Computational Methods Homework #1
CMSC / AMSC 460, Fall 2018 Due 11:59PM, September 14, 2018

Problem 1. (20) The purpose of this exercise is for you to get familiar with MATLAB and some of its capabilities. We have seen in class that, for a positive integer n and data $\{(x_j, y_j)\,|\,j = 0, \ldots n\}$, the matrix system

$$V_n a = y$$

produces the coefficients of the polynomial

$$p_n(x) = a_0 + a_1 x + \cdots + a_n x^n$$

that interpolates the data, where $[V_n]_{ij} = x_{j-1}^{i-1}$ for $1 \le i, j \le n+1$.
Let $x_j = j/n$, giving a uniformly spaced set of $n+1$ points in $[0,1]$. Write a MATLAB program that plots approximations to the functions x^r for $r = 0, 1, \ldots, n$ on the interval $[0,1]$ for $n = 1$, 5, 9, 13 and 17, using these evenly spaced points for input. For each n, the program should

- produce a single figure containing xy-plots of $\{(x_j, x_j^r)\,|\,j = 0, \ldots, n\}$ for each $r = 0, \ldots, n$,
- place the title "n=⟨value-of-n⟩" at the top of this figure
- plot x- and y-axes, in black,
- plot the data so that it is easy to read, by allowing some space to be visible around the ends of the axes, and
- format the figure in a square display.

The program should also run in an aesthetically pleasing way. For example, when it is producing the plots, you should have the program pause for a small amount of time (say, 1/2 second) so that you can see the results appear, and you should also have it pause between different values of n.
The results turned in should be your program, *properly commented*, two (of the five) figures produced, and the tabulated set of condition numbers.

Problem 2. (20) This problem concerns some aspects of accuracy of polynomial interpolation, which you will do by exploring two ways to build polynomials to interpolate the function $f(x) = \sin(4\pi x)$. The two strategies are: (1) to use part of the program from Problem 1 together with MATLAB tools to build interpolating polynomials based on the monomial basis, and (2) to use software from the text to build the polynomials in Newton form. The software, consisting of the two routines `divdif.m` and `evalnewt.m`, can be found on the web page

`https://archive.siam.org/books/cs07/.`

For polynomial degrees $n = 1, 5, 9, \ldots, 41$ (i.e., a collection of 10 integers differing by 4), do the following:

- Use the matrix built with code developed in Problem 1 to find the coefficients of the polynomial p_n of degree n in the monomial basis that interpolates f at $n+1$ uniformly distributed points in $[0,1]$.

- For each n, use the MATLAB function `polyval` to evaluate p_n at the point $x = .37$, and compute the absolute error $e_n^{mon} \equiv |p_n(.37) - f(.37)|$.
- Similarly, for each n, use the function `divdif` to find the coefficients of p_n in Newton form, use `eval_newt` to evaluate p_n at $x = .37$ and compute the absolute error e_n^{Newt} for this format.
- As you produce these results, save n, e_n^{mon} and e_n^{Newt} as well as the condition number cond(V_n) (use the MATLAB function `cond`), and afterwards print out the tabulated results. The output should be a table printed in an easy-to-read format using the MATLAB I/O function `fprintf`. Each line of the printed table should have the form

```
         n      xxx.yyye+zz       xxx.yyye+zz       xxx.yyye+zz
```

  showing cond(V_n), e_n^{mon} and e_n^{Newt} in the second, third and fourth columns, so that a reader can clearly see the condition numbers and errors.
- Can you explain the results?

Problem 3. (10) Let $x_0, x_1, \ldots, x_n$ distinct and for $j = 0, \ldots, n$, suppose that $y_j = f(x_j)$, the value of a function f at the point x_j. The Newton form of the polynomial of degree at most n that interpolates this data is

$$p_n(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots + c_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}).$$

In particular, c_n is the leading coefficient of p_n, i.e., the coefficient of x^n. We will now use the notation $c_n = f[x_0, x_1, \ldots, x_n]$.

a. Let $p_L(x)$ be the polynomial of degree at most $n-1$ such that $p_L(x_j) = y_j$ for $j = 0, 1, \ldots, n-1$, and let $p_R(x)$ be the polynomial of degree $n-1$ such that $p_R(x_j) = y_j$ for $j = 1, 2, \ldots, n$. Show that

$$p_n(x) = \frac{(x - x_0)p_R(x) - (x - x_n)p_L(x)}{x_n - x_0}.$$

b. Use this result to show that

$$f[x_0, x_1, \ldots, x_n] = \frac{f[x_1, \ldots, x_n] - f[x_0, \ldots, x_{n-1}]}{x_n - x_0}.$$

This is called the Newton divided difference formula.

Problem 4. (10) Let $f(x)$ be a function for which an estimate for a value x^* such that $f(x^*) = 0$ is sought. Suppose $n+1$ points in the plane are given, $(x_0, f(x_0)), (x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_n, f(x_n))$, where all $\{x_j\}$ are distinct and all $\{f(x_j)\}$ are distinct. Describe two different ways that polynomial interpolation can be used to estimate x^*. Which of the two approaches is easier to use?

What to hand in. You should hand in the code for Problems 1 and 2, organized and commented, together with output **neatly presented** for both problems, and the written solutions (done by hand or using `Word` or `latex`) for Problems 3 and 4.