Programming Assignments

Introduction to Programming with MATLAB

Lesson 3

- 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.
- Finally, 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.

1. Write a function called **`odd_index`** that takes a matrix, `M`, as input argument and returns a matrix that contains only those elements of `M` that are in odd rows and columns. In other words, it would return the elements of `M` at indices (1,1), (1,3), (1,5), …, (3,1), (3,3), (3,5), …, etc. Note that both the row and the column of an element must be odd to be included in the output. The following would not be returned: (1,2), (2,1), (2,2) because either the row or the column or both are even. As an example, if `M` were a 5-by-8 matrix, then the output must be 3-by-4 because the function omits rows 2 and 4 of `M` and it also omits columns 2, 4, 6, and 8 of `M`.

2. Write a function called **`int_col`** that has one input argument, a positive integer **`n`** that is greater than 1, and one output argument **`v`** that is a column vector of length **`n`** containing all the positive integers smaller than or equal to **`n`**, arranged in such a way that no element of the vector equals its own index. In other words, **`v(k)`** is not equal to **`k`** for any valid index **`k`**.

3. Suppose we have a pile of coins. Write a function called **`rich`** that computes how much money we have. It takes one input argument that is a row vector whose 4 elements specify the number of pennies (1 cent), nickels (5 cents), dimes (10 cents), and quarters (25 cents) that we have (in the order listed here). The output of the function is the value of the total in units of dollars.

4. Write a function called **`light_time`** that takes as input a row vector of distances in miles and returns two output arguments. The first output argument is a row vector of the corresponding times in minutes that light would take to travel each distance. To check your math, it takes a little more than 8 minutes for sunlight to reach Earth which is 92.9 million miles away. The second output is a vector of the input distances in kilometers. Assume that the speed of light is 300,000 km/s and that one mile equals 1.609 km.

5. Write a function called **`pitty`** that takes a matrix called **ab** as an input argument. The matrix **`ab`** has exactly two columns. The function should return a column vector **`c`** that contains positive values each of which satisfies the Pythagorean Theorem, $a^2 + b^2 = c^2$, for the corresponding row of **`ab`** assuming that the two elements on each row of **`ab`** correspond to one pair, *a* and *b,* respectively, in the theorem. Note that the built-in MATLAB function **`sqrt`** computes the square root and you are allowed to use it.

6. Write a function called **`bottom_left`** that takes two inputs: a matrix **`N`** and a scalar **`n`**, in that order, where each dimension of **`N`** is greater than or equal to **`n`**. The function returns the **`n`**-by-**`n`** square array at the bottom left corner of **`N`**.

7. Write a function called **`mean_squares`** that returns **`mm`**, which is the mean of the squares of the first **`nn`** positive integers, where **`nn`** is a positive integer and is the only input argument. For example, if **`nn`** is 5, your function needs to compute the average of the numbers 1, 4, 9, 16, and 25. You may use any built-in functions including, for example, **`sum`**.

8. Write a function called **`hulk`** that takes a row vector **`v`** as an input and returns a matrix **`H`** whose first column consist of the elements of **`v`**, whose second column consists of the squares of the elements of **`v`**, and whose third column consists of the cubes of the elements **`v`**. For example, if you call the function likes this, **`A = hulk(1:3)`** , then **`A`** will be **`[ 1 1 1; 2 4 8; 3 9 27 ].`**