Getting Started with MatLab

F. E. Wietfeldt
revised Jan. 2017 RTG

This document contains basic instructions for running MatLab on the Advanced Lab computers in Stern 2023, with a simple example of data plotting and linear regression. Keep in mind that MatLab is very powerful and can do much more than this. Many more details with examples can be found in the MatLab Help system and demos. I encourage you to explore those.

Print out your scripts and plots from this assignment, and turn it in with the Data Analysis Lab on **FRIDAY, February 01, 2019 by NOON**. If you have Matlab on your personal computer, you are free to use that (as opposed to the lab computers).

1. Log in as "Student" on any of the three Advanced Lab computers.
2. Create a folder on the desktop with your name.
3. Open Matlab R2013a from the Start Menu. You will see the MatLab default workspace.
4. Let's say you have some data that you want to fit to a straight line using regression analysis and determine the slope m and y-intercept b, and their estimated errors. Your data consists of measurements of temperature vs. time, and you have estimated uncommon errors for the temperature measurements:

time (s)	temperature (°C)	σ_i(°C)
1.0	11.3	1.0
2.0	12.9	1.0
4.0	21.5	2.0
5.0	21.5	2.0
7.0	25.3	3.0
8.0	27.0	3.0
9.0	39.2	3.0

Type the following into the MatLab command window to enter your data, hitting the Enter key after each line:

```
>> time = [1 2 4 5 7 8 9]
>> temp = [11.3 12.9 21.5 21.5 25.3 27.0 39.2]
```

```
>> ee = [1 1 2 2 3 3 3]
```

In Matlab all quantities are defined as *numbers*, *arrays*, or *vectors* (one-dimensional arrays). You have just entered your data as three separate vectors. Note that they have the same number of elements (important!). You will see that this method makes it easy to manipulate and analyze your data in MatLab. For example, type:

```
>> foo = time + 2
```

The vector `foo` contains the elements of time, plus 2. Try this:

```
>> foo = time./3
```

The vector `foo` contains the elements of time, divided by 3. Try this:

```
>> foo = time.^2
```

The vector `foo` contains the elements of time, squared. Try this:

```
>> foo = cumsum(time)
```

Each element in `foo` is the sum of all previous elements in `time`. Try this:

```
>> sum(time)
```

You obtain the sum of all elements in `time`. Try this:

```
>> sum(time.^2 + ee.^2)
```

Each element in `time` is squared, and then added to the square of the associated element in `ee`, and these are summed to make your result. Make sure that you understand all of these operations before you proceed.

5. Now make a plot of your data, temperature vs. time, with error bars. Enter the following:

```
>> errorbar(time,temp,ee)
```

A plot of your data will appear in a separate window. Float the mouse over the rightmost icon in the tool bar and it should say “Show Plot Tools”. Click on that icon. You will now see your plot in the Plotting Tools environment.

6. In the Plot Browser (right side), click on Axes. The bottom window will show “Property Editor - Axes”. Label your x axis as “time (s)” and your y axis as “temperature (C)”.

7. Click on the blue line in the Plot Browser. The bottom window will show “Property Editor - Errorbarseries”. Enter “My data” in the Display Name. Under “Line:” choose “no line”. Under “Marker:” choose the small dot. You now have a nice, properly labeled, plot of your data.

8. Now let’s do the regression analysis. Return to the MatLab Command Window and enter the following formulas (found in the Data Analysis Primer, pp. 11–12):

```
>> delta = sum(1./ee.^2) * sum(time.^2./ee.^2)
     - (sum(time./ee.^2))^2


>> m = (sum(1./ee.^2) * sum(time.*temp./ee.^2)
      - sum(time./ee.^2) * sum(temp./ee.^2)) / delta


>> b = (sum(time.^2./ee.^2) * sum(temp./ee.^2)
      - sum(time./ee.^2) * sum(time.*temp./ee.^2)) / delta


>> errm = sqrt(sum(1./ee.^2) / delta)


>> errb = sqrt(sum(time.^2./ee.^2) / delta)
```

You now have the slope (`m`) and y-intercept (`b`) of the best-fit line for your data, and their estimated errors (`errm` and `errb`).

9. Let’s add the fit line to your plot. First, generate a MatLab vector with the fit values:

```
>> fittemp = m*time + b
```

Return to the Plot Tools window. Click on your plot, and then click on the button “Add Data...” on the right. Select “time” as the X Data Source and “fittemp” as the Y Data Source. Click OK. Your best-fit line is now added to your plot. If it doesn’t look like a good fit, you probably made some mistake earlier.

10. In the Annotations window (left side) click on Text Box, then mark a text box on your plot. Type the results of your fit (using the numbers you found earlier in place of the `X`’s) into the text box:

```
fit slope = X.XX ± X.XX
fit intercept = X.XX ± X.XX
```

If you click on the text box, the Property Editor will show various tools to change the fonts, colors, etc. of your text. Play around with that a bit.

11. When you are happy with your plot, save it. Click File → Save As..., then give your plot a name and save it to your folder on the desktop (not the generic "work" folder!). **Print this plot and turn it in with the rest of the assignment.** You do not need to turn in the script or text up to this point (just the plot). Close the Plot Tools window.

12. Calculate the chi-squared for your fit. In the Command Window enter:

    ```
    >> chisq = sum((temp - fittemp).^2./ee.^2)
    ```

    What is the reduced chi-squared? Was this a good fit?

13. Finally, save your data. Click File → Save Workspace As..., then give it a name and save it to your folder on the desktop (not the generic "work" folder!).

14. Now we are going to import a data file that contains the time, temperature and error values that you previously input manually. To do this, create a new script by clicking on "New" then "Script" in the top-left of the Matlab window. Save this script as "mynewscript". Save this file in the "Matlabexample" folder on the desktop of the computer (if using the lab computers). Note that the file is saved as "mynewscript.m" as ".m" is the suffix for Matlab files. **You will print this script at the end and turn it in with the rest of the assignment.**

15. The folder "Matlabexample" contains a data file entitled "dataexample.csv". A .csv file is a comma-separated value file, in which values are separated by commas (appropriately named, right?). You can save Excel (for example) spreadsheets as csv files, which is useful for analyzing the data in more complex math software such as Matlab. **Note that if you are using any computer aside from the lab computers, you will need to email yourself the data file from any one of the three lab computers.**

16. Type "close all" in the first line of your script, and "clear all" in the second line. This will close all other Matlab windows (such as plots), and clear any previously saved variables.

17. Now import the csv file on the third line of your script by typing "data = csvread('dataexample.csv',0,0)". Note that the ' and ' here, and in the following, are just apostrophes. This will import and save the data as a matrix with the variable name "data".

18. Run the Matlab script by either hitting the “Run” button up top (make sure you are in the “Editor” tab, otherwise you won’t see this option), or hit f5 when the script window is selected. If a change folder dialog box pops up, agree to change to the appropriate folder (you need to be working in the folder that has your script, as well as the data file saved). Now, you should receive an error. Open the data file with Excel to take a look at it. Think about why you’re receiving the error.

19. Edit the import line of your script to start at row 1, and column 0. You may type “help csvread” in the Command Window and hit Enter to see a brief manual for csvread, which should tell you how to start at a certain row and column (note that when importing a file like this, Matlab calls the first row and column 0, NOT 1). This should rid you of the error, and successfully import the data.

20. Now type “data” in the Command Window and hit Enter. You should see the matrix with the data show up.

21. Now you’ll make two new vectors, in order to make plotting the temperature as a function of time data easier. Type “x = data(1:7,1:1);” in the script, somewhere after your import data line. Note that the semicolon at the end of the line suppresses the output. This saves the values of the “data” matrix from the first row to the seventh row, for the first column only (first column to first column) as the variable “x”. On the next line import the data from the second column, and save it as the variable “y”.

22. Now plot the data. To do this, type on a new line “plot(x,y)”. Run the Matlab script. A new window with the plot should appear. Note that the data points, which are discrete, are connected via lines. This is often undesirable. Edit the plot line to read “plot(x,y,‘rx‘)”. Now run the file. The old window closes and a new one pops up, this time with red x marks at the data points (this is where the rx comes in).

23. One last thing- let’s add some axes labels to the graph. Below the plot line, type “xlabel(’time’)”. Similarly, add a new line to label the y-axis as temperature. Now rerun your script. Save the resulting figure as whatever you would like to call it. **Print this figure and print the script file, and turn them in with the rest of this, and the Data Analysis Lab assignment.**

24. Exit MatLab. Feel free to email yourself your script, as you should now delete it from the lab computer (if using a lab computer), so that others may use the same file naming convention. Don’t delete the data file please!

25. Congratulations! If you fully understood this example, you should have no trouble using MatLab for the Data Analysis Lab and your Advanced Lab experiments. If you have any questions be sure to see your professor or TA.