

---

# **IBCAO\_py Documentation**

***Release 1.1***

**Gaute Hope**

May 23, 2017



<b>1</b>	<b>ibcao</b>	<b>1</b>
1.1	ibcao package . . . . .	1
<b>2</b>	<b>Indices and tables</b>	<b>5</b>
	<b>Python Module Index</b>	<b>7</b>



## ibcao package

### Submodules

#### ibcao.ibcao module

**class** `ibcao.ibcao.IBCAO` (*ibcao\_grd\_file*='home/docs/checkouts/readthedocs.org/user\_builds/ibcao-py/checkouts/latest/ibcao/IBCAO\_V3\_500m\_RR.grd')

A class for setting up a matplotlib / cartopy instance of the IBCAO. The IBCAO grd file can be retrieved from:

<http://www.ngdc.noaa.gov/mgg/bathymetry/arctic/ibcaoversion3.html>

Download file: [http://www.ngdc.noaa.gov/mgg/bathymetry/arctic/grids/version3\\_0/IBCAO\\_V3\\_500m\\_RR.grd.gz](http://www.ngdc.noaa.gov/mgg/bathymetry/arctic/grids/version3_0/IBCAO_V3_500m_RR.grd.gz)

Specification can be found in file: IBCAO\_V3\_README.txt from:

[http://www.ngdc.noaa.gov/mgg/bathymetry/arctic/grids/version3\\_0/](http://www.ngdc.noaa.gov/mgg/bathymetry/arctic/grids/version3_0/)

The specifications are hardcoded into this class, that means that the class has to be updated for new versions of the IBCAO.

The plot is set up to use the same projection as the map, Polar Stereographic, as defined in the IBCAO readme with the WGS84 datum. While working in this projection it is not necessary to transform the IBCAO image (which for the full version of the IBCAO is significant).

#### **Colormap()**

Returns a discrete colormap and norm based on the official IBCAO colormap.

**Returns:** (cmap, norm): A tuple with the colormap and the norm

Example:

```
>>> (cmap, norm) = self.Colormap ()
>>> cm = ax.pcolormesh (x, y, z, cmap = cmap, norm = norm)
>>> plt.colorbar (cm)
```

**REFERENCE** = 'Jakobsson, M., L. A. Mayer, B. Coakley, J. A. Dowdeswell, S. Forbes, B. Fridman, H. Hodnesdal, R. Noor

**VERSION** = '3.0'

#### **close()**

Closes the map file. The map is memorymapped, so this will cause a warning unless all references to the map have been removed.

**geod**

Return a *pyproj.Geoid* set up with the WGS84 ellipsoid.

**get\_cartopy()**

Deprecated: Use *ups*.

Returns a Cartopy instance set up for the IBCAO UPS variant. A separate instance is used internally.

**grid** (*div=1*)

Create position grid for IBCAO

**Args:**

**div:** Skip every div point (1 include all, default) corresponds to *div* in *template()*.

**Returns:** (x, y): A tuple with x and y grid for z.

**interp\_depth** (*x, y*)

Interpolate depth at *x* and *y* using *scipy.interpolate.RectBivariateSpline*.

This method is more accurate but slower than *map\_depth*.

The interpolation function is cached, so later interpolations should be faster.

**Args:** *x*: (1D array) coordinates (longitude) in meters on UPS *y*: (1D array) coordinates (latitude) in meters on UPS

points outside the map are set to *np.nan*.

**Returns:** *z*: depths along *x* and *y*.

**map\_depth** (*x, y, order=3*)

Map coordinates *x* and *y* onto *z* in order to retrieve depth using *scipy.ndimage.map\_coordinates*.

**Args:** *x*: (1D array) coordinates (longitude) in meters on UPS *y*: (1D array) coordinates (latitude) in meters on UPS

points outside the map are set to *np.nan*.

**Returns:** *z*: depths along *x* and *y*.

Example: Get points along great circle:

```
>>> start = [10, 78]      # positions in degrees
>>> end   = [-18, 76]
>>>
>>> ## Sample a great circle between the two points using the pyproj.Geod set up by the IBCAO
>>> i = IBCAO ()
>>> gc = np.array(i.get_geod ().npts (start[0], start[1], end[0], end[1], 100))
>>>
>>> ## Interpolate the depth along the great circle
>>> gc_xy = i.ups.transform_points (i.g, gc[:,0], gc[:,1]) # convert to UPS coordinates
>>> depth = i.map_depth (gc_xy[:,0], gc_xy[:,1])
```

**proj**

Returns a Proj.4 instance set up for the IBCAO UPS variant

**proj\_str**

Returns a Proj.4 string for the IBCAO UPS variant.

**template** (*div=1*)

Sets up and returns a figure with the IBCAO map loaded, ready for additional plotting:

**Args:** *div*: use every div point in map (1 is default, use all points)

**Returns:** matplotlib Figure

```
>>> i = IBCAO ()
>>> f = i.template ()
>>> f.show ()
```

**ups**

Returns a Cartopy Stereographic instance set up for the Universal Polar Stereographic (UPS) projection used in the IBCAO.

**x**

$x$  (longitude) arguments for depth data ( $z$ ) in meters (UPS).

**xlim**

Extent in longitude coordinates (meters) on UPS projection.

**y**

$y$  (latitude) arguments for depth data ( $z$ ) in meters (UPS).

**ylim**

Extent in latitude coordinates (meters) on UPS projection.

**z**

Depth data on *grid*.

## Module contents





---

## Indices and tables

---

- `genindex`
- `modindex`
- `search`



**i**

`ibcao`, 3

`ibcao.ibcao`, 1



## C

close() (ibcao.ibcao.IBCAO method), 1  
Colormap() (ibcao.ibcao.IBCAO method), 1

## G

geod (ibcao.ibcao.IBCAO attribute), 1  
get\_cartopy() (ibcao.ibcao.IBCAO method), 2  
grid() (ibcao.ibcao.IBCAO method), 2

## I

IBCAO (class in ibcao.ibcao), 1  
ibcao (module), 3  
ibcao.ibcao (module), 1  
interp\_depth() (ibcao.ibcao.IBCAO method), 2

## M

map\_depth() (ibcao.ibcao.IBCAO method), 2

## P

proj (ibcao.ibcao.IBCAO attribute), 2  
proj\_str (ibcao.ibcao.IBCAO attribute), 2

## R

REFERENCE (ibcao.ibcao.IBCAO attribute), 1

## T

template() (ibcao.ibcao.IBCAO method), 2

## U

ups (ibcao.ibcao.IBCAO attribute), 3

## V

VERSION (ibcao.ibcao.IBCAO attribute), 1

## X

x (ibcao.ibcao.IBCAO attribute), 3  
xlim (ibcao.ibcao.IBCAO attribute), 3

## Y

y (ibcao.ibcao.IBCAO attribute), 3

ylim (ibcao.ibcao.IBCAO attribute), 3

## Z

z (ibcao.ibcao.IBCAO attribute), 3