Dave's very quick MATLAB tutorial

You have 4 basic windows in MATLAB. There is the **command window** which allows you to type instructions directly into MATLAB. There is also a **command history window** which remembers all of your commands (WARNING if you are on a shared computer try to have the command history saved somewhere). The **current directory window** just shows which folder you are in on the computer and the **workspace window** shows you what variables that are currently active (i.e. the variables you are working with right now). In Figure 1 you can see that I am working with three variables "Dalec801", "data" and "textdata" – but you can also see lots of files in the current directory (AnnLitter.txt, Clim.txt etc.). These are text files and I could load any of them into MATLAB. We use text files to store data because they are simple, easy to store or compress, and because many different programs can use text files.

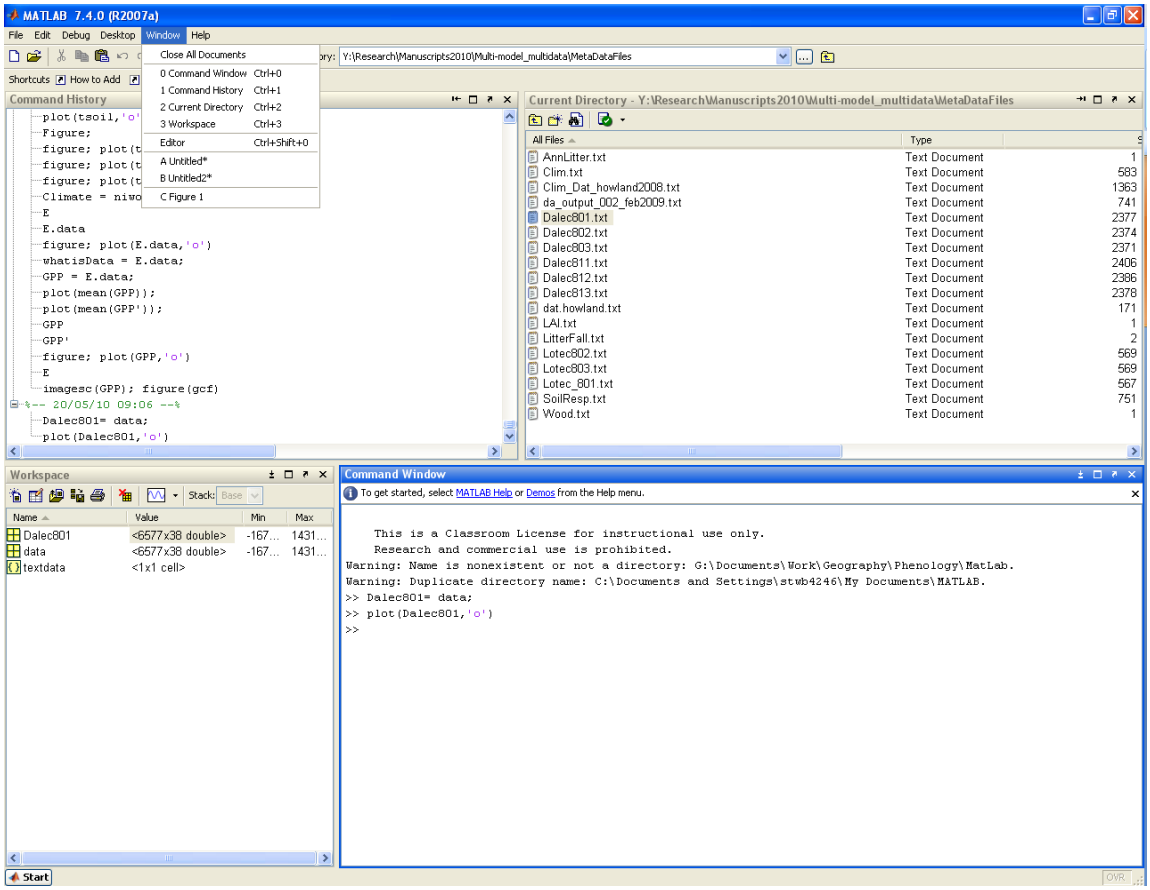

Figure 1: Matlab Windows – the layout can be configured by the user.

In the **workspace window** we see 3 matrices. It doesn't really matter what these data are at this point but Dalec801 has 6577 rows and 38 columns so it is listed as <6577x38 double>

- "double" is a datatype – it means that the value is a double precision floating point number taking up 8bytes and having a range of +/- 1.7e +/- 308 (~15 digits).

There are different data types which are used for different things. Depending on your use of MATLAB you may or may not end up using lots of data types. They are not that complicated. I bet you can guess what the "char" datatype is?

"char" is a character or small integer. You can find out more by googling "data type double" and exploring.

Open up your copy of MATLAB.

Type a simple mathematical problem in the command window and hit return – MATLAB should give you the answer.

Something like the following:

```
>> 10 +2

ans =

    12
```

Now ASSIGN the answer to a variable for example:

```
>> A = 10 + 2

A =

    12
```

Now type the name of the variable and hit return

```
>> A

A =

    12
```

Add something to the variable

```
>> C = A + 4

C =

    16
```

Can you work out how to multiply A by C ?

We can come back to assigning variables – you can read about it in the MATLAB documentation.

Since you probably want to analyse your own data with MATLAB - let's see how to import some data

The next section is From: Shoehorn on Oct17 2008 - http://www.physicsforums.com

There are lots of resources online describing how MATLAB and other

"You can get Matlab to generate almost any code you want. Here's a hint on how you might get Matlab to generate code to open a data file and assign headers to the variables.

Suppose that you have some text file, data.txt, which contains the data you talked about. For instance, the contents of data.txt might be something like the following:

Code:

```
Time,Value
1,2
2,2
3,4
4,6
5,7
```

Using the Matlab current directory window, browse to where this text file is located on your hard drive. Right-click on it and select 'Import Data'.

You'll now see a new window pop up. Since your data is in comma-separated value (CSV) form, make sure 'comma' is selected as column separator and click 'Next'.

On the next screen, make sure that 'Create vectors from each column using column names' and tick the 'Generate M-code' box also.

Click 'Finish' and you'll find that your data has been imported in just the format you asked for in the original post. You'll also notice that Matlab has generated an M-file containing all of the Matlab code that was used to import the data. Browse through the code and you'll see precisely what Matlab commands are necessary to generate variables using text data from the command line.

For instance, the M-file that Matlab generates for me when I perform the above procedure is:

Code:

```
function importfile(fileToRead1)
%IMPORTFILE(FILETOREAD1)
%  Imports data from the specified file
%  FILETOREAD1:  file to read

%  Auto-generated by MATLAB on 17-Oct-2008 22:26:03

% Import the file
newData1 = importdata(fileToRead1);

% Break the data up into a new structure with one field per column.
colheaders = genvarname(newData1.colheaders);
for i = 1:length(colheaders)
    dataByColumn1.(colheaders{i}) = newData1.data(:, i);
end
```

```
% Create new variables in the base workspace from those fields.
vars = fieldnames(dataByColumn1);
for i = 1:length(vars)
    assignin('base', vars{i}, dataByColumn1.(vars{i}));
end
```

Reading through this it should be obvious how to do what you want. If you wish, you can save this M-file so that you can repeat the entire procedure for other files simply by typing

Code:

```
importfile(<FILENAME>)
```

at the Matlab command prompt.

Now let's look at some data.

There is a file which accompanies this tutorial it's called "uptake.data"

Find it and put it in a folder that you can reach on your computer

Can you open this file by double clicking?

Try opening it with a simple text editor (EditPad Lite? WordPad? Notepad?)*

What does it look like?

Try to apply the instructions above to the data file which accompanies this document (uptake.data)

Remember to navigate to where the data is stored – so that the data file you want import is in your Current Directory Window

*For scientific computing some text editors are better than others – while MS Word might be good for word processing – programs like MATLAB, R or IDL will be confused by all the formatting tags stored in Word file – remember simple is better – keep it simple! – keep it safe! For macs try 'textedit', for windows try 'Editpad Lite' … but there are lots of options

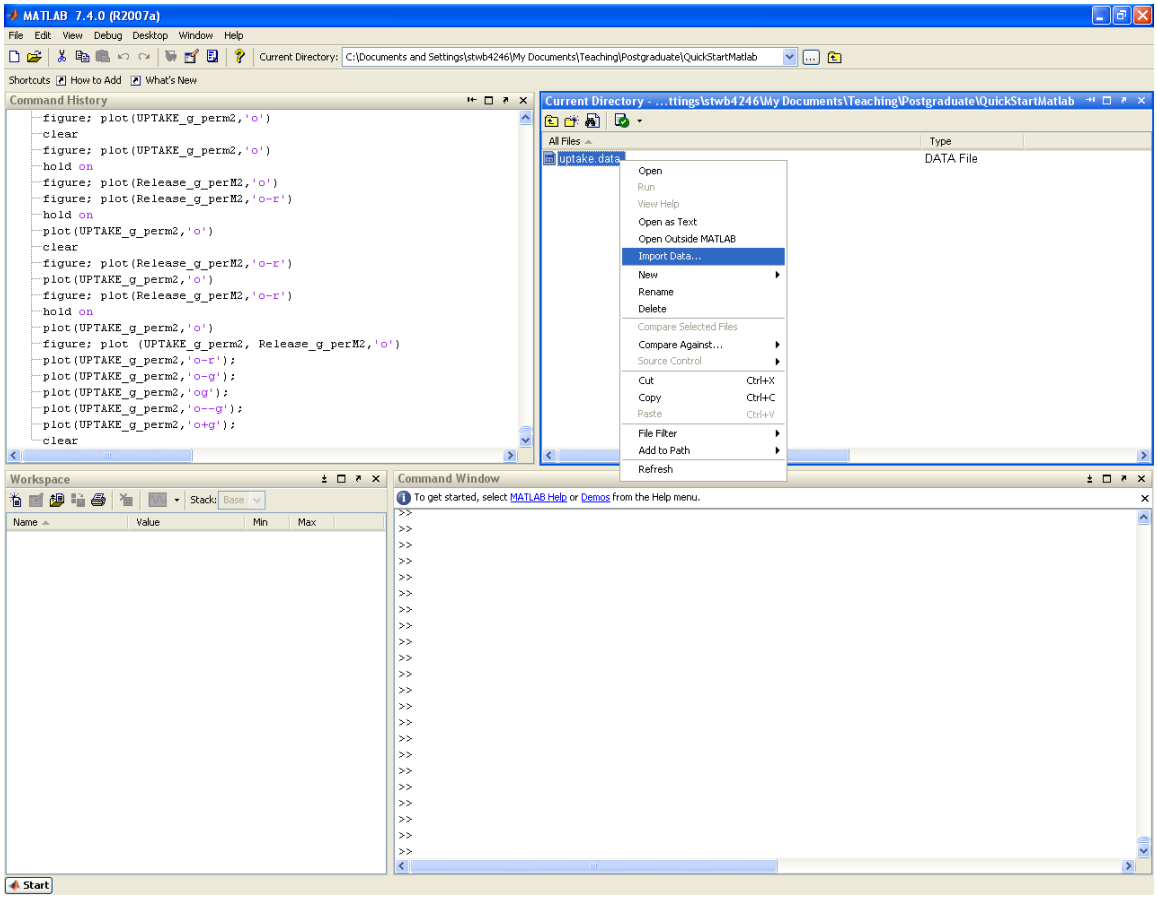

Figure 2: The file 'uptake.data' is in the Working Directory. If you right click on a data file in your working directory you get the options shown in the figure.

Navigate so that your working directory shows the file 'uptake.data'

Notice that the file extension is ".data" – I just made this up – it's not associated with any program on the computer. The file is a simple text file – I could have chosen to call it 'uptake.txt' or 'uptake.DaveFile' or 'uptake.eddardstark' if I wanted to. When we use tools like MATLAB or even a custom model to analyse our data we can use file extensions as a way to remind ourselves of what the file contains.

Right click on the file and select '*Import Data*' then follow the instructions remembering to check the box marked *'Create vectors from each column using column names'*

You should now have three new variables in your workspace. DOY, Release_g_perM2, UPTAKE_g_perm2. If you type 'clear' to clear the data and

You can manipulate these the data as you would manipulate numbers.

Type UPTAKE_g_perm2 into the command window

```
>> UPTAKE_g_perm2

UPTAKE_g_perm2 =

    0.0300
    0.3300
    0.4000
    0.2700
    0.4000
...
...
```

There should be a list of 26 values

In your workspace you should see UPTAKE_g_perm2 with some basic description beside it. It's a double float variable with 26 rows and 1 column

Now type `UPTAKE_g_perm2 + 2`

```
>>UPTAKE_g_perm2 + 2

ans =

    2.0300
    2.3300
    2.4000
    2.2700...
```

You get a similar list but each value is 2 units larger.

Try some other operations – multiply the values by 10, add DOY to Release_g_perM2 – you can explore to your heart's content.

Try this

```
NEE_matrix = [UPTAKE_g_perm2, Release_g_perM2]
```

What happened there?

If you have created a variable with 2 columns and 26 rows called NEE_matrix ... well how can we save it?

It's pretty simple

```
save NEEMatrix.txt NEE_matrix –ascii
```

If you type in the save code above you should create a Text Document in the Current Directory called "NEEMatrix.txt"

But there are lots of options

Type

```
help save
```

This should bring up some instructions on how to use the save command.

Let's plot some data

The 'figure' command creates a new (empty) figure

```
>>figure;
```

Try creating some plots by typing in the code below – one at a time:

```
plot(UPTAKE_g_perm2,'o');

figure; plot(UPTAKE_g_perm2,'o-r');

plot(UPTAKE_g_perm2,'o-g');

figure; plot(UPTAKE_g_perm2,'o--g');

plot(UPTAKE_g_perm2,'og');

figure; plot(Release_g_perM2,'o');
```

What does the bit of code in 'quotes' do in this command?

What happens when you use the figure command before the plot command? What happens when you leave it out?

Try the following

```
figure; plot(Release_g_perM2,'o-r');

hold on;

plot(UPTAKE_g_perm2,'o');
```

What happened there? Why?

And next try:

```
figure; plot (UPTAKE_g_perm2, Release_g_perM2,'o');
hold on
plot (UPTAKE_g_perm2, UPTAKE_g_perm2, '-r');
```

Type

```
help plot
```

(wait a minute)

No scroll up and read the help entry for the plot command.

This entry should explain what all the plot code from above is doing.

Remember you can also EDIT and SAVE the graphs using MATLAB's graphical user interface (GUI) – in other words you can click on bits of the graph an format it.

Give it a go.

Create a graph with Release_g_perM2 on the x-axis and UPTAKE_g_perm2 on the y-axis. Then click on 'Show Plot Tools and Dock Figure' (Fig 3).

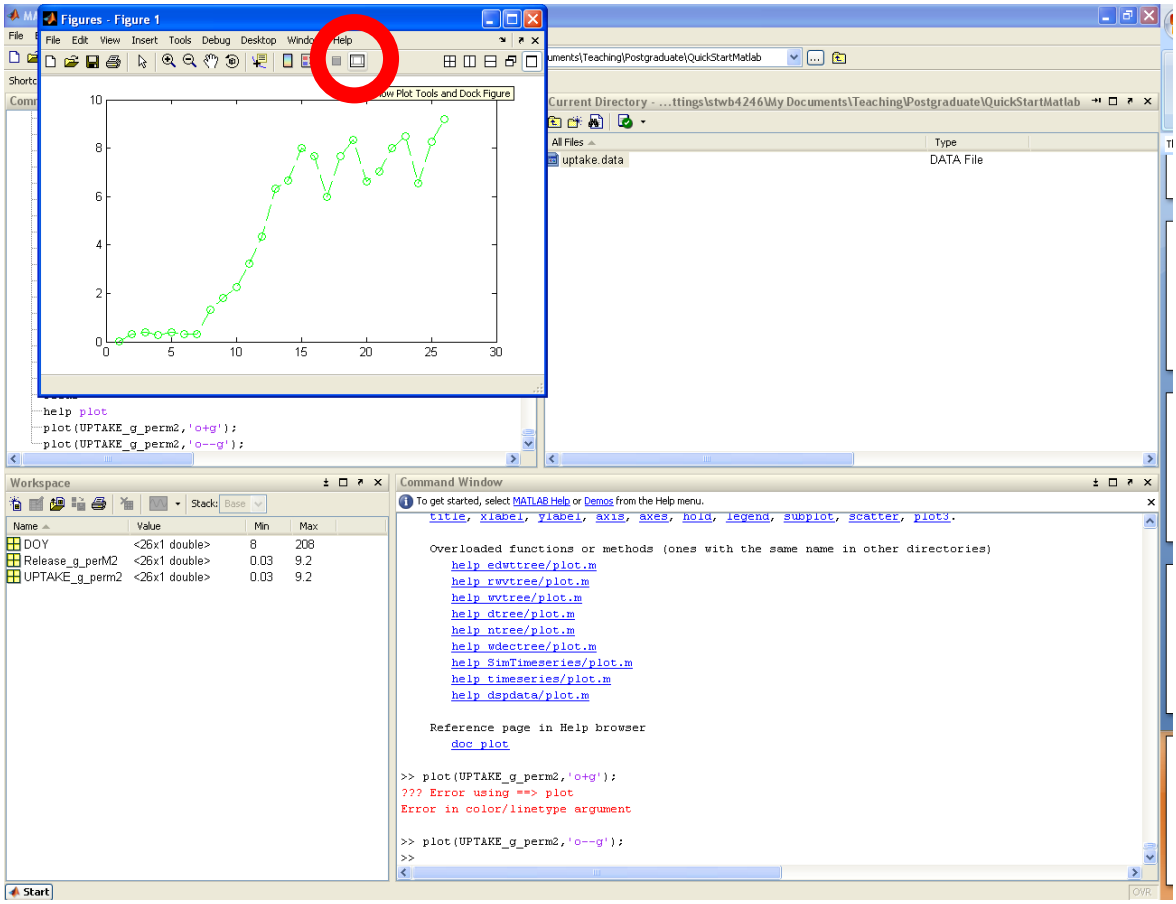

Figure 3: The big red circle shows the location of the 'Show Plot Tools and Dock Figure' icon – click it!.

When you have clicked on the *'Show Plot Tools and Dock Figure'* icon a Property Editor box will appear below the figure. At this point you can click on different elements of the graph and edit them (e.g. the axis, the symbols etc).

Play around with the GUI until you are comfortable creating a graph that would make Bruce proud.

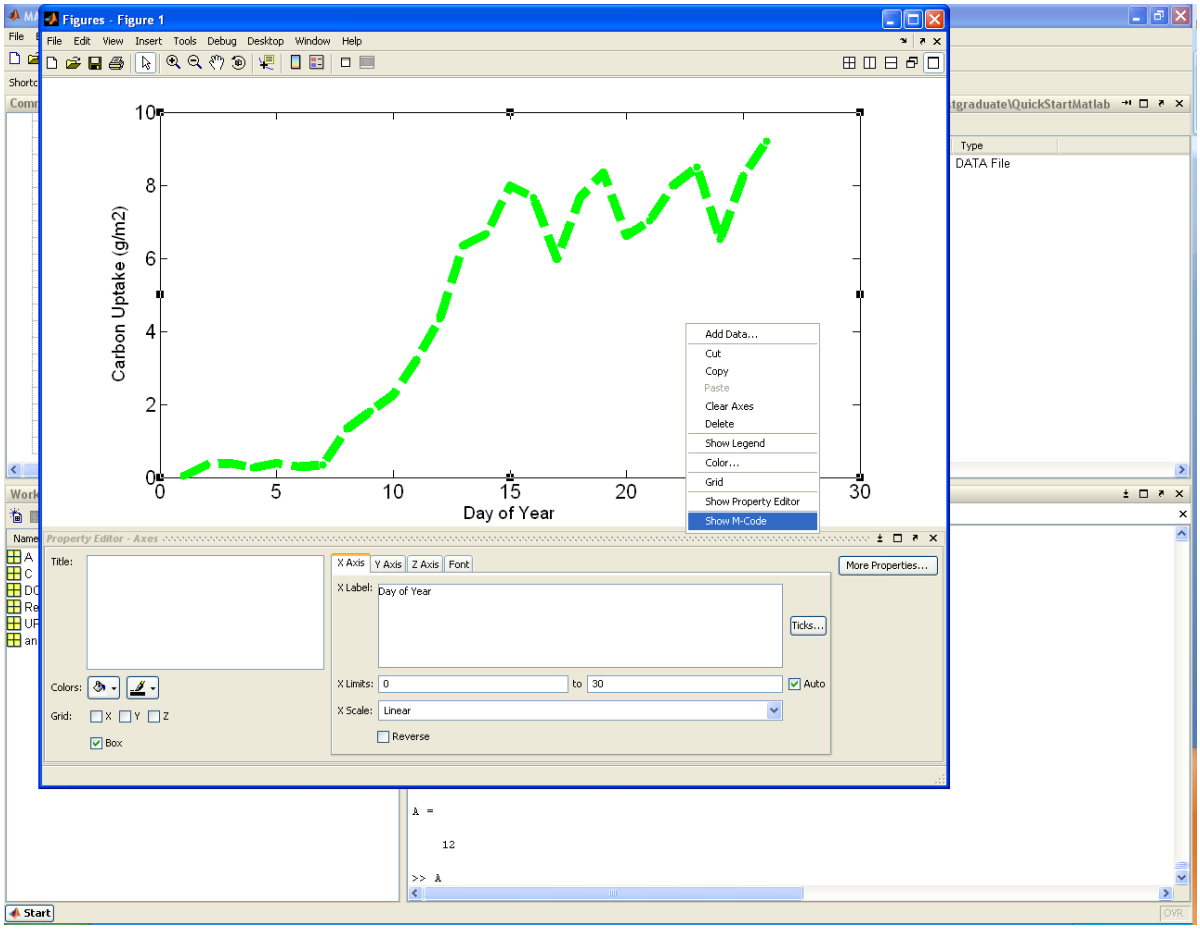

Figure 4: Expanded GUI for a MATLAB figure. When you right click on a figure you get various options including the ability to see the MATLAB code (or M-Code) which can be used to generate the figure.

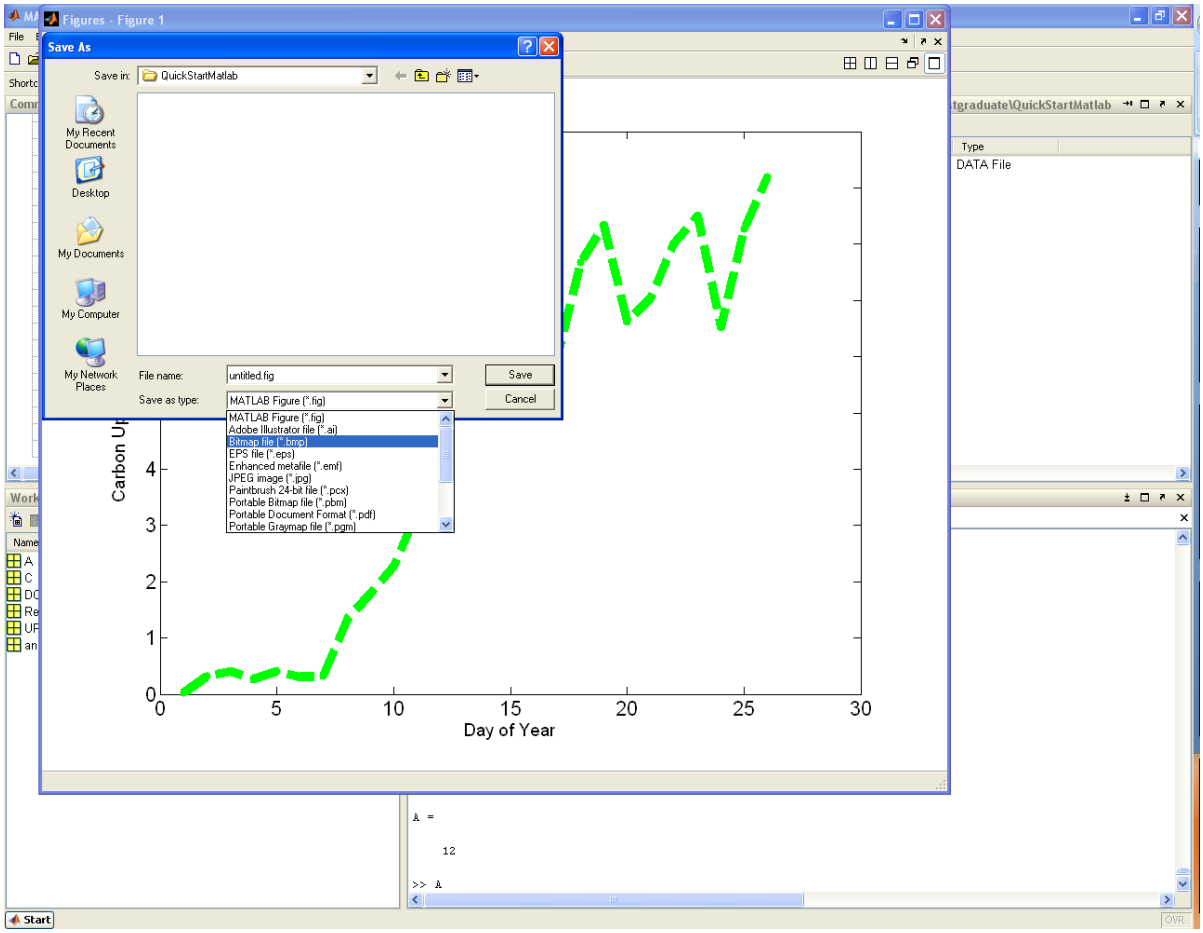

Figure 5: The 'Save as...' dialogue box allows you to save a figure in a variety of different formats. Be sure that the resolution is sufficient for your needs.

Once you are happy with your graph – save the graph as your favourite image file or save it as a ".fig" file so you can open it again in MATLAB.

There are lots of great tutorials about using MATLAB online:

http://www.mathworks.com/videos/matlab/getting-started-with-matlab.html

If your institution doesn't have a license for MATLAB you can still purchase a student license for about $100 from the MATLAB website.

If you would rather go open source you can follow John's advice and use Octave

http://www.gnu.org/software/octave/

It's a free, open source MATLAB clone and the majority of the basic MATLAB commands will function with octave.