Accessing MET Norway ocean model output from THREDDS using MATLAB

K. H. Christensen (kaihc@met.no)

November 27, 2019

Contents

1 Introduction

Most output data from MET Norway's atmospheric, surface wave, and ocean models are distributed in netCDF format via so-called THREDDS servers. This is now the *de facto* standard for geophysical model output and data dissemination, and many software tools for data analysis have native support for accessing data using this format and these server protocols, for example MATLAB, R, and Python.

1.1 The netCDF data format

NetCDF files are self-contained in the sense that metadata is part of the contents. The metadata can be accessed using software that supports reading netCDF files, and hence information about the actual data can be retrieved before accessing them. The information in a netCDF file is either in the form of *dimensions*, *variables*, or *attributes*. The attributes are the metadata descriptors that provide information about the data. They can either be *global*, and hence refer to properties of the entire data set, or associated with a specific variable. As an example we consider the temperature data from the operational NorKyst-800m ocean model. The temperature is a variable (simply called "temperature" in the netCDF file). It has dimensions "X", "Y", "depth", "time", that is, the three spatial dimensions and time. Furthermore, the variable "temperature" has attributes such as "standard name", "units" and "_FillValue", which provide information about the data type (e.g. integer or float), masking, file compression, units and so on. The command `ncdisp` is very useful, providing all the metadata about all the variables on the netCDF file.

We will get back to some examples on how to extract data in Sec. 2. Note the important global attribute "Conventions", which is listed as "CF-1.4". This attribute refer to the version of the so-called CF convention being used. The CF convention provides standard names for metadata in climate and forecast data files. A complete list of such names can be found on the Internet at http://cfconventions.org/. According to the CF convention, ocean temperature is indeed called "sea_water_potential_temperature", which explains the somewhat overly descriptive standard name. A great benefit of using the CF convention is that it facilitates generic code. Consider for example outputs from different models, in which the *variable* name for temperature is not the same (e.g. "temp" and "temperature"), a commonly occuring situation. It is easy to write code that scans the metadata and identifies what variable that has the standard name attribute given as "sea_water_potential_temperature", regardless of what the variable itself is named. Such code is used for example in MET Norway's generic ocean drift application software OpenDrift[1], which can utilize geophysical forcing from any model provided metadata have been added using the CF convention. NetCDF attributes can be read directly into MATLAB variables using the command `ncreadatt`.

1.2 OPeNDAP and THREDDS

Briefly summarized, OPeNDAP and THREDDS (TDS) are technical solutions and protocols that allow both (i) data dissemination over the Internet, and (ii) downstream clients to access subsets of the data on the servers. It is also possible to provide access to *aggregated* data sets so that the link to the data does not change, but new data are regularly added to what already exist. Such aggregated data sets are very useful for disseminating data from forecasting centers. In our example above, we access the aggregated NorKyst-800m model data set, which is updated daily with the latest forecast. At the moment of this writing, this aggregated data set contains $O(10^{11})$ data points altogether, collected over several years of running the model operationally. It is clear that downstream clients cannot easily download the entire data set, hence the usefulness of accessing subsets. The data catalog for MET Norway models and observations can be browsed at http://thredds.met.no/thredds/catalog.html. For specific data sets, the link one should use for fetching data to MATLAB or other software is given as the "Data URL" under the access menu item "OPENDAP".

The ocean model ROMS uses terrain-following coordinates[2], which means that there are a fixed number of vertical levels and that the depth of these levels depend on the local bathymetry. For downstream users, we provide files on THREDDS where all values are interpolated to pre-defined fixed depths, and the examples shown in this report are based on these files. There is obviously no data where any of the fixed depths exceed the local depth, and in such cases the data are replaced by a fill value. Most OPeNDAP clients will handle these fill values in a meaningful way, for example replacing them with not-a-number as in the case of MATLAB. The numerical solution technique in ROMS is also based on a staggered grid, which means that the velocities and tracer variables (salinity, temperature) are not co-located in the horizontal. Horizontal interpolation has been taken care of in the files with pre-defined fixed depth, however, hence in these files the (X,Y) coordinate values refer to the same location for all variables.

2 Accessing data using MATLAB

2.1 Examples using scalar variables

First we will show a simple example accessing the scalar variable temperature to demonstrate the basics. A brief description of the velocity vector fields is given later on in Sec. 2.2. Continuing from the previous code example, and by browsing through the information given by `ncdisp(infile)`, we find that

- the MATLAB variable `infile` contains the name of the link to the data,
- ocean temperature is stored in a variable named `temperature`,

[1]http://https://github.com/OpenDrift/opendrift

[2]https:www.myroms.orgwikiVertical_S-coordinate

- the horizontal extent is 2602 by 902 grid points,
- there are 17 depth levels available,
- there are 877 time steps available (at this time of writing).

As can be easily checked using `ncdisp(infile)`, both "time" and "depth" are not just dimensions, but also variables. Hence we can retrieve both of them as variables to get the depth and time ranges of the aggregated data set. Starting with the depth, we do

```
>> depth = ncread(infile, 'depth');
```

which stores the depth range in a vector, which for the NorKyst-800m model has values (0, 3, 10, 15, 25, 50, 75, 100, 150, 200, 250, 300, 500, 700, 1000, 2000, 3000). Note that the `depth` variable is stored in single precision, but can easily be converted to double precision:

```
>> whos
(...)
  depth          17x1                      68  single
>> depth = double(depth);
>> whos
(...)
  depth          17x1                     136  double
```

Using `ncdisp(infile)`, we also find that the `time` variable has units "seconds since 1970-01-01 00:00:00". To read in the time vector and convert it to MATLAB's internal date format `datenum`, we thus need to add the reference time, and to scale from seconds to days:

```
>> timeref = datenum(1970,1,1);
>> time = ncread(infile, 'time');
>> time = time/24/3600 + timeref;
>> datestr(time(end))

ans =

    '28-Nov-2019 12:00:00'
```

and we see that the latest fields have time stamp Dec. 5, 2017 at 12 (UTC) in this case. Assume that we want to retrieve the surface temperature for June 30, 2016. We can find the time index as follows:

```
>> timeindex = find(time > datenum(2018,10,31), 1);
>> datestr(time(timeindex))

ans =

    '31-Oct-2018 12:00:00'
```

When we now want to retrieve the temperature data, we need to specify the starting indices and the *increments* to these. The depth index of the surface is 1 (the first entry of the depth vector is 0). The indices are according to the dimensions [X, Y, depth, time], and the code to get the data becomes (ignore the line break)

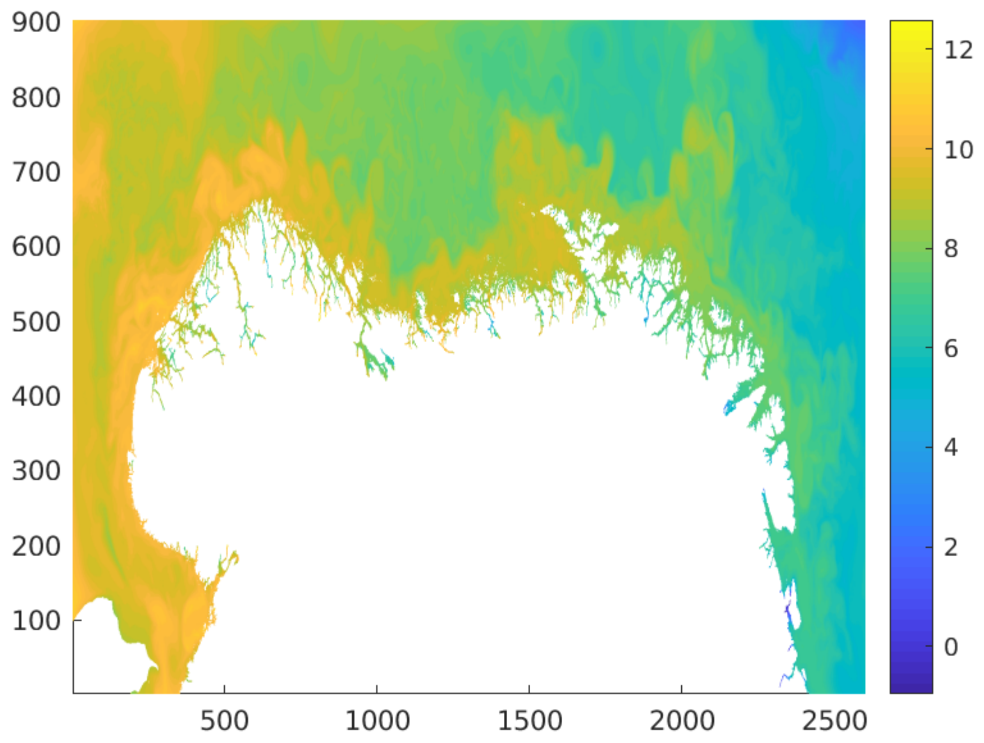

Figure 1: Surface temperature for June 30, 2016 at 12 (UTC) shown on the native grid of the ocean model NorKyst-800m. The tickmarks denote grid points.

```
>> temp = ncread(infile, 'temperature', [1 1 1 timeindex],
                                         [2602 902 1 1]);
>> whos
(...)
  temp              2602x902              18776032  double
```

which shows that we have extracted a full two-dimensional field for a specific depth and time. If, for example, we want to extract 10 successive fields starting on Oct. 31, 2018, we would need to specify the increment `[2602 902 1 10]` instead.

The data can easily be visualized using e.g. `pcolor`:

```
>> pcolor(temp'); shading flat; colorbar
```

and the result is shown in Fig. 1. Land is masked automatically in this case, since the `ncread` command checks for the presence of the attribute `_FillValue`, and replaces every data point containing this fill value with not-a-number (NaN).

In Fig. 1 the data are plotted on the native grid of the model. The data can plotted using various map projections if converted from the native grid, its specification is provided by the `proj4_string` attribute found in the metadata. For a simple latitude-longitude plot, the position of every grid point can be read from the data set, and a plot produced in this manner:

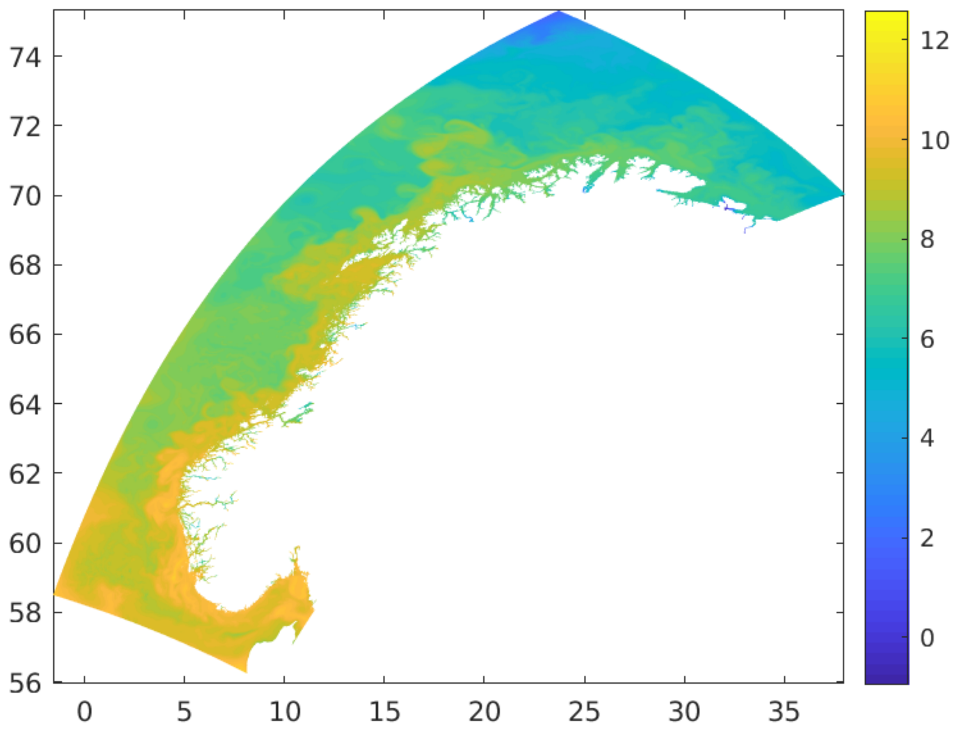

Figure 2: Surface temperature for Oct. 31, 2018 at 12 (UTC) shown on a latitude-longitude grid.

```
>> lon = ncread(infile, 'lon');
>> lat = ncread(infile, 'lat');
>> pcolor(lon, lat, temp); shading flat; colorbar
```

The result is shown in Fig. 2.

Finally, as an example of subsetting the data in space, consider the following example that extracts the same data as in the previous example, but for a smaller area around Lofoten and Vesterålen (the result is shown in Fig. 3):

```
>> lon = ncread(infile, 'lon', [1500 500], [400 300]);
>> lat = ncread(infile, 'lat', [1500 500], [400 300]);
>> temp = ncread(infile, 'temperature', [1500 500 1 timeindex],
                                  [400 300 1 1]);
>> pcolor(lon, lat, temp); shading flat; colorbar
```

In this example, we have used the starting indices (X,Y) = (1500,500), and increments of 400 and 300 grid points in the x- and y-directions, respectively, to choose our sub-domain of interest. Another option is to identify the (X,Y)-indices from latitude and longitude values, see the example in Sec. 2.2 on how to find the grid indices for this option. Note that in the example here the grid point position variables are named "longitude" and "latitude". In other data sets the variables may be named "lon" and "lat". What variable names to use can be easily checked with the command `ncdisp`.

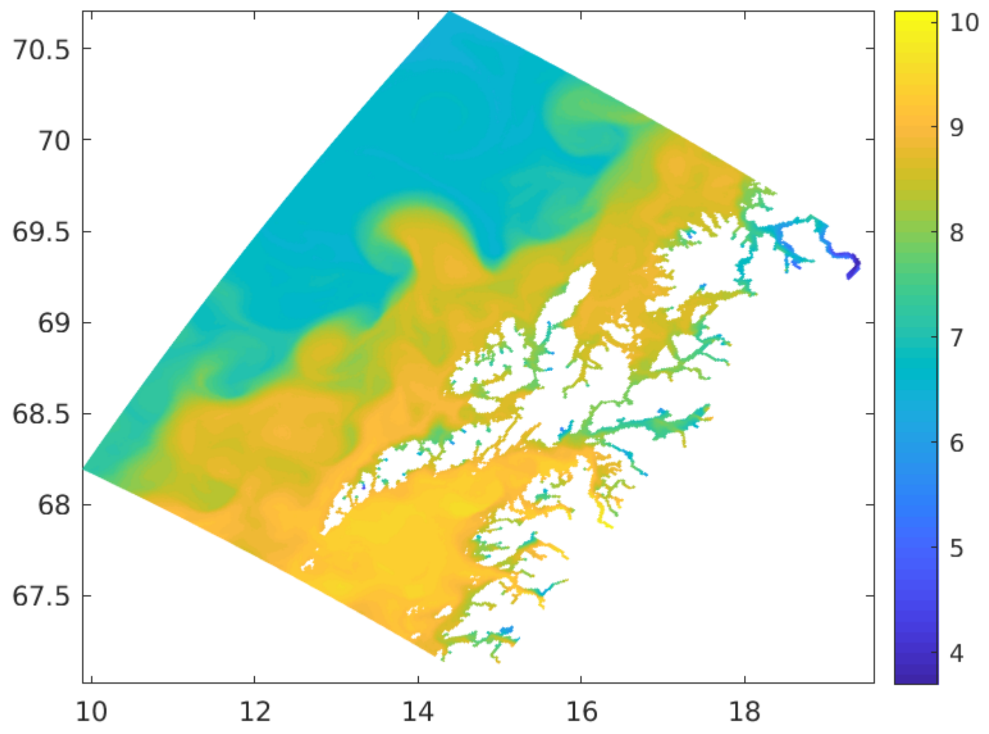

Figure 3: Surface temperature for Oct. 31, 2018 at 12 (UTC) for a sub-domain of the NorKyst-800m model.

2.2 Ocean currents

Ocean currents are stored either as components (u, v) relative to the native grid, or as components (u_e, v_n) that are rotated into standard eastward/northward components. In the former case we need to rotate the components ourselves. In the model output file there is a variable called "angle", which contains the angle needed to rotate the velocity vector. The conversion is done in the following way:

$$u_e = u\cos(\theta) - v\sin(\theta),$$
$$v_e = v\cos(\theta) + u\sin(\theta),$$

where θ is the value of the "angle" variable. The non-rotated velocity component variables are usually called "u" and "v", and their standard name attributes are "x_sea_water_velocity" and "y_sea_water_velocity", respectively. The output already contains velocities that are rotated into eastward and northward components, however, and the variable names are "u_eastward" and "v_northward" with standard name attributes are "eastward_sea_water_velocity" and "northward_sea_water_velocity", respectively.

3 A note on HF radar data

Daily aggregations of hourly HF radar radials can be accessed as netCDF files from the THREDDS servers. The dimensions in these files are "time", "bearing" and "range". The bearing is the direction in degrees of the radial pointing away from the radar, with zero value for true north and increasing clockwise. The range is the distance from the antenna in km. The position of the HF radar is provided in the global attribute "Origin" as decimal degrees. The position of a single HF radar radial observation can be calculated using the origin, range and bearing, but the position is also given in the variables "lat" and "lon". Note that for long-range HF radars, such as those deployed in the southern Barents Sea, the measurements are representative of the current at about 2 m depth. Medium-range systems (e.g. Torungen) have representative depths of about 0.8 m.

The radial velocity is defined positive when the currents move *away* from the radar, and the values are stored in the variable "speed" with units cm/s, and with standard name attribute "radial_sea_water_velocity_away_from_instrument". The fill value is -999.0, indicating observations that are dubious because of shadowing by land, excessive noise or other sources of error. Uncertainty estimates (lower bounds) are given for each observation in the variables "espc" (spatial uncertainty, σ_S) and "etmp" (temporal uncertainty, σ_T) as standard deviations in cm/s. The standard name attributes are "radial_sea_water_velocity_spatial_quality" and "radial_sea_water_velocity_temporal_quality", respectively. Again, a fill value of -999.0 is used in cases where the instrument error is such that meaningful values cannot be calculated. An estimate of the total uncertainty is then $\sigma = \sqrt{\sigma_S^2 + \sigma_T^2}$.

Assuming that the radial speed r_{HF} for a specific position is retrieved from THREDDS, an equivalent model radial speed r_{model} can be calculated as follows

$$r_{\mathrm{model}} = u_e \sin(\alpha) + v_e \cos(\alpha),$$

where α is the bearing angle.

The data are accessible from http://thredds.met.no/thredds/catalog/remotesensinghfradar/catalog.html, with folders for year, month and day per radar system. As an example of extracting hourly HF radials for one day at the observation point closest to a specific position, consider the following, with `infile` referring to the file on THREDDS and with the longitude and latitude position stored in the variables `lon_in` and `lat_in`, respectively (time has the same name and reference as in the ocean model):

```
>> lon = ncread(infile, 'lon');
>> lat = ncread(infile, 'lat');
>> rs = ncread(infile, 'speed');
>> brg = ncread(infile, 'bearing');
>> dm = (lon - lon_in).^2 + (lat - lat_in).^2;
>> [i,j] = find( dm == min(dm(:)) );
>> radialspeed = squeeze(rs(i,j,:));
>> bearing = squeeze(brg(i,j,:));
```

4 Concluding remarks

The preferred method for direct access to MET Norway's ocean model output is to retrieve data from the THREDDS servers using client software with OPeNDAP support. MATLAB is one such client, others include Python and R. There are several advantages to using THREDDS, for instance continuously updated and aggregated data sets, and the possibility to retrieve arbitrary subsets of the data. The data file format is netCDF, which is a widely used format for geophysical data. NetCDF files contain metadata with information about the content, and the metadata values can either be viewed directly on the THREDDS servers using a browser, or queried using client software such as MATLAB.

Some examples of retrieving and plotting ocean model data using MATLAB are provided in this report. Care must be taken to choose the correct files to access the model data, preferably the files in which data have been interpolated from terrain-following vertical coordinates to fixed depth levels, and from staggered to non-staggered grids. Example scripts can be found in the MET Norway git repository https://github.com/metno/fou_hi_example_scripts, in the folder "matlab", and a brief description of what each of these scripts do are listed in the "README.md" file found in that folder.

Acknowledgements

The author wish to thank A/S Norske Shell and Eni Norge for financial support for producing this manual.