ENGR 15100: Software Tools for Engineers
Spring 2015

Computer Assignment #8

Due: Tuesday, April 7, 2015, 9am CST

Departments of Engineering
School of Engineering, Mathematics, & Sciences
Purdue University Calumet

PURDUE
UNIVERSITY
CALUMET

1. OBJECTIVE

Continue working with **`if-end`** and **`for-end`** statements.

2. PROCEDURE

Task I: Maximum of a Function in an Interval [25 points]

In a script file named `LASTNAME_LAB8_TASK1.m`, write a MATLAB program that finds the maximum value of function $y(x) = (x+1)^3(x-1)(x-2)$ in the interval $x = [-2, +2]$. The program also finds the value of x in the interval that caused the maximum $y(x)$ to occur. Unless otherwise specified, suppress output to the Command Window.

(a) **[2 points]** Clear the MATLAB Workspace and the MATALB Command Window.

(b) **[6 points]** Create variables `xMax` and `yMax`. Initialize each variable to an appropriate scalar value.

(c) **[6 points]** Declare a **`for-end`** statement with a loop variable named `x` that will be assigned to every element of a row vector whose elements are equally spaced values starting with `-2.0`, ending with `+2.0`, with a increment of `0.01`.

(d) **[7 points]** The body of the **`for-end`** statement should perform the following:

- **[3 points]** Compute the current scalar value of a variable `y` in terms of the current scalar value of loop variable `x` according to the function $y(x) = (x+1)^3(x-1)(x-2)$.
- **[4 points]** When appropriate, update the scalar values of `xMax` and `yMax`. You may not use any built in functions, including built-in function **`max()`**.

(e) **[4 points]** Using multiple instances of the built-in **`fprintf()`** function, display the values of `yMax` and `xMax` according to the following formatting specifications:

- **[2 points]** Format the value of `yMax` as a fixed-point real number.
- **[2 points]** Format the value of `xMax` as a fixed-point real number.

After completing the above steps, the result of executing the script should look like the output below.

```
The maximum value of y is 2.640522.
The value of x causing the maximum value of y is 0.370000.
```

Task II: Modifying and Displaying N-elements of an Vector Per Line [60 points]

In a script file named **LASTNAME_LAB8_TASK2.m**, write a program according to the following specifications. **Unless specified, suppress all MATLAB Command Window output**. After completing the program, the result of executing your program should look similar to the output shown below.

```
>> LASTNAME_LAB8_TASK2
Enter the number of elements for a vector v: 10
The elements of vector v are:
-----------------------------
    12    60    23    40
    12    45    19    49
    52    56

V contains 2 prime numbers.
V contains 1 multiples of 3 in the range (25, 55).

After updating the multiples of 3, V now contains:
-------------------------------------------------
    -2    -4    23    40
   -10   -12    19    49
    52    56
```

(a) **[2 points]** Clear the MATLAB Workspace and the MATALB Command Window.

(b) **[2 points]** Create a variable named **N** and assign to it an integer obtained by prompting the user to enter the size of a row vector **V**. *Assume the user always enters a positive integer.*

(c) **[5 points]** Create a variable named **V** and assign to it an **N**-element row vector whose elements are randomly generated integers uniformly chosen from the closed interval **[0, 75]**.

(d) **[18 points]** Declare a first **for-end** statement with a loop-variable named **k** that will represent each index/address of row vector **V**. In the body of the first **for-end** statement, display all elements of row vector **V**, **4** elements per line. Format each integer value within a **6**-digit field-width. Ensure the cursor (**>>**) is placed at the start of the next line for any value of **N**.

(e) **[20 points]** Declare another **for-end** statement with a loop-variable named **k** that will represent each element in row vector **V**. In the body of this **for-end** statement, perform the following:

- count the number of prime numbers contained in row vector **V**. Use the built-in function **isprime()**.
- count the number of elements of row vector **V** that are multiples of **3** (i.e. evenly divisible by **3**), greater than **25**, but less than **55**.
- multiply (and update) each of the elements in vector **V** that are only multiples of 3 by two times the negative of their index/position within row vector **V**.

(f) **[4 points]** Using multiple instances of the built-in **fprintf()** function, display the following:

- Number of prime numbers contained in **V**. Format the number as an integer.
- Number of multiples of **3** contained in **V** in the range **(25, 55)**. Format the number as an integer.

(g) **[9 points]** Repeat step (d) to display all the elements contained in row vector **V**. Re-use as much of the code as possible that was as part of completing step (d).

Test your program with at least the following scalar values for **N**: **+1**, **+10**, and **+24**.

Task III: Finding the Zero Crossings of a Functions [30 Points]

In a script file named **LASTNAME_LAB8_TASK3.m**, write a program according to the following specifications. **Unless specified, suppress all MATLAB Command Window output**. After completing the program, the result of executing your program should look similar to the output shown below.

```
>> LASTNAME_LAB8_TASK3.m
Enter a particular value of F0 for the line f(x) = F0: 0.0
Function y(x) crosses the line f(x) = 0.000 at x = <x1>.
Function y(x) crosses the line f(x) = 0.000 at x = <x2>.
...
Function y(x) crosses the line f(x) = 0.000 a total of <num_times>.
```

(a) **[1 point]** Clear the MATLAB Workspace and the MATALB Command Window.

(b) **[3 points]** Create a variable named **x** and assign to it a row vector whose elements have equally spaced values starting with **-0.5**, ending with **+2.0**, with an increment of **1e-5**.

(c) **[4 points]** Create a variable named **y** and assign to it a row vector whose elements are obtained by evaluating function $y(x) = 6e^{-1.5x}cos(8\pi x)$ for each element contained in row vector **x**.

(d) **[1 point]** Create a variable named **F0** and assign to it a scalar value obtained by prompting the user with the prompt '**Enter a value of F0 for the line f(x) = F0: '**.

(e) **[4 points]** Plot **y** vs. **x** and **F0** vs. **x** an axis contained within a Figure Window named Figure 1. Color the curve of **y** vs. **x** in red and the curve of **F0** vs. **x** in blue. Hold the current plot and turn on the major grid axis lines so as to more easily visualize the resulting curves.

(f) **[15 points]** Define a **for-end** statement with a loop variable named **k** that will represent the indices/positions of row vector **y**. In the body of the **for-end** statement, perform the following:

- Count the number of times $y(x)$ crosses the line $f(x) = F_0$. ***Hint**: One may accomplish this by comparing **F0** with the value of the current value of **y** and the previous value of **y**.*
- Use built-in **text()** function to draw a *left arrow* that "points" to each data point at which $y(x)$ crosses the line $f(t) = F_0$. The syntax is **text(xPos, yPos, '\leftarrow')**
- use the built-in **fprintf()** function to display the value of each x wherein $y(x)$ crosses the line $f(x) = F_0$. Format each value of x as a fixed-point real number showing a maximum of **3** digits beyond the decimal point.

(g) **[2 points]** Using the built-in **fprintf()** function, display the number of times $y(x)$ crosses the line $f(x) = F_0$. Format the number as an integer.

Test your script with the following scalar values for **F0**: **+20**, **-12**, **+5**, **-0.5**, and **0.0**. For each value of **F0**, visually verify the results generated in steps (f) through (g) with those generated in step (e).