ECCO_v4_A_Better_Method_for_Loading_ECCOv4_NetCDF_Tile_File

February 6, 2018

1 A Better Method for Loading ECCOv4 NetCDF Tile Files

1.1 Objectives:

Introduce an alternative method for loading ECCO v4 NetCDF tile files that returns `Dataset` and `DataArray` objects with better labelling of variable coordinates with respect to *where* they are situated on the Arakawa-C grid.

1.2 Introduction

As we showed in the first tutorial, we can use the `open_dataset` method from `xarray` to load a NetCDF tile file into Python as a `Dataset` object. `open_dataset` is very convienent because it automatically parses the NetCDF file and constructs a `Dataset` object using all of the dimensions, coordinates, variables, and metadata information. However, by default the names of the coordinates are pretty generic: *i1*, *i2*, *i3*, etc. We can do a lot better.

In the last tutorial we loaded a single ECCOv4 grid tile file and examined its contents. Let's load it up again and take another look at its coordinates. This time we'll name the new `Dataset` object `grid_3_od` since we are loading the file using `open_dataset`.

```
In [1]: import matplotlib.pylab as plt
        import numpy as np
        import sys
        import xarray as xr
        from copy import deepcopy
        import ecco_v4_py as ecco
```

```
In [2]: # point to your local directory holding the nctiles_grid files
        grid_dir='/Users/ifenty/ECCOv4/R3/nctiles_grid/'
        fname = 'GRID.0003.nc'
        grid_3_od = xr.open_dataset(grid_dir + fname)
```

```
In [3]: grid_3_od
```

```
Out[3]: <xarray.Dataset>
        Dimensions:  (i1: 50, i2: 90, i3: 90)
        Coordinates:
```

```
  * i1       (i1) float64 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 ...
  * i2       (i2) float64 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 ...
  * i3       (i3) float64 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 ...
Data variables:
    hFacC    (i1, i2, i3) float64 ...
    hFacW    (i1, i2, i3) float64 ...
    hFacS    (i1, i2, i3) float64 ...
    XC       (i2, i3) float64 ...
    YC       (i2, i3) float64 ...
    XG       (i2, i3) float64 ...
    YG       (i2, i3) float64 ...
    RAC      (i2, i3) float64 ...
    RAZ      (i2, i3) float64 ...
    DXC      (i2, i3) float64 ...
    DYC      (i2, i3) float64 ...
    DXG      (i2, i3) float64 ...
    DYG      (i2, i3) float64 ...
    Depth    (i2, i3) float64 ...
    AngleCS  (i2, i3) float64 ...
    AngleSN  (i2, i3) float64 ...
    RC       (i1) float64 ...
    RF       (i1) float64 ...
    DRC      (i1) float64 ...
    DRF      (i1) float64 ...
Attributes:
    description:    C-grid parameters (see MITgcm documentation for details)...
    A:              :Format      = native grid (nctiles w. 13 tiles)
    B:              :source      = ECCO consortium (http://ecco-group.org/)
    C:              :institution = JPL/UT/MIT/AER
    D:              :history     = files revision history :
    E:                                 04/20/2017: fill in geometry info for ...
    F:                                 11/06/2016: third release of ECCO v4 (...
    G:                             estimates revision history (from second re...
    H:                                 employs bi-harmonic viscosity (enhance...
    I:                                 sea-ice parameters, updated or novel o...
    J:                                 GRACE OBP, Aquarius SSS, global mean s...
    K:                                 time-series, extended and/or expanded ...
    L:                                 revised weights including data and con...
    M:                                 to account for grid-size variation and...
    N:                                 separate time-mean and time-variable d...
    O:                                 and controls, sea-ice costs, and initi...
    P:                                 additional controls.\n
    Q:              :references  = Forget, G., J.-M. Campin, P. Heimbach, C. ...
    R:                              and C. Wunsch, 2015: ECCO version 4: an i...
    S:                              non-linear inverse modeling and global oc...
    T:                              Geoscientific Model Development, 8, 3071-...
    U:                             Forget, G., J.-M. Campin, P. Heimbach, C. ...
    V:                              ECCO version 4: Second Release, 2016, htt...
```

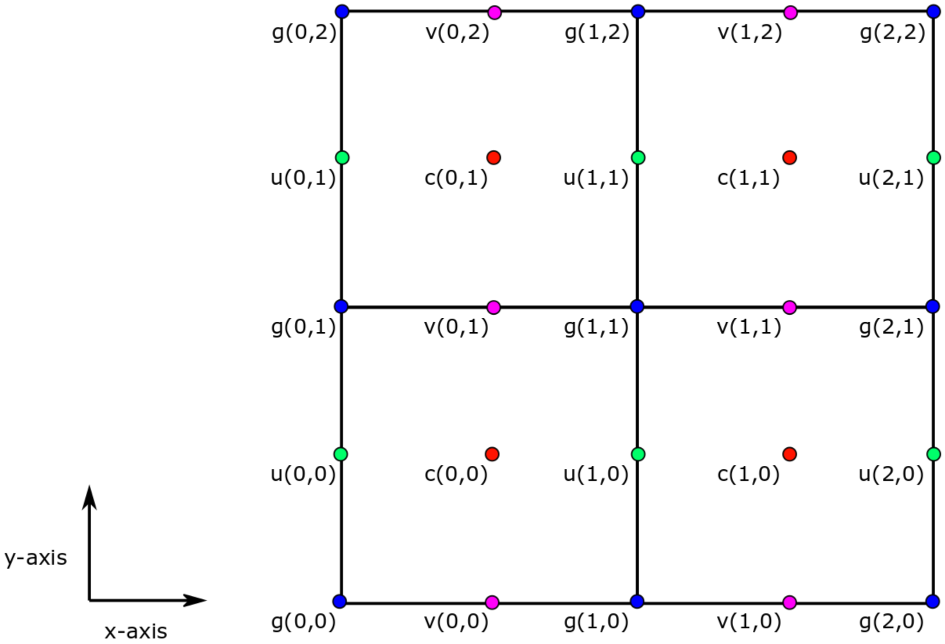
```
W:              file created using gcmfaces_IO/write2nctiles.m
date:           21-Apr-2017
Conventions:    CF-1.6
_FillValue:     nan
missing_value:  nan
```

We see that all of the Data variables in `grid_3_od` use one of three dimensions, **i1**, **i2**, and **i3**. As we saw before, some variables are 3D (e.g., hFacC), others are 2D (e.g., XC), and others are 1D (e.g., RF).

This `Dataset` object is already quite useful but it falls well short of taking full advantage of the coordinate labeling feature provided by the `Dataset` objects. Let's investigate the coordinates of the Arakawa-C grid (hereafter c-grid)

1.3 The four horizontal points of the c-grid

The c-grid is a staggered grid, model variables are not all co-located in the center of model grid cells. In the horizontal plane, model variables can be in one of four different categories of point:



The four different categories of points used in the staggered Arakawa-C grid (C-grid)

1.3.1 *c* points

Scalar variables (e.g., T, S, SSH, OBP, sea ice concentration, vertical velocity) are situated at the center of the tracer grid cell in the horizontal plane. These are c points

Define the (i, j) coordinate system for the indices of c points.

In the ECCO v4 NetCDF tile files, $c(0, 0)$ is the -x most and -y most tracer grid cell.

- In the $+y$ direction, the next c point is $c(0, 1)$.
- In the $+x$ direction, the next c point is $c(1, 0)$

1.3.2 *u* points

Vector variables related to horizontal velocity in the x direction are staggered along the edges of tracer cells between c points in the horizontal plane. Examples include horizontal velocity in the x direction ($UVEL$) and horizontal advective flux of snow in the x direction ($ADVxSNOW$). They are situated along the edges (if 2D) or faces (if 3D) of the tracer grid cells in the x direction.

Define the (i_g, j) coordinate system for u points. We use i_g as the coordinate in the x direction because u points are situated along the tracer grid cell ed**G**es. We use j for its y coordinate because u points and c points fall along the same lines in y.

In the ECCO v4 netCDF tile files, $u(0,0)$ is the -x most and -y most u point.

1.3.3 *v* points

Vector variables related to horizontal velocity in the y direction are staggered along the edges of tracer cells between c points in the horizontal plane. Examples include horizontal velocity in the y direction ($VVEL$) and horizontal advective flux of snow in the y direction ($ADVySNOW$). They are situated along the edges (if 2D) or faces (if 3D) of the tracer grid cells in the y direction.

Define the (i, j_g) coordinate system for v points. We use j_g as the coordinate in the y direction because v points are situated along the tracer grid cell ed**G**es. We use i for its x coordinate because v points and c points fall along the same lines in x.

In the ECCO v4 NetCDF tile files, $v(0,0)$ is the -x most and -y most v point.

1.3.4 *g* points

Variables that are explictly related to horizontal velocities in the model in both the x and y direction are situated at g points in the horizontal plane. g points are situated at the corners of tracer grid cells.

Define the (i_g, j_g) coordinate system for g points following the same reasoning as described above: in both the x and y directions, g points are on the ed**G**es of tracer grid cells.

In the ECCO v4 NetCDF tile files, $g(0,0)$ is the -x most and -y most g point.

1.4 The two vertical points of the c-grid

There are two different coordinates in the vertical z dimension:

1.4.1 *w* points

Variables related to vertical velocity or vertical fluxes are situated at w points in the vertical direction. These variables are situated on the upper and lower faces of the tracer grid cell.

Define the k_g coordinate system for w points by following the same reasoning as we used above: w points fall along the the ed**G**es of tracer grid cells in the z direction.

In the ECCO v4 NetCDF tile files, $k_g(0) is the sea surface.

1.4.2 *k* points

Variables that are not related to vertical velocity or vertical fluxes are situated at k points in the vertical direction. These variables are situated on the upper and lower faces of the tracer grid cell.

Define the k_g coordinate system for w points by following the same reasoning as we used above: w points fall along the the ed**G**es of tracer grid cells in the z direction.

In the ECCO v4 NetCDF tile files, $k(0) is the middle of the uppermost tracer grid cell.

1.5 Applying the C-grid coordinates to the variables

The default coordinate names in the ECCO v4 netcdf tile files do not distinguish distinguish between the four horizontal coordinates, i, i_g, j, j_g and the two vertical coordinates, k_g and k, used by our c-grid model.

To apply these more descriptive coordinates to the `Dataset` objects that are created when we load netCDF files, we provide a special routine, `load_tile_from_netcdf`.

1.5.1 `load_tile_from_netcdf`

This routine takes four arguments, 1. *data_dir*: the directory of the netCDF file 2. *var*: the name of the netCDF file without the tile number. 3. *var_type*: one of 'c','g','u','v', or 'grid' corresponding with the variables C-grid point type. 'grid' is a special case because **GRID** ECCO v4 tile files are unique in that they contain a mix of 'c','g','u','v','k', and 'w' points. 4. *tile_index*: the tile number [1 .. 13]

1.5.2 Loading an ECCO v4 netCDF tile file using `load_tile_from_netcdf`

Let's use `load_tile_from_netcdf` to load grid tile 3 again. This time we'll call the new `Dataset` object `grid_3_new`

```
In [4]: var = 'GRID'
        var_type = 'grid'
        tile_index = 3
        grid_3_new = ecco.load_tile_from_netcdf(grid_dir,
                                                var,
                                                var_type,
                                                tile_index)
```

```
loading /Users/ifenty/ECCOv4/R3/nctiles_grid/GRID.0003.nc
```

```
In [5]: grid_3_new
```

```
Out[5]: <xarray.Dataset>
        Dimensions:  (i: 90, i_g: 90, j: 90, j_g: 90, k: 50, k_l: 50, k_u: 50)
        Coordinates:
            tile     int64 3
          * k        (k) float64 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 ...
          * i        (i) float64 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 ...
          * j        (j) float64 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 ...
          * i_g      (i_g) float64 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 ...
          * j_g      (j_g) float64 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 ...
          * k_u      (k_u) float64 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 ...
          * k_l      (k_l) int64 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 ...
        Data variables:
            XC       (j, i) float64 ...
```

```
    YC       (j, i) float64 ...
    RAC      (j, i) float64 ...
    Depth    (j, i) float64 ...
    AngleCS  (j, i) float64 ...
    AngleSN  (j, i) float64 ...
    hFacC    (k, j, i) float64 ...
    land_c   (k, j, i) float64 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 ...
    XG       (j_g, i_g) float64 ...
    YG       (j_g, i_g) float64 ...
    RAZ      (j_g, i_g) float64 ...
    DXC      (j, i_g) float64 ...
    DYG      (j, i_g) float64 ...
    hFacW    (k, j, i_g) float64 ...
    land_u   (k, j, i_g) float64 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 ...
    DYC      (j_g, i) float64 ...
    DXG      (j_g, i) float64 ...
    hFacS    (k, j_g, i) float64 ...
    land_v   (k, j_g, i) float64 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 ...
    RF       (k_u) float64 ...
    DRC      (k_u) float64 ...
    RC       (k) float64 ...
    DRF      (k) float64 ...
    RB       (k_l) float64 -10.0 -20.0 -30.0 -40.0 -50.0 -60.0 -70.0 -80.01 ...
Attributes:
    description:    C-grid parameters (see MITgcm documentation for details)...
    A:              :Format      = native grid (nctiles w. 13 tiles)
    B:              :source      = ECCO consortium (http://ecco-group.org/)
    C:              :institution = JPL/UT/MIT/AER
    D:              :history     = files revision history :
    E:                                 04/20/2017: fill in geometry info for ...
    F:                                 11/06/2016: third release of ECCO v4 (...
    G:                             estimates revision history (from second re...
    H:                                 employs bi-harmonic viscosity (enhance...
    I:                                 sea-ice parameters, updated or novel o...
    J:                                 GRACE OBP, Aquarius SSS, global mean s...
    K:                                 time-series, extended and/or expanded ...
    L:                                 revised weights including data and con...
    M:                                 to account for grid-size variation and...
    N:                                 separate time-mean and time-variable d...
    O:                                 and controls, sea-ice costs, and initi...
    P:                                 additional controls.\n
    Q:              :references  = Forget, G., J.-M. Campin, P. Heimbach, C. ...
    R:                              and C. Wunsch, 2015: ECCO version 4: an i...
    S:                              non-linear inverse modeling and global oc...
    T:                              Geoscientific Model Development, 8, 3071-...
    U:                             Forget, G., J.-M. Campin, P. Heimbach, C. ...
    V:                              ECCO version 4: Second Release, 2016, htt...
    W:              file created using gcmfaces_IO/write2nctiles.m
```

```
date:           21-Apr-2017
Conventions:    CF-1.6
_FillValue:     nan
missing_value:  nan
```