Lab 2

Matlab Fundamentals; Part II

Scripting, Importing and Exporting Data

Handout – print version of Lecture on *Marine Modelling* January 15, 2019

Katja Fennel, Oceanography, Dalhousie University

2.1

1 Scripting

So far we mostly considered statements that can be entered on a single line or very few lines. For more complex tasks you will need larger collections of statements. Such a collection of statements is called "program" or "script".

Scripting

By default, all MATLAB script files end in `.m`

You can use your favorite text editor or **File > New > M-file** (you can also type `edit example.m` at the prompt) to create a script.

Type a simple program in the editor, e.g.:

```
x = 0 : pi/20 : 6*pi;
plot(x,exp(-0.2*x) .* sin(x), 'r'), grid
```

Now **copy** and **paste** from the editor window to the command line and hit enter. 2.2

Saving a program

Save contents of the Editor window (**File > Save**). Remember extension `.m` !

E.g. save as `plotexample.m`

Now enter plotexample at the command line.

```
>> plotexample
```

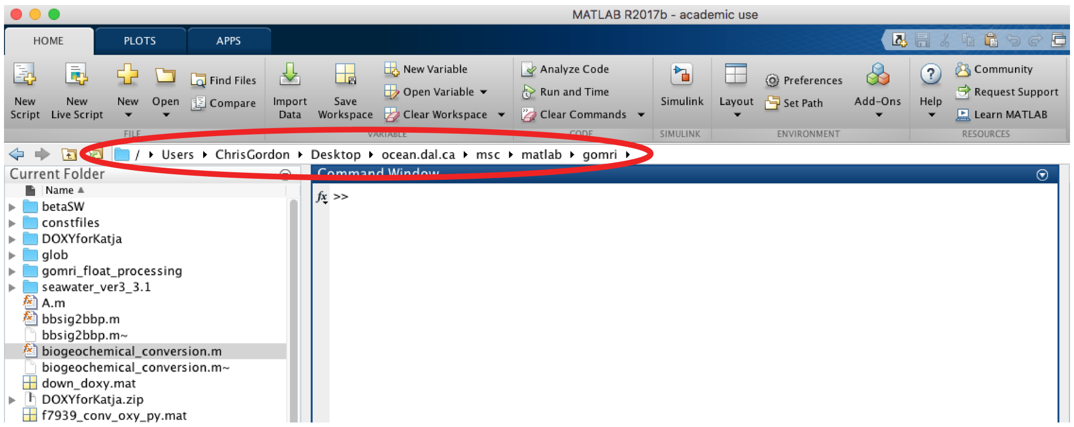
Matlab carries out the commands in your script file.

List the contents of a script file using type.

```
>> type plotexample
```

2.3

Note: Your script file has to reside in your *Current Directory* (look in the toolbar). Or, you can keep scripts in a folder that is part of the *Matlab Path*.



You can check and modify your Matlab Path using the `path` command, but we'll postpone dealing with the path. 2.4

You can also capture commands that you enter at the command line as you go (including command line output) using diary.

```
>> diary my_notes
...
>> diary off
```

2.5

2 Output

Output

Two straightforward ways:

- no semicolon
- display command: `disp(x)`

When you use disp you can also display a message with the answer:

```
>> x=2;
>> disp(['The answer is ', num2str(x)])
```

Square brackets create a vector. Apostrophes (single quotes) create a string. `num2str` converts a number into a string.

You can display more than one number (by collecting them in a vector):

```
>> x = 20; y=8; z = 3;
>> disp( [x y z])
```

2.6

3 Format

Format

Default behavior: Matlab attempts to display integers exactly (for up to 9 digits). If an integers has more than 9 digits it will be displayed in exponential notation with 5 significant digits.

Try:

```
>> 123456789
>> 1234567890
>> 1234567891
```

Numbers with decimal part are displayed with four significant digits.

The default can be changed with the format command.

```
>> format short e % will always use exp format
>> format long    % gives more accurate output
>> format % reverts back to default
```

Note: Text after % is ignored by Matlab. This is useful for commenting in scripts. 2.7

Scale factors

Enter:

```
>> x = [1e3 1 1e-4]
x =
   1.0e+03 *
    1.0000    0.0010    0.0000
```

A common scale factor is applied to the whole vector when elements differ in magnitude.

If you don't want this, try

```
>> format short e
```

or

```
>> format bank
```

2.8

More formatting:

If you like to control the format of your output exactly, `fprintf` is for you! It's more flexible and more complicated then disp.

For example:

```
>> balance = 12345;
>> rate = 0.09;
>> interest = rate * balance;
>> balance = balance + interest;
>> fprintf('Interest: %.3f  New balance: %.2f\n', ...
   rate, balance)
```

Note: The 3 dots allow you to extend your command beyond the line break. As always, for more info use `help fprintf` (not really necessary at this stage though). 2.9

4 Input

Input

The `input` statement provides a flexible way of getting data into a program while it is running (it's of no use from the command line).

General form:

```
>> variable = input('prompt');
```

Look at example script `interestexample.m` (download from:
http://memg.ocean.dal.ca/grosse/...
... Teaching/MM2019/Materials/Lab_Materials.html)

```
balance = input('Enter bank balance:');
rate = input('Enter interest rate:');
balance = balance * (1+rate);
disp('New balance:');
disp(balance)
```

Try it!

```
>> interestexample
```

2.10

5 Importing and Exporting

Importing and Exporting Data

Saving and reloading data between sessions: Commands `save` and `load`

```
>> save workspace_Lab02
>> save balance_and_interest balance interest
>> clear
>> who
>> load workspace
```

By default the saved files will be in binary format (you will not be able to read them with an editor or different program) and have the `.mat` extension. Note that during loading Matlab assumed workspace to be a mat-file by default.

To save data in a readable format use `-ascii` as option.

```
>> A = [5 6 7; 97 6 0];
>> save my_data.txt A -ascii
```

You can view it:

```
>> type my_data.txt
```

2.11

And load it:

```
>> clear
>> load my_data.txt
>> who
```

Or try instead:

```
>> A = load('my_data.txt');
```

Note that we have to include the file extension (`.txt`) in this case.

You can also load data that was not created with Matlab this way, e.g. a table, provided the number of elements per row is equal throughout the file.

```
>> data = load('lobo_data_subset.txt');
```

2.12

You can read and write files that are compatible with Excel. For example, to read an Excel file you can use

```
>> A=xlsread('filename.xls')
```

MATLAB even has a function that will tell you things about what is inside the Excel file, for example **SheetNames**, to learn more type `help xlsfinfo`.

Also useful are the MATLAB functions `csvread` and `csvwrite`. They facilitate transferring data to and from Excel.

For all kinds of different data formats use the `uiimport` or **File > Import Data**. 2.13

6 Lobo Plots

Let's plot some of the Lobo data:

`lobo_data_subset.txt`

contains the following 5 variables:

- oxygen [ml/l]
- fluorescence [micro g/l]
- nitrate [micro M]
- salinity
- days since 2001-1-1 [UTC]

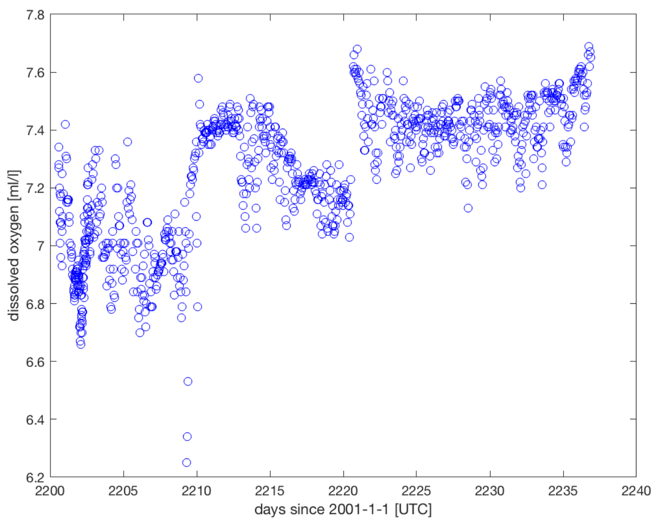
Plot properties over time, e.g.:

```
>> plot(data(:,5),data(:,1),'b*')
>> xlabel('days since 2001-1-1 [UTC]')
>> ylabel('dissolved oxygen [ml/l]')
```

You can copy and paste from:

`load_and_plot_lobosubset.m`

2.14

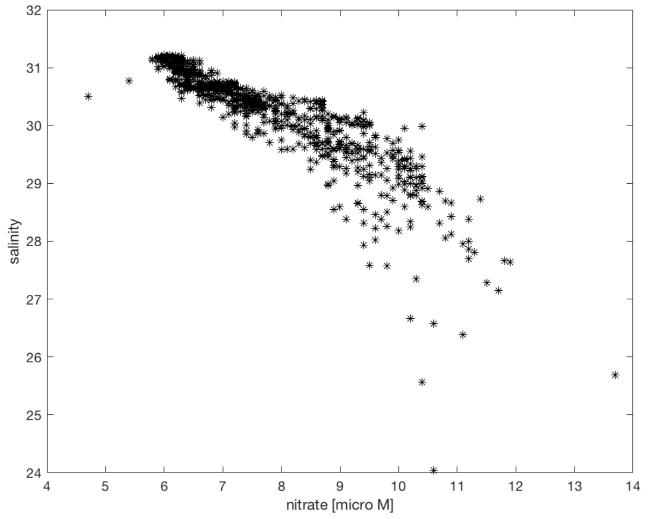


2.15

Or plot property versus property, e.g.:

```
>> plot(data(:,3),data(:,4),'g*')
>> xlabel('nitrate [micro M]')
>> ylabel('salinity')
```

2.16



2.17

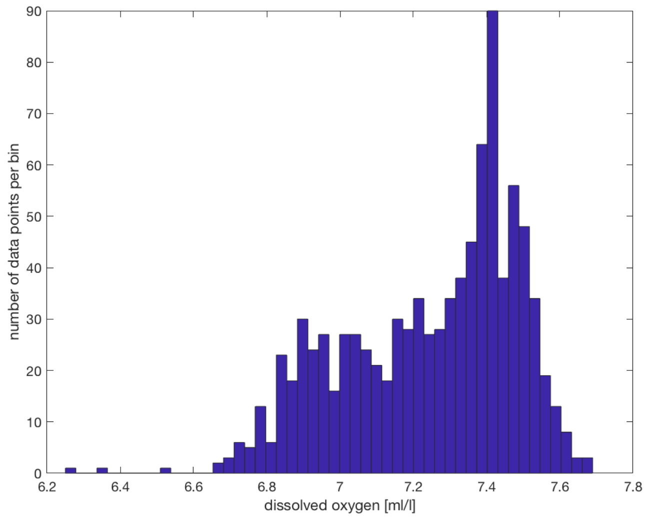
Or plot a histogram of the oxygen data:

```
>> hist(data(:,1),50)
>> xlabel('dissolved oxygen [ml/l]')
>> ylabel('number of data points per bin')
```

If you want to save the figure as png or jpg image use print, e.g.

```
>> print(gcf,'-dpng','oxygen_histogram')
```

2.18



2.19