Problem Set 1 -- Introduction to programming in Matlab

Computational Neuroscience Summer Program

May, 2011

Log into the computer with your PennKey and start Matlab. Please put your answers to this assignment in a single Microsoft Word document. This document should include your raw Matlab code, the commands you type to run the code, and its output text and plots.

Some of these questions require functions that you may not have seen before. If you have questions, please ask the instructors and feel free to consult Matlab's great built-in help functions and online tutorials at mathworks.com.

1. Concatenation

Write a script that creates the variables a = [1 2 3], b = 4, c = 5, d = [6; 9], e = [7 8]. Use only these variables to create the matrix m = [1 2 3; 4 5 6; 7 8 9] in a single line of code through horizontal & vertical concatenation. (Hint: use nested [ ] operations.) Now change the script into a function. What is the effective difference between calling the script and the function?

2. Number Classification

Write a function that will indicate whether an input number is negative, positive, or zero, as well as even or odd. The function should print these results to the command window and not return anything.

3. For loop

Write a function that creates a Fibonacci sequence (a series of numbers where each element is the sum of the previous two elements; you'll have to start it with [1 1]). It should have an input parameter that indicates the desired length of the output sequence.

4. Plotting

Download this file and load it into matlab:

http://memory.psych.upenn.edu/~josh/Q4.mat

Plot the variable 'Z' from this file using the standard plot command.

Next, create a function *plot_labeled_peak* that accepts one input vector. It should plot the vector in black. In addition, the largest data point in this vector should be labeled with a red circle, and the smallest data point should be labeled with a blue 'x'. Show the result of this function when plotting 'Z'.

5. Challenge questions: Performance analysis

A.

Here we are interested in calculating the lengths of many three-element vectors according to the pythagorean theorem (i.e., sqrt($a^2 + b^2 + c^2$)). Write a function that takes a three-column matrix, and returns the length of the vector in each row using a loop. Use *randn* to create random three-column matrices with 10, 100, 1000, & 10000 rows, and measure how long it takes your function to compute each one. Next, create a similar function that does the same calculations in a vectorized fashion. Compare the performance of the two by plotting the execution times of each of the two functions versus the number of rows in the matrix.

B.

In Matlab, preallocating a vector in a single command (by using the 'zeros' or 'ones' functions, for example) can make things operate much more quickly than repeatedly appending individual elements to the end of a matrix. Write a version of the Fibonacci function from Question 3 that preallocates the output. How fast does it calculate a 20,000 element Fibonacci sequence? How does this speed compare to the previous version of this function? Plot the Fibonacci execution times of various sequence lengths, for both versions of this procedure.

C.

A common data analysis tool is to *normalize* a dataset so that it has a mean of zero and a standard deviation of 1. Here is a function that takes in a large matrix and returns a normalized dataset where each row has a mean of zero and a standard deviation of 1:

```
function x=normData_loop(x)
for row=1:size(x,1)
    m=mean(x(row,:));
    s=std(x(row,:));
    x(row,:)=(x(row,:)-m)/s;
end
```

Can you write a vectorized version of this function? (Hint: use `repmat.`) How does the execution time of your function compare with `normData_loop` on a 1000 by 1000 matrix (e.g., `rand(1000,1000)`)?