib*, *gamma*, *x*, *y*, *eta*, *t*)

vector._compute.lorentz.boostX_gamma.**xy_eta_tau**(*lib*, *gamma*, *x*, *y*, *eta*, *tau*)

vector._compute.lorentz.boostX_gamma.**xy_theta_t**(*lib*, *gamma*, *x*, *y*, *theta*, *t*)

vector._compute.lorentz.boostX_gamma.**xy_theta_tau**(*lib*, *gamma*, *x*, *y*, *theta*, *tau*)

vector._compute.lorentz.boostX_gamma.**xy_z_t**(*lib*, *gamma*, *x*, *y*, *z*, *t*)

vector._compute.lorentz.boostX_gamma.**xy_z_tau**(*lib*, *gamma*, *x*, *y*, *z*, *tau*)

vector._compute.lorentz.boostY_beta module

```
Lorentz.boostY(self, beta=...)
```

vector._compute.lorentz.boostY_beta.**dispatch**(*beta: Any*, *v: Any*) → Any

vector._compute.lorentz.boostY_beta.**rhophi_eta_t**(*lib*, *beta*, *rho*, *phi*, *eta*, *t*)

vector._compute.lorentz.boostY_beta.**rhophi_eta_tau**(*lib*, *beta*, *rho*, *phi*, *eta*, *tau*)

vector._compute.lorentz.boostY_beta.**rhophi_theta_t**(*lib*, *beta*, *rho*, *phi*, *theta*, *t*)

vector._compute.lorentz.boostY_beta.**rhophi_theta_tau**(*lib*, *beta*, *rho*, *phi*, *theta*, *tau*)

vector._compute.lorentz.boostY_beta.**rhophi_z_t**(*lib*, *beta*, *rho*, *phi*, *z*, *t*)

vector._compute.lorentz.boostY_beta.**rhophi_z_tau**(*lib*, *beta*, *rho*, *phi*, *z*, *tau*)

vector._compute.lorentz.boostY_beta.**xy_eta_t**(*lib*, *beta*, *x*, *y*, *eta*, *t*)

vector._compute.lorentz.boostY_beta.**xy_eta_tau**(*lib*, *beta*, *x*, *y*, *eta*, *tau*)

vector._compute.lorentz.boostY_beta.**xy_theta_t**(*lib*, *beta*, *x*, *y*, *theta*, *t*)

vector._compute.lorentz.boostY_beta.**xy_theta_tau**(*lib*, *beta*, *x*, *y*, *theta*, *tau*)

vector._compute.lorentz.boostY_beta.**xy_z_t**(*lib*, *beta*, *x*, *y*, *z*, *t*)

vector._compute.lorentz.boostY_beta.**xy_z_tau**(*lib*, *beta*, *x*, *y*, *z*, *tau*)

vector._compute.lorentz.boostY_gamma module

```
Lorentz.boostY(self, gamma=...)
```

vector._compute.lorentz.boostY_gamma.**dispatch**(*gamma: Any*, *v: Any*) → Any

vector._compute.lorentz.boostY_gamma.**rhophi_eta_t**(*lib*, *gamma*, *rho*, *phi*, *eta*, *t*)

vector._compute.lorentz.boostY_gamma.**rhophi_eta_tau**(*lib*, *gamma*, *rho*, *phi*, *eta*, *tau*)

vector._compute.lorentz.boostY_gamma.**rhophi_theta_t**(*lib*, *gamma*, *rho*, *phi*, *theta*, *t*)

vector._compute.lorentz.boostY_gamma.**rhophi_theta_tau**(*lib*, *gamma*, *rho*, *phi*, *theta*, *tau*)

vector._compute.lorentz.boostY_gamma.**rhophi_z_t**(*lib*, *gamma*, *rho*, *phi*, *z*, *t*)

vector._compute.lorentz.boostY_gamma.**rhophi_z_tau**(*lib*, *gamma*, *rho*, *phi*, *z*, *tau*)

vector._compute.lorentz.boostY_gamma.**xy_eta_t**(*lib*, *gamma*, *x*, *y*, *eta*, *t*)

vector._compute.lorentz.boostY_gamma.**xy_eta_tau**(*lib*, *gamma*, *x*, *y*, *eta*, *tau*)

vector._compute.lorentz.boostY_gamma.**xy_theta_t**(*lib*, *gamma*, *x*, *y*, *theta*, *t*)

vector._compute.lorentz.boostY_gamma.**xy_theta_tau**(*lib*, *gamma*, *x*, *y*, *theta*, *tau*)

vector._compute.lorentz.boostY_gamma.**xy_z_t**(*lib*, *gamma*, *x*, *y*, *z*, *t*)

vector._compute.lorentz.boostY_gamma.**xy_z_tau**(*lib*, *gamma*, *x*, *y*, *z*, *tau*)

vector._compute.lorentz.boostZ_beta module

```
Lorentz.boostZ(self, beta=...)
```

vector._compute.lorentz.boostZ_beta.**dispatch**(*beta: Any*, *v: Any*) → Any

vector._compute.lorentz.boostZ_beta.**rhophi_eta_t**(*lib*, *beta*, *rho*, *phi*, *eta*, *t*)

vector._compute.lorentz.boostZ_beta.**rhophi_eta_tau**(*lib*, *beta*, *rho*, *phi*, *eta*, *tau*)

vector._compute.lorentz.boostZ_beta.**rhophi_theta_t**(*lib*, *beta*, *rho*, *phi*, *theta*, *t*)

vector._compute.lorentz.boostZ_beta.**rhophi_theta_tau**(*lib*, *beta*, *rho*, *phi*, *theta*, *tau*)

vector._compute.lorentz.boostZ_beta.**rhophi_z_t**(*lib*, *beta*, *rho*, *phi*, *z*, *t*)

vector._compute.lorentz.boostZ_beta.**rhophi_z_tau**(*lib*, *beta*, *rho*, *phi*, *z*, *tau*)

vector._compute.lorentz.boostZ_beta.**xy_eta_t**(*lib*, *beta*, *x*, *y*, *eta*, *t*)

vector._compute.lorentz.boostZ_beta.**xy_eta_tau**(*lib*, *beta*, *x*, *y*, *eta*, *tau*)

vector._compute.lorentz.boostZ_beta.**xy_theta_t**(*lib*, *beta*, *x*, *y*, *theta*, *t*)

vector._compute.lorentz.boostZ_beta.**xy_theta_tau**(*lib*, *beta*, *x*, *y*, *theta*, *tau*)

vector._compute.lorentz.boostZ_beta.**xy_z_t**(*lib*, *beta*, *x*, *y*, *z*, *t*)

vector._compute.lorentz.boostZ_beta.**xy_z_tau**(*lib*, *beta*, *x*, *y*, *z*, *tau*)

vector._compute.lorentz.boostZ_gamma module

```
Lorentz.boostZ(self, gamma=...)
```

vector._compute.lorentz.boostZ_gamma.**dispatch**(*gamma: Any*, *v: Any*) → Any

vector._compute.lorentz.boostZ_gamma.**rhophi_eta_t**(*lib*, *gamma*, *rho*, *phi*, *eta*, *t*)

vector._compute.lorentz.boostZ_gamma.**rhophi_eta_tau**(*lib*, *gamma*, *rho*, *phi*, *eta*, *tau*)

vector._compute.lorentz.boostZ_gamma.**rhophi_theta_t**(*lib*, *gamma*, *rho*, *phi*, *theta*, *t*)

vector._compute.lorentz.boostZ_gamma.**rhophi_theta_tau**(*lib*, *gamma*, *rho*, *phi*, *theta*, *tau*)

vector._compute.lorentz.boostZ_gamma.**rhophi_z_t**(*lib*, *gamma*, *rho*, *phi*, *z*, *t*)

vector._compute.lorentz.boostZ_gamma.**rhophi_z_tau**(*lib*, *gamma*, *rho*, *phi*, *z*, *tau*)

vector._compute.lorentz.boostZ_gamma.**xy_eta_t**(*lib*, *gamma*, *x*, *y*, *eta*, *t*)

vector._compute.lorentz.boostZ_gamma.**xy_eta_tau**(*lib*, *gamma*, *x*, *y*, *eta*, *tau*)

vector._compute.lorentz.boostZ_gamma.**xy_theta_t**(*lib*, *gamma*, *x*, *y*, *theta*, *t*)

vector._compute.lorentz.boostZ_gamma.**xy_theta_tau**(*lib*, *gamma*, *x*, *y*, *theta*, *tau*)

vector._compute.lorentz.boostZ_gamma.**xy_z_t**(*lib*, *gamma*, *x*, *y*, *z*, *t*)

vector._compute.lorentz.boostZ_gamma.**xy_z_tau**(*lib*, *gamma*, *x*, *y*, *z*, *tau*)

vector._compute.lorentz.deltaRapidityPhi module

```
Spatial.deltaRapidityPhi(self, other)
```

vector._compute.lorentz.deltaRapidityPhi.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.lorentz.deltaRapidityPhi.**make_conversion**(*azimuthal1*, *longitudinal1*, *temporal1*, *azimuthal2*, *longitudinal2*, *temporal2*)

vector._compute.lorentz.deltaRapidityPhi2 module

```
Spatial.deltaRapidityPhi2(self, other)
```

vector._compute.lorentz.deltaRapidityPhi2.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.lorentz.deltaRapidityPhi2.**make_conversion**(*azimuthal1*, *longitudinal1*, *temporal1*, *azimuthal2*, *longitudinal2*, *temporal2*)

vector._compute.lorentz.dot module

```
Lorentz.dot(self, other)
```

vector._compute.lorentz.dot.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.lorentz.dot.**make_conversion**(*azimuthal1*, *longitudinal1*, *temporal1*, *azimuthal2*, *longitudinal2*, *temporal2*)

vector._compute.lorentz.equal module

```
Lorentz.equal(self, other)
```

vector._compute.lorentz.equal.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.lorentz.equal.**make_conversion**(*azimuthal1*, *longitudinal1*, *temporal1*, *azimuthal2*, *longitudinal2*, *temporal2*)

vector._compute.lorentz.Et module

```
@property
Lorentz.Et(self)
```

vector._compute.lorentz.Et.**dispatch**(*v: Any*) → Any

vector._compute.lorentz.Et.**rhophi_eta_t**(*lib*, *rho*, *phi*, *eta*, *t*)

vector._compute.lorentz.Et.**rhophi_eta_tau**(*lib*, *rho*, *phi*, *eta*, *tau*)

vector._compute.lorentz.Et.**rhophi_theta_t**(*lib*, *rho*, *phi*, *theta*, *t*)

vector._compute.lorentz.Et.**rhophi_theta_tau**(*lib*, *rho*, *phi*, *theta*, *tau*)

vector._compute.lorentz.Et.**rhophi_z_t**(*lib*, *rho*, *phi*, *z*, *t*)

vector._compute.lorentz.Et.**rhophi_z_tau**(*lib*, *rho*, *phi*, *z*, *tau*)

vector._compute.lorentz.Et.**xy_eta_t**(*lib*, *x*, *y*, *eta*, *t*)

vector._compute.lorentz.Et.**xy_eta_tau**(*lib*, *x*, *y*, *eta*, *tau*)

vector._compute.lorentz.Et.**xy_theta_t**(*lib*, *x*, *y*, *theta*, *t*)

vector._compute.lorentz.Et.**xy_theta_tau**(*lib*, *x*, *y*, *theta*, *tau*)

vector._compute.lorentz.Et.**xy_z_t**(*lib*, *x*, *y*, *z*, *t*)

vector._compute.lorentz.Et.**xy_z_tau**(*lib*, *x*, *y*, *z*, *tau*)

vector._compute.lorentz.Et2 module

```
@property
Lorentz.Et2(self)
```

vector._compute.lorentz.Et2.**dispatch**(*v: Any*) → Any

vector._compute.lorentz.Et2.**rhophi_eta_t**(*lib*, *rho*, *phi*, *eta*, *t*)

vector._compute.lorentz.Et2.**rhophi_eta_tau**(*lib*, *rho*, *phi*, *eta*, *tau*)

vector._compute.lorentz.Et2.**rhophi_theta_t**(*lib*, *rho*, *phi*, *theta*, *t*)

vector._compute.lorentz.Et2.**rhophi_theta_tau**(*lib*, *rho*, *phi*, *theta*, *tau*)

vector._compute.lorentz.Et2.**rhophi_z_t**(*lib*, *rho*, *phi*, *z*, *t*)

vector._compute.lorentz.Et2.**rhophi_z_tau**(*lib*, *rho*, *phi*, *z*, *tau*)

vector._compute.lorentz.Et2.**xy_eta_t**(*lib*, *x*, *y*, *eta*, *t*)

vector._compute.lorentz.Et2.**xy_eta_tau**(*lib*, *x*, *y*, *eta*, *tau*)

vector._compute.lorentz.Et2.**xy_theta_t**(*lib*, *x*, *y*, *theta*, *t*)

vector._compute.lorentz.Et2.**xy_theta_tau**(*lib*, *x*, *y*, *theta*, *tau*)

vector._compute.lorentz.Et2.**xy_z_t**(*lib*, *x*, *y*, *z*, *t*)

vector._compute.lorentz.Et2.**xy_z_tau**(*lib*, *x*, *y*, *z*, *tau*)

vector._compute.lorentz.gamma module

```
@property
Lorentz.gamma(self)
```

vector._compute.lorentz.gamma.**dispatch**(*v: Any*) → Any

vector._compute.lorentz.gamma.**rhophi_eta_t**(*lib*, *rho*, *phi*, *eta*, *t*)

vector._compute.lorentz.gamma.**rhophi_eta_tau**(*lib*, *rho*, *phi*, *eta*, *tau*)

vector._compute.lorentz.gamma.**rhophi_theta_t**(*lib*, *rho*, *phi*, *theta*, *t*)

vector._compute.lorentz.gamma.**rhophi_theta_tau**(*lib*, *rho*, *phi*, *theta*, *tau*)

vector._compute.lorentz.gamma.**rhophi_z_t**(*lib*, *rho*, *phi*, *z*, *t*)

vector._compute.lorentz.gamma.**rhophi_z_tau**(*lib*, *rho*, *phi*, *z*, *tau*)

vector._compute.lorentz.gamma.**xy_eta_t**(*lib*, *x*, *y*, *eta*, *t*)

vector._compute.lorentz.gamma.**xy_eta_tau**(*lib*, *x*, *y*, *eta*, *tau*)

vector._compute.lorentz.gamma.**xy_theta_t**(*lib*, *x*, *y*, *theta*, *t*)

vector._compute.lorentz.gamma.**xy_theta_tau**(*lib*, *x*, *y*, *theta*, *tau*)

vector._compute.lorentz.gamma.**xy_z_t**(*lib*, *x*, *y*, *z*, *t*)

vector._compute.lorentz.gamma.**xy_z_tau**(*lib*, *x*, *y*, *z*, *tau*)

vector._compute.lorentz.is_lightlike module

```
Lorentz.is_lightlike(self, tolerance=...)
```

vector._compute.lorentz.is_lightlike.**dispatch**(*tolerance: Any*, *v: Any*) → Any

vector._compute.lorentz.is_lightlike.**make_function**(*azimuthal*, *longitudinal*, *temporal*)

vector._compute.lorentz.is_spacelike module

```
Lorentz.is_spacelike(self, tolerance=...)
```

vector._compute.lorentz.is_spacelike.**dispatch**(*tolerance: Any*, *v: Any*) → Any

vector._compute.lorentz.is_spacelike.**make_function**(*azimuthal*, *longitudinal*, *temporal*)

vector._compute.lorentz.is_timelike module

```
Lorentz.is_timelike(self, tolerance=...)
```

vector._compute.lorentz.is_timelike.**dispatch**(*tolerance: Any*, *v: Any*) → Any

vector._compute.lorentz.is_timelike.**make_function**(*azimuthal*, *longitudinal*, *temporal*)

vector._compute.lorentz.isclose module

```
Lorentz.isclose(self, rtol=..., atol=..., equal_nan=...)
```

vector._compute.lorentz.isclose.**dispatch**(*rtol: Any*, *atol: Any*, *equal_nan: Any*, *v1: Any*, *v2: Any*) → Any

vector._compute.lorentz.isclose.**make_conversion**(*azimuthal1*, *longitudinal1*, *temporal1*, *azimuthal2*, *longitudinal2*, *temporal2*)

vector._compute.lorentz.Mt module

```
@property
Lorentz.Mt(self)
```

vector._compute.lorentz.Mt.**dispatch**(*v: Any*) → Any

vector._compute.lorentz.Mt.**rhophi_eta_t**(*lib*, *rho*, *phi*, *eta*, *t*)

vector._compute.lorentz.Mt.**rhophi_eta_tau**(*lib*, *rho*, *phi*, *eta*, *tau*)

vector._compute.lorentz.Mt.**rhophi_theta_t**(*lib*, *rho*, *phi*, *theta*, *t*)

vector._compute.lorentz.Mt.**rhophi_theta_tau**(*lib*, *rho*, *phi*, *theta*, *tau*)

vector._compute.lorentz.Mt.**rhophi_z_t**(*lib*, *rho*, *phi*, *z*, *t*)

vector._compute.lorentz.Mt.**rhophi_z_tau**(*lib*, *rho*, *phi*, *z*, *tau*)

vector._compute.lorentz.Mt.**xy_eta_t**(*lib*, *x*, *y*, *eta*, *t*)

vector._compute.lorentz.Mt.**xy_eta_tau**(*lib*, *x*, *y*, *eta*, *tau*)

vector._compute.lorentz.Mt.**xy_theta_t**(*lib*, *x*, *y*, *theta*, *t*)

vector._compute.lorentz.Mt.**xy_theta_tau**(*lib*, *x*, *y*, *theta*, *tau*)

vector._compute.lorentz.Mt.**xy_z_t**(*lib*, *x*, *y*, *z*, *t*)

vector._compute.lorentz.Mt.**xy_z_tau**(*lib*, *x*, *y*, *z*, *tau*)

vector._compute.lorentz.Mt2 module

```
@property
Lorentz.Mt2(self)
```

vector._compute.lorentz.Mt2.**dispatch**(*v: Any*) → Any

vector._compute.lorentz.Mt2.**rhophi_eta_t**(*lib*, *rho*, *phi*, *eta*, *t*)

vector._compute.lorentz.Mt2.**rhophi_eta_tau**(*lib*, *rho*, *phi*, *eta*, *tau*)

vector._compute.lorentz.Mt2.**rhophi_theta_t**(*lib*, *rho*, *phi*, *theta*, *t*)

vector._compute.lorentz.Mt2.**rhophi_theta_tau**(*lib*, *rho*, *phi*, *theta*, *tau*)

vector._compute.lorentz.Mt2.**rhophi_z_t**(*lib*, *rho*, *phi*, *z*, *t*)

vector._compute.lorentz.Mt2.**rhophi_z_tau**(*lib*, *rho*, *phi*, *z*, *tau*)

vector._compute.lorentz.Mt2.**xy_eta_t**(*lib*, *x*, *y*, *eta*, *t*)

vector._compute.lorentz.Mt2.**xy_eta_tau**(*lib*, *x*, *y*, *eta*, *tau*)

vector._compute.lorentz.Mt2.**xy_theta_t**(*lib*, *x*, *y*, *theta*, *t*)

vector._compute.lorentz.Mt2.**xy_theta_tau**(*lib*, *x*, *y*, *theta*, *tau*)

vector._compute.lorentz.Mt2.**xy_z_t**(*lib*, *x*, *y*, *z*, *t*)

vector._compute.lorentz.Mt2.**xy_z_tau**(*lib*, *x*, *y*, *z*, *tau*)

vector._compute.lorentz.not_equal module

```
Lorentz.not_equal(self, other)
```

vector._compute.lorentz.not_equal.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.lorentz.not_equal.**make_conversion**(*azimuthal1*, *longitudinal1*, *temporal1*, *azimuthal2*, *longitudinal2*, *temporal2*)

vector._compute.lorentz.rapidity module

```
@property
Lorentz.rapidity(self)
```

vector._compute.lorentz.rapidity.**dispatch**(*v: Any*) → Any

vector._compute.lorentz.rapidity.**rhophi_eta_t**(*lib*, *rho*, *phi*, *eta*, *t*)

vector._compute.lorentz.rapidity.**rhophi_eta_tau**(*lib*, *rho*, *phi*, *eta*, *tau*)

vector._compute.lorentz.rapidity.**rhophi_theta_t**(*lib*, *rho*, *phi*, *theta*, *t*)

vector._compute.lorentz.rapidity.**rhophi_theta_tau**(*lib*, *rho*, *phi*, *theta*, *tau*)

vector._compute.lorentz.rapidity.**rhophi_z_t**(*lib*, *rho*, *phi*, *z*, *t*)

vector._compute.lorentz.rapidity.**rhophi_z_tau**(*lib*, *rho*, *phi*, *z*, *tau*)

vector._compute.lorentz.rapidity.**xy_eta_t**(*lib*, *x*, *y*, *eta*, *t*)

vector._compute.lorentz.rapidity.**xy_eta_tau**(*lib*, *x*, *y*, *eta*, *tau*)

vector._compute.lorentz.rapidity.**xy_theta_t**(*lib*, *x*, *y*, *theta*, *t*)

vector._compute.lorentz.rapidity.**xy_theta_tau**(*lib*, *x*, *y*, *theta*, *tau*)

vector._compute.lorentz.rapidity.**xy_z_t**(*lib*, *x*, *y*, *z*, *t*)

vector._compute.lorentz.rapidity.**xy_z_tau**(*lib*, *x*, *y*, *z*, *tau*)

vector._compute.lorentz.scale module

```
Lorentz.scale(self, factor)
```

vector._compute.lorentz.scale.**dispatch**(*factor: Any*, *v: Any*) → Any

vector._compute.lorentz.scale.**rhophi_eta_t**(*lib*, *factor*, *rho*, *phi*, *eta*, *t*)

vector._compute.lorentz.scale.**rhophi_eta_tau**(*lib*, *factor*, *rho*, *phi*, *eta*, *tau*)

vector._compute.lorentz.scale.**rhophi_theta_t**(*lib*, *factor*, *rho*, *phi*, *theta*, *t*)

vector._compute.lorentz.scale.**rhophi_theta_tau**(*lib*, *factor*, *rho*, *phi*, *theta*, *tau*)

vector._compute.lorentz.scale.**rhophi_z_t**(*lib*, *factor*, *rho*, *phi*, *z*, *t*)

vector._compute.lorentz.scale.**rhophi_z_tau**(*lib*, *factor*, *rho*, *phi*, *z*, *tau*)

vector._compute.lorentz.scale.**xy_eta_t**(*lib*, *factor*, *x*, *y*, *eta*, *t*)

vector._compute.lorentz.scale.**xy_eta_tau**(*lib*, *factor*, *x*, *y*, *eta*, *tau*)

vector._compute.lorentz.scale.**xy_theta_t**(*lib*, *factor*, *x*, *y*, *theta*, *t*)

vector._compute.lorentz.scale.**xy_theta_tau**(*lib*, *factor*, *x*, *y*, *theta*, *tau*)

vector._compute.lorentz.scale.**xy_z_t**(*lib*, *factor*, *x*, *y*, *z*, *t*)

vector._compute.lorentz.scale.**xy_z_tau**(*lib*, *factor*, *x*, *y*, *z*, *tau*)

vector._compute.lorentz.subtract module

```
Lorentz.subtract(self, other)
```

vector._compute.lorentz.subtract.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.lorentz.subtract.**make_conversion**(*azimuthal1*, *longitudinal1*, *temporal1*, *azimuthal2*, *longitudinal2*, *temporal2*)

vector._compute.lorentz.t module

```
@property
Lorentz.t(self)
```

vector._compute.lorentz.t.**dispatch**(*v: Any*) → Any

vector._compute.lorentz.t.**rhophi_eta_t**(*lib*, *rho*, *phi*, *eta*, *t*)

vector._compute.lorentz.t.**rhophi_eta_tau**(*lib*, *rho*, *phi*, *eta*, *tau*)

vector._compute.lorentz.t.**rhophi_theta_t**(*lib*, *rho*, *phi*, *theta*, *t*)

vector._compute.lorentz.t.**rhophi_theta_tau**(*lib*, *rho*, *phi*, *theta*, *tau*)

vector._compute.lorentz.t.**rhophi_z_t**(*lib*, *rho*, *phi*, *z*, *t*)

vector._compute.lorentz.t.**rhophi_z_tau**(*lib*, *rho*, *phi*, *z*, *tau*)

vector._compute.lorentz.t.**xy_eta_t**(*lib*, *x*, *y*, *eta*, *t*)

vector._compute.lorentz.t.**xy_eta_tau**(*lib*, *x*, *y*, *eta*, *tau*)

vector._compute.lorentz.t.**xy_theta_t**(*lib*, *x*, *y*, *theta*, *t*)

vector._compute.lorentz.t.**xy_theta_tau**(*lib*, *x*, *y*, *theta*, *tau*)

vector._compute.lorentz.t.**xy_z_t**(*lib*, *x*, *y*, *z*, *t*)

vector._compute.lorentz.t.**xy_z_tau**(*lib*, *x*, *y*, *z*, *tau*)

vector._compute.lorentz.t2 module

```
@property
Lorentz.t2(self)
```

vector._compute.lorentz.t2.**dispatch**(*v: Any*) → Any

vector._compute.lorentz.t2.**rhophi_eta_t**(*lib*, *rho*, *phi*, *eta*, *t*)

vector._compute.lorentz.t2.**rhophi_eta_tau**(*lib*, *rho*, *phi*, *eta*, *tau*)

vector._compute.lorentz.t2.**rhophi_theta_t**(*lib*, *rho*, *phi*, *theta*, *t*)

vector._compute.lorentz.t2.**rhophi_theta_tau**(*lib*, *rho*, *phi*, *theta*, *tau*)

vector._compute.lorentz.t2.**rhophi_z_t**(*lib*, *rho*, *phi*, *z*, *t*)

vector._compute.lorentz.t2.**rhophi_z_tau**(*lib*, *rho*, *phi*, *z*, *tau*)

vector._compute.lorentz.t2.**xy_eta_t**(*lib*, *x*, *y*, *eta*, *t*)

vector._compute.lorentz.t2.**xy_eta_tau**(*lib*, *x*, *y*, *eta*, *tau*)

vector._compute.lorentz.t2.**xy_theta_t**(*lib*, *x*, *y*, *theta*, *t*)

vector._compute.lorentz.t2.**xy_theta_tau**(*lib*, *x*, *y*, *theta*, *tau*)

vector._compute.lorentz.t2.**xy_z_t**(*lib*, *x*, *y*, *z*, *t*)

vector._compute.lorentz.t2.**xy_z_tau**(*lib*, *x*, *y*, *z*, *tau*)

vector._compute.lorentz.tau module

```
@property
Lorentz.tau(self)
```

vector._compute.lorentz.tau.**dispatch**(*v: Any*) → Any

vector._compute.lorentz.tau.**rhophi_eta_t**(*lib*, *rho*, *phi*, *eta*, *t*)

vector._compute.lorentz.tau.**rhophi_eta_tau**(*lib*, *rho*, *phi*, *eta*, *tau*)

vector._compute.lorentz.tau.**rhophi_theta_t**(*lib*, *rho*, *phi*, *theta*, *t*)

vector._compute.lorentz.tau.**rhophi_theta_tau**(*lib*, *rho*, *phi*, *theta*, *tau*)

vector._compute.lorentz.tau.**rhophi_z_t**(*lib*, *rho*, *phi*, *z*, *t*)

vector._compute.lorentz.tau.**rhophi_z_tau**(*lib*, *rho*, *phi*, *z*, *tau*)

vector._compute.lorentz.tau.**xy_eta_t**(*lib*, *x*, *y*, *eta*, *t*)

vector._compute.lorentz.tau.**xy_eta_tau**(*lib*, *x*, *y*, *eta*, *tau*)

vector._compute.lorentz.tau.**xy_theta_t**(*lib*, *x*, *y*, *theta*, *t*)

vector._compute.lorentz.tau.**xy_theta_tau**(*lib*, *x*, *y*, *theta*, *tau*)

vector._compute.lorentz.tau.**xy_z_t**(*lib*, *x*, *y*, *z*, *t*)

vector._compute.lorentz.tau.**xy_z_tau**(*lib*, *x*, *y*, *z*, *tau*)

vector._compute.lorentz.tau2 module

```
@property
Lorentz.tau2(self)
```

vector._compute.lorentz.tau2.**dispatch**(*v: Any*) → Any

vector._compute.lorentz.tau2.**rhophi_eta_t**(*lib*, *rho*, *phi*, *eta*, *t*)

vector._compute.lorentz.tau2.**rhophi_eta_tau**(*lib*, *rho*, *phi*, *eta*, *tau*)

vector._compute.lorentz.tau2.**rhophi_theta_t**(*lib*, *rho*, *phi*, *theta*, *t*)

vector._compute.lorentz.tau2.**rhophi_theta_tau**(*lib*, *rho*, *phi*, *theta*, *tau*)

vector._compute.lorentz.tau2.**rhophi_z_t**(*lib*, *rho*, *phi*, *z*, *t*)

vector._compute.lorentz.tau2.**rhophi_z_tau**(*lib*, *rho*, *phi*, *z*, *tau*)

vector._compute.lorentz.tau2.**xy_eta_t**(*lib*, *x*, *y*, *eta*, *t*)

vector._compute.lorentz.tau2.**xy_eta_tau**(*lib*, *x*, *y*, *eta*, *tau*)

vector._compute.lorentz.tau2.**xy_theta_t**(*lib*, *x*, *y*, *theta*, *t*)

vector._compute.lorentz.tau2.**xy_theta_tau**(*lib*, *x*, *y*, *theta*, *tau*)

vector._compute.lorentz.tau2.**xy_z_t**(*lib*, *x*, *y*, *z*, *t*)

vector._compute.lorentz.tau2.**xy_z_tau**(*lib*, *x*, *y*, *z*, *tau*)

vector._compute.lorentz.to_beta3 module

```
Lorentz.to_beta3(self)
```

vector._compute.lorentz.to_beta3.**dispatch**(*v: Any*) → Any

vector._compute.lorentz.to_beta3.**rhophi_eta_t**(*lib*, *rho*, *phi*, *eta*, *t*)

vector._compute.lorentz.to_beta3.**rhophi_eta_tau**(*lib*, *rho*, *phi*, *eta*, *tau*)

vector._compute.lorentz.to_beta3.**rhophi_theta_t**(*lib*, *rho*, *phi*, *theta*, *t*)

vector._compute.lorentz.to_beta3.**rhophi_theta_tau**(*lib*, *rho*, *phi*, *theta*, *tau*)

vector._compute.lorentz.to_beta3.**rhophi_z_t**(*lib*, *rho*, *phi*, *z*, *t*)

vector._compute.lorentz.to_beta3.**rhophi_z_tau**(*lib*, *rho*, *phi*, *z*, *tau*)

vector._compute.lorentz.to_beta3.**xy_eta_t**(*lib*, *x*, *y*, *eta*, *t*)

vector._compute.lorentz.to_beta3.**xy_eta_tau**(*lib*, *x*, *y*, *eta*, *tau*)

vector._compute.lorentz.to_beta3.**xy_theta_t**(*lib*, *x*, *y*, *theta*, *t*)

vector._compute.lorentz.to_beta3.**xy_theta_tau**(*lib*, *x*, *y*, *theta*, *tau*)

vector._compute.lorentz.to_beta3.**xy_z_t**(*lib*, *x*, *y*, *z*, *t*)

vector._compute.lorentz.to_beta3.**xy_z_tau**(*lib*, *x*, *y*, *z*, *tau*)

vector._compute.lorentz.transform4D module

```
Lorentz.transform4D(self, obj)
```

where `obj` has `obj["xx"]`, `obj["xy"]`, etc.

vector._compute.lorentz.transform4D.**cartesian_t**(*lib*, *xx*, *xy*, *xz*, *xt*, *yx*, *yy*, *yz*, *yt*, *zx*, *zy*, *zz*, *zt*, *tx*, *ty*, *tz*, *tt*, *x*, *y*, *z*, *t*)

vector._compute.lorentz.transform4D.**cartesian_tau**(*lib*, *xx*, *xy*, *xz*, *xt*, *yx*, *yy*, *yz*, *yt*, *zx*, *zy*, *zz*, *zt*, *x*, *y*, *z*, *tau*)

vector._compute.lorentz.transform4D.**dispatch**(*obj: Any*, *v: Any*) → Any

vector._compute.lorentz.transform4D.**make_conversion**(*azimuthal*, *longitudinal*, *temporal*)

vector._compute.lorentz.unit module

```
Lorentz.unit(self)
```

vector._compute.lorentz.unit.**dispatch**(*v: Any*) → Any

vector._compute.lorentz.unit.**rhophi_eta_t**(*lib*, *rho*, *phi*, *eta*, *t*)

vector._compute.lorentz.unit.**rhophi_eta_tau**(*lib*, *rho*, *phi*, *eta*, *tau*)

vector._compute.lorentz.unit.**rhophi_theta_t**(*lib*, *rho*, *phi*, *theta*, *t*)

vector._compute.lorentz.unit.**rhophi_theta_tau**(*lib*, *rho*, *phi*, *theta*, *tau*)

vector._compute.lorentz.unit.**rhophi_z_t**(*lib*, *rho*, *phi*, *z*, *t*)

vector._compute.lorentz.unit.**rhophi_z_tau**(*lib*, *rho*, *phi*, *z*, *tau*)

vector._compute.lorentz.unit.**xy_eta_t**(*lib*, *x*, *y*, *eta*, *t*)

vector._compute.lorentz.unit.**xy_eta_tau**(*lib*, *x*, *y*, *eta*, *tau*)

vector._compute.lorentz.unit.**xy_theta_t**(*lib*, *x*, *y*, *theta*, *t*)

vector._compute.lorentz.unit.**xy_theta_tau**(*lib*, *x*, *y*, *theta*, *tau*)

vector._compute.lorentz.unit.**xy_z_t**(*lib*, *x*, *y*, *z*, *t*)

vector._compute.lorentz.unit.**xy_z_tau**(*lib*, *x*, *y*, *z*, *tau*)

vector._compute.planar package

Compute functions for planar vectors, which is to say 2D, 3D, and 4D.

Each function is a module with variants for each coordinate system (or combination of coordinate systems) as functions within the module.

Each module has a `dispatch_map` (dict) that maps coordinate types to the appropriate function and its return type(s), and a `dispatch` (function) uses this information to call the right function and return the right type.

The compute functions themselves are restricted to a minimum of Python features: no statements other than assignments and one return, no assumptions about data types. In particular, if statements and loops are not allowed. The tests/test_compute_features.py suite ensures that these rules are followed (though that set of allowed features can be expanded if it doesn't prevent the addition of new backends).

Submodules

vector._compute.planar.add module

```
Planar.add(self, other)
```

vector._compute.planar.add.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.planar.add.**rectify**(*lib*, *phi*)

vector._compute.planar.add.**rhophi_rhophi**(*lib*, *rho1*, *phi1*, *rho2*, *phi2*)

vector._compute.planar.add.**rhophi_xy**(*lib*, *rho1*, *phi1*, *x2*, *y2*)

vector._compute.planar.add.**xy_rhophi**(*lib*, *x1*, *y1*, *rho2*, *phi2*)

vector._compute.planar.add.**xy_xy**(*lib*, *x1*, *y1*, *x2*, *y2*)

vector._compute.planar.deltaphi module

```
Planar.deltaphi(self, other)
```

vector._compute.planar.deltaphi.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.planar.deltaphi.**rectify**(*lib*, *phi*)

vector._compute.planar.deltaphi.**rhophi_rhophi**(*lib*, *rho1*, *phi1*, *rho2*, *phi2*)

vector._compute.planar.deltaphi.**rhophi_xy**(*lib*, *rho1*, *phi1*, *x2*, *y2*)

vector._compute.planar.deltaphi.**xy_rhophi**(*lib*, *x1*, *y1*, *rho2*, *phi2*)

vector._compute.planar.deltaphi.**xy_xy**(*lib*, *x1*, *y1*, *x2*, *y2*)

vector._compute.planar.dot module

```
Planar.dot(self, other)
```

vector._compute.planar.dot.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.planar.dot.**rhophi_rhophi**(*lib*, *rho1*, *phi1*, *rho2*, *phi2*)

vector._compute.planar.dot.**rhophi_xy**(*lib*, *rho1*, *phi1*, *x2*, *y2*)

vector._compute.planar.dot.**xy_rhophi**(*lib*, *x1*, *y1*, *rho2*, *phi2*)

vector._compute.planar.dot.**xy_xy**(*lib*, *x1*, *y1*, *x2*, *y2*)

vector._compute.planar.equal module

```
Planar.equal(self, other)
```

vector._compute.planar.equal.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.planar.equal.**rhophi_rhophi**(*lib*, *rho1*, *phi1*, *rho2*, *phi2*)

vector._compute.planar.equal.**rhophi_xy**(*lib*, *rho1*, *phi1*, *x2*, *y2*)

vector._compute.planar.equal.**xy_rhophi**(*lib*, *x1*, *y1*, *rho2*, *phi2*)

vector._compute.planar.equal.**xy_xy**(*lib*, *x1*, *y1*, *x2*, *y2*)

vector._compute.planar.is_antiparallel module

```
Planar.is_antiparallel(self, other, tolerance=...)
```

vector._compute.planar.is_antiparallel.**dispatch**(*tolerance: Any*, *v1: Any*, *v2: Any*) → Any

vector._compute.planar.is_antiparallel.**make_function**(*azimuthal1*, *azimuthal2*)

vector._compute.planar.is_parallel module

```
Planar.is_parallel(self, other, tolerance=...)
```

vector._compute.planar.is_parallel.**dispatch**(*tolerance: Any*, *v1: Any*, *v2: Any*) → Any

vector._compute.planar.is_parallel.**make_function**(*azimuthal1*, *azimuthal2*)

vector._compute.planar.is_perpendicular module

```
Planar.is_perpendicular(self, other, tolerance=...)
```

vector._compute.planar.is_perpendicular.**dispatch**(*tolerance: Any*, *v1: Any*, *v2: Any*) → Any

vector._compute.planar.is_perpendicular.**make_function**(*azimuthal1*, *azimuthal2*)

vector._compute.planar.isclose module

```
Planar.isclose(self, other, rtol=..., atol=..., equal_nan=...)
```

vector._compute.planar.isclose.**dispatch**(*rtol: Any*, *atol: Any*, *equal_nan: Any*, *v1: Any*, *v2: Any*) → Any

vector._compute.planar.isclose.**rhophi_rhophi**(*lib*, *rtol*, *atol*, *equal_nan*, *rho1*, *phi1*, *rho2*, *phi2*)

vector._compute.planar.isclose.**rhophi_xy**(*lib*, *rtol*, *atol*, *equal_nan*, *rho1*, *phi1*, *x2*, *y2*)

vector._compute.planar.isclose.**xy_rhophi**(*lib*, *rtol*, *atol*, *equal_nan*, *x1*, *y1*, *rho2*, *phi2*)

vector._compute.planar.isclose.**xy_xy**(*lib*, *rtol*, *atol*, *equal_nan*, *x1*, *y1*, *x2*, *y2*)

vector._compute.planar.not_equal module

```
Planar.not_equal(self, other)
```

vector._compute.planar.not_equal.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.planar.not_equal.**rhophi_rhophi**(*lib*, *rho1*, *phi1*, *rho2*, *phi2*)

vector._compute.planar.not_equal.**rhophi_xy**(*lib*, *rho1*, *phi1*, *x2*, *y2*)

vector._compute.planar.not_equal.**xy_rhophi**(*lib*, *x1*, *y1*, *rho2*, *phi2*)

vector._compute.planar.not_equal.**xy_xy**(*lib*, *x1*, *y1*, *x2*, *y2*)

vector._compute.planar.phi module

```
@property
Planar.phi(self)
```

vector._compute.planar.phi.**dispatch**(*v: Any*) → Any

vector._compute.planar.phi.**rhophi**(*lib*, *rho*, *phi*)

vector._compute.planar.phi.**xy**(*lib*, *x*, *y*)

vector._compute.planar.rho module

```
@property
Planar.rho(self)
```

vector._compute.planar.rho.**dispatch**(*v: Any*) → Any

vector._compute.planar.rho.**rhophi**(*lib*, *rho*, *phi*)

vector._compute.planar.rho.**xy**(*lib*, *x*, *y*)

vector._compute.planar.rho2 module

```
@property
Planar.rho2(self)
```

vector._compute.planar.rho2.**dispatch**(*v: Any*) → Any

vector._compute.planar.rho2.**rhophi**(*lib*, *rho*, *phi*)

vector._compute.planar.rho2.**xy**(*lib*, *x*, *y*)

vector._compute.planar.rotateZ module

```
Planar.rotateZ(self, angle)
```

vector._compute.planar.rotateZ.**dispatch**(*angle: Any*, *v: Any*) → Any

vector._compute.planar.rotateZ.**rectify**(*lib*, *phi*)

vector._compute.planar.rotateZ.**rhophi**(*lib*, *angle*, *rho*, *phi*)

vector._compute.planar.rotateZ.**xy**(*lib*, *angle*, *x*, *y*)

vector._compute.planar.scale module

```
Planar.scale(self, factor)
```

vector._compute.planar.scale.**dispatch**(*factor: Any*, *v: Any*) → Any

vector._compute.planar.scale.**rectify**(*lib*, *phi*)

vector._compute.planar.scale.**rhophi**(*lib*, *factor*, *rho*, *phi*)

vector._compute.planar.scale.**xy**(*lib*, *factor*, *x*, *y*)

vector._compute.planar.subtract module

```
Planar.subtract(self, other)
```

vector._compute.planar.subtract.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.planar.subtract.**rectify**(*lib*, *phi*)

vector._compute.planar.subtract.**rhophi_rhophi**(*lib*, *rho1*, *phi1*, *rho2*, *phi2*)

vector._compute.planar.subtract.**rhophi_xy**(*lib*, *rho1*, *phi1*, *x2*, *y2*)

vector._compute.planar.subtract.**xy_rhophi**(*lib*, *x1*, *y1*, *rho2*, *phi2*)

vector._compute.planar.subtract.**xy_xy**(*lib*, *x1*, *y1*, *x2*, *y2*)

vector._compute.planar.transform2D module

```
Planar.transform2D(self, obj)
```

where `obj` has `obj["xx"]`, `obj["xy"]`, etc.

vector._compute.planar.transform2D.**cartesian**(*lib*, *xx*, *xy*, *yx*, *yy*, *x*, *y*)

vector._compute.planar.transform2D.**dispatch**(*obj: Any*, *v: Any*) → Any

vector._compute.planar.transform2D.**rhophi**(*lib*, *xx*, *xy*, *yx*, *yy*, *rho*, *phi*)

vector._compute.planar.unit module

```
Planar.unit(self)
```

vector._compute.planar.unit.**dispatch**(*v: Any*) → Any

vector._compute.planar.unit.**rhophi**(*lib*, *rho*, *phi*)

vector._compute.planar.unit.**xy**(*lib*, *x*, *y*)

vector._compute.planar.x module

```
@property
Planar.x(self)
```

vector._compute.planar.x.**dispatch**(*v: Any*) → Any

vector._compute.planar.x.**rhophi**(*lib*, *rho*, *phi*)

vector._compute.planar.x.**xy**(*lib*, *x*, *y*)

vector._compute.planar.y module

```
@property
Planar.y(self)
```

vector._compute.planar.y.**dispatch**(*v: Any*) → Any

vector._compute.planar.y.**rhophi**(*lib*, *rho*, *phi*)

vector._compute.planar.y.**xy**(*lib*, *x*, *y*)

vector._compute.spatial package

Compute functions for spatial vectors, which is to say 3D and 4D.

Each function is a module with variants for each coordinate system (or combination of coordinate systems) as functions within the module.

Each module has a `dispatch_map` (dict) that maps coordinate types to the appropriate function and its return type(s), and a `dispatch` (function) uses this information to call the right function and return the right type.

The compute functions themselves are restricted to a minimum of Python features: no statements other than assignments and one return, no assumptions about data types. In particular, if statements and loops are not allowed. The tests/test_compute_features.py suite ensures that these rules are followed (though that set of allowed features can be expanded if it doesn't prevent the addition of new backends).

Submodules

vector._compute.spatial.add module

```
Spatial.add(self, other)
```

vector._compute.spatial.add.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.spatial.add.**rhophi_eta_rhophi_eta**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.add.**rhophi_eta_rhophi_theta**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.add.**rhophi_eta_rhophi_z**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.add.**rhophi_eta_xy_eta**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.add.**rhophi_eta_xy_theta**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.add.**rhophi_eta_xy_z**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.add.**rhophi_theta_rhophi_eta**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.add.**rhophi_theta_rhophi_theta**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.add.**rhophi_theta_rhophi_z**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.add.**rhophi_theta_xy_eta**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.add.**rhophi_theta_xy_theta**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.add.**rhophi_theta_xy_z**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.add.**rhophi_z_rhophi_eta**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.add.**rhophi_z_rhophi_theta**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.add.**rhophi_z_rhophi_z**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.add.**rhophi_z_xy_eta**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.add.**rhophi_z_xy_theta**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.add.**rhophi_z_xy_z**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *z2*)

vector._compute.spatial.add.**xy_eta_rhophi_eta**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.add.**xy_eta_rhophi_theta**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.add.**xy_eta_rhophi_z**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.add.**xy_eta_xy_eta**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.add.**xy_eta_xy_theta**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.add.**xy_eta_xy_z**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.add.**xy_theta_rhophi_eta**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.add.**xy_theta_rhophi_theta**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.add.**xy_theta_rhophi_z**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.add.**xy_theta_xy_eta**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.add.**xy_theta_xy_theta**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.add.**xy_theta_xy_z**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.add.**xy_z_rhophi_eta**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.add.**xy_z_rhophi_theta**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.add.**xy_z_rhophi_z**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.add.**xy_z_xy_eta**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.add.**xy_z_xy_theta**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.add.**xy_z_xy_z**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *z2*)

vector._compute.spatial.costheta module

```
@property
Spatial.costheta(self)
```

vector._compute.spatial.costheta.**dispatch**(*v: Any*) → Any

vector._compute.spatial.costheta.**rhophi_eta**(*lib*, *rho*, *phi*, *eta*)

vector._compute.spatial.costheta.**rhophi_theta**(*lib*, *rho*, *phi*, *theta*)

vector._compute.spatial.costheta.**rhophi_z**(*lib*, *rho*, *phi*, *z*)

vector._compute.spatial.costheta.**xy_eta**(*lib*, *x*, *y*, *eta*)

vector._compute.spatial.costheta.**xy_theta**(*lib*, *x*, *y*, *theta*)

vector._compute.spatial.costheta.**xy_z**(*lib*, *x*, *y*, *z*)

vector._compute.spatial.cottheta module

```
@property
Spatial.cottheta(self)
```

vector._compute.spatial.cottheta.**dispatch**(*v: Any*) → Any

vector._compute.spatial.cottheta.**rhophi_eta**(*lib*, *rho*, *phi*, *eta*)

vector._compute.spatial.cottheta.**rhophi_theta**(*lib*, *rho*, *phi*, *theta*)

vector._compute.spatial.cottheta.**rhophi_z**(*lib*, *rho*, *phi*, *z*)

vector._compute.spatial.cottheta.**xy_eta**(*lib*, *x*, *y*, *eta*)

vector._compute.spatial.cottheta.**xy_theta**(*lib*, *x*, *y*, *theta*)

vector._compute.spatial.cottheta.**xy_z**(*lib*, *x*, *y*, *z*)

vector._compute.spatial.cross module

```
Spatial.cross(self, other)
```

Note that this returns a 3D vector even for 4D inputs. The `None` at the end of the return signature indicates termination.

vector._compute.spatial.cross.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.spatial.cross.**make_conversion**(*azimuthal1*, *longitudinal1*, *azimuthal2*, *longitudinal2*)

vector._compute.spatial.cross.**xy_z_xy_z**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *z2*)

vector._compute.spatial.deltaangle module

```
Spatial.deltaangle(self, other)
```

vector._compute.spatial.deltaangle.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.spatial.deltaangle.**rhophi_eta_rhophi_eta**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.deltaangle.**rhophi_eta_rhophi_theta**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.deltaangle.**rhophi_eta_rhophi_z**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.deltaangle.**rhophi_eta_xy_eta**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.deltaangle.**rhophi_eta_xy_theta**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.deltaangle.**rhophi_eta_xy_z**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.deltaangle.**rhophi_theta_rhophi_eta**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.deltaangle.**rhophi_theta_rhophi_theta**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.deltaangle.**rhophi_theta_rhophi_z**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.deltaangle.**rhophi_theta_xy_eta**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.deltaangle.**rhophi_theta_xy_theta**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.deltaangle.**rhophi_theta_xy_z**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.deltaangle.**rhophi_z_rhophi_eta**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.deltaangle.**rhophi_z_rhophi_theta**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.deltaangle.**rhophi_z_rhophi_z**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.deltaangle.**rhophi_z_xy_eta**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.deltaangle.**rhophi_z_xy_theta**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.deltaangle.**rhophi_z_xy_z**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *z2*)

vector._compute.spatial.deltaangle.**xy_eta_rhophi_eta**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.deltaangle.**xy_eta_rhophi_theta**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.deltaangle.**xy_eta_rhophi_z**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.deltaangle.**xy_eta_xy_eta**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.deltaangle.**xy_eta_xy_theta**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.deltaangle.**xy_eta_xy_z**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.deltaangle.**xy_theta_rhophi_eta**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.deltaangle.**xy_theta_rhophi_theta**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.deltaangle.**xy_theta_rhophi_z**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.deltaangle.**xy_theta_xy_eta**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.deltaangle.**xy_theta_xy_theta**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.deltaangle.**xy_theta_xy_z**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.deltaangle.**xy_z_rhophi_eta**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.deltaangle.**xy_z_rhophi_theta**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.deltaangle.**xy_z_rhophi_z**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.deltaangle.**xy_z_xy_eta**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.deltaangle.**xy_z_xy_theta**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.deltaangle.**xy_z_xy_z**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *z2*)

vector._compute.spatial.deltaeta module

```
Spatial.deltaeta(self, other)
```

vector._compute.spatial.deltaeta.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.spatial.deltaeta.**rhophi_eta_rhophi_eta**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.deltaeta.**rhophi_eta_rhophi_theta**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.deltaeta.**rhophi_eta_rhophi_z**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.deltaeta.**rhophi_eta_xy_eta**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.deltaeta.**rhophi_eta_xy_theta**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.deltaeta.**rhophi_eta_xy_z**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.deltaeta.**rhophi_theta_rhophi_eta**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.deltaeta.**rhophi_theta_rhophi_theta**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.deltaeta.**rhophi_theta_rhophi_z**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.deltaeta.**rhophi_theta_xy_eta**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.deltaeta.**rhophi_theta_xy_theta**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.deltaeta.**rhophi_theta_xy_z**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.deltaeta.**rhophi_z_rhophi_eta**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.deltaeta.**rhophi_z_rhophi_theta**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.deltaeta.**rhophi_z_rhophi_z**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.deltaeta.**rhophi_z_xy_eta**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.deltaeta.**rhophi_z_xy_theta**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.deltaeta.**rhophi_z_xy_z**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *z2*)

vector._compute.spatial.deltaeta.**xy_eta_rhophi_eta**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.deltaeta.**xy_eta_rhophi_theta**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.deltaeta.**xy_eta_rhophi_z**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.deltaeta.**xy_eta_xy_eta**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.deltaeta.**xy_eta_xy_theta**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.deltaeta.**xy_eta_xy_z**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.deltaeta.**xy_theta_rhophi_eta**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.deltaeta.**xy_theta_rhophi_theta**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.deltaeta.**xy_theta_rhophi_z**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.deltaeta.**xy_theta_xy_eta**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.deltaeta.**xy_theta_xy_theta**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.deltaeta.**xy_theta_xy_z**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.deltaeta.**xy_z_rhophi_eta**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.deltaeta.**xy_z_rhophi_theta**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.deltaeta.**xy_z_rhophi_z**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.deltaeta.**xy_z_xy_eta**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.deltaeta.**xy_z_xy_theta**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.deltaeta.**xy_z_xy_z**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *z2*)

vector._compute.spatial.deltaR module

```
Spatial.deltaR(self, other)
```

vector._compute.spatial.deltaR.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.spatial.deltaR.**rhophi_eta_rhophi_eta**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.deltaR.**rhophi_eta_rhophi_theta**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.deltaR.**rhophi_eta_rhophi_z**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.deltaR.**rhophi_eta_xy_eta**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.deltaR.**rhophi_eta_xy_theta**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.deltaR.**rhophi_eta_xy_z**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.deltaR.**rhophi_theta_rhophi_eta**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.deltaR.**rhophi_theta_rhophi_theta**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.deltaR.**rhophi_theta_rhophi_z**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.deltaR.**rhophi_theta_xy_eta**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.deltaR.**rhophi_theta_xy_theta**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.deltaR.**rhophi_theta_xy_z**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.deltaR.**rhophi_z_rhophi_eta**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.deltaR.**rhophi_z_rhophi_theta**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.deltaR.**rhophi_z_rhophi_z**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.deltaR.**rhophi_z_xy_eta**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.deltaR.**rhophi_z_xy_theta**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.deltaR.**rhophi_z_xy_z**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *z2*)

vector._compute.spatial.deltaR.**xy_eta_rhophi_eta**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.deltaR.**xy_eta_rhophi_theta**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.deltaR.**xy_eta_rhophi_z**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.deltaR.**xy_eta_xy_eta**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.deltaR.**xy_eta_xy_theta**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.deltaR.**xy_eta_xy_z**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.deltaR.**xy_theta_rhophi_eta**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.deltaR.**xy_theta_rhophi_theta**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.deltaR.**xy_theta_rhophi_z**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.deltaR.**xy_theta_xy_eta**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.deltaR.**xy_theta_xy_theta**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.deltaR.**xy_theta_xy_z**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.deltaR.**xy_z_rhophi_eta**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.deltaR.**xy_z_rhophi_theta**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.deltaR.**xy_z_rhophi_z**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.deltaR.**xy_z_xy_eta**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.deltaR.**xy_z_xy_theta**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.deltaR.**xy_z_xy_z**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *z2*)

vector._compute.spatial.deltaR2 module

```
Spatial.deltaR2(self, other)
```

vector._compute.spatial.deltaR2.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.spatial.deltaR2.**rhophi_eta_rhophi_eta**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.deltaR2.**rhophi_eta_rhophi_theta**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.deltaR2.**rhophi_eta_rhophi_z**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.deltaR2.**rhophi_eta_xy_eta**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.deltaR2.**rhophi_eta_xy_theta**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.deltaR2.**rhophi_eta_xy_z**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.deltaR2.**rhophi_theta_rhophi_eta**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.deltaR2.**rhophi_theta_rhophi_theta**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.deltaR2.**rhophi_theta_rhophi_z**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.deltaR2.**rhophi_theta_xy_eta**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.deltaR2.**rhophi_theta_xy_theta**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.deltaR2.**rhophi_theta_xy_z**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.deltaR2.**rhophi_z_rhophi_eta**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.deltaR2.**rhophi_z_rhophi_theta**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.deltaR2.**rhophi_z_rhophi_z**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.deltaR2.**rhophi_z_xy_eta**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.deltaR2.**rhophi_z_xy_theta**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.deltaR2.**rhophi_z_xy_z**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *z2*)

vector._compute.spatial.deltaR2.**xy_eta_rhophi_eta**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.deltaR2.**xy_eta_rhophi_theta**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.deltaR2.**xy_eta_rhophi_z**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.deltaR2.**xy_eta_xy_eta**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.deltaR2.**xy_eta_xy_theta**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.deltaR2.**xy_eta_xy_z**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.deltaR2.**xy_theta_rhophi_eta**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.deltaR2.**xy_theta_rhophi_theta**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.deltaR2.**xy_theta_rhophi_z**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.deltaR2.**xy_theta_xy_eta**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.deltaR2.**xy_theta_xy_theta**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.deltaR2.**xy_theta_xy_z**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.deltaR2.**xy_z_rhophi_eta**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.deltaR2.**xy_z_rhophi_theta**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.deltaR2.**xy_z_rhophi_z**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.deltaR2.**xy_z_xy_eta**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.deltaR2.**xy_z_xy_theta**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.deltaR2.**xy_z_xy_z**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *z2*)

vector._compute.spatial.dot module

```
Spatial.dot(self, other)
```

vector._compute.spatial.dot.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.spatial.dot.**rhophi_eta_rhophi_eta**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.dot.**rhophi_eta_rhophi_theta**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.dot.**rhophi_eta_rhophi_z**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.dot.**rhophi_eta_xy_eta**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.dot.**rhophi_eta_xy_theta**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.dot.**rhophi_eta_xy_z**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.dot.**rhophi_theta_rhophi_eta**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.dot.**rhophi_theta_rhophi_theta**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.dot.**rhophi_theta_rhophi_z**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.dot.**rhophi_theta_xy_eta**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.dot.**rhophi_theta_xy_theta**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.dot.**rhophi_theta_xy_z**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.dot.**rhophi_z_rhophi_eta**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.dot.**rhophi_z_rhophi_theta**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.dot.**rhophi_z_rhophi_z**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.dot.**rhophi_z_xy_eta**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.dot.**rhophi_z_xy_theta**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.dot.**rhophi_z_xy_z**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *z2*)

vector._compute.spatial.dot.**xy_eta_rhophi_eta**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.dot.**xy_eta_rhophi_theta**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.dot.**xy_eta_rhophi_z**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.dot.**xy_eta_xy_eta**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.dot.**xy_eta_xy_theta**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.dot.**xy_eta_xy_z**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.dot.**xy_theta_rhophi_eta**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.dot.**xy_theta_rhophi_theta**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.dot.**xy_theta_rhophi_z**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.dot.**xy_theta_xy_eta**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.dot.**xy_theta_xy_theta**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.dot.**xy_theta_xy_z**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.dot.**xy_z_rhophi_eta**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.dot.**xy_z_rhophi_theta**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.dot.**xy_z_rhophi_z**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.dot.**xy_z_xy_eta**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.dot.**xy_z_xy_theta**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.dot.**xy_z_xy_z**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *z2*)

vector._compute.spatial.equal module

```
Spatial.equal(self, other)
```

vector._compute.spatial.equal.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.spatial.equal.**rhophi_eta_rhophi_eta**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.equal.**rhophi_eta_rhophi_theta**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.equal.**rhophi_eta_rhophi_z**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.equal.**rhophi_eta_xy_eta**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.equal.**rhophi_eta_xy_theta**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.equal.**rhophi_eta_xy_z**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.equal.**rhophi_theta_rhophi_eta**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.equal.**rhophi_theta_rhophi_theta**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.equal.**rhophi_theta_rhophi_z**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.equal.**rhophi_theta_xy_eta**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.equal.**rhophi_theta_xy_theta**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.equal.**rhophi_theta_xy_z**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.equal.**rhophi_z_rhophi_eta**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.equal.**rhophi_z_rhophi_theta**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.equal.**rhophi_z_rhophi_z**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.equal.**rhophi_z_xy_eta**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.equal.**rhophi_z_xy_theta**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.equal.**rhophi_z_xy_z**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *z2*)

vector._compute.spatial.equal.**xy_eta_rhophi_eta**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.equal.**xy_eta_rhophi_theta**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.equal.**xy_eta_rhophi_z**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.equal.**xy_eta_xy_eta**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.equal.**xy_eta_xy_theta**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.equal.**xy_eta_xy_z**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.equal.**xy_theta_rhophi_eta**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.equal.**xy_theta_rhophi_theta**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.equal.**xy_theta_rhophi_z**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.equal.**xy_theta_xy_eta**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.equal.**xy_theta_xy_theta**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.equal.**xy_theta_xy_z**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.equal.**xy_z_rhophi_eta**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.equal.**xy_z_rhophi_theta**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.equal.**xy_z_rhophi_z**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.equal.**xy_z_xy_eta**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.equal.**xy_z_xy_theta**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.equal.**xy_z_xy_z**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *z2*)

vector._compute.spatial.eta module

```
@property
Spatial.eta(self)
```

vector._compute.spatial.eta.**dispatch**(*v: Any*) → Any

vector._compute.spatial.eta.**rhophi_eta**(*lib*, *rho*, *phi*, *eta*)

vector._compute.spatial.eta.**rhophi_theta**(*lib*, *rho*, *phi*, *theta*)

vector._compute.spatial.eta.**rhophi_z**(*lib*, *rho*, *phi*, *z*)

vector._compute.spatial.eta.**xy_eta**(*lib*, *x*, *y*, *eta*)

vector._compute.spatial.eta.**xy_theta**(*lib*, *x*, *y*, *theta*)

vector._compute.spatial.eta.**xy_z**(*lib*, *x*, *y*, *z*)

vector._compute.spatial.is_antiparallel module

```
Spatial.is_antiparallel(self, other, tolerance=...)
```

vector._compute.spatial.is_antiparallel.**dispatch**(*tolerance: Any*, *v1: Any*, *v2: Any*) → Any

vector._compute.spatial.is_antiparallel.**make_function**(*azimuthal1*, *longitudinal1*, *azimuthal2*, *longitudinal2*)

vector._compute.spatial.is_parallel module

```
Spatial.is_parallel(self, other, tolerance=...)
```

vector._compute.spatial.is_parallel.**dispatch**(*tolerance: Any*, *v1: Any*, *v2: Any*) → Any

vector._compute.spatial.is_parallel.**make_function**(*azimuthal1*, *longitudinal1*, *azimuthal2*, *longitudinal2*)

vector._compute.spatial.is_perpendicular module

```
Spatial.is_perpendicular(self, other, tolerance=...)
```

vector._compute.spatial.is_perpendicular.**dispatch**(*tolerance: Any*, *v1: Any*, *v2: Any*) → Any

vector._compute.spatial.is_perpendicular.**make_function**(*azimuthal1*, *longitudinal1*, *azimuthal2*, *longitudinal2*)

vector._compute.spatial.isclose module

```
Spatial.isclose(self, other, rtol=..., atol=..., equal_nan=...)
```

vector._compute.spatial.isclose.**dispatch**(*rtol: Any*, *atol: Any*, *equal_nan: Any*, *v1: Any*, *v2: Any*) → Any

vector._compute.spatial.isclose.**rhophi_eta_rhophi_eta**(*lib*, *rtol*, *atol*, *equal_nan*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.isclose.**rhophi_eta_rhophi_theta**(*lib*, *rtol*, *atol*, *equal_nan*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.isclose.**rhophi_eta_rhophi_z**(*lib*, *rtol*, *atol*, *equal_nan*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.isclose.**rhophi_eta_xy_eta**(*lib*, *rtol*, *atol*, *equal_nan*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.isclose.**rhophi_eta_xy_theta**(*lib*, *rtol*, *atol*, *equal_nan*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.isclose.**rhophi_eta_xy_z**(*lib*, *rtol*, *atol*, *equal_nan*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.isclose.**rhophi_theta_rhophi_eta**(*lib*, *rtol*, *atol*, *equal_nan*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.isclose.**rhophi_theta_rhophi_theta**(*lib*, *rtol*, *atol*, *equal_nan*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.isclose.**rhophi_theta_rhophi_z**(*lib*, *rtol*, *atol*, *equal_nan*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.isclose.**rhophi_theta_xy_eta**(*lib*, *rtol*, *atol*, *equal_nan*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.isclose.**rhophi_theta_xy_theta**(*lib*, *rtol*, *atol*, *equal_nan*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.isclose.**rhophi_theta_xy_z**(*lib*, *rtol*, *atol*, *equal_nan*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.isclose.**rhophi_z_rhophi_eta**(*lib*, *rtol*, *atol*, *equal_nan*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.isclose.**rhophi_z_rhophi_theta**(*lib*, *rtol*, *atol*, *equal_nan*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.isclose.**rhophi_z_rhophi_z**(*lib*, *rtol*, *atol*, *equal_nan*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.isclose.**rhophi_z_xy_eta**(*lib*, *rtol*, *atol*, *equal_nan*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.isclose.**rhophi_z_xy_theta**(*lib*, *rtol*, *atol*, *equal_nan*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.isclose.**rhophi_z_xy_z**(*lib*, *rtol*, *atol*, *equal_nan*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *z2*)

vector._compute.spatial.isclose.**xy_eta_rhophi_eta**(*lib*, *rtol*, *atol*, *equal_nan*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.isclose.**xy_eta_rhophi_theta**(*lib*, *rtol*, *atol*, *equal_nan*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.isclose.**xy_eta_rhophi_z**(*lib*, *rtol*, *atol*, *equal_nan*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.isclose.**xy_eta_xy_eta**(*lib*, *rtol*, *atol*, *equal_nan*, *x1*, *y1*, *eta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.isclose.**xy_eta_xy_theta**(*lib*, *rtol*, *atol*, *equal_nan*, *x1*, *y1*, *eta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.isclose.**xy_eta_xy_z**(*lib*, *rtol*, *atol*, *equal_nan*, *x1*, *y1*, *eta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.isclose.**xy_theta_rhophi_eta**(*lib*, *rtol*, *atol*, *equal_nan*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.isclose.**xy_theta_rhophi_theta**(*lib*, *rtol*, *atol*, *equal_nan*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.isclose.**xy_theta_rhophi_z**(*lib*, *rtol*, *atol*, *equal_nan*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.isclose.**xy_theta_xy_eta**(*lib*, *rtol*, *atol*, *equal_nan*, *x1*, *y1*, *theta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.isclose.**xy_theta_xy_theta**(*lib*, *rtol*, *atol*, *equal_nan*, *x1*, *y1*, *theta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.isclose.**xy_theta_xy_z**(*lib*, *rtol*, *atol*, *equal_nan*, *x1*, *y1*, *theta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.isclose.**xy_z_rhophi_eta**(*lib*, *rtol*, *atol*, *equal_nan*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.isclose.**xy_z_rhophi_theta**(*lib*, *rtol*, *atol*, *equal_nan*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.isclose.**xy_z_rhophi_z**(*lib*, *rtol*, *atol*, *equal_nan*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.isclose.**xy_z_xy_eta**(*lib*, *rtol*, *atol*, *equal_nan*, *x1*, *y1*, *z1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.isclose.**xy_z_xy_theta**(*lib*, *rtol*, *atol*, *equal_nan*, *x1*, *y1*, *z1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.isclose.**xy_z_xy_z**(*lib*, *rtol*, *atol*, *equal_nan*, *x1*, *y1*, *z1*, *x2*, *y2*, *z2*)

vector._compute.spatial.mag module

```
@property
Spatial.mag(self)
```

vector._compute.spatial.mag.**dispatch**(*v: Any*) → Any

vector._compute.spatial.mag.**rhophi_eta**(*lib*, *rho*, *phi*, *eta*)

vector._compute.spatial.mag.**rhophi_theta**(*lib*, *rho*, *phi*, *theta*)

vector._compute.spatial.mag.**rhophi_z**(*lib*, *rho*, *phi*, *z*)

vector._compute.spatial.mag.**xy_eta**(*lib*, *x*, *y*, *eta*)

vector._compute.spatial.mag.**xy_theta**(*lib*, *x*, *y*, *theta*)

vector._compute.spatial.mag.**xy_z**(*lib*, *x*, *y*, *z*)

vector._compute.spatial.mag2 module

```
@property
Spatial.mag2(self)
```

vector._compute.spatial.mag2.**dispatch**(*v: Any*) → Any

vector._compute.spatial.mag2.**rhophi_eta**(*lib*, *rho*, *phi*, *eta*)

vector._compute.spatial.mag2.**rhophi_theta**(*lib*, *rho*, *phi*, *theta*)

vector._compute.spatial.mag2.**rhophi_z**(*lib*, *rho*, *phi*, *z*)

vector._compute.spatial.mag2.**xy_eta**(*lib*, *x*, *y*, *eta*)

vector._compute.spatial.mag2.**xy_theta**(*lib*, *x*, *y*, *theta*)

vector._compute.spatial.mag2.**xy_z**(*lib*, *x*, *y*, *z*)

vector._compute.spatial.not_equal module

```
Spatial.not_equal(self, other)
```

vector._compute.spatial.not_equal.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.spatial.not_equal.**rhophi_eta_rhophi_eta**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.not_equal.**rhophi_eta_rhophi_theta**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.not_equal.**rhophi_eta_rhophi_z**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.not_equal.**rhophi_eta_xy_eta**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.not_equal.**rhophi_eta_xy_theta**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.not_equal.**rhophi_eta_xy_z**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.not_equal.**rhophi_theta_rhophi_eta**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.not_equal.**rhophi_theta_rhophi_theta**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.not_equal.**rhophi_theta_rhophi_z**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.not_equal.**rhophi_theta_xy_eta**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.not_equal.**rhophi_theta_xy_theta**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.not_equal.**rhophi_theta_xy_z**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.not_equal.**rhophi_z_rhophi_eta**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.not_equal.**rhophi_z_rhophi_theta**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.not_equal.**rhophi_z_rhophi_z**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.not_equal.**rhophi_z_xy_eta**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.not_equal.**rhophi_z_xy_theta**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.not_equal.**rhophi_z_xy_z**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *z2*)

vector._compute.spatial.not_equal.**xy_eta_rhophi_eta**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.not_equal.**xy_eta_rhophi_theta**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.not_equal.**xy_eta_rhophi_z**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.not_equal.**xy_eta_xy_eta**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.not_equal.**xy_eta_xy_theta**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.not_equal.**xy_eta_xy_z**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.not_equal.**xy_theta_rhophi_eta**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.not_equal.**xy_theta_rhophi_theta**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.not_equal.**xy_theta_rhophi_z**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.not_equal.**xy_theta_xy_eta**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.not_equal.**xy_theta_xy_theta**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.not_equal.**xy_theta_xy_z**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.not_equal.**xy_z_rhophi_eta**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.not_equal.**xy_z_rhophi_theta**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.not_equal.**xy_z_rhophi_z**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.not_equal.**xy_z_xy_eta**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.not_equal.**xy_z_xy_theta**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.not_equal.**xy_z_xy_z**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *z2*)

vector._compute.spatial.rotate_axis module

```
Spatial.rotate_axis(self, axis, angle)
```

vector._compute.spatial.rotate_axis.**cartesian**(*lib*, *angle*, *x1*, *y1*, *z1*, *x2*, *y2*, *z2*)

vector._compute.spatial.rotate_axis.**dispatch**(*angle: Any*, *v1: Any*, *v2: Any*) → Any

vector._compute.spatial.rotate_axis.**make_conversion**(*azimuthal1*, *longitudinal1*, *azimuthal2*, *longitudinal2*)

vector._compute.spatial.rotate_euler module

```
Spatial.rotate_euler(self, phi, theta, psi, order=...)
```

vector._compute.spatial.rotate_euler.**cartesian_xyx**(*lib*, *phi*, *theta*, *psi*, *x*, *y*, *z*)

vector._compute.spatial.rotate_euler.**cartesian_xyz**(*lib*, *phi*, *theta*, *psi*, *x*, *y*, *z*)

vector._compute.spatial.rotate_euler.**cartesian_xzx**(*lib*, *phi*, *theta*, *psi*, *x*, *y*, *z*)

vector._compute.spatial.rotate_euler.**cartesian_xzy**(*lib*, *phi*, *theta*, *psi*, *x*, *y*, *z*)

vector._compute.spatial.rotate_euler.**cartesian_yxy**(*lib*, *phi*, *theta*, *psi*, *x*, *y*, *z*)

vector._compute.spatial.rotate_euler.**cartesian_yxz**(*lib*, *phi*, *theta*, *psi*, *x*, *y*, *z*)

vector._compute.spatial.rotate_euler.**cartesian_yzx**(*lib*, *phi*, *theta*, *psi*, *x*, *y*, *z*)

vector._compute.spatial.rotate_euler.**cartesian_yzy**(*lib*, *phi*, *theta*, *psi*, *x*, *y*, *z*)

vector._compute.spatial.rotate_euler.**cartesian_zxy**(*lib*, *phi*, *theta*, *psi*, *x*, *y*, *z*)

vector._compute.spatial.rotate_euler.**cartesian_zxz**(*lib*, *phi*, *theta*, *psi*, *x*, *y*, *z*)

vector._compute.spatial.rotate_euler.**cartesian_zyx**(*lib*, *phi*, *theta*, *psi*, *x*, *y*, *z*)

vector._compute.spatial.rotate_euler.**cartesian_zyz**(*lib*, *phi*, *theta*, *psi*, *x*, *y*, *z*)

vector._compute.spatial.rotate_euler.**dispatch**(*phi: Any*, *theta: Any*, *psi: Any*, *order: Any*, *v: Any*) → Any

vector._compute.spatial.rotate_euler.**make_conversion**(*azimuthal*, *longitudinal*, *order*)

vector._compute.spatial.rotate_quaternion module

```
Spatial.rotate_quaternion(self, u, i, j, k)
```

vector._compute.spatial.rotate_quaternion.**cartesian**(*lib*, *u*, *i*, *j*, *k*, *x*, *y*, *z*)

vector._compute.spatial.rotate_quaternion.**dispatch**(*u: Any*, *i: Any*, *j: Any*, *k: Any*, *vec: Any*) → Any

vector._compute.spatial.rotate_quaternion.**make_conversion**(*azimuthal*, *longitudinal*)

vector._compute.spatial.rotateX module

```
Spatial.rotateX(self, angle)
```

vector._compute.spatial.rotateX.**dispatch**(*angle: Any*, *v: Any*) → Any

vector._compute.spatial.rotateX.**rhophi_eta**(*lib*, *angle*, *rho*, *phi*, *eta*)

vector._compute.spatial.rotateX.**rhophi_theta**(*lib*, *angle*, *rho*, *phi*, *theta*)

vector._compute.spatial.rotateX.**rhophi_z**(*lib*, *angle*, *rho*, *phi*, *z*)

vector._compute.spatial.rotateX.**xy_eta**(*lib*, *angle*, *x*, *y*, *eta*)

vector._compute.spatial.rotateX.**xy_theta**(*lib*, *angle*, *x*, *y*, *theta*)

vector._compute.spatial.rotateX.**xy_z**(*lib*, *angle*, *x*, *y*, *z*)

vector._compute.spatial.rotateY module

```
Spatial.rotateY(self, angle)
```

vector._compute.spatial.rotateY.**dispatch**(*angle: Any*, *v: Any*) → Any

vector._compute.spatial.rotateY.**rhophi_eta**(*lib*, *angle*, *rho*, *phi*, *eta*)

vector._compute.spatial.rotateY.**rhophi_theta**(*lib*, *angle*, *rho*, *phi*, *theta*)

vector._compute.spatial.rotateY.**rhophi_z**(*lib*, *angle*, *rho*, *phi*, *z*)

vector._compute.spatial.rotateY.**xy_eta**(*lib*, *angle*, *x*, *y*, *eta*)

vector._compute.spatial.rotateY.**xy_theta**(*lib*, *angle*, *x*, *y*, *theta*)

vector._compute.spatial.rotateY.**xy_z**(*lib*, *angle*, *x*, *y*, *z*)

vector._compute.spatial.scale module

```
Spatial.scale(self, factor)
```

vector._compute.spatial.scale.**dispatch**(*factor: Any*, *v: Any*) → Any

vector._compute.spatial.scale.**rectify**(*lib*, *phi*)

vector._compute.spatial.scale.**rhophi_eta**(*lib*, *factor*, *rho*, *phi*, *eta*)

vector._compute.spatial.scale.**rhophi_theta**(*lib*, *factor*, *rho*, *phi*, *theta*)

vector._compute.spatial.scale.**rhophi_z**(*lib*, *factor*, *rho*, *phi*, *z*)

vector._compute.spatial.scale.**xy_eta**(*lib*, *factor*, *x*, *y*, *eta*)

vector._compute.spatial.scale.**xy_theta**(*lib*, *factor*, *x*, *y*, *theta*)

vector._compute.spatial.scale.**xy_z**(*lib*, *factor*, *x*, *y*, *z*)

vector._compute.spatial.subtract module

```
Spatial.subtract(self, angle)
```

vector._compute.spatial.subtract.**dispatch**(*v1: Any*, *v2: Any*) → Any

vector._compute.spatial.subtract.**rhophi_eta_rhophi_eta**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.subtract.**rhophi_eta_rhophi_theta**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.subtract.**rhophi_eta_rhophi_z**(*lib*, *rho1*, *phi1*, *eta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.subtract.**rhophi_eta_xy_eta**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.subtract.**rhophi_eta_xy_theta**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.subtract.**rhophi_eta_xy_z**(*lib*, *rho1*, *phi1*, *eta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.subtract.**rhophi_theta_rhophi_eta**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.subtract.**rhophi_theta_rhophi_theta**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.subtract.**rhophi_theta_rhophi_z**(*lib*, *rho1*, *phi1*, *theta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.subtract.**rhophi_theta_xy_eta**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.subtract.**rhophi_theta_xy_theta**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.subtract.**rhophi_theta_xy_z**(*lib*, *rho1*, *phi1*, *theta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.subtract.**rhophi_z_rhophi_eta**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.subtract.**rhophi_z_rhophi_theta**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.subtract.**rhophi_z_rhophi_z**(*lib*, *rho1*, *phi1*, *z1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.subtract.**rhophi_z_xy_eta**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.subtract.**rhophi_z_xy_theta**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.subtract.**rhophi_z_xy_z**(*lib*, *rho1*, *phi1*, *z1*, *x2*, *y2*, *z2*)

vector._compute.spatial.subtract.**xy_eta_rhophi_eta**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.subtract.**xy_eta_rhophi_theta**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.subtract.**xy_eta_rhophi_z**(*lib*, *x1*, *y1*, *eta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.subtract.**xy_eta_xy_eta**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.subtract.**xy_eta_xy_theta**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.subtract.**xy_eta_xy_z**(*lib*, *x1*, *y1*, *eta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.subtract.**xy_theta_rhophi_eta**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.subtract.**xy_theta_rhophi_theta**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.subtract.**xy_theta_rhophi_z**(*lib*, *x1*, *y1*, *theta1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.subtract.**xy_theta_xy_eta**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.subtract.**xy_theta_xy_theta**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.subtract.**xy_theta_xy_z**(*lib*, *x1*, *y1*, *theta1*, *x2*, *y2*, *z2*)

vector._compute.spatial.subtract.**xy_z_rhophi_eta**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *eta2*)

vector._compute.spatial.subtract.**xy_z_rhophi_theta**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *theta2*)

vector._compute.spatial.subtract.**xy_z_rhophi_z**(*lib*, *x1*, *y1*, *z1*, *rho2*, *phi2*, *z2*)

vector._compute.spatial.subtract.**xy_z_xy_eta**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *eta2*)

vector._compute.spatial.subtract.**xy_z_xy_theta**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *theta2*)

vector._compute.spatial.subtract.**xy_z_xy_z**(*lib*, *x1*, *y1*, *z1*, *x2*, *y2*, *z2*)

vector._compute.spatial.theta module

```
@property
Spatial.theta(self)
```

vector._compute.spatial.theta.**dispatch**(*v: Any*) → Any

vector._compute.spatial.theta.**rhophi_eta**(*lib*, *rho*, *phi*, *eta*)

vector._compute.spatial.theta.**rhophi_theta**(*lib*, *rho*, *phi*, *theta*)

vector._compute.spatial.theta.**rhophi_z**(*lib*, *rho*, *phi*, *z*)

vector._compute.spatial.theta.**xy_eta**(*lib*, *x*, *y*, *eta*)

vector._compute.spatial.theta.**xy_theta**(*lib*, *x*, *y*, *theta*)

vector._compute.spatial.theta.**xy_z**(*lib*, *x*, *y*, *z*)

vector._compute.spatial.transform3D module

```
Spatial.transform3D(self, obj)
```

where `obj` has ``obj["xx"]``, ``obj["xy"]``, etc.

vector._compute.spatial.transform3D.**cartesian**(*lib*, *xx*, *xy*, *xz*, *yx*, *yy*, *yz*, *zx*, *zy*, *zz*, *x*, *y*, *z*)

vector._compute.spatial.transform3D.**dispatch**(*obj: Any*, *v: Any*) → Any

vector._compute.spatial.transform3D.**rhophi_eta**(*lib*, *xx*, *xy*, *xz*, *yx*, *yy*, *yz*, *zx*, *zy*, *zz*, *rho*, *phi*, *eta*)

vector._compute.spatial.transform3D.**rhophi_theta**(*lib*, *xx*, *xy*, *xz*, *yx*, *yy*, *yz*, *zx*, *zy*, *zz*, *rho*, *phi*, *theta*)

vector._compute.spatial.transform3D.**rhophi_z**(*lib*, *xx*, *xy*, *xz*, *yx*, *yy*, *yz*, *zx*, *zy*, *zz*, *rho*, *phi*, *z*)

vector._compute.spatial.transform3D.**xy_eta**(*lib*, *xx*, *xy*, *xz*, *yx*, *yy*, *yz*, *zx*, *zy*, *zz*, *x*, *y*, *eta*)

vector._compute.spatial.transform3D.**xy_theta**(*lib*, *xx*, *xy*, *xz*, *yx*, *yy*, *yz*, *zx*, *zy*, *zz*, *x*, *y*, *theta*)

vector._compute.spatial.unit module

```
Spatial.unit(self)
```

vector._compute.spatial.unit.**dispatch**(*v: Any*) → Any

vector._compute.spatial.unit.**rhophi_eta**(*lib*, *rho*, *phi*, *eta*)

vector._compute.spatial.unit.**rhophi_theta**(*lib*, *rho*, *phi*, *theta*)

vector._compute.spatial.unit.**rhophi_z**(*lib*, *rho*, *phi*, *z*)

vector._compute.spatial.unit.**xy_eta**(*lib*, *x*, *y*, *eta*)

vector._compute.spatial.unit.**xy_theta**(*lib*, *x*, *y*, *theta*)

vector._compute.spatial.unit.**xy_z**(*lib*, *x*, *y*, *z*)

vector._compute.spatial.z module

```
@property
Spatial.z(self)
```

vector._compute.spatial.z.**dispatch**(*v: Any*) → Any

vector._compute.spatial.z.**rhophi_eta**(*lib*, *rho*, *phi*, *eta*)

vector._compute.spatial.z.**rhophi_theta**(*lib*, *rho*, *phi*, *theta*)

vector._compute.spatial.z.**rhophi_z**(*lib*, *rho*, *phi*, *z*)

vector._compute.spatial.z.**xy_eta**(*lib*, *x*, *y*, *eta*)

vector._compute.spatial.z.**xy_theta**(*lib*, *x*, *y*, *theta*)

vector._compute.spatial.z.**xy_z**(*lib*, *x*, *y*, *z*)

Submodules

vector._methods module

class vector._methods.**Azimuthal**

Bases: *Coordinates*

property elements: tuple[Any, Any]

Azimuthal coordinates as a tuple.

Each coordinate may be a scalar, a NumPy array, an Awkward Array, etc., but they are not vectors.

class vector._methods.**AzimuthalRhoPhi**

Bases: *Azimuthal*

rho

The ρ coordinate(s).

Type
scalar, np.ndarray, ak.Array, etc.

phi

The ϕ coordinate(s).

Type
scalar, np.ndarray, ak.Array, etc.

phi: Any

rho: Any

class vector._methods.**AzimuthalXY**

Bases: *Azimuthal*

x

The x coordinate(s).

Type
scalar, np.ndarray, ak.Array, etc.

y

The y coordinate(s).

Type
scalar, np.ndarray, ak.Array, etc.

x: Any

y: Any

class vector._methods.**Coordinates**

Bases: object

class vector._methods.**Longitudinal**

Bases: *Coordinates*

property elements: tuple[Any]

Longitudinal coordinates as a tuple.

Each coordinate may be a scalar, a NumPy array, an Awkward Array, etc., but they are not vectors.

class vector._methods.**LongitudinalEta**

Bases: *Longitudinal*

eta

The η coordinate(s).

Type
scalar, np.ndarray, ak.Array, etc.

eta: Any

class vector._methods.**LongitudinalTheta**

Bases: *Longitudinal*

theta

The θ coordinate(s).

Type
scalar, `np.ndarray`, `ak.Array`, etc.

theta: Any

class vector._methods.**LongitudinalZ**

Bases: *Longitudinal*

z

The z coordinate(s).

Type
scalar, `np.ndarray`, `ak.Array`, etc.

z: Any

class vector._methods.**Lorentz**

Bases: *Spatial*, *VectorProtocolLorentz*

add(*other:* VectorProtocol) → *VectorProtocol*

Sum of `self` and `other`.

This method is equivalent to the + operator.

property beta: Any

The speed(s) of the Lorentz vector or array of vectors, in which lightlike vectors have `beta == 1`.

boost(*booster:* VectorProtocolSpatial | VectorProtocolLorentz) → SameVectorType

Boosts the vector or array of vectors using the 3D or 4D `booster`.

If `booster` is 3D, it is interpreted as a velocity (in which lightlike velocities have `mag == 1`) and *vector._methods.VectorProtocolLorentz.boost_beta3()* is called.

If `booster` is 4D, it is interpreted as a Lorentz vector and *vector._methods.VectorProtocolLorentz.boost_p4()* is called.

Note that `v.boost(v)` does not boost into the center-of-mass (CM) frame of `v`; it boosts *away* from its CM frame. Neither does `v.boost(-v)`, since that negates the time component of `v` as well.

To boost to the center-of-mass frame of a vector `v`, use *vector._methods.VectorProtocolLorentz.boostCM_of()*. For instance, `v.boostCM_of(v)` is guaranteed to have spatial components close to zero and a temporal component close to `v.tau`.

boostCM_of(*booster:* VectorProtocolSpatial | VectorProtocolLorentz) → SameVectorType

Boosts the vector or array of vectors to the center-of-mass (CM) frame of the 3D or 4D `booster`.

If `booster` is 3D, it is interpreted as a velocity (in which lightlike velocities have `mag == 1`) and *vector._methods.VectorProtocolLorentz.boostCM_of_beta3()* is called.

If `booster` is 4D, it is interpreted as a Lorentz vector and *vector._methods.VectorProtocolLorentz.boostCM_of_p4()* is called.

Note that `v.boostCM_of(v)` is guaranteed to have spatial components close to zero and a temporal component close to `v.tau`.

boostCM_of_beta3(*beta3:* VectorProtocolSpatial) → SameVectorType

Boosts the vector or array of vectors to the center-of-mass (CM) frame of the 3D velocity or array of velocity vectors `beta3`.

Note that `v.boostCM_of_beta3(v.to_beta3())` is guaranteed to have spatial components close to zero and a temporal component close to `v.tau`.

boostCM_of_p4(*p4:* VectorProtocolLorentz) → SameVectorType

Boosts the vector or array of vectors to the center-of-mass (CM) frame of the 4D vector or array of vectors `p4`.

This function is equivalent to but more numerically stable than

```
boostCM_of_beta3(p4.to_beta3())
```

Note that `v.boostCM_of_p4(v)` is guaranteed to have spatial components close to zero and a temporal component close to `v.tau`.

boostX(*beta: Any | None = None*, *gamma: Any | None = None*) → SameVectorType

Boosts the vector or array of vectors in the x direction by a speed `beta` (in which lightlike boosts have `beta == 1`) or time dilation/length contraction factor `gamma`.

Either `beta` xor `gamma` must be specified, not both or neither.

If `beta` or `gamma` is negative, it is taken as a boost in the $-x$ direction.

boostY(*beta: Any | None = None*, *gamma: Any | None = None*) → SameVectorType

Boosts the vector or array of vectors in the y direction by a speed `beta` (in which lightlike boosts have `beta == 1`) or time dilation/length contraction factor `gamma`.

Either `beta` xor `gamma` must be specified, not both or neither.

If `beta` or `gamma` is negative, it is taken as a boost in the $-y$ direction.

boostZ(*beta: Any | None = None*, *gamma: Any | None = None*) → SameVectorType

Boosts the vector or array of vectors in the z direction by a speed `beta` (in which lightlike boosts have `beta == 1`) or time dilation/length contraction factor `gamma`.

Either `beta` xor `gamma` must be specified, not both or neither.

If `beta` or `gamma` is negative, it is taken as a boost in the $-z$ direction.

boost_beta3(*beta3:* VectorProtocolSpatial) → SameVectorType

Boosts the vector or array of vectors in a direction and magnitude given by the 3D velocity or array of velocity vectors `beta3`.

Note that `v.boost_beta3(v.to_beta3())` does not boost into the center-of-mass (CM) frame of `v`; it boosts *away* from its CM frame. Neither does `v.boost_beta3((-v).to_beta3())`, since that negates the time component of `v` as well. On the other hand, `v.boost_beta3(-(v.to_beta3()))` *would* boost to the center-of-mass frame.

However, there's a function for that: *vector._methods.VectorProtocolLorentz.boostCM_of_beta3()* is explicit about boosting to a center-of-mass (CM) frame and it handles the negative sign for you: `v.boostCM_of_beta3(v.to_beta3())` is guaranteed to have spatial components close to zero and a temporal component close to `v.tau`.

boost_p4(*p4:* VectorProtocolLorentz) → SameVectorType

Boosts the vector or array of vectors in a direction and magnitude given by the 4D vector or array of vectors `p4`.

This function is equivalent to but more numerically stable than

```
boost_beta3(p4.to_beta3())
```

where *vector._methods.VectorProtocolLorentz.to_beta3()* converts a 4D Lorentz vector into a 3D velocity (in which lightlike velocities have `mag == 1`).

Note that `v.boost_p4(v)` does not boost into the center-of-mass (CM) frame of `v`; it boosts *away* from its CM frame. Neither does `v.boost_p4(-v)`, since that negates the time component of `v` as well.

To boost to the center-of-mass frame of a vector `v`, use *vector._methods.VectorProtocolLorentz.boostCM_of_p4()*. For instance, `v.boostCM_of_p4(v)` is guaranteed to have spatial components close to zero and a temporal component close to `v.tau`.

deltaRapidityPhi(*other:* VectorProtocolLorentz) → Any

Sum in quadrature of *vector._methods.VectorProtocolPlanar.deltaphi()* and the difference in *vector._methods.VectorProtocolLorentz.rapidity* of the two vectors:

$$\Delta R_{\text{rapidity}} = \sqrt{\Delta\phi^2 + \Delta\text{rapidity}^2}$$

deltaRapidityPhi2(*other:* VectorProtocolLorentz) → Any

Square of the sum in quadrature of *vector._methods.VectorProtocolPlanar.deltaphi()* and the difference in *vector._methods.VectorProtocolLorentz.rapidity* of the two vectors:

$$\Delta R_{\text{rapidity}} = \Delta\phi^2 + \Delta\text{rapidity}^2$$

dot(*other:* VectorProtocol) → Any

Vector dot product of `self` with `other`.

This method is equivalent to the `@` operator.

equal(*other:* VectorProtocol) → Any

Returns True if `self` is exactly equal to `other` (possibly for arrays of vectors), False otherwise.

This method is equivalent to the == operator.

Typically, you'll want to check *vector._methods.VectorProtocol.isclose()* to allow for numerical errors.

property gamma: Any

The time dilation/length contraction factor(s) of the Lorentz vector or array of vectors: t/τ.

is_lightlike(*tolerance: Any = 1e-05*) → Any

Returns True if the vector or a vector in the array is pointing in a lightlike direction, `t**2 == mag**2`, False otherwise.

The `tolerance` is in units of `t` and `mag`. Note that

- the default `tolerance` for *vector._methods.VectorProtocolLorentz.is_timelike()* is `0`
- the default `tolerance` for *vector._methods.VectorProtocolLorentz.is_spacelike()* is `0`
- the default `tolerance` for *vector._methods.VectorProtocolLorentz.is_lightlike()* is `1e-5`

If you want to use these methods to divide space-time into non-overlapping regions (the light-cone), use the same `tolerance` for each.

is_spacelike(*tolerance: Any = 0*) → Any

Returns True if the vector or a vector in the array is pointing in a spacelike direction, `t**2 < mag**2`, False otherwise.

The `tolerance` is in units of `t` and `mag`. Note that

- the default `tolerance` for *vector._methods.VectorProtocolLorentz.is_timelike()* is `0`
- the default `tolerance` for *vector._methods.VectorProtocolLorentz.is_spacelike()* is `0`
- the default `tolerance` for *vector._methods.VectorProtocolLorentz.is_lightlike()* is `1e-5`

If you want to use these methods to divide space-time into non-overlapping regions (the light-cone), use the same `tolerance` for each.

is_timelike(*tolerance: Any = 0*) → Any

Returns True if the vector or a vector in the array is pointing in a timelike direction, `t**2 > mag**2`, False otherwise.

The `tolerance` is in units of `t` and `mag`. Note that

- the default `tolerance` for *vector._methods.VectorProtocolLorentz.is_timelike()* is `0`
- the default `tolerance` for *vector._methods.VectorProtocolLorentz.is_spacelike()* is `0`
- the default `tolerance` for *vector._methods.VectorProtocolLorentz.is_lightlike()* is `1e-5`

If you want to use these methods to divide space-time into non-overlapping regions (the light-cone), use the same `tolerance` for each.

isclose(*other:* VectorProtocol, *rtol: Any = 1e-05*, *atol: Any = 1e-08*, *equal_nan: Any = False*) → Any

Returns True if `self` is approximately equal to `other` (possibly for arrays of vectors), False otherwise.

The relative tolerance (`rtol`) and absolute tolerance (`atol`) are interpreted as in `np.isclose`:

```
close_enough = abs(self - other) <= atol + rtol * abs(other)
```

property neg2D: SameVectorType

Returns vector(s) with the 2D part negated, not affecting any longitudinal or temporal parts.

property neg3D: SameVectorType

Returns vector(s) with the 3D part negated, not affecting any longitudinal or temporal parts.

property neg4D: SameVectorType

Same as multiplying by -1.

not_equal(*other:* VectorProtocol) → Any

Returns False if `self` is exactly equal to `other` (possibly for arrays of vectors), True otherwise.

This method is equivalent to the `!=` operator.

Typically, you'll want to check *vector._methods.VectorProtocol.isclose()* to allow for numerical errors.

property rapidity: Any

The rapidity relative to the longitudinal axis of the Lorentz vector or array of vectors.

```
0.5 * log((t + z) / (t - z))
```

scale(*factor: Any*) → SameVectorType

Returns vector(s) scaled by a `factor`, changing the length(s) but not the direction(s).

This method is equivalent to the * operator.

scale2D(*factor: Any*) → SameVectorType

Returns vector(s) with the 2D part scaled by a `factor`, not affecting any longitudinal or temporal parts.

scale3D(*factor: Any*) → SameVectorType

Returns vector(s) with the 3D part scaled by a `factor`, not affecting any longitudinal or temporal parts.

scale4D(*factor: Any*) → SameVectorType

Same as `scale`.

subtract(*other:* VectorProtocol) → *VectorProtocol*

Difference of `self` minus `other`.

This method is equivalent to the - operator.

property t: Any

The Cartesian t (time) coordinate of the vector or every vector in the array.

If t is derived from τ, it is not allowed to be `NaN`.

```
t = sqrt(max(copysign(tau**2, tau) + mag**2, 0))
```

property t2: Any

The Cartesian t (time) coordinate squared of the vector or every vector in the array.

If t^2 is derived from τ, it is not allowed to be negative.

```
t2 = max(copysign(tau**2, tau) + mag**2, 0)
```

property tau: Any

The Lorentz magnitude τ (proper time) of the vector or every vector in the array.

If τ is derived from t, spacelike vectors are represented by negative proper times.

```
tau = copysign(sqrt(abs(t**2 - mag**2)), t**2 - mag**2)
```

property tau2: Any

The Lorentz magnitude τ (proper time) squared of the vector or every vector in the array.

```
tau2 = t**2 - mag**2
```

to_beta3() → *VectorProtocolSpatial*

Converts the 4D Lorentz vector or array of vectors into a 3D velocity vector or array of vectors, in which lightlike velocities have `mag == 1`.

transform4D(*obj:* TransformProtocol4D) → SameVectorType

Arbitrarily transforms the vector(s) by

```
obj["xx"] obj["xy"] obj["xz"] obj["xt"]
obj["yx"] obj["yy"] obj["yz"] obj["yt"]
obj["zx"] obj["zy"] obj["zz"] obj["zt"]
obj["tx"] obj["ty"] obj["tz"] obj["tt"]
```

There is no restriction on the type of `obj`; it just has to provide those components (which can be arrays if the vectors are in an array).

unit() → SameVectorType

Returns vector(s) normalized to unit length, which is *rho == 1* for 2D vectors, *mag == 1* for 3D vectors, and *tau == 1* for 4D vectors.

class vector._methods.**LorentzMomentum**

Bases: *SpatialMomentum*, *MomentumProtocolLorentz*

property E: Any

Momentum-synonyor *vector._methods.VectorProtocolLorentz.t*.

property E2: Any

Momentum-synonym for `vector._methods.VectorProtocolLorent2`.

property Et: Any

Transverse energy of the four-momentum vector or array of vectors: $E_T = E \sin\theta$.

property Et2: Any

Transverse energy squared of the four-momentum vector or array of vectors: $E_T^2 = E^2 \sin^2\theta$.

property M: Any

Momentum-synonym for *vector._methods.VectorProtocolLorentz.tau*.

property M2: Any

Momentum-synonym for *vector._methods.VectorProtocolLorentz.tau2*.

property Mt: Any

Transverse mass of the four-momentum vector or array of vectors: $M_T = \sqrt{t^2 - z^2}$.

property Mt2: Any

Transverse mass squared of the four-momentum vector or array of vectors: $M_T^2 = t^2 - z^2$.

property e: Any

Momentum-synonym for *vector._methods.VectorProtocolLorentz.t*.

property e2: Any

Momentum-synonym for *vector._methods.VectorProtocolLorentz.t2*.

property energy: Any

Momentum-synonym for *vector._methods.VectorProtocolLorentz.t*.

property energy2: Any

Momentum-synonym for *vector._methods.VectorProtocolLorentz.t2*.

property et: Any

Transverse energy of the four-momentum vector or array of vectors: $E_T = E \sin\theta$.

property et2: Any

Transverse energy squared of the four-momentum vector or array of vectors: $E_T^2 = E^2 \sin^2\theta$.

property m: Any

Momentum-synonym for *vector._methods.VectorProtocolLorentz.tau*.

property m2: Any

Momentum-synonym for *vector._methods.VectorProtocolLorentz.tau2*.

property mass: Any

Momentum-synonym for *vector._methods.VectorProtocolLorentz.tau*.

property mass2: Any

Momentum-synonym for *vector._methods.VectorProtocolLorentz.tau2*.

property mt: Any

Transverse mass of the four-momentum vector or array of vectors: $M_T = \sqrt{t^2 - z^2}$.

property mt2: Any

Transverse mass squared of the four-momentum vector or array of vectors: $M_T^2 = t^2 - z^2$.

property transverse_energy: Any

Synonym for *vector._methods.MomentumProtocolLorentz.Et*.

property transverse_energy2: Any

Synonym for *vector._methods.MomentumProtocolLorentz.Et2*.

property transverse_mass: Any

Synonym for *vector._methods.MomentumProtocolLorentz.Mt*.

property transverse_mass2: Any

Synonym for *vector._methods.MomentumProtocolLorentz.Mt2*.

class vector._methods.**Momentum**

Bases: object

class vector._methods.**MomentumProtocolLorentz**

Bases: *VectorProtocolLorentz*, *MomentumProtocolSpatial*

property E: Any

Momentum-synonyor *vector._methods.VectorProtocolLorentz.t*.

property E2: Any

Momentum-synonym for vector._methods.VectorProtocolLorent2.

property Et: Any

Transverse energy of the four-momentum vector or array of vectors: $E_T = E \sin\theta$.

property Et2: Any

Transverse energy squared of the four-momentum vector or array of vectors: $E_T^2 = E^2 \sin^2\theta$.

property M: Any

Momentum-synonym for *vector._methods.VectorProtocolLorentz.tau*.

property M2: Any

Momentum-synonym for *vector._methods.VectorProtocolLorentz.tau2*.

property Mt: Any

Transverse mass of the four-momentum vector or array of vectors: $M_T = \sqrt{t^2 - z^2}$.

property Mt2: Any

Transverse mass squared of the four-momentum vector or array of vectors: $M_T^2 = t^2 - z^2$.

property e: Any

Momentum-synonym for *vector._methods.VectorProtocolLorentz.t*.

property e2: Any

Momentum-synonym for *vector._methods.VectorProtocolLorentz.t2*.

property energy: Any

Momentum-synonym for *vector._methods.VectorProtocolLorentz.t*.

property energy2: Any

Momentum-synonym for *vector._methods.VectorProtocolLorentz.t2*.

property et: Any

Transverse energy of the four-momentum vector or array of vectors: $E_T = E \sin\theta$.

property et2: Any

Transverse energy squared of the four-momentum vector or array of vectors: $E_T^2 = E^2 \sin^2\theta$.

property m: Any

Momentum-synonym for *vector._methods.VectorProtocolLorentz.tau*.

property m2: Any

Momentum-synonym for *vector._methods.VectorProtocolLorentz.tau2*.

property mass: Any

Momentum-synonym for *vector._methods.VectorProtocolLorentz.tau*.

property mass2: Any

Momentum-synonym for *vector._methods.VectorProtocolLorentz.tau2*.

property mt: Any

Transverse mass of the four-momentum vector or array of vectors: $M_T = \sqrt{t^2 - z^2}$.

property mt2: Any

Transverse mass squared of the four-momentum vector or array of vectors: $M_T^2 = t^2 - z^2$.

property transverse_energy: Any

Synonym for *vector._methods.MomentumProtocolLorentz.Et*.

property transverse_energy2: Any

Synonym for *vector._methods.MomentumProtocolLorentz.Et2*.

property transverse_mass: Any

Synonym for *vector._methods.MomentumProtocolLorentz.Mt*.

property transverse_mass2: Any

Synonym for *vector._methods.MomentumProtocolLorentz.Mt2*.

class vector._methods.**MomentumProtocolPlanar**

Bases: *VectorProtocolPlanar*

property pt: Any

Momentum-synonym for *vector._methods.VectorProtocolPlanar.rho*.

property pt2: Any

Momentum-synonym for *vector._methods.VectorProtocolPlanar.rho2*.

property px: Any

Momentum-synonym for *vector._methods.VectorProtocolPlanar.x*.

property py: Any

Momentum-synonym for *vector._methods.VectorProtocolPlanar.y*.

class vector._methods.**MomentumProtocolSpatial**

Bases: *VectorProtocolSpatial*, *MomentumProtocolPlanar*

property p: Any

Momentum-synonym for *vector._methods.VectorProtocolSpatial.mag*.

property p2: Any

Momentum-synonym for *vector._methods.VectorProtocolSpatial.mag2*.

property pseudorapidity: Any

Momentum-synonym for *vector._methods.VectorProtocolSpatial.eta*.

property pz: Any

Momentum-synonym for *vector._methods.VectorProtocolSpatial.z*.

class vector._methods.**Planar**

Bases: *VectorProtocolPlanar*

add(*other:* VectorProtocol) → *VectorProtocol*

Sum of `self` and `other`.

This method is equivalent to the + operator.

deltaphi(*other:* VectorProtocol) → Any

Signed difference in ϕ of `self` minus `other` (in radians).

dot(*other:* VectorProtocol) → Any

Vector dot product of `self` with `other`.

This method is equivalent to the `@` operator.

equal(*other:* VectorProtocol) → Any

Returns True if `self` is exactly equal to `other` (possibly for arrays of vectors), False otherwise.

This method is equivalent to the == operator.

Typically, you'll want to check *vector._methods.VectorProtocol.isclose()* to allow for numerical errors.

is_antiparallel(*other:* VectorProtocol, *tolerance: Any = 1e-05*) → Any

Returns True if `self` and `other` are pointing in opposite directions (i.e. dot product is nearly `-abs(self) * abs(other)`).

The `tolerance` is measured in units of $\cos(\Delta\alpha)$ where $\Delta\alpha$ is `self.deltaangle(other)`.

is_parallel(*other:* VectorProtocol, *tolerance: Any = 1e-05*) → Any

Returns True if `self` and `other` are pointing in the same direction (i.e. not "antiparallel"; dot product is nearly `abs(self) * abs(other)`).

The `tolerance` is measured in units of $\cos(\Delta\alpha)$ where $\Delta\alpha$ is `self.deltaangle(other)`.

is_perpendicular(*other:* VectorProtocol, *tolerance: Any = 1e-05*) → Any

Returns True if `self` and `other` are pointing in perpendicular directions (i.e. dot product is nearly `0`).

The `tolerance` is measured in units of $\cos(\Delta\alpha)$ where $\Delta\alpha$ is `self.deltaangle(other)`.

isclose(*other:* VectorProtocol, *rtol: Any = 1e-05*, *atol: Any = 1e-08*, *equal_nan: Any = False*) → Any

Returns True if `self` is approximately equal to `other` (possibly for arrays of vectors), False otherwise.

The relative tolerance (`rtol`) and absolute tolerance (`atol`) are interpreted as in `np.isclose`:

```
close_enough = abs(self - other) <= atol + rtol * abs(other)
```

property neg2D: SameVectorType

Returns vector(s) with the 2D part negated, not affecting any longitudinal or temporal parts.

not_equal(*other:* VectorProtocol) → Any

Returns False if `self` is exactly equal to `other` (possibly for arrays of vectors), True otherwise.

This method is equivalent to the `!=` operator.

Typically, you'll want to check *vector._methods.VectorProtocol.isclose()* to allow for numerical errors.

property phi: Any

The polar ϕ coordinate of the vector or every vector in the array (in radians, always between $-\pi$ and π).

property rho: Any

The polar ρ coordinate of the vector or every vector in the array.

This is also the magnitude of the 2D azimuthal part of the vector (not including any longitudinal or temporal parts).

property rho2: Any

The polar ρ coordinate squared of the vector or every vector in the array.

rotateZ(*angle: Any*) → SameVectorType

Rotates the vector(s) by a given `angle` (in radians) around the longitudinal axis.

Note that the `angle` can be an array with the same length as the vectors, if the vectors are in an array.

scale(*factor: Any*) → SameVectorType

Returns vector(s) scaled by a `factor`, changing the length(s) but not the direction(s).

This method is equivalent to the `*` operator.

scale2D(*factor: Any*) → SameVectorType

Returns vector(s) with the 2D part scaled by a `factor`, not affecting any longitudinal or temporal parts.

subtract(*other:* VectorProtocol) → *VectorProtocol*

Difference of `self` minus `other`.

This method is equivalent to the `-` operator.

transform2D(*obj:* TransformProtocol2D) → SameVectorType

Arbitrarily transforms the vector(s) by

```
obj["xx"] obj["xy"]
obj["yx"] obj["yy"]
```

leaving any longitudinal or temporal coordinates unchanged. There is no restriction on the type of `obj`; it just has to provide those components (which can be arrays if the vectors are in an array).

unit() → SameVectorType

Returns vector(s) normalized to unit length, which is *rho == 1* for 2D vectors, *mag == 1* for 3D vectors, and *tau == 1* for 4D vectors.

property x: Any

The Cartesian x coordinate of the vector or every vector in the array.

property y: Any

The Cartesian y coordinate of the vector or every vector in the array.

class vector._methods.**PlanarMomentum**

Bases: *Momentum*, *MomentumProtocolPlanar*

property pt: Any

Momentum-synonym for *vector._methods.VectorProtocolPlanar.rho*.

property pt2: Any

Momentum-synonym for *vector._methods.VectorProtocolPlanar.rho2*.

property px: Any

Momentum-synonym for *vector._methods.VectorProtocolPlanar.x*.

property py: Any

Momentum-synonym for *vector._methods.VectorProtocolPlanar.y*.

class vector._methods.**Spatial**

Bases: *Planar*, *VectorProtocolSpatial*

add(*other:* VectorProtocol) → *VectorProtocol*

Sum of `self` and `other`.

This method is equivalent to the + operator.

property costheta: Any

The $\cos\theta$ coordinate of the vector or every vector in the array (has the same sign as z).

property cottheta: Any

The $\cot\theta$ coordinate of the vector or every vector in the array (has the same sign as z).

cross(*other:* VectorProtocolSpatial) → *VectorProtocolSpatial*

The 3D cross-product of `self` with `other`.

Even if `self` or `other` is 4D, the resulting vector(s) is/are 3D.

deltaR(*other:* VectorProtocolSpatial | VectorProtocolLorentz) → Any

Sum in quadrature of *vector._methods.VectorProtocolPlanar.deltaphi()* and *vector._methods.VectorProtocolSpatial.deltaeta()*:

$$\Delta R = \sqrt{\Delta\phi^2 + \Delta\eta^2}$$

deltaR2(*other:* VectorProtocolSpatial | VectorProtocolLorentz) → Any

Square of the sum in quadrature of *vector._methods.VectorProtocolPlanar.deltaphi()* and *vector._methods.VectorProtocolSpatial.deltaeta()*:

$$\Delta R^2 = \Delta\phi^2 + \Delta\eta^2$$

deltaangle(*other:* VectorProtocolSpatial | VectorProtocolLorentz) → Any

Angle in 3D space between `self` and `other`, which is always positive, between 0 and π.

deltaeta(*other:* VectorProtocolSpatial | VectorProtocolLorentz) → Any

Signed difference in η of `self` minus `other`.

dot(*other:* VectorProtocol) → Any

Vector dot product of `self` with `other`.

This method is equivalent to the `@` operator.

equal(*other:* VectorProtocol) → Any

Returns True if `self` is exactly equal to `other` (possibly for arrays of vectors), False otherwise.

This method is equivalent to the == operator.

Typically, you'll want to check *vector._methods.VectorProtocol.isclose()* to allow for numerical errors.

property eta: Any

The pseudorapidity η coordinate of the vector or every vector in the array (in radians, always between 0 ($+z$) and π ($-z$)).

is_antiparallel(*other:* VectorProtocol, *tolerance: Any = 1e-05*) → Any

Returns True if `self` and `other` are pointing in opposite directions (i.e. dot product is nearly `-abs(self) * abs(other)`).

The `tolerance` is measured in units of $\cos(\Delta\alpha)$ where $\Delta\alpha$ is `self.deltaangle(other)`.

is_parallel(*other:* VectorProtocol, *tolerance: Any = 1e-05*) → Any

Returns True if `self` and `other` are pointing in the same direction (i.e. not "antiparallel"; dot product is nearly `abs(self) * abs(other)`).

The `tolerance` is measured in units of $\cos(\Delta\alpha)$ where $\Delta\alpha$ is `self.deltaangle(other)`.

is_perpendicular(*other:* VectorProtocol, *tolerance: Any = 1e-05*) → Any

Returns True if `self` and `other` are pointing in perpendicular directions (i.e. dot product is nearly `0`).

The `tolerance` is measured in units of $\cos(\Delta\alpha)$ where $\Delta\alpha$ is `self.deltaangle(other)`.

isclose(*other:* VectorProtocol, *rtol: Any = 1e-05*, *atol: Any = 1e-08*, *equal_nan: Any = False*) → Any

Returns True if `self` is approximately equal to `other` (possibly for arrays of vectors), False otherwise.

The relative tolerance (`rtol`) and absolute tolerance (`atol`) are interpreted as in `np.isclose`:

```
close_enough = abs(self - other) <= atol + rtol * abs(other)
```

property mag: Any

The magnitude of the vector(s) in 3D (not including any temporal parts).

property mag2: Any

The magnitude-squared of the vector(s) in 3D (not including any temporal parts).

property neg2D: SameVectorType

Returns vector(s) with the 2D part negated, not affecting any longitudinal or temporal parts.

property neg3D: SameVectorType

Returns vector(s) with the 3D part negated, not affecting any longitudinal or temporal parts.

not_equal(*other:* VectorProtocol) → Any

Returns False if `self` is exactly equal to `other` (possibly for arrays of vectors), True otherwise.

This method is equivalent to the != operator.

Typically, you'll want to check *vector._methods.VectorProtocol.isclose()* to allow for numerical errors.

rotateX(*angle: Any*) → SameVectorType

Rotates the vector(s) by a given `angle` (in radians) around the x axis.

Note that the `angle` can be an array with the same length as the vectors, if the vectors are in an array.

rotateY(*angle: Any*) → SameVectorType

Rotates the vector(s) by a given `angle` (in radians) around the y axis.

Note that the `angle` can be an array with the same length as the vectors, if the vectors are in an array.

rotate_axis(*axis:* VectorProtocolSpatial, *angle: Any*) → SameVectorType

Rotates the vector(s) by a given `angle` (in radians) around the axis indicated by another vector, `axis`. The magnitude of `axis` is ignored.

Note that the `axis` and `angle` can be arrays with the same length as the vectors, if the vectors are in an array.

rotate_euler(*phi: Any*, *theta: Any*, *psi: Any*, *order: str = 'zxz'*) → SameVectorType

Rotates the vector(s) by three given angles: `phi`, `theta`, and `psi` (in radians). The `order` string determines which axis each rotation is applied around:

- `"zxz"`, `"xyx"`, `"yzy"`, `"zyz"`, `"xzx"`, and `"yxy"` are proper Euler angles
- `"zxz"`, `"xyx"`, `"yzy"`, `"zyz"`, `"xzx"`, and `"yxy"` are Tait-Bryan angles (see *vector._methods.VectorProtocolSpatial.rotate_nautical()*)

The names `phi`, `theta`, and `psi` agree with Wikipedia's terminology, and both the names and order agree with ROOT's Math::EulerAngles. The default `order = "zxz"` is also ROOT's convention.

Note that the angles can be arrays with the same lengths as the vectors, if the vectors are in an array.

rotate_nautical(*yaw: Any*, *pitch: Any*, *roll: Any*) → SameVectorType

Rotates the vector(s) by three given angles: `yaw`, `pitch`, and `roll` (in radians). These are Tait-Bryan angles often used for boats and planes (see this lesson and this lesson).

This function is entirely equivalent to

```
rotate_euler(roll, pitch, yaw, order="zyx")
```

Note that the angles can be arrays with the same lengths as the vectors, if the vectors are in an array.

rotate_quaternion(*u: Any*, *i: Any*, *j: Any*, *k: Any*) → SameVectorType

Rotates the vector(s) by four angles as quaternion coefficients (in radians). Four angles are sometimes preferred over three because the latter has a pathology known as "gimbal lock."

This function follows the same conventions as ROOT's Math::Quaternion.

Note that the angles can be arrays with the same lengths as the vectors, if the vectors are in an array.

scale(*factor: Any*) → SameVectorType

Returns vector(s) scaled by a `factor`, changing the length(s) but not the direction(s).

This method is equivalent to the * operator.

scale2D(*factor: Any*) → SameVectorType

Returns vector(s) with the 2D part scaled by a `factor`, not affecting any longitudinal or temporal parts.

scale3D(*factor: Any*) → SameVectorType

Returns vector(s) with the 3D part scaled by a `factor`, not affecting any longitudinal or temporal parts.

subtract(*other:* VectorProtocol) → *VectorProtocol*

Difference of `self` minus `other`.

This method is equivalent to the `-` operator.

property theta: Any

The spherical θ coordinate (polar angle) of the vector or every vector in the array (in radians, always between 0 ($+z$) and π ($-z$)).

transform3D(*obj:* TransformProtocol3D) → SameVectorType

Arbitrarily transforms the vector(s) by

```
obj["xx"] obj["xy"] obj["xz"]
obj["yx"] obj["yy"] obj["yz"]
obj["zx"] obj["zy"] obj["zz"]
```

leaving any temporal coordinate unchanged. There is no restriction on the type of `obj`; it just has to provide those components (which can be arrays if the vectors are in an array).

unit() → SameVectorType

Returns vector(s) normalized to unit length, which is *rho == 1* for 2D vectors, *mag == 1* for 3D vectors, and *tau == 1* for 4D vectors.

property z: Any

The Cartesian z coordinate of the vector or every vector in the array.

class vector._methods.**SpatialMomentum**

Bases: *PlanarMomentum*, *MomentumProtocolSpatial*

property p: Any

Momentum-synonym for *vector._methods.VectorProtocolSpatial.mag*.

property p2: Any

Momentum-synonym for *vector._methods.VectorProtocolSpatial.mag2*.

property pseudorapidity: Any

Momentum-synonym for *vector._methods.VectorProtocolSpatial.eta*.

property pz: Any

Momentum-synonym for *vector._methods.VectorProtocolSpatial.z*.

class vector._methods.**Temporal**

Bases: *Coordinates*

property elements: tuple[Any]

Temporal coordinates as a tuple.

Each coordinate may be a scalar, a NumPy array, an Awkward Array, etc., but they are not vectors.

class vector._methods.**TemporalT**

Bases: *Temporal*

t

The t coordinate(s).

Type
scalar, `np.ndarray`, `ak.Array`, etc.

t: Any

class vector._methods.**TemporalTau**

Bases: *Temporal*

tau

The τ coordinate(s).

Type
scalar, np.ndarray, ak.Array, etc.

tau: Any

class vector._methods.**Vector**(*, *x: float*, *y: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *z: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *eta: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *theta: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *z: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *eta: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *theta: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*)
class vector._methods.**Vector**(*, *pt: float*, *phi: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *pz: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *z: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *pz: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *z: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *pz: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *z: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *pz: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *pz: float*)
class vector._methods.**Vector**(*, *pt: float*, *phi: float*, *z: float*)
class vector._methods.**Vector**(*, *pt: float*, *phi: float*, *pz: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *theta: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *theta: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *theta: float*)
class vector._methods.**Vector**(*, *pt: float*, *phi: float*, *theta: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *eta: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *eta: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *eta: float*)
class vector._methods.**Vector**(*, *pt: float*, *phi: float*, *eta: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *pz: float*, *t: float*)

class vector._methods.**Vector**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector._methods.**Vector**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector._methods.**Vector**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector._methods.**Vector**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector._methods.**Vector**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector._methods.**Vector**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector._methods.**Vector**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector._methods.**Vector**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector._methods.**Vector**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)

class vector._methods.**Vector**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector._methods.**Vector**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector._methods.**Vector**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector._methods.**Vector**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector._methods.**Vector**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector._methods.**Vector**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector._methods.**Vector**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector._methods.**Vector**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *eta: float*, *e: float*)

class vector._methods.**Vector**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector._methods.**Vector**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector._methods.**Vector**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector._methods.**Vector**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector._methods.**Vector**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector._methods.**Vector**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector._methods.**Vector**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector._methods.**Vector**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *theta: float*, *M: float*)

class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector._methods.**Vector**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector._methods.**Vector**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector._methods.**Vector**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector._methods.**Vector**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector._methods.**Vector**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector._methods.**Vector**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector._methods.**Vector**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector._methods.**Vector**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)

class vector._methods.**Vector**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector._methods.**Vector**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector._methods.**Vector**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector._methods.**Vector**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector._methods.**Vector**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector._methods.**Vector**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector._methods.**Vector**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector._methods.**Vector**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector._methods.**Vector**(*__azumthal:* Azimuthal)
class vector._methods.**Vector**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector._methods.**Vector**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *VectorProtocol*

like(*other:* VectorProtocol) → *VectorProtocol*

Projects the vector into the geometric coordinates of the *other* vector.

Value(s) of 0 is/are imputed while transforming vector from a lower geometric dimension to a higher geometric dimension.

```
vec_4d + vec_3d.like(vec_4d)
```

For more flexibility (passing new coordinate values), see *vector._methods.Vector2D.to_Vector3D()*, *vector._methods.Vector2D.to_Vector4D()*, and *vector._methods.Vector3D.to_Vector4D()*, which can be used as:

```
vec_2d.to_Vector3D(z=3.0)
vec_2d.to_Vector4D(z=3.0, t=4.0)
vec_3d.to_Vector4D(t=4.0)
```

to_ptphi() → *VectorProtocolPlanar*

Converts to pt-ϕ coordinates, possibly eliminating dimensions with a projection.

to_ptphieta(*, *eta: float | ndarray = 0.0*) → *VectorProtocolSpatial*

Converts to pt-ϕ-η coordinates, possibly eliminating or imputing dimensions with a projection.

The eta coordinate can be passed as a named argument.

to_ptphietaenergy(*, *eta: float | ndarray = 0.0*, *energy: float | ndarray = 0.0*) → *VectorProtocolLorentz*

Converts to pt-ϕ-η-$energy$ coordinates, possibly imputing dimensions with a projection.

The eta and $energy$ coordinates can be passed as a named argument.

to_ptphietamass(*, *eta: float | ndarray = 0.0*, *mass: float | ndarray = 0.0*) → *VectorProtocolLorentz*

Converts to pt-ϕ-θ-$mass$ coordinates, possibly imputing dimensions with a projection.

The eta and $mass$ coordinates can be passed as a named argument.

to_ptphipz(*, *pz: float | ndarray = 0.0*) → *VectorProtocolSpatial*

Converts to pt-ϕ-pz coordinates, possibly eliminating or imputing dimensions with a projection.

The pz coordinate can be passed as a named argument.

to_ptphipzenergy(*, *pz: float | ndarray = 0.0*, *energy: float | ndarray = 0.0*) → *VectorProtocolLorentz*

Converts to pt-ϕ-pz-$energy$ coordinates, possibly imputing dimensions with a projection.

The pz and $energy$ coordinates can be passed as a named argument.

to_ptphipzmass(*, *pz: float | ndarray = 0.0*, *mass: float | ndarray = 0.0*) → *VectorProtocolLorentz*

Converts to pt-ϕ-pz-$mass$ coordinates, possibly imputing dimensions with a projection.

The pz and $mass$ coordinates can be passed as a named argument.

to_ptphitheta(*, *theta: float | ndarray = 0.0*) → *VectorProtocolSpatial*

Converts to pt-ϕ-θ coordinates, possibly eliminating or imputing dimensions with a projection.

The $theta$ coordinate can be passed as a named argument.

to_ptphithetaenergy(*, *theta: float | ndarray = 0.0*, *energy: float | ndarray = 0.0*) → *VectorProtocolLorentz*

Converts to pt-ϕ-θ-$energy$ coordinates, possibly imputing dimensions with a projection.

The $theta$ and $energy$ coordinates can be passed as a named argument.

to_ptphithetamass(*, *theta: float | ndarray = 0.0*, *mass: float | ndarray = 0.0*) → *VectorProtocolLorentz*

Converts to pt-ϕ-θ-$mass$ coordinates, possibly imputing dimensions with a projection.

The $theta$ and $mass$ coordinates can be passed as a named argument.

to_pxpy() → *VectorProtocolPlanar*

Converts to px-py coordinates, possibly eliminating dimensions with a projection.

to_pxpyeta(*, *eta: float | ndarray = 0.0*) → *VectorProtocolSpatial*

Converts to px-py-η coordinates, possibly eliminating or imputing dimensions with a projection.

The eta coordinate can be passed as a named argument.

to_pxpyetaenergy(*, *eta: float | ndarray = 0.0*, *energy: float | ndarray = 0.0*) → *VectorProtocolLorentz*

Converts to px-py-η-$energy$ coordinates, possibly imputing dimensions with a projection.

The eta and $energy$ coordinates can be passed as a named argument.

to_pxpyetamass(*, *eta: float | ndarray = 0.0*, *mass: float | ndarray = 0.0*) → *VectorProtocolLorentz*

Converts to px-py-η-$mass$ coordinates, possibly imputing dimensions with a projection.

The eta and $mass$ coordinates can be passed as a named argument.

to_pxpypz(*, *pz: float | ndarray = 0.0*) → *VectorProtocolSpatial*

Converts to px-py-pz coordinates, possibly eliminating or imputing dimensions with a projection.

The pz coordinate can be passed as a named argument.

to_pxpypzenergy(*, *pz: float | ndarray = 0.0*, *energy: float | ndarray = 0.0*) → *VectorProtocolLorentz*

Converts to px-py-pz-$energy$ coordinates, possibly imputing dimensions with a projection.

The pz and $energy$ coordinates can be passed as a named argument.

to_pxpypzmass(*, *pz: float | ndarray = 0.0*, *mass: float | ndarray = 0.0*) → *VectorProtocolLorentz*

Converts to px-py-pz-$mass$ coordinates, possibly imputing dimensions with a projection.

The pz and $mass$ coordinates can be passed as a named argument.

to_pxpytheta(*, *theta: float | ndarray = 0.0*) → *VectorProtocolSpatial*

Converts to px-py-θ coordinates, possibly eliminating or imputing dimensions with a projection.

The $theta$ coordinate can be passed as a named argument.

to_pxpythetaenergy(*, *theta: float | ndarray = 0.0*, *energy: float | ndarray = 0.0*) → *VectorProtocolLorentz*

Converts to px-py-θ-$energy$ coordinates, possibly imputing dimensions with a projection.

The $theta$ and $energy$ coordinates can be passed as a named argument.

to_pxpythetamass(*, *theta: float | ndarray = 0.0*, *mass: float | ndarray = 0.0*) → *VectorProtocolLorentz*

Converts to px-py-θ-$energy$ coordinates, possibly imputing dimensions with a projection.

The $theta$ and $mass$ coordinates can be passed as a named argument.

to_rhophi() → *VectorProtocolPlanar*

Converts to ρ-ϕ coordinates, possibly eliminating dimensions with a projection.

to_rhophieta(*, *eta: float | ndarray = 0.0*) → *VectorProtocolSpatial*

Converts to ρ-ϕ-η coordinates, possibly eliminating or imputing dimensions with a projection.

The eta coordinate can be passed as a named argument.

to_rhophietat(*, *eta: float | ndarray = 0.0*, *t: float | ndarray = 0.0*) → *VectorProtocolLorentz*

Converts to ρ-ϕ-η-t coordinates, possibly imputing dimensions with a projection.

The eta and t coordinates can be passed as a named argument.

to_rhophietatau(*, *eta: float | ndarray = 0.0*, *tau: float | ndarray = 0.0*) → *VectorProtocolLorentz*

Converts to ρ-ϕ-η-τ coordinates, possibly imputing dimensions with a projection.

The eta and tau coordinates can be passed as a named argument.

to_rhophitheta(*, *theta: float | ndarray = 0.0*) → *VectorProtocolSpatial*

Converts to ρ-ϕ-θ coordinates, possibly eliminating or imputing dimensions with a projection.

The $theta$ coordinate can be passed as a named argument.

to_rhophithetat(*, *theta: float | ndarray = 0.0*, *t: float | ndarray = 0.0*) → *VectorProtocolLorentz*

Converts to ρ-ϕ-θ-t coordinates, possibly imputing dimensions with a projection.

The $theta$ and t coordinates can be passed as a named argument.

to_rhophithetatau(*, *theta: float | ndarray = 0.0*, *tau: float | ndarray = 0.0*) → *VectorProtocolLorentz*

Converts to ρ-ϕ-θ-τ coordinates, possibly imputing dimensions with a projection.

The $theta$ and tau coordinates can be passed as a named argument.

to_rhophiz(*, *z: float | ndarray = 0.0*) → *VectorProtocolSpatial*

Converts to ρ-ϕ-z coordinates, possibly eliminating or imputing dimensions with a projection.

The z coordinate can be passed as a named argument.

to_rhophizt(*, *z: float | ndarray = 0.0*, *t: float | ndarray = 0.0*) → *VectorProtocolLorentz*

Converts to ρ-ϕ-z-t coordinates, possibly imputing dimensions with a projection.

The z and t coordinates can be passed as a named argument.

to_rhophiztau(*, *z: float | ndarray = 0.0*, *tau: float | ndarray = 0.0*) → *VectorProtocolLorentz*

Converts to ρ-ϕ-z-τ coordinates, possibly imputing dimensions with a projection.

The z and tau coordinates can be passed as a named argument.

to_xy() → *VectorProtocolPlanar*

Converts to x-y coordinates, possibly eliminating dimensions with a projection.

to_xyeta(*, *eta: float | ndarray = 0.0*) → *VectorProtocolSpatial*

Converts to x-y-η coordinates, possibly eliminating or imputing dimensions with a projection.

The eta coordinate can be passed as a named argument.

to_xyetat(*, *eta: float | ndarray = 0.0*, *t: float | ndarray = 0.0*) → *VectorProtocolLorentz*

Converts to x-y-η-t coordinates, possibly imputing dimensions with a projection.

The eta and t coordinates can be passed as a named argument.

to_xyetatau(*, *eta: float | ndarray = 0.0*, *tau: float | ndarray = 0.0*) → *VectorProtocolLorentz*

Converts to x-y-η-τ coordinates, possibly imputing dimensions with a projection.

The eta and tau coordinates can be passed as a named argument.

to_xytheta(*, *theta: float | ndarray = 0.0*) → *VectorProtocolSpatial*

Converts to x-y-θ coordinates, possibly eliminating or imputing dimensions with a projection.

The $theta$ coordinate can be passed as a named argument.

to_xythetat(*, *theta: float | ndarray = 0.0*, *t: float | ndarray = 0.0*) → *VectorProtocolLorentz*

Converts to x-y-θ-t coordinates, possibly imputing dimensions with a projection.

The $theta$ and t coordinates can be passed as a named argument.

to_xythetatau(*, *theta: float | ndarray = 0.0*, *tau: float | ndarray = 0.0*) → *VectorProtocolLorentz*

Converts to x-y-θ-τ coordinates, possibly imputing dimensions with a projection.

The $theta$ and tau coordinates can be passed as a named argument.

to_xyz(*, *z: float | ndarray = 0.0*) → *VectorProtocolSpatial*

Converts to x-y-z coordinates, possibly eliminating or imputing dimensions with a projection.

The z coordinate can be passed as a named argument.

to_xyzt(*, *z: float | ndarray = 0.0*, *t: float | ndarray = 0.0*) → *VectorProtocolLorentz*

Converts to x-y-z-t coordinates, possibly imputing dimensions with a projection.

The z and t coordinates can be passed as a named argument.

to_xyztau(*, *z: float | ndarray = 0.0*, *tau: float | ndarray = 0.0*) → *VectorProtocolLorentz*

Converts to x-y-z-τ coordinates, possibly imputing dimensions with a projection.

The z and tau coordinates can be passed as a named argument.

class vector._methods.**Vector2D**(*, *x: float*, *y: float*)

class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*)

class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *z: float*)

class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *eta: float*)

class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *theta: float*)

class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *z: float*)

class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *eta: float*)

class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *theta: float*)

class vector._methods.**Vector2D**(*, *px: float*, *py: float*)

class vector._methods.**Vector2D**(*, *x: float*, *py: float*)

class vector._methods.**Vector2D**(*, *px: float*, *y: float*)

class vector._methods.**Vector2D**(*, *pt: float*, *phi: float*)

class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *pz: float*)

class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *z: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *pz: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *z: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *pz: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *z: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *pz: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector._methods.**Vector2D**(*, *pt: float*, *phi: float*, *z: float*)
class vector._methods.**Vector2D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *theta: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *theta: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *theta: float*)
class vector._methods.**Vector2D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *eta: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *eta: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *eta: float*)
class vector._methods.**Vector2D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector._methods.**Vector2D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector._methods.**Vector2D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector._methods.**Vector2D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector._methods.**Vector2D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)

class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector._methods.**Vector2D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector._methods.**Vector2D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector._methods.**Vector2D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector._methods.**Vector2D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector._methods.**Vector2D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector._methods.**Vector2D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector._methods.**Vector2D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)

class vector._methods.**Vector2D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector._methods.**Vector2D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector._methods.**Vector2D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector._methods.**Vector2D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector._methods.**Vector2D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector._methods.**Vector2D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector._methods.**Vector2D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector._methods.**Vector2D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)

class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector._methods.**Vector2D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector._methods.**Vector2D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector._methods.**Vector2D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector._methods.**Vector2D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector._methods.**Vector2D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector._methods.**Vector2D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector._methods.**Vector2D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)

class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector._methods.**Vector2D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector._methods.**Vector2D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector._methods.**Vector2D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector._methods.**Vector2D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector._methods.**Vector2D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector._methods.**Vector2D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector._methods.**Vector2D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector._methods.**Vector2D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector._methods.**Vector2D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector._methods.**Vector2D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector._methods.**Vector2D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector._methods.**Vector2D**(*__azumthal:* Azimuthal)
class vector._methods.**Vector2D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector._methods.**Vector2D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *Vector*, *VectorProtocolPlanar*

to_2D() → *VectorProtocolPlanar*

Alias for *vector._methods.Vector2D.to_Vector2D()*.

to_3D(**kwargs: float | None*) → *VectorProtocolSpatial*

Alias for *vector._methods.Vector2D.to_Vector3D()*.

to_4D(**kwargs: *float* | *None*) → *VectorProtocolLorentz*

Alias for *vector._methods.Vector2D.to_Vector4D()*.

to_Vector2D() → *VectorProtocolPlanar*

Projects this vector/these vectors onto azimuthal coordinates only.

to_Vector3D(*, *z: float | ndarray | None = None*, *pz: float | ndarray | None = None*, *theta: float | ndarray | None = None*, *eta: float | ndarray | None = None*) → *VectorProtocolSpatial*

Converts a 2D vector to 3D vector.

The scalar longitudinal coordinate is broadcasted for NumPy and Awkward vectors. Only a single longitudinal coordinate should be provided.

Examples

```
>>> import vector
>>> vec = vector.VectorObject2D(x=1, y=2)
>>> vec.to_Vector3D(z=1)
VectorObject3D(x=1, y=2, z=1)
>>> vec = vector.MomentumObject2D(px=1, py=2)
>>> vec.to_Vector3D(pz=4)
MomentumObject3D(px=1, py=2, pz=4)
```

to_Vector4D(*, *z: float | ndarray | None = None*, *pz: float | ndarray | None = None*, *theta: float | ndarray | None = None*, *eta: float | ndarray | None = None*, *t: float | ndarray | None = None*, *e: float | ndarray | None = None*, *E: float | ndarray | None = None*, *energy: float | ndarray | None = None*, *tau: float | ndarray | None = None*, *m: float | ndarray | None = None*, *M: float | ndarray | None = None*, *mass: float | ndarray | None = None*) → *VectorProtocolLorentz*

Converts a 2D vector to 4D vector.

The scalar longitudinal and temporal coordinates are broadcasted for NumPy and Awkward vectors. Only a single longitudinal and temporal coordinate should be provided.

Examples

```
>>> import vector
>>> vec = vector.VectorObject2D(x=1, y=2)
>>> vec.to_Vector4D(z=3, t=4)
VectorObject4D(x=1, y=2, z=3, t=4)
>>> vec = vector.MomentumObject2D(px=1, py=2)
>>> vec.to_Vector4D(pz=4, energy=4)
MomentumObject4D(px=1, py=2, pz=4, E=4)
```

class vector._methods.**Vector3D**(*, *x: float*, *y: float*)

class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*)

class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *z: float*)

class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *eta: float*)

class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *theta: float*)

class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *z: float*)

class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *eta: float*)

class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *theta: float*)

class vector._methods.**Vector3D**(*, *px: float*, *py: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*)
class vector._methods.**Vector3D**(*, *pt: float*, *phi: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *pz: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *z: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *pz: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *z: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *pz: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *z: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *pz: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector._methods.**Vector3D**(*, *pt: float*, *phi: float*, *z: float*)
class vector._methods.**Vector3D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *theta: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *theta: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *theta: float*)
class vector._methods.**Vector3D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *eta: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *eta: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *eta: float*)
class vector._methods.**Vector3D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector._methods.**Vector3D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector._methods.**Vector3D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector._methods.**Vector3D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)

class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector._methods.**Vector3D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector._methods.**Vector3D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector._methods.**Vector3D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector._methods.**Vector3D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector._methods.**Vector3D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector._methods.**Vector3D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector._methods.**Vector3D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector._methods.**Vector3D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)

class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector._methods.**Vector3D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector._methods.**Vector3D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector._methods.**Vector3D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector._methods.**Vector3D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector._methods.**Vector3D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector._methods.**Vector3D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector._methods.**Vector3D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)

class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector._methods.**Vector3D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector._methods.**Vector3D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector._methods.**Vector3D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector._methods.**Vector3D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector._methods.**Vector3D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector._methods.**Vector3D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)

class vector._methods.**Vector3D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector._methods.**Vector3D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector._methods.**Vector3D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector._methods.**Vector3D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector._methods.**Vector3D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector._methods.**Vector3D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector._methods.**Vector3D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector._methods.**Vector3D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector._methods.**Vector3D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector._methods.**Vector3D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector._methods.**Vector3D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector._methods.**Vector3D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector._methods.**Vector3D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector._methods.**Vector3D**(*__azumthal:* Azimuthal)
class vector._methods.**Vector3D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector._methods.**Vector3D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *Vector*, *VectorProtocolSpatial*

to_2D() → *VectorProtocolPlanar*

Alias for *vector._methods.Vector3D.to_Vector2D()*.

to_3D() → *VectorProtocolSpatial*

Alias for *vector._methods.Vector3D.to_Vector3D()*.

to_4D(***kwargs: float | None*) → *VectorProtocolLorentz*

Alias for *vector._methods.Vector3D.to_Vector4D()*.

to_Vector2D() → *VectorProtocolPlanar*

Projects this vector/these vectors onto azimuthal coordinates only.

to_Vector3D() → *VectorProtocolSpatial*

Projects this vector/these vectors onto azimuthal and longitudinal coordinates only.

If 2D, a default z component of 0 is imputed.

The longitudinal coordinate can be passed as a named argument.

to_Vector4D(**, t: float | ndarray | None = None, e: float | ndarray | None = None, E: float | ndarray | None = None, energy: float | ndarray | None = None, tau: float | ndarray | None = None, m: float | ndarray | None = None, M: float | ndarray | None = None, mass: float | ndarray | None = None*) → *VectorProtocolLorentz*

Converts a 3D vector to 4D vector.

The scalar temporal coordinate are broadcasted for NumPy and Awkward vectors. Only a single temporal coordinate should be provided.

Examples

```
>>> import vector
>>> vec = vector.VectorObject3D(x=1, y=2, z=3)
>>> vec.to_Vector4D(t=4)
VectorObject4D(x=1, y=2, z=3, t=4)
>>> vec = vector.MomentumObject3D(px=1, py=2, pz=3)
>>> vec.to_Vector4D(M=4)
MomentumObject4D(px=1, py=2, pz=3, mass=4)
```

class vector._methods.**Vector4D**(**, x: float, y: float*)

class vector._methods.**Vector4D**(**, rho: float, phi: float*)

class vector._methods.**Vector4D**(**, x: float, y: float, z: float*)

class vector._methods.**Vector4D**(**, x: float, y: float, eta: float*)

class vector._methods.**Vector4D**(**, x: float, y: float, theta: float*)

class vector._methods.**Vector4D**(**, rho: float, phi: float, z: float*)

class vector._methods.**Vector4D**(**, rho: float, phi: float, eta: float*)

class vector._methods.**Vector4D**(**, rho: float, phi: float, theta: float*)

class vector._methods.**Vector4D**(**, px: float, py: float*)

class vector._methods.**Vector4D**(**, x: float, py: float*)

class vector._methods.**Vector4D**(**, px: float, y: float*)

class vector._methods.**Vector4D**(**, pt: float, phi: float*)

class vector._methods.**Vector4D**(**, x: float, y: float, pz: float*)

class vector._methods.**Vector4D**(**, x: float, py: float, z: float*)

class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *pz: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *z: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *pz: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *z: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *pz: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *pz: float*)
class vector._methods.**Vector4D**(*, *pt: float*, *phi: float*, *z: float*)
class vector._methods.**Vector4D**(*, *pt: float*, *phi: float*, *pz: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *theta: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *theta: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *theta: float*)
class vector._methods.**Vector4D**(*, *pt: float*, *phi: float*, *theta: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *eta: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *eta: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *eta: float*)
class vector._methods.**Vector4D**(*, *pt: float*, *phi: float*, *eta: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *z: float*, *t: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *pz: float*, *t: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *z: float*, *t: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *pz: float*, *t: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *z: float*, *t: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *pz: float*, *t: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *z: float*, *t: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *pz: float*, *t: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *z: float*, *t: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *pz: float*, *t: float*)
class vector._methods.**Vector4D**(*, *pt: float*, *phi: float*, *z: float*, *t: float*)
class vector._methods.**Vector4D**(*, *pt: float*, *phi: float*, *pz: float*, *t: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *theta: float*, *t: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *theta: float*, *t: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *theta: float*, *t: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *theta: float*, *t: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *theta: float*, *t: float*)
class vector._methods.**Vector4D**(*, *pt: float*, *phi: float*, *theta: float*, *t: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *eta: float*, *t: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *eta: float*, *t: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *eta: float*, *t: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *eta: float*, *t: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *eta: float*, *t: float*)
class vector._methods.**Vector4D**(*, *pt: float*, *phi: float*, *eta: float*, *t: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *z: float*, *tau: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *pz: float*, *tau: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *z: float*, *tau: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *pz: float*, *tau: float*)

class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *z: float*, *tau: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *pz: float*, *tau: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *z: float*, *tau: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *pz: float*, *tau: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *z: float*, *tau: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *pz: float*, *tau: float*)
class vector._methods.**Vector4D**(*, *ptau: float*, *phi: float*, *z: float*, *tau: float*)
class vector._methods.**Vector4D**(*, *ptau: float*, *phi: float*, *pz: float*, *tau: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *theta: float*, *tau: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *theta: float*, *tau: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *theta: float*, *tau: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *theta: float*, *tau: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *theta: float*, *tau: float*)
class vector._methods.**Vector4D**(*, *ptau: float*, *phi: float*, *theta: float*, *tau: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *eta: float*, *tau: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *eta: float*, *tau: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *eta: float*, *tau: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *eta: float*, *tau: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *eta: float*, *tau: float*)
class vector._methods.**Vector4D**(*, *ptau: float*, *phi: float*, *eta: float*, *tau: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *z: float*, *E: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *pz: float*, *E: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *z: float*, *E: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *pz: float*, *E: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *z: float*, *E: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *pz: float*, *E: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *z: float*, *E: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *pz: float*, *E: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *z: float*, *E: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *pz: float*, *E: float*)
class vector._methods.**Vector4D**(*, *pE: float*, *phi: float*, *z: float*, *E: float*)
class vector._methods.**Vector4D**(*, *pE: float*, *phi: float*, *pz: float*, *E: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *theta: float*, *E: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *theta: float*, *E: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *theta: float*, *E: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *theta: float*, *E: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *theta: float*, *E: float*)
class vector._methods.**Vector4D**(*, *pE: float*, *phi: float*, *theta: float*, *E: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *eta: float*, *E: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *eta: float*, *E: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *eta: float*, *E: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *eta: float*, *E: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *eta: float*, *E: float*)
class vector._methods.**Vector4D**(*, *pE: float*, *phi: float*, *eta: float*, *E: float*)

class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *z: float*, *e: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *pz: float*, *e: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *z: float*, *e: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *pz: float*, *e: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *z: float*, *e: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *pz: float*, *e: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *z: float*, *e: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *pz: float*, *e: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *z: float*, *e: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *pz: float*, *e: float*)
class vector._methods.**Vector4D**(*, *pe: float*, *phi: float*, *z: float*, *e: float*)
class vector._methods.**Vector4D**(*, *pe: float*, *phi: float*, *pz: float*, *e: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *theta: float*, *e: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *theta: float*, *e: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *theta: float*, *e: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *theta: float*, *e: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *theta: float*, *e: float*)
class vector._methods.**Vector4D**(*, *pe: float*, *phi: float*, *theta: float*, *e: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *eta: float*, *e: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *eta: float*, *e: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *eta: float*, *e: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *eta: float*, *e: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *eta: float*, *e: float*)
class vector._methods.**Vector4D**(*, *pe: float*, *phi: float*, *eta: float*, *e: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *z: float*, *energy: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *pz: float*, *energy: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *z: float*, *energy: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *pz: float*, *energy: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *z: float*, *energy: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *pz: float*, *energy: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *z: float*, *energy: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *pz: float*, *energy: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *z: float*, *energy: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *pz: float*, *energy: float*)
class vector._methods.**Vector4D**(*, *pt: float*, *phi: float*, *z: float*, *energy: float*)
class vector._methods.**Vector4D**(*, *pt: float*, *phi: float*, *pz: float*, *energy: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *theta: float*, *energy: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *theta: float*, *energy: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *theta: float*, *energy: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *theta: float*, *energy: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *theta: float*, *energy: float*)
class vector._methods.**Vector4D**(*, *pt: float*, *phi: float*, *theta: float*, *energy: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *eta: float*, *energy: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *eta: float*, *energy: float*)

class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *eta: float*, *energy: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *eta: float*, *energy: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *eta: float*, *energy: float*)
class vector._methods.**Vector4D**(*, *pt: float*, *phi: float*, *eta: float*, *energy: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *z: float*, *M: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *pz: float*, *M: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *z: float*, *M: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *pz: float*, *M: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *z: float*, *M: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *pz: float*, *M: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *z: float*, *M: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *pz: float*, *M: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *z: float*, *M: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *pz: float*, *M: float*)
class vector._methods.**Vector4D**(*, *pM: float*, *phi: float*, *z: float*, *M: float*)
class vector._methods.**Vector4D**(*, *pM: float*, *phi: float*, *pz: float*, *M: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *theta: float*, *M: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *theta: float*, *M: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *theta: float*, *M: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *theta: float*, *M: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *theta: float*, *M: float*)
class vector._methods.**Vector4D**(*, *pM: float*, *phi: float*, *theta: float*, *M: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *eta: float*, *M: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *eta: float*, *M: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *eta: float*, *M: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *eta: float*, *M: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *eta: float*, *M: float*)
class vector._methods.**Vector4D**(*, *pM: float*, *phi: float*, *eta: float*, *M: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *z: float*, *m: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *pz: float*, *m: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *z: float*, *m: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *pz: float*, *m: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *z: float*, *m: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *pz: float*, *m: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *z: float*, *m: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *pz: float*, *m: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *z: float*, *m: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *pz: float*, *m: float*)
class vector._methods.**Vector4D**(*, *pm: float*, *phi: float*, *z: float*, *m: float*)
class vector._methods.**Vector4D**(*, *pm: float*, *phi: float*, *pz: float*, *m: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *theta: float*, *m: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *theta: float*, *m: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *theta: float*, *m: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *theta: float*, *m: float*)

class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *theta: float*, *m: float*)
class vector._methods.**Vector4D**(*, *pm: float*, *phi: float*, *theta: float*, *m: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *eta: float*, *m: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *eta: float*, *m: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *eta: float*, *m: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *eta: float*, *m: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *eta: float*, *m: float*)
class vector._methods.**Vector4D**(*, *pm: float*, *phi: float*, *eta: float*, *m: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *z: float*, *mass: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *pz: float*, *mass: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *z: float*, *mass: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *pz: float*, *mass: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *z: float*, *mass: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *pz: float*, *mass: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *z: float*, *mass: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *pz: float*, *mass: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *z: float*, *mass: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *pz: float*, *mass: float*)
class vector._methods.**Vector4D**(*, *pt: float*, *phi: float*, *z: float*, *mass: float*)
class vector._methods.**Vector4D**(*, *pt: float*, *phi: float*, *pz: float*, *mass: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *theta: float*, *mass: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *theta: float*, *mass: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *theta: float*, *mass: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *theta: float*, *mass: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *theta: float*, *mass: float*)
class vector._methods.**Vector4D**(*, *pt: float*, *phi: float*, *theta: float*, *mass: float*)
class vector._methods.**Vector4D**(*, *x: float*, *y: float*, *eta: float*, *mass: float*)
class vector._methods.**Vector4D**(*, *x: float*, *py: float*, *eta: float*, *mass: float*)
class vector._methods.**Vector4D**(*, *px: float*, *y: float*, *eta: float*, *mass: float*)
class vector._methods.**Vector4D**(*, *px: float*, *py: float*, *eta: float*, *mass: float*)
class vector._methods.**Vector4D**(*, *rho: float*, *phi: float*, *eta: float*, *mass: float*)
class vector._methods.**Vector4D**(*, *pt: float*, *phi: float*, *eta: float*, *mass: float*)
class vector._methods.**Vector4D**(*__azumthal:* Azimuthal)
class vector._methods.**Vector4D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal)
class vector._methods.**Vector4D**(*__azumthal:* Azimuthal, *__longitudinal:* Longitudinal, *__temporal:* Temporal)

Bases: *Vector*, *VectorProtocolLorentz*

to_2D() → *VectorProtocolPlanar*

Alias for *vector._methods.Vector4D.to_Vector2D()*.

to_3D() → *VectorProtocolSpatial*

Alias for *vector._methods.Vector4D.to_Vector3D()*.

to_4D() → *VectorProtocolLorentz*

Alias for *vector._methods.Vector4D.to_Vector4D()*.

to_Vector2D() → *VectorProtocolPlanar*

Projects this vector/these vectors onto azimuthal coordinates only.

to_Vector3D() → *VectorProtocolSpatial*

Projects this vector/these vectors onto azimuthal and longitudinal coordinates only.

If 2D, a default z component of 0 is imputed.

The longitudinal coordinate can be passed as a named argument.

to_Vector4D() → *VectorProtocolLorentz*

Projects this vector/these vectors onto azimuthal, longitudinal, and temporal coordinates.

If 3D, a default t component of 0 is imputed.

If 2D, a z component of 0 is imputed along with a default t component of 0.

The longitudinal and temporal coordinates can be passed as named arguments.

class vector._methods.**VectorProtocol**

Bases: object

lib

The module used for functions used in compute functions (such as sqrt, sin, cos). Usually numpy.

Type
module

ProjectionClass2D

The class that would result from projecting this vector onto azimuthal coordinates only.

Type
type

ProjectionClass3D

The class that would result from projecting this vector onto azimuthal and longitudinal coordinates only.

Type
type

ProjectionClass4D

The class that would result from projecting this vector onto azimuthal, longitudinal, and temporal coordinates.

Type
type

GenericClass

The most generic concrete class for this type, for vectors without momentum-synonyms.

Type
type

MomentumClass

The momentum class for this type, for vectors with momentum-synonyms.

Type
type

GenericClass: type[*VectorProtocol*]

MomentumClass: type[*VectorProtocol*]

ProjectionClass2D: type[*VectorProtocolPlanar*]

ProjectionClass3D: type[*VectorProtocolSpatial*]

ProjectionClass4D: type[*VectorProtocolLorentz*]

_wrap_result(*cls: Any*, *result: Any*, *returns: Any*, *num_vecargs: Any*) → Any

Parameters

- **result** – Value or tuple of values from a compute function.
- **returns** – Signature from a `dispatch_map`.
- **num_vecargs** (*int*) – Number of vector arguments in the function that would be treated on an equal footing (i.e. `add` has two, but `rotate_axis` has only one: the `axis` is secondary).

Wraps the raw result of a compute function as a scalar, an array of scalars, a vector, or an array of vectors.

add(*other:* VectorProtocol) → *VectorProtocol*

Sum of `self` and `other`.

This method is equivalent to the + operator.

dot(*other:* VectorProtocol) → Any

Vector dot product of `self` with `other`.

This method is equivalent to the `@` operator.

equal(*other:* VectorProtocol) → Any

Returns True if `self` is exactly equal to `other` (possibly for arrays of vectors), False otherwise.

This method is equivalent to the == operator.

Typically, you'll want to check *vector._methods.VectorProtocol.isclose()* to allow for numerical errors.

isclose(*other:* VectorProtocol, *rtol: Any = 1e-05*, *atol: Any = 1e-08*, *equal_nan: Any = False*) → Any

Returns True if `self` is approximately equal to `other` (possibly for arrays of vectors), False otherwise.

The relative tolerance (`rtol`) and absolute tolerance (`atol`) are interpreted as in `np.isclose`:

```
close_enough = abs(self - other) <= atol + rtol * abs(other)
```

property lib: Any

like(*other:* VectorProtocol) → *VectorProtocol*

Projects the vector into the geometric coordinates of the *other* vector.

Value(s) of 0 is/are imputed while transforming vector from a lower geometric dimension to a higher geometric dimension.

```
vec_4d + vec_3d.like(vec_4d)
```

For more flexibility (passing new coordinate values), see *vector._methods.Vector2D.to_Vector3D()*, *vector._methods.Vector2D.to_Vector4D()*, and *vector._methods.Vector3D.to_Vector4D()*, which can be used as:

```
vec_2d.to_Vector3D(z=3.0)
vec_2d.to_Vector4D(z=3.0, t=4.0)
vec_3d.to_Vector4D(t=4.0)
```

not_equal(*other:* VectorProtocol) → Any

Returns False if `self` is exactly equal to `other` (possibly for arrays of vectors), True otherwise.

This method is equivalent to the `!=` operator.

Typically, you'll want to check *vector._methods.VectorProtocol.isclose()* to allow for numerical errors.

scale(*factor: Any*) → SameVectorType

Returns vector(s) scaled by a `factor`, changing the length(s) but not the direction(s).

This method is equivalent to the `*` operator.

subtract(*other:* VectorProtocol) → *VectorProtocol*

Difference of `self` minus `other`.

This method is equivalent to the `-` operator.

to_2D() → *VectorProtocolPlanar*

Projects this vector/these vectors onto azimuthal coordinates only.

Alias for *vector._methods.VectorProtocol.to_Vector2D()*.

to_3D() → *VectorProtocolSpatial*

Projects this vector/these vectors onto azimuthal and longitudinal coordinates only.

If 2D, a default z component of 0 is imputed.

The longitudinal coordinate can be passed as a named argument.

Alias for *vector._methods.VectorProtocol.to_Vector3D()*.

to_4D() → *VectorProtocolLorentz*

Projects this vector/these vectors onto azimuthal, longitudinal, and temporal coordinates.

If 3D, a default t component of 0 is imputed.

If 2D, a z component of 0 is imputed along with a default t component of 0.

The longitudinal and temporal coordinates can be passed as named arguments.

Alias for *vector._methods.VectorProtocol.to_Vector4D()*.

to_Vector2D() → *VectorProtocolPlanar*

Projects this vector/these vectors onto azimuthal coordinates only.

to_Vector3D() → *VectorProtocolSpatial*

Projects this vector/these vectors onto azimuthal and longitudinal coordinates only.

If 2D, a default z component of 0 is imputed.

The longitudinal coordinate can be passed as a named argument.

to_Vector4D() → *VectorProtocolLorentz*

Projects this vector/these vectors onto azimuthal, longitudinal, and temporal coordinates.

If 3D, a default t component of 0 is imputed.

If 2D, a z component of 0 is imputed along with a default t component of 0.

The longitudinal and temporal coordinates can be passed as named arguments.

to_ptphi() → *VectorProtocolPlanar*

Converts to pt-ϕ coordinates, possibly eliminating dimensions with a projection.

to_ptphieta() → *VectorProtocolSpatial*

Converts to pt-ϕ-η coordinates, possibly eliminating or imputing dimensions with a projection.

The eta coordinate can be passed as a named argument.

to_ptphietaenergy() → *VectorProtocolLorentz*

Converts to pt-ϕ-η-$energy$ coordinates, possibly imputing dimensions with a projection.

The eta and $energy$ coordinates can be passed as a named argument.

to_ptphietamass() → *VectorProtocolLorentz*

Converts to pt-ϕ-θ-$mass$ coordinates, possibly imputing dimensions with a projection.

The eta and $mass$ coordinates can be passed as a named argument.

to_ptphipz() → *VectorProtocolSpatial*

Converts to pt-ϕ-pz coordinates, possibly eliminating or imputing dimensions with a projection.

The pz coordinate can be passed as a named argument.

to_ptphipzenergy() → *VectorProtocolLorentz*

Converts to pt-ϕ-pz-$energy$ coordinates, possibly imputing dimensions with a projection.

The pz and $energy$ coordinates can be passed as a named argument.

to_ptphipzmass() → *VectorProtocolLorentz*

Converts to pt-ϕ-pz-$mass$ coordinates, possibly imputing dimensions with a projection.

The pz and $mass$ coordinates can be passed as a named argument.

to_ptphitheta() → *VectorProtocolSpatial*

Converts to pt-ϕ-θ coordinates, possibly eliminating or imputing dimensions with a projection.

The $theta$ coordinate can be passed as a named argument.

to_ptphithetaenergy() → *VectorProtocolLorentz*

Converts to pt-ϕ-θ-$energy$ coordinates, possibly imputing dimensions with a projection.

The $theta$ and $energy$ coordinates can be passed as a named argument.

to_ptphithetamass() → *VectorProtocolLorentz*

Converts to pt-ϕ-θ-$mass$ coordinates, possibly imputing dimensions with a projection.

The $theta$ and $mass$ coordinates can be passed as a named argument.

to_pxpy() → *VectorProtocolPlanar*

Converts to px-py coordinates, possibly eliminating dimensions with a projection.

to_pxpyeta() → *VectorProtocolSpatial*

Converts to px-py-η coordinates, possibly eliminating or imputing dimensions with a projection.

The eta coordinate can be passed as a named argument.

to_pxpyetaenergy() → *VectorProtocolLorentz*

Converts to px-py-η-$energy$ coordinates, possibly imputing dimensions with a projection.

The eta and $energy$ coordinates can be passed as a named argument.

to_pxpyetamass() → *VectorProtocolLorentz*

Converts to px-py-η-$mass$ coordinates, possibly imputing dimensions with a projection.

The eta and $mass$ coordinates can be passed as a named argument.

to_pxpypz() → *VectorProtocolSpatial*

Converts to px-py-pz coordinates, possibly eliminating or imputing dimensions with a projection.

The pz coordinate can be passed as a named argument.

to_pxpypzenergy() → *VectorProtocolLorentz*

Converts to px-py-pz-$energy$ coordinates, possibly imputing dimensions with a projection.

The pz and $energy$ coordinates can be passed as a named argument.

to_pxpypzmass() → *VectorProtocolLorentz*

Converts to px-py-pz-$mass$ coordinates, possibly imputing dimensions with a projection.

The pz and $mass$ coordinates can be passed as a named argument.

to_pxpytheta() → *VectorProtocolSpatial*

Converts to px-py-θ coordinates, possibly eliminating or imputing dimensions with a projection.

The $theta$ coordinate can be passed as a named argument.

to_pxpythetaenergy() → *VectorProtocolLorentz*

Converts to px-py-θ-$energy$ coordinates, possibly imputing dimensions with a projection.

The $theta$ and $energy$ coordinates can be passed as a named argument.

to_pxpythetamass() → *VectorProtocolLorentz*

Converts to px-py-θ-$energy$ coordinates, possibly imputing dimensions with a projection.

The $theta$ and $mass$ coordinates can be passed as a named argument.

to_rhophi() → *VectorProtocolPlanar*

Converts to ρ-ϕ coordinates, possibly eliminating dimensions with a projection.

to_rhophieta() → *VectorProtocolSpatial*

Converts to ρ-ϕ-η coordinates, possibly eliminating or imputing dimensions with a projection.

The eta coordinate can be passed as a named argument.

to_rhophietat() → *VectorProtocolLorentz*

Converts to ρ-ϕ-η-t coordinates, possibly imputing dimensions with a projection.

The eta and t coordinates can be passed as a named argument.

to_rhophietatau() → *VectorProtocolLorentz*

Converts to ρ-ϕ-η-τ coordinates, possibly imputing dimensions with a projection.

The eta and tau coordinates can be passed as a named argument.

to_rhophitheta() → *VectorProtocolSpatial*

Converts to ρ-ϕ-θ coordinates, possibly eliminating or imputing dimensions with a projection.

The $theta$ coordinate can be passed as a named argument.

to_rhophithetat() → *VectorProtocolLorentz*

Converts to ρ-ϕ-θ-t coordinates, possibly imputing dimensions with a projection.

The $theta$ and t coordinates can be passed as a named argument.

to_rhophithetatau() → *VectorProtocolLorentz*

Converts to ρ-ϕ-θ-τ coordinates, possibly imputing dimensions with a projection.

The $theta$ and tau coordinates can be passed as a named argument.

to_rhophiz() → *VectorProtocolSpatial*

Converts to ρ-ϕ-z coordinates, possibly eliminating or imputing dimensions with a projection.

The z coordinate can be passed as a named argument.

to_rhophizt() → *VectorProtocolLorentz*

Converts to ρ-ϕ-z-t coordinates, possibly imputing dimensions with a projection.

The z and t coordinates can be passed as a named argument.

to_rhophiztau() → *VectorProtocolLorentz*

Converts to ρ-ϕ-z-τ coordinates, possibly imputing dimensions with a projection.

The z and tau coordinates can be passed as a named argument.

to_xy() → *VectorProtocolPlanar*

Converts to x-y coordinates, possibly eliminating dimensions with a projection.

to_xyeta() → *VectorProtocolSpatial*

Converts to x-y-η coordinates, possibly eliminating or imputing dimensions with a projection.

The eta coordinate can be passed as a named argument.

to_xyetat() → *VectorProtocolLorentz*

Converts to x-y-η-t coordinates, possibly imputing dimensions with a projection.

The eta and t coordinates can be passed as a named argument.

to_xyetatau() → *VectorProtocolLorentz*

Converts to x-y-η-τ coordinates, possibly imputing dimensions with a projection.

The eta and tau coordinates can be passed as a named argument.

to_xytheta() → *VectorProtocolSpatial*

Converts to x-y-θ coordinates, possibly eliminating or imputing dimensions with a projection.

The $theta$ coordinate can be passed as a named argument.

to_xythetat() → *VectorProtocolLorentz*

Converts to x-y-θ-t coordinates, possibly imputing dimensions with a projection.

The $theta$ and t coordinates can be passed as a named argument.

to_xythetatau() → *VectorProtocolLorentz*

Converts to x-y-θ-τ coordinates, possibly imputing dimensions with a projection.

The $theta$ and tau coordinates can be passed as a named argument.

to_xyz() → *VectorProtocolSpatial*

Converts to x-y-z coordinates, possibly eliminating or imputing dimensions with a projection.

The z coordinate can be passed as a named argument.

to_xyzt() → *VectorProtocolLorentz*

Converts to x-y-z-t coordinates, possibly imputing dimensions with a projection.

The z and t coordinates can be passed as a named argument.

to_xyztau() → *VectorProtocolLorentz*

Converts to x-y-z-τ coordinates, possibly imputing dimensions with a projection.

The z and tau coordinates can be passed as a named argument.

unit() → SameVectorType

Returns vector(s) normalized to unit length, which is *rho == 1* for 2D vectors, *mag == 1* for 3D vectors, and *tau == 1* for 4D vectors.

class vector._methods.**VectorProtocolLorentz**

Bases: *VectorProtocolSpatial*

property beta: Any

The speed(s) of the Lorentz vector or array of vectors, in which lightlike vectors have `beta == 1`.

boost(*booster:* VectorProtocolSpatial | VectorProtocolLorentz) → SameVectorType

Boosts the vector or array of vectors using the 3D or 4D `booster`.

If `booster` is 3D, it is interpreted as a velocity (in which lightlike velocities have `mag == 1`) and *vector._methods.VectorProtocolLorentz.boost_beta3()* is called.

If `booster` is 4D, it is interpreted as a Lorentz vector and *vector._methods.VectorProtocolLorentz.boost_p4()* is called.

Note that `v.boost(v)` does not boost into the center-of-mass (CM) frame of `v`; it boosts *away* from its CM frame. Neither does `v.boost(-v)`, since that negates the time component of `v` as well.

To boost to the center-of-mass frame of a vector `v`, use *vector._methods.VectorProtocolLorentz.boostCM_of()*. For instance, `v.boostCM_of(v)` is guaranteed to have spatial components close to zero and a temporal component close to `v.tau`.

boostCM_of(*booster:* VectorProtocolSpatial | VectorProtocolLorentz) → SameVectorType

Boosts the vector or array of vectors to the center-of-mass (CM) frame of the 3D or 4D `booster`.

If `booster` is 3D, it is interpreted as a velocity (in which lightlike velocities have `mag == 1`) and *vector._methods.VectorProtocolLorentz.boostCM_of_beta3()* is called.

If `booster` is 4D, it is interpreted as a Lorentz vector and *vector._methods.VectorProtocolLorentz.boostCM_of_p4()* is called.

Note that `v.boostCM_of(v)` is guaranteed to have spatial components close to zero and a temporal component close to `v.tau`.

boostCM_of_beta3(*beta3:* VectorProtocolSpatial) → SameVectorType

Boosts the vector or array of vectors to the center-of-mass (CM) frame of the 3D velocity or array of velocity vectors `beta3`.

Note that `v.boostCM_of_beta3(v.to_beta3())` is guaranteed to have spatial components close to zero and a temporal component close to `v.tau`.

boostCM_of_p4(*p4:* VectorProtocolLorentz) → SameVectorType

Boosts the vector or array of vectors to the center-of-mass (CM) frame of the 4D vector or array of vectors `p4`.

This function is equivalent to but more numerically stable than

```
boostCM_of_beta3(p4.to_beta3())
```

Note that `v.boostCM_of_p4(v)` is guaranteed to have spatial components close to zero and a temporal component close to `v.tau`.

boostX(*beta: Any | None = None*, *gamma: Any | None = None*) → SameVectorType

Boosts the vector or array of vectors in the x direction by a speed `beta` (in which lightlike boosts have `beta == 1`) or time dilation/length contraction factor `gamma`.

Either `beta` xor `gamma` must be specified, not both or neither.

If `beta` or `gamma` is negative, it is taken as a boost in the $-x$ direction.

boostY(*beta: Any | None = None*, *gamma: Any | None = None*) → SameVectorType

Boosts the vector or array of vectors in the y direction by a speed `beta` (in which lightlike boosts have `beta == 1`) or time dilation/length contraction factor `gamma`.

Either `beta` xor `gamma` must be specified, not both or neither.

If `beta` or `gamma` is negative, it is taken as a boost in the $-y$ direction.

boostZ(*beta: Any | None = None*, *gamma: Any | None = None*) → SameVectorType

Boosts the vector or array of vectors in the z direction by a speed `beta` (in which lightlike boosts have `beta == 1`) or time dilation/length contraction factor `gamma`.

Either `beta` xor `gamma` must be specified, not both or neither.

If `beta` or `gamma` is negative, it is taken as a boost in the $-z$ direction.

boost_beta3(*beta3:* VectorProtocolSpatial) → SameVectorType

Boosts the vector or array of vectors in a direction and magnitude given by the 3D velocity or array of velocity vectors `beta3`.

Note that `v.boost_beta3(v.to_beta3())` does not boost into the center-of-mass (CM) frame of `v`; it boosts *away* from its CM frame. Neither does `v.boost_beta3((-v).to_beta3())`, since that negates the time component of `v` as well. On the other hand, `v.boost_beta3(-(v.to_beta3()))` *would* boost to the center-of-mass frame.

However, there's a function for that: *vector._methods.VectorProtocolLorentz.boostCM_of_beta3()* is explicit about boosting to a center-of-mass (CM) frame and it handles the negative sign for you: `v.boostCM_of_beta3(v.to_beta3())` is guaranteed to have spatial components close to zero and a temporal component close to `v.tau`.

boost_p4(*p4:* VectorProtocolLorentz) → SameVectorType

Boosts the vector or array of vectors in a direction and magnitude given by the 4D vector or array of vectors `p4`.

This function is equivalent to but more numerically stable than

```
boost_beta3(p4.to_beta3())
```

where *vector._methods.VectorProtocolLorentz.to_beta3()* converts a 4D Lorentz vector into a 3D velocity (in which lightlike velocities have `mag == 1`).

Note that `v.boost_p4(v)` does not boost into the center-of-mass (CM) frame of `v`; it boosts *away* from its CM frame. Neither does `v.boost_p4(-v)`, since that negates the time component of `v` as well.

To boost to the center-of-mass frame of a vector `v`, use *vector._methods.VectorProtocolLorentz.boostCM_of_p4()*. For instance, `v.boostCM_of_p4(v)` is guaranteed to have spatial components close to zero and a temporal component close to `v.tau`.

deltaRapidityPhi(*other:* VectorProtocolLorentz) → Any

Sum in quadrature of *vector._methods.VectorProtocolPlanar.deltaphi()* and the difference in *vector._methods.VectorProtocolLorentz.rapidity* of the two vectors:

$$\Delta R_{\text{rapidity}} = \sqrt{\Delta\phi^2 + \Delta\text{rapidity}^2}$$

deltaRapidityPhi2(*other:* VectorProtocolLorentz) → Any

Square of the sum in quadrature of *vector._methods.VectorProtocolPlanar.deltaphi()* and the difference in *vector._methods.VectorProtocolLorentz.rapidity* of the two vectors:

$$\Delta R_{\text{rapidity}} = \Delta\phi^2 + \Delta\text{rapidity}^2$$

property gamma: Any

The time dilation/length contraction factor(s) of the Lorentz vector or array of vectors: t/τ.

is_lightlike(*tolerance: Any = 1e-05*) → Any

Returns True if the vector or a vector in the array is pointing in a lightlike direction, `t**2 == mag**2`, False otherwise.

The `tolerance` is in units of `t` and `mag`. Note that

- the default `tolerance` for *vector._methods.VectorProtocolLorentz.is_timelike()* is `0`
- the default `tolerance` for *vector._methods.VectorProtocolLorentz.is_spacelike()* is `0`
- the default `tolerance` for *vector._methods.VectorProtocolLorentz.is_lightlike()* is `1e-5`

If you want to use these methods to divide space-time into non-overlapping regions (the light-cone), use the same `tolerance` for each.

is_spacelike(*tolerance: Any = 0*) → Any

Returns True if the vector or a vector in the array is pointing in a spacelike direction, `t**2 < mag**2`, False otherwise.

The `tolerance` is in units of `t` and `mag`. Note that

- the default `tolerance` for *vector._methods.VectorProtocolLorentz.is_timelike()* is `0`
- the default `tolerance` for *vector._methods.VectorProtocolLorentz.is_spacelike()* is `0`
- the default `tolerance` for *vector._methods.VectorProtocolLorentz.is_lightlike()* is `1e-5`

If you want to use these methods to divide space-time into non-overlapping regions (the light-cone), use the same `tolerance` for each.

is_timelike(*tolerance: Any = 0*) → Any

Returns True if the vector or a vector in the array is pointing in a timelike direction, `t**2 > mag**2`, False otherwise.

The `tolerance` is in units of `t` and `mag`. Note that

- the default `tolerance` for *vector._methods.VectorProtocolLorentz.is_timelike()* is `0`
- the default `tolerance` for *vector._methods.VectorProtocolLorentz.is_spacelike()* is `0`
- the default `tolerance` for *vector._methods.VectorProtocolLorentz.is_lightlike()* is `1e-5`

If you want to use these methods to divide space-time into non-overlapping regions (the light-cone), use the same `tolerance` for each.

property neg4D: SameVectorType

Same as multiplying by -1.

property rapidity: Any

The rapidity relative to the longitudinal axis of the Lorentz vector or array of vectors.

```
0.5 * log((t + z) / (t - z))
```

scale4D(*factor: Any*) → SameVectorType

Same as `scale`.

property t: Any

The Cartesian t (time) coordinate of the vector or every vector in the array.

If t is derived from τ, it is not allowed to be `NaN`.

```
t = sqrt(max(copysign(tau**2, tau) + mag**2, 0))
```

property t2: Any

The Cartesian t (time) coordinate squared of the vector or every vector in the array.

If t^2 is derived from τ, it is not allowed to be negative.

```
t2 = max(copysign(tau**2, tau) + mag**2, 0)
```

property tau: Any

The Lorentz magnitude τ (proper time) of the vector or every vector in the array.

If τ is derived from t, spacelike vectors are represented by negative proper times.

```
tau = copysign(sqrt(abs(t**2 - mag**2)), t**2 - mag**2)
```

property tau2: Any

The Lorentz magnitude τ (proper time) squared of the vector or every vector in the array.

```
tau2 = t**2 - mag**2
```

property temporal: *Temporal*

Container of temporal coordinates, for use in dispatching to compute functions or to identify coordinate system with `isinstance`.

to_beta3() → *VectorProtocolSpatial*

Converts the 4D Lorentz vector or array of vectors into a 3D velocity vector or array of vectors, in which lightlike velocities have `mag == 1`.

transform4D(*obj:* TransformProtocol4D) → SameVectorType

Arbitrarily transforms the vector(s) by

```
obj["xx"] obj["xy"] obj["xz"] obj["xt"]
obj["yx"] obj["yy"] obj["yz"] obj["yt"]
obj["zx"] obj["zy"] obj["zz"] obj["zt"]
obj["tx"] obj["ty"] obj["tz"] obj["tt"]
```

There is no restriction on the type of `obj`; it just has to provide those components (which can be arrays if the vectors are in an array).

class vector._methods.VectorProtocolPlanar

Bases: *VectorProtocol*

property azimuthal: *Azimuthal*

Container of azimuthal coordinates, for use in dispatching to compute functions or to identify coordinate system with `isinstance`.

deltaphi(*other:* VectorProtocol) → Any

Signed difference in ϕ of `self` minus `other` (in radians).

is_antiparallel(*other:* VectorProtocol, *tolerance: Any = 1e-05*) → Any

Returns True if `self` and `other` are pointing in opposite directions (i.e. dot product is nearly `-abs(self) * abs(other)`).

The `tolerance` is measured in units of $\cos(\Delta\alpha)$ where $\Delta\alpha$ is `self.deltaangle(other)`.

is_parallel(*other:* VectorProtocol, *tolerance: Any = 1e-05*) → Any

Returns True if `self` and `other` are pointing in the same direction (i.e. not "antiparallel"; dot product is nearly `abs(self) * abs(other)`).

The `tolerance` is measured in units of $\cos(\Delta\alpha)$ where $\Delta\alpha$ is `self.deltaangle(other)`.

is_perpendicular(*other:* VectorProtocol, *tolerance: Any = 1e-05*) → Any

Returns True if `self` and `other` are pointing in perpendicular directions (i.e. dot product is nearly `0`).

The `tolerance` is measured in units of $\cos(\Delta\alpha)$ where $\Delta\alpha$ is `self.deltaangle(other)`.

property neg2D: SameVectorType

Returns vector(s) with the 2D part negated, not affecting any longitudinal or temporal parts.

property phi: Any

The polar ϕ coordinate of the vector or every vector in the array (in radians, always between $-\pi$ and π).

property rho: Any

The polar ρ coordinate of the vector or every vector in the array.

This is also the magnitude of the 2D azimuthal part of the vector (not including any longitudinal or temporal parts).

property rho2: Any

The polar ρ coordinate squared of the vector or every vector in the array.

rotateZ(*angle: Any*) → SameVectorType

Rotates the vector(s) by a given `angle` (in radians) around the longitudinal axis.

Note that the `angle` can be an array with the same length as the vectors, if the vectors are in an array.

scale2D(*factor: Any*) → SameVectorType

Returns vector(s) with the 2D part scaled by a `factor`, not affecting any longitudinal or temporal parts.

transform2D(*obj:* TransformProtocol2D) → SameVectorType

Arbitrarily transforms the vector(s) by

```
obj["xx"] obj["xy"]
obj["yx"] obj["yy"]
```

leaving any longitudinal or temporal coordinates unchanged. There is no restriction on the type of `obj`; it just has to provide those components (which can be arrays if the vectors are in an array).

property x: Any

The Cartesian x coordinate of the vector or every vector in the array.

property y: Any

The Cartesian y coordinate of the vector or every vector in the array.

class vector._methods.**VectorProtocolSpatial**

Bases: *VectorProtocolPlanar*

property costheta: Any

The $\cos\theta$ coordinate of the vector or every vector in the array (has the same sign as z).

property cottheta: Any

The $\cot\theta$ coordinate of the vector or every vector in the array (has the same sign as z).

cross(*other:* VectorProtocolSpatial) → *VectorProtocolSpatial*

The 3D cross-product of `self` with `other`.

Even if `self` or `other` is 4D, the resulting vector(s) is/are 3D.

deltaR(*other:* VectorProtocolSpatial | VectorProtocolLorentz) → Any

Sum in quadrature of *vector._methods.VectorProtocolPlanar.deltaphi()* and *vector._methods.VectorProtocolSpatial.deltaeta()*:

$$\Delta R = \sqrt{\Delta\phi^2 + \Delta\eta^2}$$

deltaR2(*other:* VectorProtocolSpatial | VectorProtocolLorentz) → Any

Square of the sum in quadrature of *vector._methods.VectorProtocolPlanar.deltaphi()* and *vector._methods.VectorProtocolSpatial.deltaeta()*:

$$\Delta R^2 = \Delta\phi^2 + \Delta\eta^2$$

deltaangle(*other:* VectorProtocolSpatial | VectorProtocolLorentz) → Any

Angle in 3D space between `self` and `other`, which is always positive, between 0 and π.

deltaeta(*other:* VectorProtocolSpatial | VectorProtocolLorentz) → Any

Signed difference in η of `self` minus `other`.

property eta: Any

The pseudorapidity η coordinate of the vector or every vector in the array (in radians, always between 0 ($+z$) and π ($-z$)).

is_antiparallel(*other:* VectorProtocol, *tolerance: Any = 1e-05*) → Any

Returns True if `self` and `other` are pointing in opposite directions (i.e. dot product is nearly `-abs(self) * abs(other)`).

The `tolerance` is measured in units of $\cos(\Delta\alpha)$ where $\Delta\alpha$ is `self.deltaangle(other)`.

is_parallel(*other:* VectorProtocol, *tolerance: Any = 1e-05*) → Any

Returns True if `self` and `other` are pointing in the same direction (i.e. not "antiparallel"; dot product is nearly `abs(self) * abs(other)`).

The `tolerance` is measured in units of $\cos(\Delta\alpha)$ where $\Delta\alpha$ is `self.deltaangle(other)`.

is_perpendicular(*other:* VectorProtocol, *tolerance: Any = 1e-05*) → Any

Returns True if `self` and `other` are pointing in perpendicular directions (i.e. dot product is nearly `0`).

The `tolerance` is measured in units of $\cos(\Delta\alpha)$ where $\Delta\alpha$ is `self.deltaangle(other)`.

property longitudinal: *Longitudinal*

Container of longitudinal coordinates, for use in dispatching to compute functions or to identify coordinate system with `isinstance`.

property mag: Any

The magnitude of the vector(s) in 3D (not including any temporal parts).

property mag2: Any

The magnitude-squared of the vector(s) in 3D (not including any temporal parts).

property neg3D: SameVectorType

Returns vector(s) with the 3D part negated, not affecting any longitudinal or temporal parts.

rotateX(*angle: Any*) → SameVectorType

Rotates the vector(s) by a given `angle` (in radians) around the x axis.

Note that the `angle` can be an array with the same length as the vectors, if the vectors are in an array.

rotateY(*angle: Any*) → SameVectorType

Rotates the vector(s) by a given `angle` (in radians) around the y axis.

Note that the `angle` can be an array with the same length as the vectors, if the vectors are in an array.

rotate_axis(*axis:* VectorProtocolSpatial, *angle: Any*) → SameVectorType

Rotates the vector(s) by a given `angle` (in radians) around the axis indicated by another vector, `axis`. The magnitude of `axis` is ignored.

Note that the `axis` and `angle` can be arrays with the same length as the vectors, if the vectors are in an array.

rotate_euler(*phi: Any*, *theta: Any*, *psi: Any*, *order: str = 'zxz'*) → SameVectorType

Rotates the vector(s) by three given angles: `phi`, `theta`, and `psi` (in radians). The `order` string determines which axis each rotation is applied around:

- `"zxz"`, `"xyx"`, `"yzy"`, `"zyz"`, `"xzx"`, and `"yxy"` are proper Euler angles
- `"zxz"`, `"xyx"`, `"yzy"`, `"zyz"`, `"xzx"`, and `"yxy"` are Tait-Bryan angles (see *vector._methods.VectorProtocolSpatial.rotate_nautical()*)

The names `phi`, `theta`, and `psi` agree with Wikipedia's terminology, and both the names and order agree with ROOT's Math::EulerAngles. The default `order = "zxz"` is also ROOT's convention.

Note that the angles can be arrays with the same lengths as the vectors, if the vectors are in an array.

rotate_nautical(*yaw: Any*, *pitch: Any*, *roll: Any*) → SameVectorType

Rotates the vector(s) by three given angles: `yaw`, `pitch`, and `roll` (in radians). These are Tait-Bryan angles often used for boats and planes (see this lesson and this lesson).

This function is entirely equivalent to

```
rotate_euler(roll, pitch, yaw, order="zyx")
```

Note that the angles can be arrays with the same lengths as the vectors, if the vectors are in an array.

rotate_quaternion(*u: Any*, *i: Any*, *j: Any*, *k: Any*) → SameVectorType

Rotates the vector(s) by four angles as quaternion coefficients (in radians). Four angles are sometimes preferred over three because the latter has a pathology known as "gimbal lock."

This function follows the same conventions as ROOT's Math::Quaternion.

Note that the angles can be arrays with the same lengths as the vectors, if the vectors are in an array.

scale3D(*factor: Any*) → SameVectorType

Returns vector(s) with the 3D part scaled by a `factor`, not affecting any longitudinal or temporal parts.

property theta: Any

The spherical θ coordinate (polar angle) of the vector or every vector in the array (in radians, always between 0 ($+z$) and π ($-z$)).

transform3D(*obj:* TransformProtocol3D) → SameVectorType

Arbitrarily transforms the vector(s) by

```
obj["xx"] obj["xy"] obj["xz"]
obj["yx"] obj["yy"] obj["yz"]
obj["zx"] obj["zy"] obj["zz"]
```

leaving any temporal coordinate unchanged. There is no restriction on the type of `obj`; it just has to provide those components (which can be arrays if the vectors are in an array).

property z: Any

The Cartesian z coordinate of the vector or every vector in the array.

vector._methods.**_aztype**(*obj:* VectorProtocolPlanar) → type[*Coordinates*]

Determines the Azimuthal type of a vector for use in looking up a dispatched function.

vector._methods.**_check_instance**(*any_or_all: Callable[[Iterable[bool]], bool]*, *objects: tuple[*VectorProtocol*, ...]*, *clas: type[*VectorProtocol*]*) → bool

vector._methods.**_compute_module_of**(*one:* VectorProtocol, *two:* VectorProtocol, *nontemporal: bool = False*) → Any

Determines which compute module to use for functions of two vectors (the one with minimum dimension).

If `nontemporal`, use a spatial module even if both vectors are 4D.

vector._methods.**_flavor_of**(**objects:* VectorProtocol) → type[*VectorProtocol*]

Determines the flavor of the output of a dispatched function, where "flavor" is generic vs momentum.

vector._methods.**_from_signature**(*name: str*, *dispatch_map: dict[Any, Any]*, *signature: tuple[Any, ...]*) → tuple[Any, ...]

Gets a function and its return type from a `dispatch_map` and the `signature` to search for (complaining if none is found).

vector._methods.**_get_handler_index**(*obj:* VectorProtocol) → int

Returns the index of the first valid handler checking the list of parent classes

vector._methods.**_handler_of**(**objects:* VectorProtocol) → *VectorProtocol*

Determines which vector should wrap the output of a dispatched function.

Awkward Arrays have higher priority than NumPy arrays, which have higher priority than Python objects, which has the effect of "promoting" Python objects to NumPy arrays to Awkward Arrays whenever two are used in the same formula.

vector._methods._lib_of(**objects:* VectorProtocol) → Any

Determines the lib of a vector or set of vectors, complaining if they're incompatible.

vector._methods._ltype(*obj:* VectorProtocolSpatial) → type[*Coordinates*]

Determines the Longitudinal type of a vector for use in looking up a dispatched function.

vector._methods._maybe_same_dimension_error(*v1:* VectorProtocol, *v2:* VectorProtocol, *operation: str*) → None

Raises an error if the vectors are not of the same dimension.

vector._methods._ttype(*obj:* VectorProtocolLorentz) → type[*Coordinates*]

Determines the Temporal type of a vector for use in looking up a dispatched function.

vector._methods.dim(*v:* VectorProtocol) → int

Returns the number of dimensions in a vector: 2, 3, or 4.

vector._typeutils module

vector._typeutils.BoolCollection

alias of Any

vector._typeutils.FloatArray

alias of ndarray

class vector._typeutils.Protocol

Bases: Generic

Base class for protocol classes.

Protocol classes are defined as:

```
class Proto(Protocol):
    def meth(self) -> int:
        ...
```

Such classes are primarily used with static type checkers that recognize structural subtyping (static duck-typing).

For example:

```
class C:
    def meth(self) -> int:
        return 0

def func(x: Proto) -> int:
    return x.meth()

func(C())  # Passes static type check
```

See PEP 544 for details. Protocol classes decorated with @typing.runtime_checkable act as simple-minded runtime protocols that check only the presence of given attributes, ignoring their type signatures. Protocol classes can be generic, they are defined as:

```
class GenProto[T](Protocol):
    def meth(self) -> T:
        ...
```

_abc_impl = <_abc._abc_data object>

_is_protocol = True

_is_runtime_protocol = False

vector._typeutils.ScalarCollection
alias of Any

class vector._typeutils.TransformProtocol2D
Bases: TypedDict

xx: Any

xy: Any

yx: Any

yy: Any

class vector._typeutils.TransformProtocol3D
Bases: TypedDict

xx: Any

xy: Any

xz: Any

yx: Any

yy: Any

yz: Any

zx: Any

zy: Any

zz: Any

class vector._typeutils.TransformProtocol4D
Bases: TypedDict

tt: Any

tx: Any

ty: Any

tz: Any

xt: Any

xx: Any

xy: Any

xz: Any

yt: Any

yx: Any

yy: Any

yz: Any

zt: Any

zx: Any

zy: Any

zz: Any

PYTHON MODULE INDEX

V

INDEX

Symbols

A

B

C

D

E

F

G

I

L

M

N

O

P

R

S

T

U

V

X

Y

Z