**CSCI-130 (Wayne Wall): MATLAB Script Requirements**

The following are requirements for all MATLAB scripts. Points will be deducted if the script is lacking a requirement and/or the requirement is poorly implemented. The number of points deducted is at my sole discretion.

* Scripts must start with a comment line with the programmer’s name, immediately followed by a few comment lines briefly (but concisely) stating the problem that’s being solved. For example:  
    
  **% Programmer: . . .  
  % Description: Use gravitation law to calculate force exerted by the earth on the moon.**
* To start MATLAB in a known state when a script is run, the first executable statements must be:  
    
  **clear; clc; close all; format;**
* Important constants should be placed in named storage and the names (not the numeric values) should be used in subsequent calculations. Meaningful descriptive names should be used. Any measurement units associated with a name should be indicated in an associated comment. For example:  
    
  **univ\_grav\_constant = 6.673e-11; % newton meters^2 / kg^2  
  earth\_mass = 6.0e24; % kg  
  moon\_mass = 7.4e22; % kg  
  distance\_apart = 3.9e8; % meters**
* Use intermediate named variables to break down long and/or complex calculations into simpler to understand steps. Meaningful descriptive names should be used. If measurement units are associated with a variable, they should be indicated in an associated comment. For example:  
    
  **mass = 10; % kg  
  acceleration = 9.8; % meters/sec^2  
  force = mass \* acceleration; % newtons**
* Always store the final result in a variable with a meaningful descriptive name. Then display the result by placing the variable name on a line by itself with no “**;**” (semi-colon) at the end. For example:  
    
  **pounds\_CO2\_per\_year = gallons\_per\_year \* pounds\_CO2\_per\_gallon;  
  pounds\_CO2\_per\_year**  
    
  [No comment indicating the units is needed here as the units are self-evident from the variable name.]
* In general, when a script is run, there should be minimal output in the Command Window. Always display the variable holding the final answer last. Only display the results of an intermediate calculation if it will enhance the reader’s understanding of the solution process. For example:  
    
  **% Matrix of distances apart (10 equally spaced distances)  
    
  distances\_apart = linspace(3.8e8, 4.0e8, 10); % meters  
  distances\_apart  
    
  force = univ\_grav\_constant \* earth\_mass \* moon\_mass ./ (distances\_apart .^ 2) % newtons**
* Use horizontal and vertical “whitespace” (i.e., spaces and blank lines) to separate and delineate the structure of your script programming so as to make it easier for the human eye-brain combination to identify and comprehend. Don’t scrunch things close together just for the sake of reducing the amount of typing. Doing so may (or may not) save a little time when you’re coding the initial script, but the important goal is to make the script easy for someone (perhaps your boss) to understand later on. (See any of my homework solutions as examples.)
* Your script should show your understanding and correct use of MATLAB concepts and techniques. For example, don’t do a series of individual similar calculations when the quantities can be stored in a vector/matrix and then MATLAB matrix operations can be performed on the whole lot at once.
* In addition to the above, the script, when run, must NOT result in any compile or execution errors AND it must produce a correct final result.
* All words should be spelled correctly! Spelling errors reflect poorly on both programmer and employer.
* Plots and graphs should provide the viewer with enough information to understand what’s drawn without need to reference the script. Specifically, these things are mandatory:  
  + Descriptive title and axis labels.
  + Axis labels should indicate the quantities represented and their measurement units.
  + If more than one graph is on a set of axis, then a legend (with descriptive text) is required.
  + If multiple figure windows, they should be offset so they’re not stacked directly on top of each other.

In the future, in order to reduce the volume of comments I have to put in your graded scripts, I’ve assigned three letter codes to each requirement. Here are the codes and their meaning:

**NPR** – Name, problem number and concise problem statement as comments at beginning of script.  
**CLR** – **clear**; **clc; close all; format;**  commands as first executable statements.  
**UNC** – Use named constants.  
**MSN** – Meaningful symbolic names.  
**SMU** – Specify measurement units.  
**IVU** – Intermediate variable usage to break complex ops into simpler pieces.  
**SFR** – Store final result in a meaningful name.  
**DFR** – Display final result.  
**FRC** – Final result is correct.  
**MCW** – Minimal command window output.  
**WSU** – White space usage to enhance readability.  
**PUC** – Proper use of MATLAB concepts.  
**NCE** – No compile or runtime errors.  
**PLT** – Missing plot/graph title.  
**PLA** – Missing plot/graph axis label.  
**PLL** – Missing plot/graph legend.  
**SFW** – Stacked figure windows.  
**SPL** – Spelling error.

From now on, when I deduct points for a missing or poorly implemented requirement, I’ll simply list the code followed by the number of points deducted. For example:

**% (WW) MSN (-3), CLR (-5)**