region_mean

January 11, 2024

[ ]:
```
import numpy as np
import sys, os

sys.path.append("..")
from qgbaroclinic.model.ocebaroclinicmodes import OceBaroclinicModes
```

[ ]:
```
# Define Baroclinic Modes object
obm = OceBaroclinicModes()
```

[ ]:
```
# Set region domain
# NOTE: dimensions name "x", "y" can be defined by the user.
obm.region(x=[-34, -30], y=[34.5, 50.2])
```

[ ]:
```
# Extract OCEAN variables form NetCDF file.
# NOTE: "latitude", "longitude" are the dimension names in NetCDF file.
current_path = os.path.dirname(__vsc_ipynb_file__)
temp, sal, depth, lat = obm.read(
    os.path.join(current_path, "../data/reanalysis/"),
    "thetao",
    "so",
    "depth",
    "latitude",
    longitude=obm.domain["x"],
    latitude=obm.domain["y"],
)
```

[ ]:
```
# Extract Bathymetry dataset and compute mean reagion depth
# NOTE: "lat", "lon" are the dimension names in NetCDF file.
elevation = obm.read(
    os.path.join(current_path, "../data/bathymetry/GEBCO_2023.nc"),
    "elevation",
    lat=obm.domain["y"],
    lon=obm.domain["x"],
)
mean_depth = np.abs(np.nanmean(elevation))
```

[ ]:
```
"region mean depth", mean_depth, type(mean_depth)
```

[ ]:
```
('region mean depth', 3017.216260416178, numpy.float64)
```

[ ]:
```
# Set mean depth as the bottom depth
obm.bottomdepth(mean_depth)
```

[ ]:
```
lat.values
```

[ ]:
```
array([34.5     , 34.583332, 34.666668, 34.75    , 34.833332, 34.916668,
       35.      , 35.083332, 35.166668, 35.25    , 35.333332, 35.416668,
       35.5     , 35.583332, 35.666668, 35.75    , 35.833332, 35.916668,
       36.      , 36.083332, 36.166668, 36.25    , 36.333332, 36.416668,
       36.5     , 36.583332, 36.666668, 36.75    , 36.833332, 36.916668,
       37.      , 37.083332, 37.166668, 37.25    , 37.333332, 37.416668,
       37.5     , 37.583332, 37.666668, 37.75    , 37.833332, 37.916668,
       38.      , 38.083332, 38.166668, 38.25    , 38.333332, 38.416668,
       38.5     , 38.583332, 38.666668, 38.75    , 38.833332, 38.916668,
       39.      , 39.083332, 39.166668, 39.25    , 39.333332, 39.416668,
       39.5     , 39.583332, 39.666668, 39.75    , 39.833332, 39.916668,
       40.      , 40.083332, 40.166668, 40.25    , 40.333332, 40.416668,
       40.5     , 40.583332, 40.666668, 40.75    , 40.833332, 40.916668,
       41.      , 41.083332, 41.166668, 41.25    , 41.333332, 41.416668,
       41.5     , 41.583332, 41.666668, 41.75    , 41.833332, 41.916668,
       42.      , 42.083332, 42.166668, 42.25    , 42.333332, 42.416668,
       42.5     , 42.583332, 42.666668, 42.75    , 42.833332, 42.916668,
       43.      , 43.083332, 43.166668, 43.25    , 43.333332, 43.416668,
       43.5     , 43.583332, 43.666668, 43.75    , 43.833332, 43.916668,
       44.      , 44.083332, 44.166668, 44.25    , 44.333332, 44.416668,
       44.5     , 44.583332, 44.666668, 44.75    , 44.833332, 44.916668,
       45.      , 45.083332, 45.166668, 45.25    , 45.333332, 45.416668,
       45.5     , 45.583332, 45.666668, 45.75    , 45.833332, 45.916668,
       46.      , 46.083332, 46.166668, 46.25    , 46.333332, 46.416668,
       46.5     , 46.583332, 46.666668, 46.75    , 46.833332, 46.916668,
       47.      , 47.083332, 47.166668, 47.25    , 47.333332, 47.416668,
       47.5     , 47.583332, 47.666668, 47.75    , 47.833332, 47.916668,
       48.      , 48.083332, 48.166668, 48.25    , 48.333332, 48.416668,
       48.5     , 48.583332, 48.666668, 48.75    , 48.833332, 48.916668,
       49.      , 49.083332, 49.166668, 49.25    , 49.333332, 49.416668,
       49.5     , 49.583332, 49.666668, 49.75    , 49.833332, 49.916668,
       50.      , 50.083332, 50.166668], dtype=float32)
```

[ ]:
```
# Set model input variables
obm.sawater_prop(temperature=temp.values, salinity=sal.values,␣
 ↪insitu_temperature=False)
```

[ ]:
```
# Run model
obm.run(n_modes=3)
```

```
---------------------------------------------------------------------------
TypingError                                 Traceback (most recent call last)
Cell In[18], line 2
      1 # Run model
----> 2 obm.run(n_modes=3)

File ~/physics/numerical_models/ocean-baroclinic-modes/examples/../qgbaroclinic
 ↪model/ocebaroclinicmodes.py:94, in OceBaroclinicModes.run(self, n_modes)
     92 else:
     93     pot_temperature = self.temperature
---> 94 (pot_density, depth_levels) = OceBaroclinicModes.pot_density(
     95     pot_temperature, self.salinity, depth, self.grid_step, self.
 ↪bottom_depth
     96 )
     97 pot_density = np.nanmean(pot_density, axis = (0,1,2))
     98 bv_freq = OceBaroclinicModes.compute_bruntvaisala_freq(
     99     depth_levels, pot_density, self.grid_step
    100 )

File ~/physics/numerical_models/ocean-baroclinic-modes/examples/../qgbaroclinic
 ↪model/ocebaroclinicmodes.py:127, in OceBaroclinicModes.
 ↪pot_density(pot_temperature, salinity, depth, grid_step, bottom_depth)
    125 assert pot_temperature.shape == salinity.shape
    126 ref_pressure = 0  # reference pressure [dbar]
--> 127 pot_density = EoS.compute_density(salinity, pot_temperature,␣
 ↪ref_pressure)
    129 # VERTICAL INTERPOLATION (default to 1m grid step)
    130 interpolation = Interpolation(depth, pot_density)

File ~/physics/numerical_models/ocean-baroclinic-modes/examples/../qgbaroclinic
 ↪tool/eos.py:63, in EoS.compute_density(sal, pot_temp, ref_press)
     53 ref_press /= 10
     54 # ========================================================================
     55 # Compute reference density at atmospheric pressure
     56 #
   (…)
     60 # of seawater' (Millero and Poisson, 1981).
     61 # ========================================================================
---> 63 rho = EoS.__compute_rho(sal, pot_temp)
     65 # ========================================================================
     66 # Compute coefficients in the bulk modulus of seawater expression
     67 #
   (…)
     78
     79 # Bulk modulus of seawater at atmospheric pressure.
     80 K_0 = EoS.__compute_K_0(sal, pot_temp)
```

```
File ~/anaconda3/envs/obm/lib/python3.11/site-packages/numba/core/dispatcher.py
 ↪468, in _DispatcherBase._compile_for_args(self, *args, **kws)
    464         msg = (f"{str(e).rstrip()} \n\nThis error may have been caused
    465                f"by the following argument(s):\n{args_str}\n")
    466         e.patch_message(msg)
--> 468     error_rewrite(e, 'typing')
    469 except errors.UnsupportedError as e:
    470     # Something unsupported is present in the user code, add help info
    471     error_rewrite(e, 'unsupported_error')

File ~/anaconda3/envs/obm/lib/python3.11/site-packages/numba/core/dispatcher.py
 ↪409, in _DispatcherBase._compile_for_args.<locals>.error_rewrite(e, issue_typ
    407     raise e
    408 else:
--> 409     raise e.with_traceback(None)

TypingError: Failed in nopython mode pipeline (step: nopython frontend)
non-precise type pyobject
During: typing of argument at /Users/francesco/physics/numerical_models/
 ↪ocean-baroclinic-modes/examples/../qgbaroclinic/tool/eos.py (152)

File "../qgbaroclinic/tool/eos.py", line 152:
    def depth2press(depth: float) -> float:
        <source elided>

    @staticmethod
    ^

This error may have been caused by the following argument(s):
- argument 0: Cannot determine Numba type of <class 'xarray.core.variable.
 ↪Variable'>
- argument 1: Cannot determine Numba type of <class 'xarray.core.variable.
 ↪Variable'>

This error may have been caused by the following argument(s):
- argument 0: Cannot determine Numba type of <class 'xarray.core.variable.
 ↪Variable'>
- argument 1: Cannot determine Numba type of <class 'xarray.core.variable.
 ↪Variable'>
```

[ ]:
```
# The output result is stored as attributes
rossby_rad = obm.rossby_rad
vert_structfunc = obm.vert_structfunc
print(rossby_rad)
```