Programming Project PART 1
CS 323 - Numerical Analysis

1 Introduction

This project will be divided in two parts, the first part (this handout) contains the first algorithms covered in class that you must implement in Matlab. The second part will be assigned after we have covered in class the remaining algorithms that you will need to implement.

- For this project you will work individually.
- You must turn in your own work.
- All your programs should run. No partial credit will be given for programs that do not run.
- You must use at least five test cases for each program. You must turn in the input and output of each one of your test cases.

Please start working as soon as possible.

2 About the programs

2.1 Programming Language

The programs must be written in Matlab (m files), and the input should be read from a file. Please refer to the following link:
https://www.mathworks.com/help/matlab/ref/fscanf.html
for information on how to read files in matlab.

2.2 Style

All your programs must be correctly indented and must include comments (use % in matlab) describing the major parts of the algorithm. If you use any other variables that are not the usual ones described in the lecture notes, please explain in your comments the meaning of each one of those variables.

3 Algorithms

The following is the list of all the algorithms that you must implement:

1. **Horner:** Given the coefficients of a polynomial $a_0, a_1, \ldots, a_n$, and a real number x_0, find $P(x_0), P'(x_0), P''(x_0), P^{(3)}(x_0), \ldots, P^{(n)}(x_0)$
   Sample input representing $P(x) = 2 + 3x - x^2 + 2x^3$, $x_0 = 3.5$:

   ```
   3
   2
   3
   -1
   2
   3.5
   ```

   the first number is the degree of the polynomial (n), the coefficients are in order $a_0, a_1, \ldots, a_n$, the last number is x_0.

2. **Newton with Horner:** Given the coefficients of a polynomial $a_0, a_1, \ldots, a_n$, an initial guess $x_0 \in \mathbb{R}$, an error tolerance ϵ, and a maximum number of iterations N, find **one root** of the polynomial.

   Input format is the same as above, plus two more lines for ϵ and N
   Important: no credit will be given if the method does not use Horner.

   The output is just one number representing the solution within the desired error tolerance, or a message saying that the solution was not found.

3. **Cramer's Rule:** Given a matrix A representing a system of equations, and a vector b of independent terms, use Gaussian Elimination to find all the determinants needed to solve the system. The program must output the value of each of the $n+1$ determinants as well as the solution $x_1, x_2, \ldots, x_n$.

   Sample input representing the system:

$$\begin{pmatrix} 3 & 4 & 2 \\ -1 & 3 & -4 \\ 2 & 2 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 5 \\ 2 \\ -6 \end{pmatrix}$$

   is:

```
3
3
4
2
-1
3
-4
2
2
5
5
2
-6
```

   The expected output should be:

```
determinant A = 41
determinant A1 = 215
determinant A2 = -53
determinant A3 = -114
x1 =  5.24390
x2 = -1.29268
x3 = -2.78049
```

4. **Neville's Method:** Given a set of $n+1$ points

$$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n) \text{ and } x_0 \in \mathbb{R}$$

use Neville's Algorithm to find $P_n(x_0)$ where $P_n(x)$ is the unique polynomial that goes through all of the given points.

Sample input representing the points $(-1, 1), (0, 0), (1, 1)$ and $x_0 = 0.5$:

```
2
-1
1
0
0
1
1
0.5
```

where the first number is n, then the pairs x_i, y_i, and the last number is x_0

The expected output should be:

```
P(0.5) = 0.25
```