ENGR 105 – Introduction to Scientific Computing

Assignment 3

Due by 11:59 pm on Weds. 2/2/2022 on Gradescope

Problem 1 (24 pts): Translate the following mathematical expressions into MATLAB statements. The output of each expression should be assigned to variable names of your choice. Assume all symbolic variables represent arrays (in particular "vectors" or "lists" of numbers) with the same dimensions of 1 x *n* where *n* is some integer greater than 1. Assume that all computations are intended to be performed element-by-element (i.e. not matrix multiplication of division) such that the expressions resolve to a 1 x *n* array. Only utilize array operations (e.g. `.*` and `./`) when absolutely necessary and loops should not be employed. You are encouraged to check the output of the statements "by hand" and using the values in your test vectors to make sure that your statements are correct.

As an example, the expression pv might be translated to `result_ix = p.*v`.

Submit your work for this problem in a single script with the name `Assignment3_problem1.m`.

i) $p + \frac{2}{u}$

ii) ue^{p} (where e is Euler's number)

iii) $1 - \frac{1}{\cos(x)}$

iv) $\frac{p + \frac{w}{u+v}}{p + \frac{w}{u-v}}$

v) $x^{1/2}$

vi) $50y^{y+z}$

vii) $x^{y^{z}}$

viii) $x - \frac{x^3}{3!} + \frac{x^5}{5!}$

Problem 2 (30 pts): You are fond of another student in the ENGR 105 course and wish to take them on a date. However, your potential date will only go out with you if you demonstrate superior MATLAB abilities! They challenged you to extract the phone number of your meeting location from a vector of numbers using a descrambling function you will create in MATLAB.

You potential date has told you that they will provide this number as a vector of numbers. You must perform the following operations on the vector (in order of appearance) to retrieve the correct phone number.

(1) Delete every 3rd index starting with the first index (e.g. indices 1, 4, 7, 10, etc.)

(2) Swap even and odd indices (e.g. index 1 goes to 2, index 2 goes to 1, index 3 goes to 4, index 4 goes to 3, etc)

(3) Take the last 6 indices and move them to the front of the vector

(4) Subtract 1 from every even index (e.g. indices 2, 4, 6, 8, etc.)

(5) Remove an equal number of indices from the left and right hand side of the vector to yield the 10-digit phone number

Your algorithm should be contained within the following function declaration where `dNum` and `sNum` represent the descrambled and scrambled number vectors, respectively.

```
function dNum = descrambler(sNum)
```

As a test, you will want to check if the following vector yields a Philadelphia area code.

```
[1, 9, 9, 1, 3, 3, 9, 3, 8, 5, 6, 1, 3, 1, 4, 7, 2, 2, 1, 9, 5, 1]
```

Your code should be generalized enough that it could process vectors of variable lengths to produce a 10-digit number. Note that not all vector lengths will work given the need to swap even and odd indices in step 2, removal of an equal number of indices in step 5, and output of a 10-digit number. For example, your code should work with vectors of length 15, 16, 18, 19, 21, 22, 24, 25, and 27.

You may find the `length()` command useful in your descrambler function.

Submit your function as `descrambler.m`.

Problem 3 (40 pts): Infinite series are often used in the sciences to represent mathematical functions. In computers, such series are computed to a finite number of terms as the time available to perform computations is *not* infinite. Even a couple of computed terms may be sufficient to faithfully represent the function within a fraction of a percent.

Consider the following infinite series representation of $\cos(x)$, which to the limit of infinite terms ($n = \infty$) will exactly match the value of $\cos(x)$.

$$\cos(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}$$

Create two functions that find the value of $\cos(x)$ based on its series representation that utilize - (1) a `for` loop and (2) vectorization - for any `n+1` number of computed terms. For the `for` loop variation of your code (method 1), you should initialize `val = 0` and then add the computed value of each term to the running total of `val` within the `for` loop. These functions should have the following function declarations.

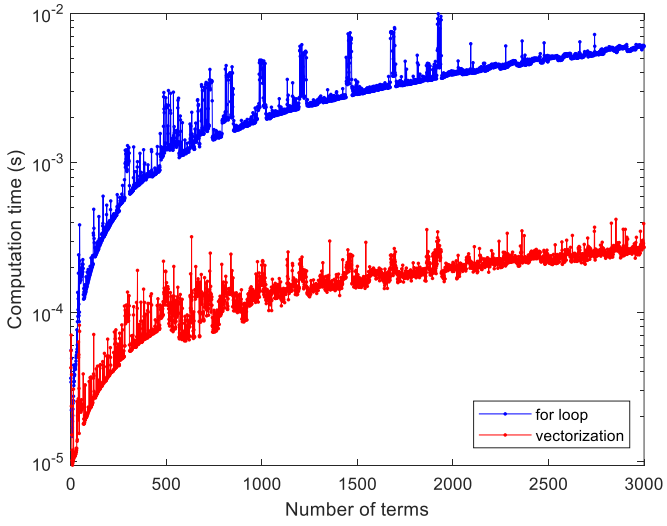
```
function [t,val] = cosSeriesFor(x,n)

function [t,val] = cosSeriesVect(x,n)
```

The output `t` is the computation time as measured between the function declaration and all statements in the function. The output `val` is the sum of the series for `n+1` terms. All inputs and outputs should be scalars. Your code may assume that `n` is an integer greater than or equal to 0. You will want to check your code (inspect `val` for a number of conditions of `x` and `n`) to ensure that your computations are correct. The following table lists the series components of $\cos(x)$ for increasing numbers of terms.

Input `n`	Number of terms	Series representation
0	1	1
1	2	$1-\frac{x^2}{2!}$
2	3	$1-\frac{x^2}{2!}+\frac{x^4}{4!}$
3	4	$1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}$
4	5	$1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\frac{x^8}{8!}$

Create a script that calls `cosSeriesFor()` and `cosSeriesVect()` for $0 \le n \le 3000$ by increments of `n=1` and plot the computation time as a function of `n`. Your plot should look similar to the following plot. Note that your computation times may vary depending on the speed of your computer and current activity of your central processing unit.



Your plot should include a linear x-axis, a logarithmic y-axis, axis titles, and a legend. See the MATLAB functions `semilogy()`, `xlabel()`, `ylabel()`, and `legend()` for help with this. You can overlay one plot over another by indicating `hold on` before invoking the next plot. It is best practice to indicate `hold off` after you have finished overlaying plots.

You may also find the following MATLAB functions useful: `sum()`, `tic`, and `toc`.

Answer the following questions:

i) What do you conclude about the computing time necessary when performing calculations using loops versus vectorization?

ii) How many terms are necessary to approximate the value of $\cos(\pi/6)$ to within 0.1% of its true value?

Submit your functions as `cosSeriesFor.m` and `cosSeriesVect.m` and your script that calls the functions and performs plotting as `Assignment3_problem3_compareAndPlot.m`. Upload your plot as `Assignment3_problem3_compTime.jpg`. Answer the questions posed above in your `README.txt` submission.