Learning Goals

- To understand the process behind the nine rules of debugging
- To read and understand someone else's code.
- To use the 5 methods of finding errors.
- To use DEBUGGER to step through code.
- To solve a problem useful in the earth sciences.

Useful MATLAB info - the Debugger

The DEBUGGER is a program that allows you to run code a bit at a time so that you can see how variables change and the flow of control occurs (or goes wrong).

There is a useful tutorial called Debug a matlab program which you should look at. The important steps in using the debugger are:

- SET (one or more) breakpoints. These can be placed at specific lines of code clicking right beside the line numbers so a red dot shows. They can also be set to occur only when certain things happen, like a warning, or a divide-by-zero (using the "Breakpoints" Menu pull-down).
- RUN the code. When it reaches a breakpoint execution stops, a red dot in the editor will show the line to be executed next, and a special command line (`K>>` appears in the command window. You can now investigate the status of variables by
  - typing out their names (or looking in the workspace variable window).
  - plotting them (using the `plot()` function).
  - doing a little math with them.
  - viewing the values of arrays in the array editor, or by moving the mouse near a variable name in the editor window...
  - ...or really doing anything that you are used to doing in the command window.
  - and you can look in the workspace of the calling function using `K>>dbup` (returning with `K>>dbdown`), or use the Function Call Stack in the Debug menu.
- CONTINUE running the code either by running until the next breakpoint (menu or `K>>dbcont`), or by stepping through lines of code one by one (Step or Step In to "step into" a function).
- Quit debugging mode (menu or `K>>dbquit`)

Background Information:

I want to make a map of ocean surface currents. Looking around on the web, I found an oceanographer who has produced a set of gridded current data, derived from ship drift records. The dataset is a `.csv` (comma-separated value file called mgsva_MJJ.csv May/June/July". I have also started to write code (two functions `plotarrows.m` and `mean2d.m`) to make some maps of this data, but the code has errors. It will be YOUR job to fix these errors.

Algorithm description

First we have to make up matrices holding the data. Download the lab9.zip file and unzip it. It contains `mgsva_MJJ.csv`, plotarrows.m, mean2d.m and the driver main script called testmap.m Take a look look at `msgsva_MJJ.csv` in the Matlab editor. It is just a list of 42730 lat/long points, and for each point there is an associated pair $\{u, v\}$ (in columns 4 and 5) of eastward and northward velocities (in m/s). Fortunately it seems that these are not at random locations. Instead they appear to be at points on a regular grid, spaced at 1 degree increments in latitude and longitude (columns 3 and 2). However, the land points are missing (because there is no data there), which is why there are only 42730 points rather than $360 \times 180 = 64800$ points.

So, the first thing done in `plotarrows()` (i.e., putting the data into rectangular matrices) is done by calling the subfunction `move_to_grid()`. Individual longitudes will be along the columns and individual latitudes along the rows of each matrix (like the `topo` data in previous labs). Land locations should contain `NaN`.

Then, we want to smooth the data a bit. We have dealt with running mean windows in 1D (for time series). This is generalized to a 2D case, which I am implementing with the function `mean2d()`. For a 'window size' of `N` (again with `N` odd) average the points in an `NxN` square centered on the location of the actual data point (draw a picture for yourself to make sure you understand how this works). For example, for a window size of 3, the 9 points within a 3×3 square surrounding the data point are averaged together.

The code is supposed to take care of 3 kinds of 'edge effects':

- We have to wrap in longitude in the same way that we did in lab 5.
- We do NOT wrap in latitude (if you go too far north or south). The easiest thing to do here is to exclude the 'nonexistent' row of points - i.e. if you are on the northern or southern boundary with a window size of 3, use a 6 point (2 rows by 3 cols) instead of a 9 point (3 by 3) average. This is a little like the edge effects you worked on in lab 6.
- If your window contains land squares, you ALSO want to exclude these from the average. That is, if (say) the top left corner of a 3×3 square is on land, you calculate the average over only the other 8 points which are not over land.

Finally, back in `plotarrows()`, we use `imagesc()` to make a colour image of the velocity

MAGNITUDE

$$M = \sqrt{u^2 + v^2}$$

and then overplot (*A*) a coastline, and (*B*) a SUBSAMPLED set of the velocity vectors as little sticks that begin at the lat/long location of the velocity vector, and then point away from it in the direction of the current. The data file `m_coasts.mat` needed to draw the coastline is also available from the class website.

To implement the subsampling, we use a 'decimating factor' to index into an evenly spaced subset of the data. The decimating factor `decfac` is an input argument to the function `plotarrows()` and is simultaneously used as the window size for the 2D average.

The Lab:

The code in `plotarrows.m` and `mean2d.m` attempts to implement the algorithm. But right now it contains a number of bugs. Some of these are SYNTAX errors, some are ALGORITHM errors, and some are INPUT errors. In this lab you should:

1. Review the nine rules of debugging
2. Carefully examine the code. Does it match the description of what I have said it should do?
3. Make it fail – i.e. comment out enough code so that something runs, then uncomment code until the script fails and you have a minimal reproducible example.
4. Save your working version–this is the next rung in your ladder.
5. Debug the code until the new addition works as you expect
6. Save this working version
7. Uncomment until you get another failure and repeat the process
8. Hand in the debugged code.

To Hand In:

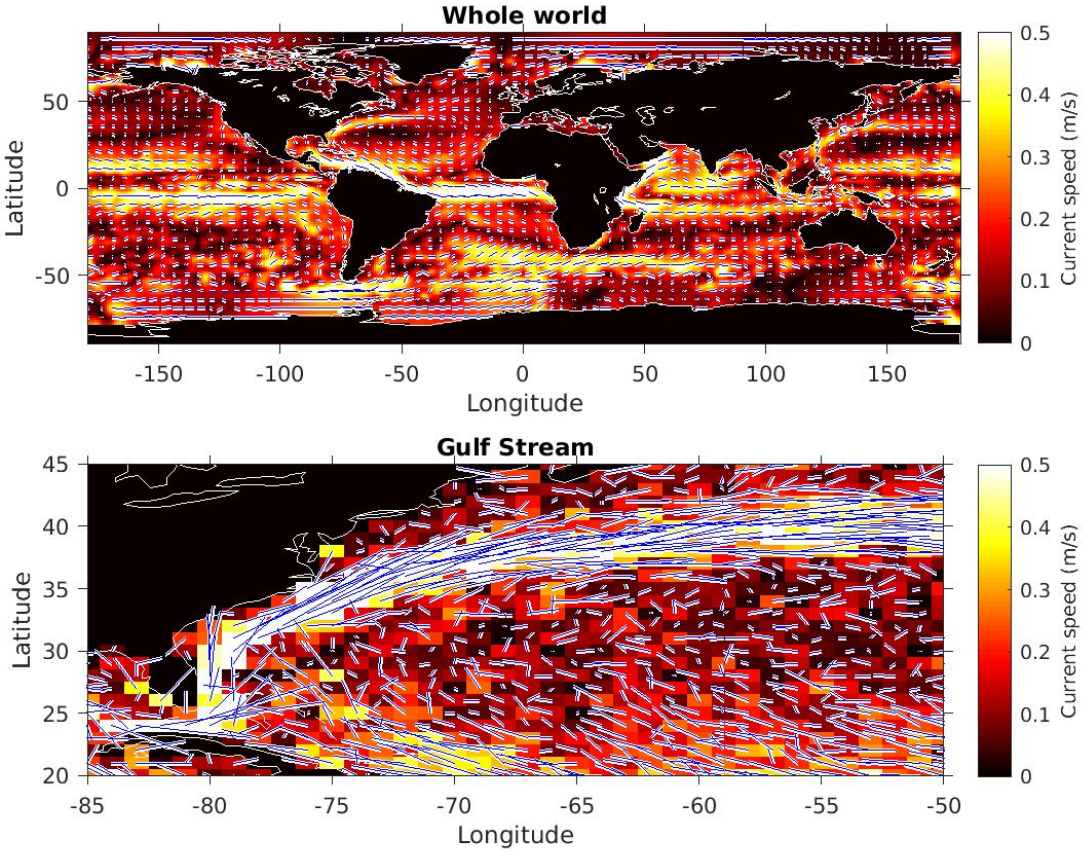
Submit your corrected files `plotarrows.m` and `mean2d.m` on Connect by Friday at 4pm. **COMMENT YOUR CORRECTIONS!** I will test them with the following code. It should make the attached plot.

```
% Test the 'debugging lab'  code
clf;
subplot(211);

plotarrows('mgsva_MJJ.csv',5,10);
title('Whole world');

subplot(212);

plotarrows('mgsva_MJJ.csv',1,7);
axis([-85 -50 20 45]);
title('Gulf Stream');
```

Include the following two lines in `plotarrows.m` as comments, at the bottom of the header comment lines:

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

with the correct information substituted. Remember to rerun your code after adding the partner information to make sure this doesn't break your code!


Whole world
Latitude
Longitude
Current speed (m/s)
50
0
-50
-150
-100
-50
0
50
100
150
0.5
0.4
0.3
0.2
0.1
0
Gulf Stream
Latitude
Longitude
Current speed (m/s)
45
40
35
30
25
20
-85
-80
-75
-70
-65
-60
-55
-50