Lab 10

Reading NetCDF data in MatLab

Marine Modelling March 25, 2019

Katja Fennel
Oceanography
Dalhousie University

Outline

Plan for today:

- Overview of different functions to read NetCDF data
- Example: Bottom oxygen in northern Gulf of Mexico

Overview of different functions to read NetCDF data

NetCDF (Network Common Data Form) is a widely used data format in earth sciences, incl. oceanography, meteorology etc. NetCDF files (`.nc`) can contain data for one or more variables and corresponding meta data, e.g. longitude, latitude and time.

There are two different sets of functions for accessing NetCDF data in MatLab:

- high-level functions: `ncinfo`, `ncread`, `ncreadatt`
- low-level functions: `netcdf`

More information here:
https://www.mathworks.com/help/matlab/import_export/importing-network-common-data-form-netcdf-files-and-opendap-data.html

Important high-level functions

High-level functions are convenient to use, but can be slower.

`ncinfo` is used to get information on file content – try it out:

```
fileName = 'roms854_MCH_bottom_O2_2000-2016.nc';
fileInfo = ncinfo(fileName)
```

Access, e.g. variable names, using:
`fileInfo.Variables.Name`

Important high-level functions

High-level functions are convenient to use, but can be slower.

`ncinfo` is used to get information on file content – try it out:

```
fileName = 'roms854_MCH_bottom_O2_2000-2016.nc';
fileInfo = ncinfo(fileName)
```

Access, e.g. variable names, using:
`fileInfo.Variables.Name`

`ncread` is used to read data for selected variable:

```
varName = 'oxygen';
varData = ncread(fileName, varName);
```


Important high-level functions

High-level functions are convenient to use, but can be slower.

`ncinfo` is used to get information on file content – try it out:

```
fileName = 'roms854_MCH_bottom_O2_2000-2016.nc';
fileInfo = ncinfo(fileName)
```

Access, e.g. variable names, using:
`fileInfo.Variables.Name`

`ncread` is used to read data for selected variable:

```
varName = 'oxygen';
varData = ncread(fileName, varName);
```

`ncreadatt` is used to read an attribute of a variable:

```
attName = 'units';
varUnit = ncreadatt(fileName, varName, attName)
```


Important low-level functions

Low-level functions require more coding, but can be significantly faster, especially when dealing with large datasets.

`netcdf.open` is used to open a NetCDF file for reading. Always the first step when using low-level functions:

```
ncID = netcdf.open(fileName)
```


Important low-level functions

Low-level functions require more coding, but can be significantly faster, especially when dealing with large datasets.

`netcdf.open` is used to open a NetCDF file for reading. Always the first step when using low-level functions:

```
ncID = netcdf.open(fileName)
```

`netcdf.inqVarID` is used to retrieve the ID of a variable:

```
varID = netcdf.inqVarID(ncID, varName)
```


Important low-level functions

Low-level functions require more coding, but can be significantly faster, especially when dealing with large datasets.

`netcdf.open` is used to open a NetCDF file for reading. Always the first step when using low-level functions:

```
ncID = netcdf.open(fileName)
```

`netcdf.inqVarID` is used to retrieve the ID of a variable:

```
varID = netcdf.inqVarID(ncID, varName)
```

`netcdf.getVar` is used to read variable data:

```
varData = netcdf.getVar(ncID, varID);
```


Important low-level functions

`netcdf.getAtt` is used to read variable attribute:

```
varUnit = netcdf.getAtt(ncID, varID, attName)
```


Important low-level functions

`netcdf.getAtt` is used to read variable attribute:

```
varUnit = netcdf.getAtt(ncID, varID, attName)
```

`netcdf.inqVarFill` is used to retrieve a variable's fill value (equivalent of `NaN` in MatLab):

```
[~, fillVal] = netcdf.inqVarFill(ncID, varID)
```

See also: https://www.mathworks.com/help/matlab/ref/netcdf.html

Example: Reading simulated bottom oxygen from `.nc` file

We will use both the high-level and the low-level functions and compare them with respect to time spent reading data from `roms854_MCH_bottom_O2_2000-2016.nc`, which contains daily model output for bottom oxygen in the northern Gulf of Mexico for 2000–2016 (i.e. 17 years).

Model grid size: 128 $\times$ 64 (x $\times$ y)
Time steps: 6211 days

Follow along the script: `netcdf_oxygen.m`

Read bottom oxygen using high-level functions

```
%% 1) first: using high-level functions

%% 1.1) read meta information
lonData = ncread(oxyFile, lonVar);
latData = ncread(oxyFile, latVar);
timeData = ncread(oxyFile, timeVar);
timeUnit = ncreadatt(oxyFile, timeVar, 'units');
```


```
%% 1.2) convert time unit
switch timeUnit(1:3)
    case 'sec'
        timeFac = 1/86400;
    case 'min'
        timeFac = 1/1440;
    case {'hou', 'hrs'}
        timeFac = 1/24;
    case 'day'
        timeFac = 1;
    otherwise
        error('Invalid time unit.');
end
```


```
%% 1.2) convert time unit
switch timeUnit(1:3)
    case 'sec'
        timeFac = 1/86400;
    case 'min'
        timeFac = 1/1440;
    case {'hou', 'hrs'}
        timeFac = 1/24;
    case 'day'
        timeFac = 1;
    otherwise
        error('Invalid time unit.');
end

% get the reference time used in the NetCDF file
i = strfind(timeUnit, 'since') + length('since');
refTimeStr = strtrim(timeUnit(i:end));
refTimeNum = datenum(refTimeStr);
% do time conversion
timeData = refTimeNum + timeData*timeFac;
% convert the day counter into a date vector
timeVec = datevec(timeData);
```


```
%% 1.3) read oxygen data for selected period
iTime = timeVec(:,1)>=yearStart & ...
        timeVec(:,1)<=yearEnd;
tStart = find(iTime==1, 1, 'first');
tCount = sum(iTime);
oxyTime = timeVec(iTime,:);
```


```
%% 1.3) read oxygen data for selected period
iTime = timeVec(:,1)>=yearStart & ...
        timeVec(:,1)<=yearEnd;
tStart = find(iTime==1, 1, 'first');
tCount = sum(iTime);
oxyTime = timeVec(iTime,:);

%% 1.4) define the start and number of indices
%       for each dimension to be read
%       NOTE: time is often the last dimension of
%             NetCDF variables, so it is here
ncStart = [  1,   1,   1, tStart];
ncCount = [Inf, Inf, Inf, tCount];
```


```
%% 1.5) read all data at once and time it
time2 = tic;
oxyData1 = ncread(oxyFile,oxyVar,ncStart,ncCount);
fprintf('Single read command: %.1fs\n', ...
        toc(time2));

%% 1.6) for comparison read data for all days
%       individually and time it
oxySize = size(oxyData1);
oxyData1 = NaN(oxySize);
time2 = tic;
for it = 1:tCount
    ncStart = [  1,   1,   1, tStart + it - 1];
    ncCount = [Inf, Inf, Inf, 1];
    oxyData1(:,:,:,it) = ...
       ncread(oxyFile, oxyVar, ncStart, ncCount);
end
fprintf('Data read sequentially: %.1fs\n', toc(time2));
```


Read bottom oxygen using low-level functions

```
%% 2) second: using low-level functions
%% 2.1) open NetCDF file
ncID = netcdf.open(oxyFile, 'NOWRITE');

%% 2.2) read meta data: longitude, latitude and time
lonID = netcdf.inqVarID(ncID, lonVar);
lonData = netcdf.getVar(ncID, lonID);
latID = netcdf.inqVarID(ncID, latVar);
latData = netcdf.getVar(ncID, latID);
timeID = netcdf.inqVarID(ncID, timeVar);
timeData = netcdf.getVar(ncID, timeID);
```


```
%% 2.4) get information about the variable we
%  to read
oxyID = netcdf.inqVarID(ncID, oxyVar);
[~, ~, oxyDimIDs, ~] = netcdf.inqVar(ncID, oxyID);

% check the variables dimensions
nDims = length(oxyDimIDs);
dimLength = zeros(1,nDims);
dimName = cell(1,nDims);
for i = 1:nDims
    [dimName{i}, dimLength(i)] = ...
        netcdf.inqDim(ncID, oxyDimIDs(i));
end
```


```
%% 2.4) get information about the variable we
%  to read
oxyID = netcdf.inqVarID(ncID, oxyVar);
[~, ~, oxyDimIDs, ~] = netcdf.inqVar(ncID, oxyID);

% check the variables dimensions
nDims = length(oxyDimIDs);
dimLength = zeros(1,nDims);
dimName = cell(1,nDims);
for i = 1:nDims
    [dimName{i}, dimLength(i)] = ...
        netcdf.inqDim(ncID, oxyDimIDs(i));
end

%% 2.5) get the variable's 'FillValue'
%       (needed to set land values to NaN
%        => not always done automatically)
[~, fillVal] = netcdf.inqVarFill(ncID, oxyID);
```


```
%% 2.6) Find the time indices we're interested in.
iTime = timeVec(:,1)>=yearStart & ...
        timeVec(:,1)<=yearEnd;
% NOTE: subtract 1 as netcdf toolbox starts
%       indexing at 0
tStart = find(iTime==1, 1, 'first') - 1;
tCount = sum(iTime);
oxyTime = timeVec(iTime,:);
```



```
%% 2.6) Find the time indices we're interested in.
iTime = timeVec(:,1)>=yearStart & ...
        timeVec(:,1)<=yearEnd;
% NOTE: subtract 1 as netcdf toolbox starts
%       indexing at 0
tStart = find(iTime==1, 1, 'first') - 1;
tCount = sum(iTime);
oxyTime = timeVec(iTime,:);

%% 2.7) set the starting indices and number of
%       indices for each dimension to be read
ncStart = zeros(1,nDims);
ncCount = dimLength;
ncStart(strcmp(dimName,timeVar)) = tStart;
ncCount(strcmp(dimName,timeVar)) = tCount;
```

```
%% 2.8) read all relevant data at once and time
time2 = tic;
oxyData2 = ...
   netcdf.getVar(ncID, oxyID, ncStart, ncCount);
fprintf('Single read command: %.1fs\n', ...
        toc(time2));
```


```
%% 2.8) read all relevant data at once and time
time2 = tic;
oxyData2 = ...
   netcdf.getVar(ncID, oxyID, ncStart, ncCount);
fprintf('Single read command: %.1fs\n', ...
        toc(time2));

%% 2.9) for comparison read data for all days
%       individually and time it
oxySize = size(oxyData2);
oxyData2 = NaN(oxySize);
time2 = tic;
for it = 1:tCount
    ncStart = [   0,   0,   0, tStart + it - 1];
    ncCount = [dimLength(1:3), 1];
    oxyData2(:,:,:,it) = ...
       netcdf.getVar(ncID, oxyID, ncStart, ncCount);
end
fprintf('Data read sequentially: %.1fs\n', toc(time2));
netcdf.close(ncID); % close NetCDF file
```


```
%% 3) in biogeochemical models, oxygen can often
%  become negative representing production of H2S
%  (hydrogen sulfide) under anoxic conditions
%  => set negative values to ZERO
oxyData1(oxyData1<0) = 0;
oxyData2(oxyData2<0) = 0;
```

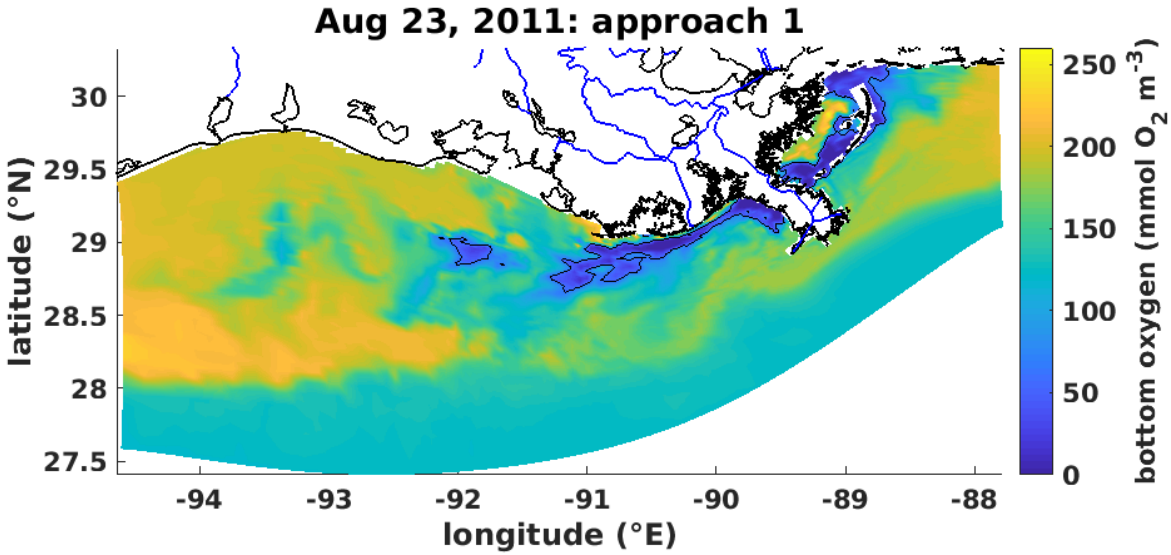

Getting coastline from global database

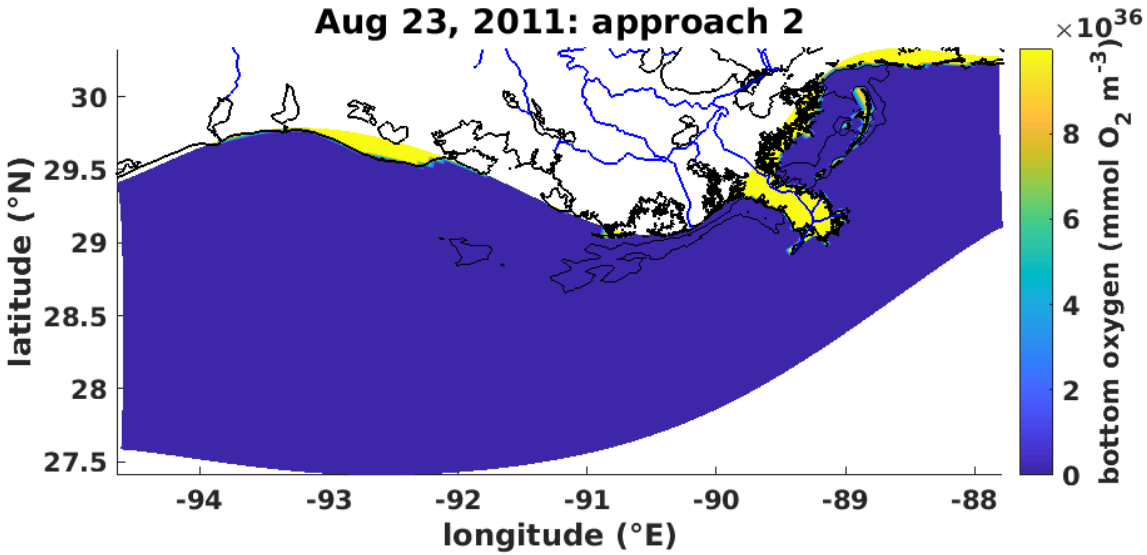
```
%% 4) get coastline (using 'gshhs' function)
lonRange = [nanmin(lonData(:)),nanmax(lonData(:))];
latRange = [nanmin(latData(:)),nanmax(latData(:))];
if  ~exist(coastFile, 'file')
    % extract coastline and land mask for region
    % of interest from GSHHG data
    mch_coast = gshhs('gshhs_h.b', latRange, lonRange);
    mch_river = ...
       gshhs('wdb_rivers_h.b', latRange, lonRange);
    save(coastFile, 'mch_coast', 'mch_river', '-v7.3');
else
    load(coastFile); % use existing file
end
```

See also:
https://www.ngdc.noaa.gov/mgg/shorelines/gshhs.html
https://www.ngdc.noaa.gov/mgg/shorelines/data/gshhg/latest/gshhg-bin-2.3.7.zip (files used by MatLab function `gshhs`)
https://www.mathworks.com/help/map/ref/gshhs.html

Plot data read with high-level and low-level functions

```
%% 5) Plot figure with original data
```





```
%% 6) remove land values and plot again
oxyData2(oxyData2>=fillVal) = NaN;
```