Matlab Exercises_Recitation 2[†] 2.086 Spring 2012

Recitation 2: Wednesday, 15 February / Friday, 17 February
MATLAB Exercises_Recitation 2 due: *Tuesday, 21 February 2012 at 5 PM by upload to Stellar*

Format for upload: Students should upload to the course Stellar website a folder

`YOURNAME_MatlabExercises_Rec2`

which contains the completed scripts and functions for the assigned MATLAB Exercises_Recitation 2: all the scripts should be in a single file, with each script preceded by a comment line which indicates the exercise number; each function `.m` file should contain a comment line which indicates the exercise number.

---

1. (6.094 1.2) Write a script to make the following single-index arrays. Leave off the semicolons so that your results are displayed.

   (*a*) $p = [3.14, 15, 9, 26]$
   (*b*) $r = [100, 99, 98, \ldots, 2, 1]$
   (*c*) $s = [0,\ \frac{1}{99},\ \frac{2}{99},\ \ldots,\ \frac{98}{99},\ 1]$

   Note you should find the `colon` operator helpful in constructing these single-index arrays.

2. (This is an extension to Exercise 4. of Recitation 1.) Write a script which finds N^* such that $F_{N^*} < 1000$ and $F_{N^*+1} \geq 1000$ and *also* constructs the single-index array `Ffib` of the associated Fibonacci numbers $[F_1, F_2, \ldots, F_{N^*}]$. This should be a relatively simple modification to your `while` loop of Exercise 4. of Recitation 1: initialize (say) `Ffib` outside the loop, and then use horizontal concatenation to "grow" the `Ffib` array each time through the loop; note you can eliminate your `Nstar_tmp` variable from earlier and instead evaluate the `length` of the final `Ffib` array. Note a "quick and dirty" debug plot can be performed after (and outside) the loop with `plot(Ffib,'o')`.

   Comment: When we get to double-index arrays we will emphasize the importance of initializing arrays (with `zeros` and later `spalloc`). Initialization ensures that you control the size/shape of the array and also is the most efficient way to allocate memory. On the other hand, concatenation can be very useful in dynamic contexts (in which array size may not be known *a priori*) or in situations in which a large array is most easily expressed in terms of several smaller arrays. But use concatenation sparingly in particular in computationally intensive codes.

3. Write a script to calculate the sum of a geometric series with $N + 1$ terms,

$$S = \sum_{i=0}^{N} r^i = 1 + r + r^2 + r^3 + \ldots + r^N \ ,$$

[†]Some of the questions were derived from *Learning* MATLAB by Tobin Driscoll, *Numerical Computing With* MATLAB by Cleve Moler, *Getting Started With* MATLAB by Rudra Pratap, *The Art of* MATLAB by Loren Shure, and the MIT 2010 IAP course 6.094; these are attributed where applicable. These exercises were initially assembled by Dr. Justin Kao.

for the particular case of $N = 10$ and $r = 1/2$. Use `ones(1,N+1)` to set up a single-index (row) array of all r's, the `colon` operator to set up a single-index array of exponents, array "dotted" operators to perform the exponentiation, and the MATLAB built-in function `sum` to perform the summation — no `for` or `while` loops allowed! (In fact, a single line of MATLAB code should suffice, though you may wish to break this into a few lines for improved readability.)

4. Write a script which given a vector of distinct points `xvec` $= [x_1, x_2, \ldots, x_N]$ and a point x finds the index i^* such that $x_{i^*} \leq x$ and $x_{i^*+1} > x$. You may assume that the points are ordered and distinct, $x_i < x_{i+1}$, $1 \leq i \leq N-1$, but you should not assume that the points are equidistantly spaced. You may also assume that $x_1 \leq x \leq x_N$.

   To write your code, you should use array relational operators, the MATLAB built-in `find`, and then (say) the MATLAB built-in `max` (or `length`) — no `for` loops allowed.

   Run your script for two cases: `x = 1./sqrt(2)` and `xvec = 0.01*[0:100]`; `x = 0.5` and `xvec = sort(rand(1,100))`. (In each case, include these assignment statements as the first two lines of your script.)

5. (Driscoll 5.1) Write a script to do the following: On a single figure, plot the functions $\sinh x$, $\cosh x$, $\tanh x$, and e^x for $-1 \leq x \leq 1$, with point spacing $\Delta x = 1/10$. Make sinh a red line, cosh a black dotted line, tanh a blue line with circles at each point, and e^x just green $\times$'s with no line. Make a legend. Label your axes and give the figure a title. Use `linspace` to create a vector of x values and call each MATLAB mathematical function with vector arguments to create the corresponding vector of "y" values. (See Section 5.4 of the text for a plotting "template.")

2.086 Numerical Computation for Mechanical Engineers
Fall 2012