COMPUTER SCIENCE 349A, Spring 2017
ASSIGNMENT #4 - 20 MARKS

DUE THURSDAY March 9, 2017 (11:30 p.m. PST)

This is a really large class and the logistics of grading assignments are challenging. Me and the markers require your help in making this process go smoothly. Please ensure that your assignments conform to the following requirements - any violation will result in getting a zero for the particular assignment.

- All assignments should be submitted electronically through the ConneX course website and shoud be **SINGLE PDF FILES**. No other formats will be accepted. Handwritten answers are ok but they will need to be scanned and merged into a single pdf file together with any code examples and associated plots.
- The assignment number, student name and student number should be clearly visible on the top of every page of your assignment submission.
- **PLEASE DO NOT COPY THE ASSIGNMENT DESCRIPTION IN YOUR SUBMISSION**
- The asnwers to the questions should be in the same order as in the assignment specification.
- Some of the questions of the assignments are recycled from previous years but typically with small changes in either the description or the numbers. Any submission that contains numbers from previous years in any questions will be immediately graded with zero.
- Any assignment related email questions should have a subject line of the form CSC349A Assignment X, where X is the number of the corresponding assignment.
- The total number of points for this assignment is 20.

1. (a) **(4 points)** Given input consisting of

   - a positive integer n,
   - a vector a with $n+1$ entries $a_1, a_2, \ldots, a_{n+1}$,
   - a vector y with n entries $y_1, y_2, \ldots, y_n$, and
   - a scalar x,

   write a MATLAB function with header

```
function p = PolyEval ( n, a, y, x )
```

   to evaluate the polynomial

$$\begin{aligned} p(x) &= a_1 + a_2(x+y_1) + a_3(x+y_1)(x+y_2) + a_4(x+y_1)(x+y_2)(x+y_3) + \cdots \\ &\quad \cdots + a_{n+1}(x+y_1)(x+y_2)(x+y_3)\cdots(x+y_n) \\ &= \sum_{j=1}^{n+1} a_j \prod_{k=1}^{j-1}(x+y_k) \end{aligned}$$

   using Horner's algorithm. Include a copy of your function M-file in your solution pdf.

   Note. Your function should use exactly $3n$ flops (floating-point operations).

   (b) **(2 points)** Use your function M-file from (a) to evaluate $p(1.234)$ when $n = 4$ and

$$a = [-1,\ 0,\ 2.33,\ -1.2,\ 2.2]$$
$$y = [-1,\ 1,\ -2,\ 2]$$

   Include the call to the function you used and the result.

2. MATLAB has several functions for standard polynomial operations, including `roots`, `poly` and `polyval`. If the coefficients of a polynomial are stored in a vector p (starting with the coefficient of the highest power of x), then

```
r = roots(p)
```

   will compute and store all of the roots of the polynomial in r. For example, if $p(x) = x^3 - 2x - 5$, in MATLAB you would enter

```
p = [1 0 -2 -5];
r = roots(p)
```

   or simply `r = roots( [1 0 -2 -5] )`

   in order to compute the 3 roots of $p(x)$.

   On the other hand, if r is any vector, then

```
p = poly(r)
```

will result in p being a vector whose entries are the coefficients of a polynomial $p(x)$ that has as its roots the entries of r. For example,

```
r = [1  2  3  4];
p = poly(r)
```

will result in `p = [1 -10 35 -50 24]`, since $p(x) = x^4 - 10x^3 + 35x^2 - 50x + 24$ has roots equal to 1, 2, 3 and 4. Finally, the function `polyval` evaluates a polynomial at a specified value. If `p = [1 -10 35 -50 24]`, then

```
polyval(p,-1)
```

gives the value of $p(x) = x^4 - 10x + 35x^2 - 50x + 24$ at $x = -1$, namely $p(-1) = 120$. Note that MATLAB uses HORNER'S ALGORITHM for polyval.

(a) **(3 points)** It is known that polynomial zeros that have multiplicity greater than 1 are ill-conditioned. Verify this for the polynomial $p(x)$ of degree 8 that has its 8 zeros all equal to 1, by letting `r = [1 1 1 1 1 1 1 1 ]` and using `poly` and `roots` to compute approximations to the 8 zeros of $p(x)$. Then add 0.001 to the coefficient of x^3 in $p(x)$ to obtain a new polynomial, say $q(x)$, and use `roots` to approximate the 8 zeros of $q(x)$.

Note: You can do this by executing

```
q = p;
```

in MATLAB and then modifying the appropriate entry of the vector p. For example,

```
q(k) = q(k) + 0.001
```

will increase the k-th entry of the vector q by 0.001.

Note: MATLAB doesn't compute the zeros of $p(x)$ very accurately (because the zeros are very ill-conditioned), but they are all close to 1, whereas the zeros of $q(x)$ are not.

(b) **(3 points)** Use `polyval` to evaluate $q(x)$ at the first zero of $q(x)$ computed by MATLAB in (a). This should give a result very close to 0 (although it is complex).

NOTE: If above you computed the roots of $q(x)$ with the statement

```
roots(q)
```

then the zeros of $q(x)$ will be stored as `ans(1), ans(2), ... , ans(8)` and you can use this to answer (b).

3. **(4 points)** The exact solution of the following system of linear equations $Mx = c$

$$\begin{bmatrix} -1 & 2 \\ 0.5 & -0.9995 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -20.5 \\ 10.245 \end{bmatrix}$$

is $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.5 \\ -10 \end{bmatrix}$. This problem is **ill-conditioned** with respect to perturbations in the vector c since, for example, the exact solution of the perturbed linear system

$$\begin{bmatrix} -1 & 2 \\ 0.5 & -0.9995 \end{bmatrix} \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix} = \begin{bmatrix} -20.55 \\ 10.24 \end{bmatrix}$$

is

$$\hat{x} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix} = M^{-1} \begin{bmatrix} -20.55 \\ 10.24 \end{bmatrix} = \begin{bmatrix} 1999 & 4000 \\ 1000 & 2000 \end{bmatrix} \begin{bmatrix} -20.55 \\ 10.24 \end{bmatrix} = \begin{bmatrix} -119.45 \\ -70 \end{bmatrix}$$

If $A = \begin{bmatrix} 1.5 & 2 \\ -1 & -2 \end{bmatrix}$, then $A^{-1} = \begin{bmatrix} 2 & 2 \\ -1 & -1.5 \end{bmatrix}$. Consider the problem of determining $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ such that $Ax = b$, where $b = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$.

Show that this problem is **well-conditioned** (with respect to perturbations only in the vector b) by considering a linear system $A\hat{x} = b + e$ with right hand side vector $b + e = \begin{bmatrix} 1 + e_1 \\ -2 + e_2 \end{bmatrix}$.

4. **(4 points)**

Solve (by hand, no programming) the linear system

$$\begin{bmatrix} -2 & 0 & 2 & 4 \\ 1 & 1 & -2 & -2 \\ 0 & 3 & -1 & -3 \\ 4 & -2 & -1 & -9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 4 \\ -7 \\ -9 \\ 5 \end{bmatrix}$$

using Naive Gaussian Elimination. Use the algorithm as it was presented (i.e. everybody should do the same steps in the same order) and show each of the derived linear systems in the forward elimination and the back substitution.