Matlab Exercises_Recitation 1[†] — 2.086 Spring 2012

Recitation 1: Wednesday, 8 February / Friday, 10 February
MATLAB Exercises_Recitation 1 due: *Monday, 13 February 2012 at 5 PM by upload to Stellar*

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

`YOURNAME_MatlabExercises_Rec1`

which contains the completed scripts and functions for the assigned MATLAB Exercises_Recitation 1: 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. (Driscoll 1.1) Evaluate the following mathematical expressions in MATLAB. Display your evaluations by omitting semicolons at the end of your lines.

   (a) 2^8
   (b) $\frac{22}{7} - \pi$
   (c) $\sqrt[4]{9^2 + \frac{19^2}{22}} - \pi$
   (d) $\pi - e^{\pi}$
   (e) $\log_{10}(2)$
   (f) $\tanh(e)$

   Note it is crucial to write arithmetic operations in a transparent and de-buggable form. Good practices include breaking a single formula into several pieces each of which is evaluated in a separate MATLAB statement, breaking a single MATLAB statement into multiple lines with the ellipsis continuation syntax, and including parentheses and spaces. (Order of precedence is not an excuse for impossibly dense lines of MATLAB code.)

2. Evaluate the following expressions, omitting semicolons at the ends of your lines. You should have the `format short` (the default) in effect for all but the last two items.

   (a) $1/0$
   (b) $0/0$
   (c) $1 - 10^{-8}$
   (d) $1 - 10^{-20}$
   (e) $1 - 10^{-8}$ with `format long` in effect
   (f) $1 - 10^{-20}$ with `format long` in effect

[†]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.

Note that this exercise highlights two different points: there is an internal representation of floating point numbers with only a finite number of bits of precision; there are a number of different formats possible for presentation of floating point numbers.

Background: The Fibonacci sequence is defined by,

$$F_n = \begin{cases} 1, & n = 1; \\ 1, & n = 2; \\ F_{n-1} + F_{n-2}, & n \geq 3. \end{cases}$$

We will take advantage of this sequence in the following exercises (and an exercise for next week).

Cartoon by Sidney Harris

---

3. Write a script which calculates F_{20}. Use a `for` loop. Note that at any given time you need only store the three active members of the sequence, say `F_curr`, `F_old`, and `F_older`, which you will "shuffle" appropriately.

   Note you can also use some "quick and dirty" plotting to help debug and confirm correct behavior: add a `hold on` and `plot(n,F_curr,'o')` in your loop. Note that "quick and dirty" plotting for debugging is different from "presentation" plotting for consumption by others; the latter must contain labels and titles and legends to be useful (we review this in Recitation 2).

4. Write a script which finds N^* such that $F_{N^*} < 1000$ and $F_{N^*+1} \geq 1000$. Use a `while` loop. Note the "interior" block of your `while` loop will be quite similar to the block of your `for` loop of Exercise 3 but now you must include a (say) `Nstar_tmp` variable which is initialized outside the loop and incremented each time through the loop.

5. Write a script which finds the sum of the first 40 Fibonacci numbers F_n, $1 \leq n \leq 40$, for

which F_n is divisible by either 2 or 5,

$$\sum_{n=1}^{40} \begin{cases} F_n \text{ if } F_n \text{ is divisible by 2 or 5} \\ 0 \text{ otherwise .} \end{cases}$$

(For example, the first Fibonacci number to be included in the sum will be $F_3 = 2$, and the second Fibonacci number to be included in the sum will be $F_5 = 5$.) Use a `for` statement similar to Exercise 3 but now add the necessary relational and logical operations, an `if`, and also a `fibsum` summation variable which is initialized outside the loop and appropriately incremented inside the loop. You should find the MATLAB built-in function `mod` helpful.