

Introduction to Computational Neuroscience

Practice I: Introduction to Matlab/Octave

Ilya Kuzovkin, Raul Vicente

February 15, 2015

A request: Please track how long it will take to complete this set of exercises. Add this time to your final report.

In this course we will use Matlab programming language. The choice is due to the popularity of this language (and the environment) in the neuroscience community. You can use any other programming language if you want to, but some practice session will have basecode in Matlab and data files we will use often come in Matlab format. There are plenty of tutorials online, this practice covers some basics and the most common tasks we will encounter.

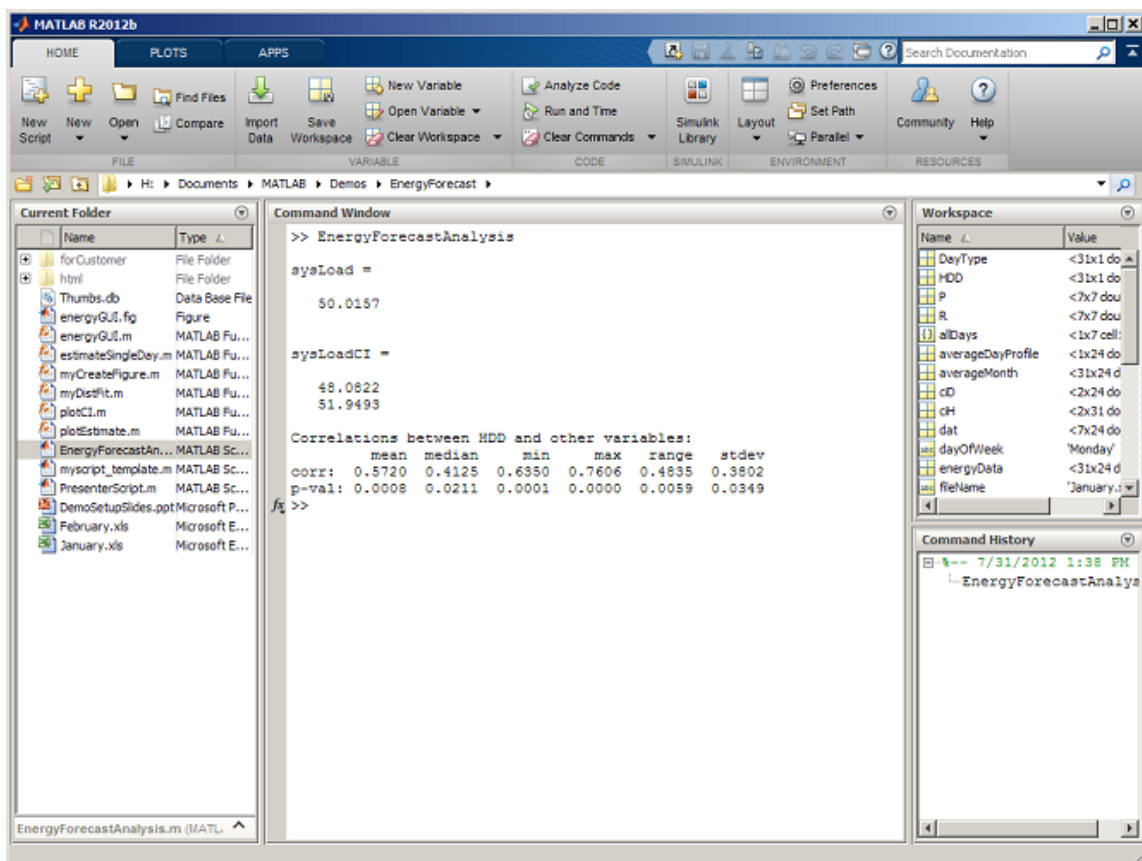


Figure 1: Matlab main window

On Figure 1 you can see Matlab main window. For Matlab it is very important to know which directory is the current *working directory*. So the first order of business for you is to pick a directory (in the upper part of the screen) where you will complete this practice exercises. Once you unpack the 01 - Matlab.zip archive you will have 01 - Matlab/code folder in it. I suggest

you point Matlab to that folder.

The main part of the screen occupies the command line, where you can run your commands. The second essential part of the Matlab is a text editor (which is not shown in default layout). There are also such panels as command log, list of variables in your current workspace and file browser. You can realign the panels to your own liking. The most common way to work in Matlab is to write your code in editor and sometimes run and test it in the command line.

The free alternative to Matlab is called Octave¹. You can see it on the Figure 2 and there are similar panels available: file browser, list of variables, command history and the command line.

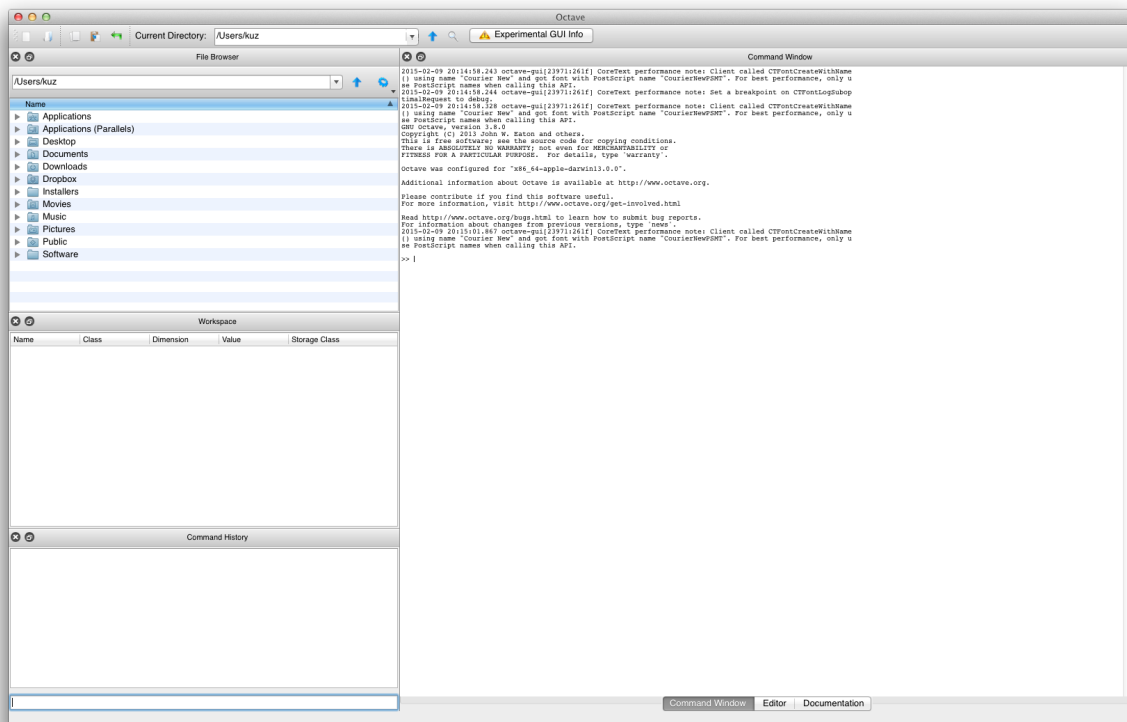


Figure 2: Octave main window

Some of the functions in Octave require installing additional packages to use them (somewhat like Matlab Toolboxes). Here is the list of available packages: <http://octave.sourceforge.net/packages.php> and here are the instructions how to install: <https://www.gnu.org/software/octave/doc/interpreter/Installing-and-Removing-Packages.html>. We will need `io` and `statistics` packages for this tutorial. Once installed a package has to be loaded by calling `pkg load package_name`.

Now you are all set and we briefly go through basic things you do in any programming language. Read the code and run the pieces which are not immediately clear.

Listing 1 Comments

```
1 % This is a comment, write comments to explain tricky parts of code and outline
2 % what is happening
```

¹Windows: http://wiki.octave.org/Octave_for_Windows (take the first option: MXE Builds), Mac: http://wiki.octave.org/Octave_for_MacOS_X (take the binary installer), Linux: distribution packages.

Listing 2 Variables

```
1 % variable can hold any kind of data, you do not need to specify the type
2 x = 5;
3
4 % semicolon at the end of a line suppresses the output, run and compare
5 >> x = 5
6 >> x = 5;
7
8 % an array is a bunch of numbers in one variable
9 v = [1, 2, 3, 4, 5];
10
11 % a matrix can be defined as
12 >> B = [[1, 2, 3]; [4, 5, 6]; [7, 8, 9]; [10, 11, 12]; [13, 14, 15]];
13 % you can take one element from the matrix by specifying row and column
14 >> B(1, 1)
15 % or you can take the whole second row
16 >> B(2, :)
17 % or the second column
18 >> B(:, 2)
19 % or rows 2, 3, 4 and 5 from columns 2 and 3
20 >> B(2:5, 2:3)
21 % to transpose a matrix do
22 >> B'
23
24 % You can also have matrixes with more dimensions than two
25 >> M = repmat(magic(3), [2, 3, 4, 5]);
26 % will produce a 4D matrix of size
27 >> size(M)
28 ans =      6      9      4      5
29 % and you can navigate along each dimension, for example
30 >> M(2, 1, 4, 3)
31 % which is powerful, but it is easy to loose track of those dimensions, so
32 % people often use cells
33
34 % Cells are most generic stuctures to store anything
35 >> C = {'here', 'we', 'add', [1, 2, 3, 4, 5], 'whatever we like', 5};
36 % and then you can access a certain element as
37 >> C{4}
38 ans =      1      2      3      4      5
39 % indexing starts form 1
```

Listing 3 Arithmetics

```
1 % You can add, subtract, multiply, divide numbers
2 >> 2 + 3 - 4 * 5 / 6
3 % raise them to power
4 >> 2^8
5 % add and subtract vectors
6 >> [1, 2, 3] + [4, 5, 6] - [7, 8, 9]
7 % multiply and divide vectors element-wise
8 >> [1, 2, 3] .* [4, 5, 6] ./ [7, 8, 9]
9 % multiply matrices
```

```
10 >> [[1, 2]; [3, 4]] * [[5, 6]; [7, 8]]
```

Listing 4 IF - THEN - ELSE

```
1 number = 1;
2 if number == 1           % as you can see comparison is done using ==
3     disp('The number is one') % this way you can produce output messages
4 elseif number == 2
5     disp('The number is two')
6 else
7     disp('The number is something else')
8 end
```

Listing 5 Loops

```
1 % FOR loop is the main guy
2 for i = 1:100
3     disp(i)
4 end
5
6 % it can also go over elements in a variable (the 'foreach' behaviour)
7 C = {'here', 'we', 'add', [1, 2, 3, 4, 5], 'whatever we like', 5};
8 for c = C
9     disp(c{1})
10 end
11
12 % the WHILE loop is also available
13 i = 0;
14 while i < 5
15     disp(i)
16     i = i + 1;
17 end
18
19 % it is often much more efficient to use vectorised code instead of loops
20 % for example to raise numbers 1 to 100 to power 2 and sum them we can do
21 total = 0;
22 for i = 1:100
23     total = total + i^2;
24 end
25 disp(total)
26
27 % but the Matlab way to do it is
28 total = sum([1:100] .^ 2)
```

You will see weird functions now and then, like this `sum` in the last example. And you may wonder *what does it do exactly?*. To answer that there is a Matlab documentation <http://se.mathworks.com/help/> where you can enter `sum` and read all about it. Or just Google `matlab sum` or `octave sum`

Listing 6 Creating a function, file dothings.m

```
1 % Each function in Matlab lives in a separate file (you can have several
2 % functions in one file, but only the first one will be callable from the outside
   )
```

```

3 % So this is an example of a function, which should be saved as
4 % a separate file 'dothings.m'
5
6 % Matlab function can return several values
7 function [s, d, p, q] = dothings(a, b)
8     s = a + b;
9     d = a - b;
10    p = a * b;
11    q = a / b;
12 end
13
14 % this function will compute sum, difference, product and quotient of two
15 % numbers a and b

```

Listing 7 Calling a function

```

1 % Now once we have our function defined and saved we can call it
2 >> [s, d, p, q] = dothings(10, 20)

```



Exercise 1: Language basics

Write a function which multiplies two matrices A and B and returns the resulting matrix as an output. If matrix dimensions do not match your program should output appropriate message. In your report please give the full code and the example of how to call the function.

(1PT)

Listing 8 Working with data

```

1 % There are two great functions in Matlab: 'save' and 'load'.
2 % Suppose you have very important piece of information
3 >> important = 'I had a cookie today';
4 % and you want to save it
5 >> save('mydata.mat', 'important')
6 % it will save the variable 'important' into file named 'mydata.mat'
7 >> clear important % deletes our variable
8 >> load('mydata.mat') % loads it back
9 >> important % will print out the content of the variable
10
11 % load() can also deal with formats other than .mat, such as .csv or .dat
12 >> load ../data/erp_trials.csv % assuming 01 - Matlab/code is working directory
13 % will load a matrix in your workspace
14 >> size(erp_trials)
15 ans = 79 2000
16 % where we have 79 recordings of neural data 2000 data points per each

```



Exercise 2: Manipulating data

Load the `erp_trials.csv` dataset as described in Listing 8. Create a subsample of observations by taking first 30 rows (and all of the columns) of the `erp_trials` matrix. Save the resulting matrix of size 30×2000 as `one.mat` file. The remaining 49 samples save as `two.mat`. Use `clear` command to remove both subsets from your workspace. Now load them both back using `load`. Glue two subsets together to be one whole set as it was before. Assuming that the original

dataset is called `erp trials` and the resulting one is called `reconstructed` confirm that they are identical by running `all(erp trials(:)==reconstructed(:))`. Include the code in your PDF file.

(1PT)

Listing 9 Plotting

```
1 % One of the most important abilities in any data analysis task is the ability to
2 % visualise the results.
3
4 % generate some a random vector
5 >> data = rand(1, 100) * 10;
6
7 % the simplest way to plot is
8 >> plot(data)
9
10 % there are various ways to plot your data depending on what you have there
11 >> bar(data)
12 >> hist(data)
13
14 % for two dimensional data
15 >> data = rand(100, 100) * 10;
16 % you might find useful
17 >> area(data)
18 >> imagesc(data)
19 >> boxplot(data(:, 1:20))
20 >> scatter(data(1, :), data(2, :))
21 % etc...
22
23 % by the way if some operation is taking too long you can interrupt
24 % it by pressing Ctrl + C
25
26 % By default Matlab will create one canvas (called figure) and plot everything
27 % on it, if you want to produce several plots and open them each separately you
28 % should call 'figure(N)' before each one
29 >> figure(1)
30 >> plot(data(1, :))
31 >> figure(2)
32 >> plot(data(2, :))
33
34 % You can also put several plots onto one canvas
35 >> subplot(3, 2, 1)
36 % will create a grid with 3 rows and 2 columns and will plot the next picture
37 % into cell 1 on this grid
38 >> plot(data(1, :))
39 % next we want to put into the second cell
40 >> subplot(3, 2, 2)
41 >> plot(data(2, :))
42 % and so on (better done with a FOR cycle)
43 >> subplot(3, 2, 3)
44 >> plot(data(3, :))
45 >> subplot(3, 2, 4)
46 >> plot(data(4, :))
```

```
47 >> subplot(3, 2, 5)
48 >> plot(data(5, :))
49 >> subplot(3, 2, 6)
50 >> plot(data(6, :))
```



Exercise 3: Plotting

Create a plot of 17th row (all columns) of our `erpdata` dataset. Whenever you produce a plot make sure that it is explained what your plot demonstrates. Go through `plot` function documentation ²³ and make the following changes to your plot:

- Add caption, so that everybody would know what this plot is about
- Set X-axis label (try to guess what it is or come up with something)
- Set Y-axis label (try to guess what it is or come up with something)
- Change the color of the line
- Change the type of the line

(1PT)

Please submit a pdf report with answers to the questions and comments about your solutions. Include figures, explanations and essential pieces of code. Do not include the code itself as a separate file, your report should give good understanding of what you have done. Please mark how long it took to complete this set of exercises. Upload the pdf to the practice session page on the course website.

²<http://se.mathworks.com/help/matlab/ref/plot.html>

³https://www.gnu.org/software/octave/doc/interpreter/Two_002dDimensional-Plots.html