ENGR 160 Algorithms and Applications Sec 02 Spring 2018

Homework Assignment #7 Due 9:00am Feb 23 (F)

Problem 7.1

Create a MATLAB script file to print multiplication table in the format shown in exercise 16 of Chapter 5. Your program should prompt the user to enter a positive integer number N, as the number of rows in the result. The example given in the book shows the result when N=5. Your program should be able to perform error-checking for the user inputted N value. If the user inputs a value that is not a positive integer, print an error message ‘Invalid N value!’ and prompt the user to enter N again. If the user could not enter a valid value of N for 5 times, print an error message ‘You have entered invalid values for 5 times!’, and stop the program. Test your program using the following two sequences of input: 1) -5; 3.5; 0; 10. 2) -1, -2.5; 0; 3.5; -2. Attach your results.

Problem 7.2

Create a MATLAB function called ‘geomser’ to calculate the sum of the following geometric sequence. $1 + r + r^2 + r^3 + r^4 + ... + r^n$

Your function should receive two input arguments, r and n, where r is a real number, and n is a positive integer. Your function should return the sum as the output argument if the inputs are valid. Otherwise, it should return a 0. Use the following input values to test your function. Attach your results. (1) r=1, n =5; (2)r=0.5, n = 9; (3) r = 0.3, n = 9.5; (4) r = 2, n = -5

Problem 7.3

The electricity accounts of residents in a very small rural community are calculated as follows:

- If 500 or fewer units are used, the cost is 2 cents per unit.
- If more than 500, but not more than 1000, units are used, the cost is $10 for the first 500 units and 5 cents for every unit in excess of 500.
- If more than 1000 units are used, the cost is $35 for the first 1000 units plus 10 cents for every unit in excess of 1000.
- A basic service fee of $5 is charged no matter how much electricity is used.

Write a MATLAB function named **billing**, which takes a vector of all residents’ electricity usages (in units) as input argument, and generate a vector of all residents’ payments as output argument. To test your function, write another script file that calls the function **billing** and plot the payments versus the usages for the range of 0< =usages<=1500 units with step size of 1 unit. Attach your figures.

Problem 7.4 (10points)

A moving-average filter is a basic digital signal processing tool to remove noise. Assume x is a vector of measured data samples with random noises, we can estimate the real data samples as a vector y as follows,

$$y(n) = \frac{1}{M}\sum_{l=0}^{M-1} x(n-l), \qquad for \quad n = 1:length(x)$$

That is, every sample in y is an M-point average over the last M-1 samples and the current sample in x. This is called an M-th order moving-average filter. For the first M-1 samples, we only average from the first element in x up to the current value. Create a Matlab function named MAfilter to implement the M-th order moving average filter, the input argument should be a vector x and a filter order M, and the output should be the filtered vector y, with exactly same number of samples as in x. Use the following input to test your function. Attach your results.

(1) x=[1,2,3,4,5]; M = 2;
(2) x = [1, 2.1, 2.9, 4.3, 4.8]; M = 3;

Note: You can NOT use any built-in filter functions in Matlab.

Problem 7.5 (10 points)
Create a script file named denoising.m to do the following:
Step 1: generate the noisy vector x as follows:

1. First, generate a one-period sinusoidal signal s(n) with 100 linearly spaced samples from 0 to 2π,
2. Then, generate 100 random noise samples w(n) ranging from -0.1 to 0.1.
3. Last, add each noise sample to the corresponding signal sample to generate x.

Step 2: Call your MAfilter function written in Problem 7.4 with two different M values: 2 and 5.
Step 3: Plot the original clear sinusoidal signal, the noisy signal, and both output signals generated in Step 2 in one single figure. Use figure titles, labels and legends to show the comparisons.
Note: If you could not make MAfilter work in problem 7.4, you can use the built-in function filter() in the Digital Signal Processing toolbox to do Step 2 in the following way:

```
y  = filter([0.5, 0.5], 1, x)   for M = 2
y = filter(0.2*ones(1,5), 1, x) for M=3
```