1 Introduction

In this lab we will investigate arrays, do some simple kinds of array indexing, and make different kinds of plots (read pages 443-456 of the text for info on plotting!). We will give you brief code "snippets" which can be copied. You should learn what is going on by using the `help` command (or the browser-accessible help pages).

At the end of the lab you will write a short script which will make some plots and this script will be handed in. Note that this script will require you to put together ideas you have developed in the earlier parts of the lab.

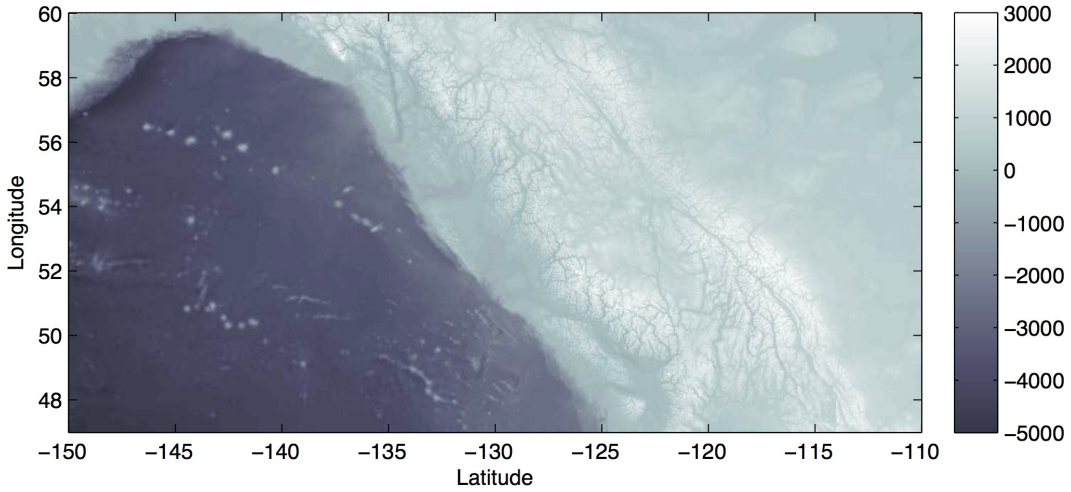

Figure 1: West coast Topography

Learning goals

(1) To download raster image data from the Web and import it into MATLAB
(2) describe the basic parent/child structure in handle graphics
(3) To write short snippets of code that will perform various actions. These include:
  (1) -displaying raw images and changing the colourmap
  (2) -retrieving subsections of an array
  (3) -performing simple mathematical operations on a vector
  (4) -displaying a contour plot
  (5) -using handles, get and set property/value pairs to customize graphics.
(6) To build up a larger program by continually modifying a small 'starter' program.
(7) To write a short program that performs specified actions.

NOTE: When people run into problems in this lab, it is generally because they have skipped ahead of early sections, or failed to answer questions in some part because they seem "too easy".

2 Ocean Colour

There is a vast quantity of earth sciences data online. Many servers provide a web interface to plot the data, and often there is also a way to download the actual data. Open your web browser to the NASA Ocean Colour Web site (`https://oceancolor.gsfc.nasa.gov`) and go to the "Data" link at the top of the left hand menu bar, then go to the "LEVEL 3 Browser" link. Thumbnail images will appear with 4 pulldown menus. Leave the leftmost one set to its default ("standard"), and the 3rd one at "monthly".

The major products of ocean colour satellites (CZCS, OCTS, SeaWiFS, MODIS, and now VIIRS/SNPP satellites) are chlorophyll (a measure of phytoplankton biomass) and sea-surface temperature (SST). Look around and find a nice SNPP VIIRS Chlorophyll or Sea Surface Temperature image at some interesting date (month of your birthday?). **One of the MONTHLY images tends to work better in this lab because you can easily see the difference between land and water regions**.

Select the 9km resolution (right hand menu bar), and click on the thumbnail to open a rather large image in your browser. Right-click on the image to save it to your disk (if you accidentally click near the lower left or right you will be asked if you want a binary file with a .nc extension - this is NOT what you want).

Now, load and display the file using

```
[A,CMAP]=imread('your_filename.png'); % use your file name here.
image(A);
xlabel('Column Index');
ylabel('Row index');
colormap(CMAP)
```

The matrix `A` is pretty large (how large? - check using the `size` function, or look in the Workspace window). Points on the same latitude are in the same row (first dimension), and points with the same longitude are in the same column (2nd dimension). But what does the data in `A` actually represent? Where do those greens and blues come from?

All colours on the computer screen are generated as a combination of only 3 colours - red, green, and blue (RGB)*. We can represent any colour as some combination of the 3, each with a fraction between 0 and 1 - say, by a 3-element vector `[.3 .6 .2]` which means 30% red, 60% green, and 20% blue. Black would be `[0 0 0]` and white is `[1 1 1]`. The 256×3 variable `CMAP` holds 256 of these colours, and the values in `A`, which are between 0 and 255 (inclusive) are an index into this color table. The `colormap()` function sets the colours to be used in the current figure. To see the colours plotted out by their index in a horizontal 'colour bar', type:

```
figure(2)
image(uint8(0:255))
colormap(CMAP);
```

The ":" is a special OPERATOR which is very useful in MATLAB. We saw it in class and lab last week, but type `help colon` to understand what it does if you've forgotten.

You can look at smaller areas by ZOOMING in and out using the GUI controls at the top of the figure. Instead, let us make a figure of a smaller area by **extracting a subset** of the whole global image. In the plot, the row and column indexes are marked on the axes. So, for example, we can show only the western North Atlantic in a new figure window using

```
WNA=A(300:650,1350:1700);
```

*other colour models such as HSV and CMYK also exist

```
figure(3)
image(WNA);
colormap(CMAP);
```

(Why those numbers?). Now extract and display an image of the ocean near Vancouver, instead of the western North Atlantic. Hint: think about what rows and columns you need. Notice what happens if you don't include the line `colormap(CMAP)` when you make your figure.

Now you know how to pull out pieces of an image. But this isn't just an image, it's actually geographic data. It would be nice if we could have axes labelled in longitude and latitude.

How is this done? By "telling" the `image` command what the x- and y- values of the rows and columns in the matrix are, instead of letting it default to labelling things by the row and column indexes. For the x-axis, we can make a vector with the same number of elements as there are columns in `A`. The first element of the vector will contain the numerical value of the longitude of the westernmost column, and the last element will contain the longitude of the easternmost. Similarly, for the y-axis we need a vector containing the numerical latitudes of all rows, with the first element giving the latitude of the northernmost row.

```
nA=size(A);
lat=  90-180*[1:nA(1)]/nA(1);
lon=-180+360*[1:nA(2)]/nA(2);
image(lon,lat,A);
xlabel('Longitude');
ylabel('Latitude');
% The next line changes the direction of the y axis so numbers increase upwards
set(gca,'ydir','normal');
```

What is happening in the first line? The `size` function RETURNS an array of values - the number of rows and columns of `A` - and assigns these values to the 1×2 variable `nA`. How about the next two lines? What numbers end up in the `lat` and `lon` variables?

Also, what's up with that line with the `set()` command? We will talk about `set()` again later in the term when we look at making more sophisticated plots, but for now here's a summary:

In MATLAB, figures are organized as OBJECTS, which can be addressed with a HANDLE. Each object can have PROPERTIES and CHILDREN. Properties affect the look-and-feel of an object. Children are other objects, associated with the current object (huh?).

The children of a figure window are the different subplots in it, and the children of individual subplots will be the lines and images displayed on it.

Properties of a figure include the colour of the window (default gray). Properties of a subplot include the axis limits, and the ticklabels. Properties of a line include its colour and thickness.

Properties are accessed using property/value pairs - a NAME, followed by its VALUE. The `set` command lets you change properties. `gca` is a function returning the HANDLE of the "current axis" (get-current-axis). `ydir` is a property of the axis, and `'normal'` is a string telling which direction you want y-axis values to increase upwards (the default with `image` is to increase downwards).

`get(gca)`	tells you the current setting of axes properties
`set(gca)`	tells you the different options for axes properties

Now make a plot of the Vancouver area with the correct longitude and latitudes on the axes (**Hint: you have to**

extract subsections of the `lat` **and** `lon` **arrays as well**). Vancouver is near 49°N, 123°W (or -123°E).

You can provide a title for the plot using the `title` function, and labels for the x and y axes using `xlabel` and `ylabel`. Do so. It is good practice to properly label axes and you should always do so in this course (especially when handing things in!). Strictly speaking we also probably want to label longitudes as E and W rather than positive and negative but this is a more advanced topic.

3 Global Topography

The png image we used in the last part was already *scaled*, so e.g. if you chose a chlorophyll map, the real world values with units of concentration between 0 and 30 mg m^{-3} had already been turned into integers between 0 and 255. To do any science with these maps we would need to refer to the colorbar that showed the original maximum and minimum values (see this bar at the bottom of the L3 data browser). In this section we will deal with *unscaled* data.

First, we will examine the topography in our part of the world. Clear the workspace using the `clear` command. Now download the data file `Bathyfile.mat` from the course web site and read it into MATLAB (recall loading `.mat` files in the last lab). Note that you have one new variable, the STRUCTURE `bath`. What are its subfields?

`bath.height` contains a matrix of elevations (heights above sea level), not RGB values. How will matlab show this? The easiest way is to use a variant of `image`:

```
figure(1)
imagesc(bath.long,bath.lat,bath.height);
xlabel('Longitude');
ylabel('Latitude');
colorbar;
set(gca,'ydir','normal');
```

The default colourmap in MATLAB versions 2014a and earlier is `jet` which goes from dark blue for lowest values to dark red for largest values. In MATLAB versions 2014b and later the default is `parula` which goes from dark blue for the lowest values to yellow for the highest values. Try some others, e.g. the `hsv` colormap:

```
colormap(hsv)
```

and try switching between `parula` and `jet` and see if you think MATLAB should have changed the default! (MATLAB also has other predefined colourmaps, see if you can find and use them).

The `imagesc` function SCALES the data values into the range of colourmap indices. We can modify this mapping with the `caxis` function. For example, this will show only the mountains:

```
caxis([0 4200]);
```

What are the actual heights in different places? What we want to do here is to a) find the row/column INDEX of a desired location, and b) extract that value from the array. The `ginput` function lets you use the mouse to get a location on the figure axes. `ginput` works with row and column numbers NOT with real-world x,y or in this case longitude and latitude values. So first make a figure showing the elevations with just row numbers and column numbers plotted. Notice that as before row numbers increase from top to bottom .

```
figure(2)
imagesc(bath.height);
```

Now cut and paste this snippet into the command window.

```
% Pick some points and label them
  [I,J]=ginput(1);
  I=round(I);
  J=round(J);
  bath.height(J,I)
  text(I,J,num2str(bath.height(J,I)));
```

What does it do? (Hint: Click a mouse button when the mouse is over the image). Now go through this line by line and decide what is going on. What is contained in the variables `I, J, bath.height(I,J)`? We will see more of `ginput` in future labs. What matlab code would you write to get the latitude and longitude of the point that you have clicked? Check your answer looks about right by looking at Figure 1.

Finally, extract a row from the grid of elevations, and make a line plot

```
plot(bath.long,bath.height(256,:));
```

What latitude is this row? Put it in a title and add x-axis and y-axis labels. Lastly, add a zero line to your plot to show where sea-level is. To do this you'll first need to turn on the hold to prevent the figure from being redrawn, and then draw a line at the zero-level:

```
hold on;
plot([bath.long(1) bath.long(end)], [0 0], 'k')
```

Experiment to see what happens if you don't include the `hold on` command, or if you replace `'k'` with `'r'`.

4 Ocean Salinity

First, `clear` the workspace. Then, get the file `salt.txt` from the course website. Look at it in an editor. It is made up of columns of numbers, some of which are `NaN`. You can load this text table into a matrix S as follows:

```
S=load('salt.txt');
```

`S` has 34 rows and 42 columns, organized into a table. Columns `2:42` of row 1 are the longitudes of data below them in the same column. Rows `2:34` of column 1 are the depths of the data in the rest of the row. Columns `2:42` of rows `2:34` are salinity values (in grams per kg seawater), one for each particular depth and longitude, at a latitude of $51.5°$N.

Extract the depth, longitude, and salinity values into subfields of a structure `sal` so that

`plot(sal.salinity,sal.depth)` shows depth profiles of salinity, and

`plot(sal.long,sal.salinity)` shows curves of salinity at different depths as a function of longitude.

NOTE: that this is a special case of plotting. We talked last week about the fact that typically when plotting two quantities x and y, the arrays `x, y` must have the same dimensions. This is obviously an exception because `sal.longitude, sal.depth and sal.salinity` do not have the same dimensions but MATLAB has still produced a plot that makes sense. What do you think has happened here? We will see more of this later in the course.

Next, make a contour map of this data using

```
contour(sal.long,sal.depth,sal.salinity,[31.5:.1:35]);
```

or (perhaps more clear)

```
contourf(sal.long,sal.depth,sal.salinity,[31.5:.1:35]);
```

What does the last argument to the `contour` command do? Notice that the contours don't extend over the whole region of your plot. This is because `sal.salinity` contains `NaN` in some elements (why do you think this is in this problem?) and so MATLAB does not attempt to draw contours in these regions.

Finally, add the curve of elevations you plotted at the end of the last section using

```
plot(bath.long,bath.height(256,:));
```

Since you cleared the workspace, you will have to reload `Bathyfile.mat` before this works. Also, you will have to use the `hold on` command to prevent the `plot` function from erasing the contour plot. You can manually set the axis limits using the `axis` command, or using `ylim` and `xlim` if you get it wrong `axis auto` may be useful in tracking down the problem.

5 To Hand In by 4:00 pm on Friday, September 21st

You **NEED NOT** hand in **EVERYTHING** you have done. Instead, you must gather together the **relevant** bits of code you have already written and write and hand in a script file called (exactly) `lab3.m` which will do EXACTLY the following (no less and no more):

(1) load the file `Bathyfile.mat`
(2) load the file `salt.txt`, and extract its data into a structure
(3) Make a plot with axis labels and a title showing contours of salinity (as a function of longitude and depth). Use `contourf`
(4) Overplot on it lines showing bathymetry (heights and depths) as a function of longitude at $48°$N, $51.5°$N, and $55°$N latitudes.
(5) `lab3.m` should also contain the following (and exactly the following) lines of working code:

```
partner.name='YYYYYY';
Time_spent= XX;
```

where YYYYYY (string) is the person with whom you were paired in the labs (remember to put single quotes around the name) and and XX (number) is your estimate of the **number of hours** spent both in the scheduled lab period and outside it to complete this lab.

It is important to follow these instructions because your code will be semi-automatically "run-tested" by a program that expects your code to follow these specifications.

Hint: To make sure your script works after adding the partner info - **save**, quit Matlab, restart Matlab, and type `lab3` at the command line.