COMPUTER SCIENCE 349A, SPRING 2018
ASSIGNMENT #4 - 20 MARKS

DUE THURSDAY, MARCH 8 2018 (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.

Question #1 - 7 marks.
Let R denote any positive number. Newton's method is often used to determine an iterative formula for calculating frequently-used functions of R on computers and calculators; for example, to compute $\sqrt{R}$, $\sqrt[3]{R}$, or $\sqrt[m]{R}$ for any given value of m, or $1/R$ (using only addition and multiplication), and so on. The basic idea is to find an equation $f(x) = 0$ that has the value you want to compute (for example, $\sqrt{R}$) as one of its roots, and then apply Newton's method to compute a root of this equation. If you run Newton's method and it converges, then it will converge to the desired answer. For example, $x = \pm\sqrt{R}$ are the only zeros of the function $f(x) = x^2 - R$. If Newton's method is applied using this particular $f(x)$, the following iterative formula is obtained:

$$(*) \qquad x_{i+1} \;=\; x_i - \frac{f(x_i)}{f'(x_i)} \;=\; x_i - \frac{x_i^2 - R}{2x_i} \;=\; \left(x_i + \frac{R}{x_i}\right)/2.$$

If this sequence of values $\{x_i\}$ converges to a value $\hat{x}$, then $\hat{x}$ must be a zero of $f(x)$; that is, $\hat{x} = \pm\sqrt{R}$. Thus, the iterative formula (*) can be used to compute square roots using only addition and division.

(a) **(3 points)** Let $m \geq 2$ denote any positive integer. Apply Newton's method to

$$f(x) = x^m - R$$

in order to determine an iterative formula (similar to (*)) for computing $\sqrt[m]{R}$. Simplify your iterative formula so that it is in a form that looks exactly like the rightmost part of (*) when $m = 2$.

(b) **(3 points)** Write a MATLAB function M-file to compute $\sqrt[m]{R}$ based on the iterative formula in (a). This M-file can be obtained by modifying the `Newton` pseudocode given in the lecture 8 notes as follows:

change the function header to

```
function root = mth_root(m, R, x0, eps, imax)
```

and replace the line

$$root \leftarrow x_0 - f(x_0)/f'(x_0)$$

by an expression for computing *root* by the formula in (a) above. Save this function as `mth_root.m`. Note that `mth_root.m` does not need to call any functions `f` or `fp`. Your iterative formula from (a) contains the particular functions `f` and `fp` that are required to compute $\sqrt[m]{R}$. If you modify `Newton` as described here, then each successive computed approximation to $\sqrt[m]{R}$ will be output to your screen. Print a copy of your MATLAB function `mth_root.m` and hand it in.

(c) **(2 points)** Use `mth_root.m` with $m = 5$, $R = 65.25$, $x0 = 1$, $eps = 10^{-10}$ and $imax = 20$ to compute an approximation to $\sqrt[5]{65.25}$. Print and hand in the MATLAB output.

Question #2 - 6 Marks.

(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 $p(1.53)$ when $n = 5$ and

$$a = [-1,\ 3.3,\ 0,\ -2.2,\ 5,\ -1.6]$$
$$y = [-1,\ 1,\ -1,\ 1,\ -1]$$

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

Question #3 - 6 Marks

Let A =

$$\begin{pmatrix} 2 & -2 & 4 & -2 \\ 2 & 0 & -2 & 2 \\ 4 & 2 & 6 & -8 \\ 0 & -2 & 2 & -2 \end{pmatrix}$$

.

(a) **(4 points)** Use Gaussian elimination with partial pivoting (as given on page 3 of Handout #16) to compute the third column vector of A^{-1}. See also Handout #17. Do this by hand computation, no programming. (Do not compute all of A^{-1}.) Explicitly interchange rows as required, and show all of the derived linear systems and the back substitution. Do not do any scaling.

(b) **(2 points)** From the computations in (a), what is the determinant of A? (Show explicitly how you obtain this.)