Computational Methods and Applications (AMS 147)

Homework 1 - Due Jan 20, 2019

Please submit to CANVAS a .zip file that includes the following four Matlab functions:

`compute_factorial.m`
`compute_Euclidean_norm.m`
`matrix_times_vector.m`
`pi_series.m`

Exercise 1 The factorial of a natural number is defined as

$$n! = n(n-1)(n-2)\cdots 1, \qquad 0! = 1. \tag{1}$$

Write a Matlab/Octave function `compute_factorial.m` that takes an integer number as input and returns (1). The function should be of the following form

```
function [b] = compute_factorial(n)
```

Input:
`n`: natural number (possibly including 0)

Output:
`b`: factorial of `n`

You are not allowed to use the Matlab function `factorial(n)` in your code.

Exercise 2 The Euclidean norm of an n-dimensional vector is defined as

$$\|\boldsymbol{x}\| = \sqrt{\sum_{k=1}^{n} x_k^2}. \tag{2}$$

Write a Matlab/Octave function `compute_Euclidean_norm.m` that computes the norm (2), for an arbitrary input vector $\boldsymbol{x}$. The function should be of the following form

```
function [z] = compute_Euclidean_norm(x)
```

Input:
`x`: n-dimensional vector (either column vector or row vector)

Output:
`z`: norm of the vector

Hint: You can use the `for` loop. The number of components of the input vector can be determined by using the Matlab command `length(x)` (see the Matlab/Octave documentation). You can compare the output of your function with the Matlab/Octave function `norm(x)`.

Exercise 3 Write a Matlab/Octave program `matrix_times_vector.m` that computes the product between an n-dimensional square matrix $\boldsymbol{A}$ and an n-dimensional (column) vector $\boldsymbol{x}$. The components of the (column) vector $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ are defined

$$y_i = \sum_{j=1}^{n} A_{ij} x_j \qquad i = 1, ..., n. \tag{3}$$

The function should be of the following form

```
function [y] = matrix_times_vector(A,x)
```

Input:
`A`: $n \times n$ matrix
`x`: $n \times 1$ vector

Output:
`y`: $n \times 1$ vector

You are not allowed to use the Matlab expression `A*x` within your code.

Hint: You can use two nested `for` loops to compute the vector $\boldsymbol{y}$ (one loop computes the sum (3) while the other one controls the index i in (3)). The size of the matrix `A` can be determined by using the Matlab command `size(A)` (see the Matlab/Octave documentation). You can debug your function by comparing the output with the Matlab expression `A*x`, for simple matrices `A` and vectors `x`.

Exercise 4 The number π can be defined as a limit of various converging series of numbers. For example,

$$\pi = \lim_{n\to\infty} P_n, \qquad \text{where} \qquad P_n = 3\sum_{k=0}^{n}(-1)^k\left(\frac{1}{6k+1}+\frac{1}{6k+5}\right). \tag{4}$$

Write a Matlab/Octave function `pi_series.m` retuns the first 15 partial sums of the series (4), i.e., the row vector `P` with components

$$\texttt{P} = [P_0 \quad P_1 \quad P_2 \quad \cdots \quad P_{14}]. \tag{5}$$

The function should be of the following form

```
function [P,n,p]=pi_series()
```

Output:
`P`: row vector (5) with components defined by the first 15 partial sums in (4).
`n`, `p`: see the Extra Credit exercises hereafter. If you do not want to code the extra credit parts, just set `n=0` and `p=0` at the end of your function `pi_series.m`.

Extra Credit 1 At the end of the Matlab function `pi_series.m`, write a code that returns the smallest integer number n such that

$$|P_n - \pi| < 10^{-4}. \tag{6}$$

To determine n, you can use a `for` loop combined with an `if`/`return` statement or, equivalently, a `while` loop (see the Matlab/Octave documentation).

Extra Credit 2 At the end of the Matlab function `pi_series.m`, write a code that estimates numerically the convergence order p of the series (4) (see LECTURE_1.pdf in CANVAS). To this end, define

$$e_n = |P_n - \pi| \qquad n = 0, 1, ... \tag{7}$$

To estimate the order of converge numerically, it is sufficient to compute the slope of the line passing through the points $A = (\log(e_n), \log(e_{n+1}))$ and $B = (\log(e_{n+1}), \log(e_{n+2}))$, for sufficiently large n. For example, you can use $n = 1000$ in your code.