AAE 637 Lab 2: Debugging

TA: Charng-Jiun Yu

January 31, 2018

1 Assignment

1. For each assignment, make sure that your submission contains:

   1) MATLAB command files and outputs. I recommend using the “publish” button in the MATLAB editor to generate outputs. This creates a document containing both your outputs and codes. Note that you should also submit the codes for your function.
   2) A writeup on your process and answers. This includes words and equations necessary to get your results. A clear explanation on your work shows that you understand the problems well .

2. Please follow the naming convention: “FI_LN_Assign#X(q=Y)”, where FI is your first initial, LN is your last name, X is the assignment number, and Y is the question number if applicable. Example: C_Yu_Assign#1(q=1).pdf.

3. Submit your assignment to the assignment dropbox on Learn@UW.

2 Debugging Strategies

No matter how experienced you are in programming, you will always find yourself debugging. Fortunately, MATLAB has some good features that make debugging easier:

1. **Breakpoints**: You can set a breakpoint at any executable line so that when you run the program, it will pause at the location you specify. With breakpoints, you can run your codes step by step and check your results. Note that the program will not pause if you are running by section rather than the whole script.

   When the program is paused by a breakpoint, it enters the debugging mode. To run the program line by line, press the “step” bottom. You can also set multiple breakpoints. If you want your program to pause at the next breakpoint, press the “continue” bottom. In the debugging mode, you are still able to enter commands.

   Breakpoints can be used inside a function. In this case, you can check the intermediate results in the function and see where errors occur. You can also put a breakpoint inside a loop to check the results in each iteration. If you want the program to pause at a specific iteration, right click the breakpoint and set the stopping condition.

2. **Run By Section**: By putting “%%”, you are able to run the specific section using the “Run Section” button. This is useful for debugging as you do not have to run the whole program every time after you make some changes.

Here are some useful tips for debugging in general:

1. Don’t change your codes drastically. You might find other errors that make things more complicated. Try to change only one thing, run the program, and see if it works.
2. Take special attention to the “wrong input” type of mistakes. Did you use a wrong variable in the equation? Did you do “find & replace” and accidentally replace something you are not intent to? Did you put a hyphen in your variable name? This kind of mistakes is common but often ignored because we don’t usually pay too much attention on them while coding.
3. When you are really having trouble, try to sit back and think about what your program is doing instead of simply tracking down the errors. This will give you some new perspectives.

3 GOOD PROGRAMMING HABITS

Good programming habits will not only make debugging easier but also reduce the chance that errors occur:

1. Run the codes often. If an error occurs, you are more likely to track it down as you just successfully run the program a minute ago.
2. Break a long equation into several smaller parts. It is harder to detect an error if the equations are long and messy.
3. Comment your codes, so that you know what you are doing when you get back to it. This is especially important when you are doing a long project.

4 USEFUL MATLAB COMMANDS

- clear: Remove all variables in the workspace. You can do “clear X Y” if you just want to remove X and Y.
- clc: Clear the command window.
- %: Write comments. Things followed by “%” will not be run.
- %%: Create a section that can be run separately.
- ... (three dots): Continue your codes to the next line.
- size(X): Return the dimension of X. You can do “[nrow,ncol]=size(X)” to store the number of rows and columns as variables. You can also do “[nrow,~]=size(X)” if you only want to store the number of rows.
- disp(X): Display variable X. You can also use this to display words by adding single quotation marks. ex: dsip(’AAE637’) displays *AAE637*. disp(’ ’) with blank inside creates a blank line.
- ones(x,y): Create a matrix of 1 with x rows and y column.

- horzcat(X,Y): Horizontally concatenate X and Y. Use vertcat(X,Y) for vertical concatenation.
- fprintf(X,Y): Display words and variables with a specific format. ex: Suppose that Y=2018.637. fprintf('The mean income is %3.2f', Y) will display *The mean income is 2018.64.* The number 3.2 tells MATLAB to display at least 3 digits with 2 of them to the right of the decimal sign.
- \n (within single quotation marks): Create blank line. You can add "\n" in the formal example as "fprintf('\nThe mean income is %3.2f', Y)" to make sure that this output shows up in a new line.
- X': Create the transpose of X.
- inv(X): Create the inverse of X.
- X = (Y>1): Create X where X=1 if Y is greater than 1, otherwise X=0. This is an efficient way of creating dummy variables.
- find(X): Return a vector containing indices of non-zero element in X. You can also do "find(X>5)" to get the indices of elements in X that are higher than 5.
- global: Declare variables in a global scope. You can use this to show the variables in a function without setting them as outputs.

5 Other Tips for Surviving in the Class

1. This class uses matrix notations heavily. Most of the formulas you use in the problem sets will be in matrix forms as well. Make sure that you are comfortable with matrix algebra. Can you write the formula for linear OLS estimator in matrix form? What about the variance-covariance matrix?

2. I highly recommend the econometric textbook by Prof. Bruce Hansen. It presents the fundamentals very well in a conversative tone. Check out Chapter 3 and 4 if you are not familiar with the linear OLS, including matrix notation. For hypothesis testing, take a look at Chapter 9. The text can be downloaded at the following link:
https://www.ssc.wisc.edu/~bhansen/econometrics/Econometrics.pdf