CSCI 1320 Computer Science I: Engineering Applications – Fall 2018
Instructor: Zagrodzki
Assignment 4
Due Sunday, October 7 by 6pm

New Matrices

1. (30 points) Write a function called *makemat* that will receive two column vectors as input arguments, and from them create and return a matrix with two columns (n x 2). If either of both of the vectors passed in are row vectors, transpose them into column vectors. If either input is a matrix, your program should throw a unique error (create your own error message.) You may not assume that the length of the vectors is known. Also, the vectors may be of two different lengths. If that is the case, add 0's to the end of one vector first to make it as long as the other (this is often referred to as padding the matrix with 0s - do not use any built-in functions such as padarray()). Some examples of calling the function and the result you should see:

```
>> makemat([2;8;7],[9;0;3])

ans =

     2     9
     8     0
     7     3


>> makemat([2 8 7],[9 0 3 4 3])

ans =

     2     9
     8     0
     7     3
     0     4
     0     3
>> makemat([4 8;3 6],[4 9 0])
Error using makemat (line 17)
Can't process a matrix, try again. Undefined for
matrix input arguments.
```

Smallest number adjacent to a zero

2. (40 points) Write a function named *smallest* that will receive one vector *vec* as input argument. The elements of *vec* are of type double, and they can be positive or negative numbers. The function will return the smallest element of *vec* that comes before an element with the value 0(zero).

If the input argument vector is empty, or if there is no element with the value zero (hence, no element adjacent to a zero), then the function should return an **empty vector**.

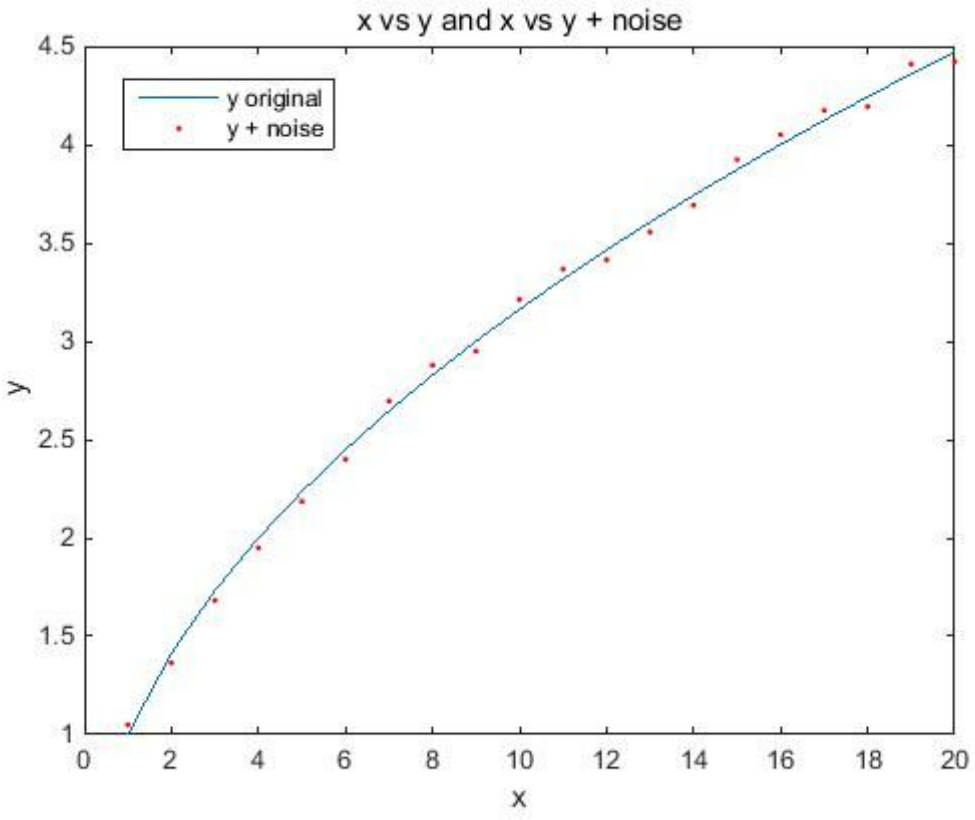
You are **NOT** allowed to use the following built-in Matlab functions: **imdilate, unique, diff** (or any other exotic built-in functions not covered in lecture or recitation). You are allowed to use the **find** and **min** built-in Matlab functions. The function should return the desired output regardless of the input vector size.

Some examples of calling the function:

```
>> vec = [1 5 3 0 2 7 0 8 9 1 0];
% Here the elements that come before 0 are [3 7 1] and
hence % the smallest number among these elements is 1
>> smallest(vec)
ans =
     1
>> smallest([1 2 3 2 4])
ans =
     []


>> smallest([])
ans =
     []

>> smallest([11])
ans =
     []

>> smallest([0, 0])
ans =
     0

>> smallest([3 -4 0 0 -6])
% Here the elements adjacent to 0 are [-4 0 ] and hence
% the smallest number among these elements is -4
ans =
     -4
```

Adding Noise to Data

3. (30 points) Write a script *noisify.m* where you first create a vector *x* that has 20 linearly spaced elements with values ranging from 1 to 20. Second, set a vector *y* equal to $\sqrt{x}$. Plot this curve with x on the horizontal axis and y on the vertical axis. Finally, create a new vector, call it *y2*, and add random noise of amplitude between -0.05 and 0.05 (i.e. -0.05<noise<0.05) to the original vector y. (Hint: *y2* is of the same length as *y*. Each element of *y2* is either larger or smaller than the corresponding element of y.) Plot the original curve y and the noisy-y (using red dot for the noisy data points). Give appropriate axis labels, title, and legend (use *legend()* function). The resulting plot should look similar to this:



Submitting the assignment:
Make sure all your .m files are well commented and they include a comment block with your name, student ID, course number, assignment number and recitation section. Zip all the files together and submit the resulting *.zip* file through Moodle as Assignment 4 by due date.