Programming Assignments

Introduction to Programming with MATLAB

Lesson 4

- Unless otherwise indicated, you may assume that each function will be given the correct number of inputs and that those inputs have the correct dimensions. For example, if the input is stated to be three row vectors of four elements each, your function is not required to determine whether the input consists of three two-dimensional arrays, each with one row and four columns.
- Unless otherwise indicated, your function should not print anything to the Command Window, but your function will not be counted incorrect if it does.
- Note that you are not required to use the suggested names of input variables and output variables, but you must use the specified function names.
- Also, read the instructions on the web page on how to test your functions with the auto-grader program provided, and what to submit to Coursera to get credit.
- Note that starred problems, marked by ***, are harder than usual, so do not get discouraged if you have difficulty solving them.
- Note that we have not covered if-statements or loops, so they are neither needed nor allowed!
- You need MATLAB r2012a or newer or MATLAB Online to run the grader! Older versions are not supported.

1. Write a function called **`intquad`** that takes as its input arguments two scalar positive integers named **`n`** and **`m`** in that order. The function returns **`Q`**, a **`2n`**-by-**`2m`** matrix. **`Q`** consists of four **`n`**-by-**`m`** submatrices. The elements of the submatrix in the top left corner are all 0s, the elements of the submatrix at the top right are 1s, the elements in the bottom left are 2s, and the elements in the bottom right are 3s.

2. Write a function called **`sindeg`** that takes a matrix input called **`deg`**. The function returns a matrix of the same size as **`deg`** with each of its elements containing the sine of the corresponding element of **`deg`**. Note that the elements of **`deg`** are given in degrees and not radians. As a second output, the function returns a scalar that contains the average value of the first output. You are not allowed to use the **`sind`** and **`cosd`** built-in functions, but the use of any other function is acceptable.

3. Write a function called **`simple_stats`** that takes a matrix **`N`** as an input and returns the matrix **`S`** as the output. **`S`** has the same number of rows as N. Each element of the first column of S contains the mean of the corresponding row of **`N`**. Similarly, the second column contains the median values; while the third column has the minimums. Finally, each element of the fourth column of **`S`** is equal to the maximum value of given row of **`N`**. (Hint: note that the grader will not test this with column vectors, but you should try to solve it for that case too. Remember, **`help`** is your friend.)

4. Write a function called **`odd_rms`** that returns **`orms`**, which is the square root of the mean of the squares of the first **`nn`** positive odd integers, where **`nn`** is a positive integer and is the only input argument. For example, if **`nn`** is 3, your function needs to compute and return the square root of the average of the numbers 1, 9, and 25. You may use built-in functions including, for example, **`sum`** and **`sqrt`**, except for the built-in function **`rms`**, which is not allowed.

5. Write a function called **`fence`** that takes two scalar inputs: **`lng`**, the length of a straight fence we need to build and **`seg`**, the length of one segment of fencing material. A segment needs to have a pole at both ends, but two neighboring segments always share a pole. The function returns two scalar outputs: the number of segments we need for the given length of fence and the number of poles needed. Note that a segment can be cut shorter if needed. For example, to build a 75m long straight fence using 10m segments, we need 8 segments. You may find the **`ceil`** built-in function handy.

6. Write a function called **`zero_stat`** that takes a matrix as an input that only has 0 and 1 elements. The function needs to compute and return the percentage of 0 elements in the matrix. For example, if there are 10 zeros and 15 ones in a 5-by-5 matrix that is provided as an input to **`zero_stat`**, it would return 40 because 40% of the elements are zero.

7. *** Write a function called **`reverse_diag`** that creates a square matrix whose elements are 0 except for 1s on the reverse diagonal from top right to bottom left. The reverse diagonal of an n-by-n matrix consists of the elements at the following indexes: (1, n), (2, n-1), (3, n-2), … (n, 1). The function takes one positive integer input argument named **`n`**, which is the size of the matrix, and returns the matrix itself as an output argument. Note that using the built-in functions **`eye`** and **`diag`** are not allowed. (Hint: you can index into a matrix with a single index and MATLAB will handle it as if it was a vector using column-major order. Note that the grader will not test for n = 1, but try to solve it for that case too.)

8. If we list all the natural numbers up to 15 that are multiples of 3 or 5, we get 3, 5, 6, 9, 10, 12 and 15. The sum of these multiples is 60. Write a function called **`sum3and5muls`** that returns the sum of all the unique multiples of 3 or 5 up to **`n`** where **`n`** is a positive integer and the only input argument of the function. (Credit: Project Euler)