Exercise: Debugging in Matlab

Stockholm Doctoral Program in Economics
Computational Bootcamp

Kathrin Schlafmann

Questions

1. Debugging in a script:
   Orthogonal polynomials are a useful tool when you want to approximate a function numerically. One such set of polynomials are Chebychev polynomials.[1] They are defined recursively according to the following equations:

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_{n+1}(x) &= 2x\ T_n(x) - T_{n-1}(x) \quad \forall\ n \geq 1 \end{aligned}$$

   The code for this question computes the Chebychev polynomials recursively on a grid from -1 to 1. However, there is a bug in this code - can you find it?

   (a) Execute this cell: What does the error say?

   (b) In which line does the error occur?

   (c) What is the problem and how can you fix the it? (Hint: use the tools you have already used to learn about the properties of all the terms in the problematic line!)

2. Debugging in a script, cont'd:
   You want to look at the utility functions for different values of constant relative risk aversion (CRRA):

$$u(c) = \begin{cases} \log(c) & \text{if } \sigma = 1 \\ \frac{c^{1-\sigma}}{1-\sigma} & \text{otherwise} \end{cases}$$

   The code aims to compute this utility for various values of risk aversion σ and plot the utility functions. But there is a problem - what is it?

3. Debugging in a script, cont'd:
   The code for this question runs a Monte Carlo Simulation to find out how precise the estimate of the mean of a random variable is if we simply draw repeatedly from its distribution. Run the code - is the outcome unexpected? Is there a problem in the code?

4. Debugging when using functions:

[1] If you are not familiar with what these polynomials do or why they are useful don't worry about it for now, all you need to know is how they are constructed (the 3 equations above).

(a) Copy the contents of `firstFunction.m` from the last exercise sheet into a new function and call it `secondFunction.m`. Now run the code corresponding to question 4(a). Make sure you can execute without error.

(b) Now you realize that you will need to execute this function for several values of the variables `x1`, `x2` and `x3`. To avoid inefficient loops you want to use vectorization. You replace your input variables with vectors. Execute this code. What happens?

(c) Click on the line reference in the error message. This brings you right to the point within the function file where the error occurred. What is the problem here?

(d) Fix the function declaration to avoid this error. Re-execute this part of the code. Does it work now?

5. Debugging when using functions, cont'd:

(a) Look at the code in the cell for question 5 and the function that is called from within that code. What do you expect it to do?

(b) Execute the code in this cell. What happens?

(c) Go to the exact line where the error occurs. Try to use the `size()`-function to detect the bug. Why doesn't this work here?

(d) Set a breakpoint in the line where the error occurs and and re-execute the cell. Can you now use the `size()`-function to detect the bug? Note: You can set breakpoints in the editor window by clicking to the left of a line. This will show a red dot next to the line. You can remove a single breakpoint by clicking on it again. All breakpoints can be removed at once by selecting "clear all" in "Breakpoints"-menu at the top of the editor window.

(e) Once you have fixed the bug, re-execute the code in the cell to ensure that the code runs correctly now. Before you do that you will have to exit the "Debug"-mode - you can do this by clicking on the "Quit Debugging" button on top of the editor window. Note: You can tell whether you are in "Debug"-mode or not by looking at the command window: the command line will start with `K>>` in "Debug"-mode, but only with `>>` in normal mode.

6. Debugging when using functions, cont'd: This exercise looks at the development of two species:

- Species 1 are predators: they eat parts of the population of species 2 as long as there is enough of species 2 and grow in numbers. However, once they outnumber species 2 they start fighting each other and half of their population is killed.
- Species 2 are very peaceful and all members who are not eaten multiply over time.

The function simulates the population growth of both species.

Execute the cell for this question. Find the bug.