ECCO_v4_Improving_the_GRID_Dataset_Object

January 25, 2018

1 An Improved Method for Loading ECCOv4 netCDF files

1.1 Objectives:

To introduce a method for loading data from the ECCO v4 netCDF tile files that returns `Dataset` and `DataArray` objects that properly distinguish *where* on the Arakawa-C grid the variables are situated.

This custom routine written for the *ecco_v4_py* package is: `load_tile_from_netcdf`.

1.2 Introduction

As we showed in the first tutorial, we can use the `xr.open_dataset(data_dir + fname)` routine from xarray to load a netCDF tile file into Python. This routine parses the netCDF file and extracts the dimensions, coordinates, variables, and metadata information and uses those to construct the `Dataset` object.

In the last tutorial we read in a single GRID netCDF tile file and examined its contents. We found that it listed three dimensions, i1, i2, and i3, for its Data variables. Let's load it up again and take a closer look. This time we'll name the `Dataset` object as `grid_default` since we are loading it with the default method from `xarray`.

```
In [26]: import matplotlib.pylab as plt
         import numpy as np
         import sys
         import xarray as xr
         from copy import deepcopy
         sys.path.append('/Users/ifenty/git_repo/ECCOv4-py')
         import ecco_v4_py as ecco
```

```
In [27]: data_dir='/Volumes/ECCO_BASE/ECCO_v4r3/nctiles_grid/'
         fname = 'GRID.0003.nc'
         grid_default = xr.open_dataset(data_dir + fname)
```

```
In [28]: grid_default
```

```
Out[28]: <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_default` use one of three dimensions, **i1**,**i2**,**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).

Now, while the default format of this Dataset object is already quite useful, it falls short of taking full advantage of the 'coordinate' feature afforded by the Dataset object.

Model variables on Arakawa-C grids are staggered in space. On the horizontal plane, model variables can be situated at one of four different classes of point.

figure

1.3 The four horizontal points of the Arakawa-C grid

1.3.1 c points

Variables that do not have a horizontal velocity component (e.g., T, S, SSH, OBP, sea ice concentration, vertical velocity) are situated at c points in the horizontal plane. c points are at the center of the *tracer* grid cell in the horizontal plane.

Let us define the (i, j) coordinate system for the indices of c points.

$c(0, 0)$ is the -x most and -y most tracer grid cell.

Moving in the +y direction, the next grid cell is $c(0, 1)$ Moving in the +x direction, the next grid cell is $c(1, 0)$

1.3.2 u points

Variables that are explictly related to horizontal velocity or horizontal fluxes in the model's x direction are situated at u points in the horizontal plane. Examples include horizontal velocity in the x direction ($UVEL$) and horizontal advective flux of snow in the x direction ($ADVxSNOW$).

In the x direction they are situated on the edges or faces of the tracer grid cell. In the y direction they are at the same locations as the c points.

Let us 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 the j for its y coordinate because it is the same as the y coordinate of the c points.

$u(0, 0)$ is the first u point situated in the $-x$ direction of $c(0,0).

Moving in the +y direction, the next grid cell is $u(0, 1)$ Moving in the +x direction, the next grid cell is $u(1, 0)$

1.3.3 v points

Variables that are explictly related to horizontal velocity or horizontal fluxes in the model's y direction are situated at v points in the horizontal plane. Examples include horizontal velocity in the y direction ($VVEL$) and horizontal advective flux of snow in the y direction ($ADVySNOW$).

In the x direction they are at the same locations as the c points. In the y direction they are situated on the edges (or faces) of the tracer grid cell.

Let us define the (i, j_g) coordinate system for v points. We use the i for its x coordinate because it is the same as the x coordinate of the c 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.

$v(0,0)$ is the first v point situated in the $-y$ direction of $c(0,0).

Moving in the +y direction, the next grid cell is $v(0,1)$ Moving in the +x direction, the next grid cell is $v(1,0)$

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. Vorticity is an example.

g points are situated along the edges of the grid cell in both x and y. In other words, they are at the **corners** of tracer grid cells.

Let us 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.

$g(0,0)$ is the first g point situated in the $-y$ and $-x$ directions of $c(0,0).

Moving in the +y direction, the next grid cell is $g(0,1)$ Moving in the +x direction, the next grid cell is $g(1,0)$

1.4 The two vertical points of the Arakawa-C gird

There are two coordinates in the vertical z dimension:

1.4.1 *w* points

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

Let us 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.

Indexing begins at the sea surface, k_g=0.

1.4.2 *k* points

Let us define the k coordinate system for variables situated in the middle of a tracer grid cell in the vertical z direction. Examples include all tracers.

Indexing begins in the uppermost grid cell surface, k=0.

The default coordinate names in the GRID netcdf tile files are not sufficiently descriptive to distinguish between the four horizontal coordinates ('i','i_g','j','j_g') and the three vertical coordinates.

Therefore, we provide a routine especially for reading in ECCOv4 llc90 GRID netcdf files and giving the default i1, i2, and i3 coordinates more descriptive names.

1.5 Load the file containing the grid parameter information for one tile.

To load ECCO v4's netcdf files we will use the *open_dataset* command from the xarray Python package. *open_dataset* creates a **Dataset** object and loads the contents of the netcdf file, including its metadata, into a data structure.

Let's open the grid file for *tile 3* (North East Atlantic Ocean), of the 13 ECCO v4 llc90 grid files.

Change `data_dir` to match the location of your `nctiles_grid` directory.

```
In [41]: data_dir='/Volumes/ECCO_BASE/ECCO_v4r3/nctiles_grid/'
         var = 'GRID'
         var_type = 'grid'
         tile_index = 3
         grid_tile_3 = ecco.load_tile_from_netcdf(data_dir,
                                                  var,
                                                  var_type,
                                                  tile_index)

loading /Volumes/ECCO_BASE/ECCO_v4r3/nctiles_grid/GRID.0003.nc


In [32]: grid_tile_3

Out[32]: <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
```