NENG 685 Fall 2017
HW 1 Due Oct. 11, 2017

Name:

On homework:

- If you work with anyone else, document what you worked on together.
- Show your work.
- Always clearly label plots (axis labels, a title, and a legend if applicable).
- Homework should be done "by hand" (i.e. not with a numerical program such as MATLAB, Python, or Wolfram Alpha) unless otherwise specified. You may use a numerical program to check your work.
- If you use a numerical program to solve a problem, submit the associated code, input, and output (email submission is fine).

Do not write in the table to the right.

Run LaTeX again to produce the table

1. (20 points) Using four points on the interval $[x_0, x_3]$, do the following:
   (a) (4 points) Construct all of the Lagrange polynomials $L_j(x)$ that correspond to the points x_0, x_1, x_2, and x_3 by hand.
   (b) (4 points) Use these Lagrange polynomials to construct the interpolating polynomial, $P_3(x)$, that interpolates the function $f(x)$ at the points x_0, x_1, x_2, and x_3 by hand.
   (c) (8 points) Using the $P_3(x)$ you derived, create an interpolant for

$$f(x) = \sin(\frac{\pi}{2}x) + \frac{x^2}{4}$$

   over $[x_0, x_3]$ with $x_0 = 0, x_1 = 2, x_2 = 3$, and $x_3 = 4$. You may do this using something like Python or MATLAB, but *write your own functions rather than using the built in ones.*

   Plot the actual function and your interpolant using 100 equally spaced points for x between -0.5 and 4.5.

   (d) (4 points) Repeat what you did in part **??** but instead use $x_0 = 0, x_1 = 1, x_2 = 2.5$, and $x_3 = 4$.

   Discuss the differences in how well the function is interpolated using the different point sets.

Solution:

(a) Recall that

$$L_k(x) = \prod_{i=0, i\neq k}^{n} \frac{(x - x_i)}{(x_k - x_i)}$$

In our problem we have $n = 3$, giving the following Lagrange polynomials:

$$L_0(x) = \frac{(x - x_1)(x - x_2)(x - x_3)}{(x_0 - x_1)(x_0 - x_2)(x_0 - x_3)}$$
$$L_1(x) = \frac{(x - x_0)(x - x_2)(x - x_3)}{(x_1 - x_0)(x_1 - x_2)(x_1 - x_3)}$$
$$L_2(x) = \frac{(x - x_0)(x - x_1)(x - x_3)}{(x_2 - x_0)(x_2 - x_1)(x_2 - x_3)}$$
$$L_3(x) = \frac{(x - x_0)(x - x_1)(x - x_2)}{(x_3 - x_0)(x_3 - x_1)(x_3 - x_2)}$$

(b) To combine the polynomials and actually compute an approximation, we use

the formula:

$$P_3(x) = \sum_{k=0}^{3} f(x_k) L_k(x) \,,$$

giving:

$$\begin{aligned} P_3(x) =& \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} f(x_0) + \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} f(x_1) \\ &+ \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} f(x_2) + \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)} f(x_3) \end{aligned}$$

(c) Plugging in the function and all of our values gives

$$\begin{aligned} P_3(x) =& \frac{(x-2)(x-3)(x-4)}{(0-2)(0-3)(0-4)} \big(\sin(\frac{\pi}{2}0) + \frac{0^2}{4}\big) + \frac{(x-0)(x-3)(x-4)}{(2-0)(2-3)(2-4)} \big(\sin(\frac{\pi}{2}2) + \frac{2^2}{4}\big) \\ &+ \frac{(x-0)(x-2)(x-4)}{(3-0)(3-x_1)(3-4)} \big(\sin(\frac{\pi}{2}3) + \frac{3^2}{4}\big) + \frac{(x-0)(x-2)(x-3)}{(4-0)(4-x_1)(4-3)} \big(\sin(\frac{\pi}{2}4) + \frac{4^2}{4}\big) \end{aligned}$$

See the iPython solution for the plot.

(d) See the iPython solution for the plot.

We can see that with the first set of points the upward slope behavior is captured very well, but the interpolant is pretty innacurate around $x = 1$. This make sense as there is a large gap between $x = 0$ and $x = 2$ and it turns out the function changes quite a bit in that area.

Conversely, the second point set has more points in the area with the greatest function change and the overall function behavior is captured better. In this case we trade off accuracy in the right region.

2. (15 points) Using the interpolant $P_3(x)$ derived in question **??**:

   (a) (3 points) Write the general expression for the error term, $err(x) = |f(x) - P_3(x)|$.

   (b) (4 points) Given

   $$f(x) = \sin(\frac{\pi}{2}x) + \frac{x^2}{4} \,,$$

   use information about the function to bound the error expression.

   (c) (8 points) Use the values $x_0 = 0, x_1 = 2, x_2 = 3$, and $x_3 = 4$ to get the upper bound of $err(x)$ over this interval. That is, insert the points into the expression from part **??**, find an expression for x that maximizes error, and then find the x that gives the maximum. Present one final number. You may use a mathematical package to assist you in solving for x.

Solution:

(a) Recall the theorem that if $x_0, x_1, \ldots, x_n$ are $(n+1)$ distinct points $\in [a,b]$ and f is $C^{n+1} \in [a,b]$ then for each $x \in [a,b]$ $\exists\, \mu(x) \in [a,b]$ s.t.

$$|f(x) - P_n(x)| = |\frac{f^{n+1}(\xi)}{(n+1)!}(x-x_0)(x-x_1)\cdots(x-x_n)|$$

$$err = |\frac{f^{n+1}(\xi)}{(n+1)!}\prod_{i=0}^{n}(x-x_i)|$$

We have $n = 3$ and our $f(x)$ is 4 times differentiable, so we can use this theorem.

$$\boxed{err(x) = |\frac{f^{(4)}(\xi)}{4!}(x-x_0)(x-x_1)(x-x_2)(x-x_3)|}$$

(b) To bound the error expression, we need to find the 4th derivative of f as a function of ξ:

$$f'(\xi) = \frac{\pi}{2}\cos(\frac{\pi}{2}\xi) + \frac{x}{2}$$
$$f''(\xi) = -(\frac{\pi}{2})^2\sin(\frac{\pi}{2}\xi) + \frac{1}{2}$$
$$f'''(\xi) = -(\frac{\pi}{2})^3\cos(\frac{\pi}{2}\xi)$$
$$f^{(4)}(\xi) = (\frac{\pi}{2})^4\sin(\frac{\pi}{2}\xi)$$

We know the sin term is between -1 and 1, and thus the max value for the derivative term is $\frac{\pi^4}{16}$. We can use this to get a bounded expression for err:

$$\boxed{err(x) \leq |\frac{\pi^4}{16}\frac{1}{4!}(x-x_0)(x-x_1)(x-x_2)(x-x_3)|}$$

(c) Plugging in our evaluation values we get

$$err(x) \leq |\frac{\pi^4}{16}\frac{1}{4!}(x-0)(x-2)(x-3)(x-4)|$$
$$err(x) \leq |\frac{\pi^4}{16}\frac{1}{4!}(x^4 - 9x^3 + 26x^2 - 24x)|$$

To find the x that will cause this to be largest, we

$$\frac{d}{dx}(x^4 - 9x^3 + 26x^2 - 24x) = 0$$
$$4x^3 - 27x^2 + 52x - 24 = 0$$
$$x \approx 0.67365, 2.4691, 3.6073$$

Now we check which gives the maximum, and that gives us a bound:

$$\begin{array}{rr} 0.67365 & -6.9141 \\ 2.4691 & 0.94138 \\ 3.6073 & -1.3827 \end{array}$$

This means that

$$\boxed{err \leq \frac{\pi^4}{16}\frac{1}{4!} * 6.9141 \leq 1.7539}$$

3. (15 points) We have the following data:

$$x = [1, 2, 3, 4, 5, 6, 7]\,,$$

$$f(x) = [1, 4, 10, 12, 5, 4, 0]\,.$$

   (a) (10 points) Using *built in* Python or MATLAB functions, interpolate this data using

   - Piecewise linear interpolation
   - Lagrange polynomial interpolation
   - Spline interpolation

   Create a subplot for each of your interpolants over $[0.75, 7.25]$ using a fine mesh spacing, e.g. 0.05 (note that to use scipy's piecewise linear polynomial interpolation you will need to restrict the range to the exact endpoints, 1.0 and 7.0). Include the data points on the interpolation plots.

   (b) (5 points) Briefly discuss the differences between the resulting interpolations.

Solution: See the corresponding iPython notebook.

4. (30 points) The errors generated by a numerical method on a test problem with various grid resolutions have been recorded in the following table:

Grid Spacing (h)	Error (E)
5.00000e-02	1.036126e-01
2.50000e-02	3.333834e-02
1.25000e-02	1.375409e-02
6.25000e-03	4.177237e-03
3.12500e-03	1.103962e-03
1.56250e-03	2.824698e-04
7.81250e-04	7.185644e-05
3.90625e-04	1.813937e-05

For this numerical method, the error should be of the form

$$E = kh^p$$

(a) (3 points) Write this problem as a linear system $\mathbf{A}\vec{x} = \vec{b}$, where $x = \begin{pmatrix} \ln(k) \\ p \end{pmatrix}$ is the vector of unknowns.

(b) (5 points) Derive the normal equations for this over-determined system: write the matrices in $\mathbf{A}\vec{x} = \vec{b}$ form, where you include formulas / values for each entry.

(c) (9 points) Solve, using the program/language of your choice, the normal equations to obtain a least squares estimate to the parameters k and p.

(d) (6 points) Solve for the parameters k and p using SciPy's CurveFit function

(e) (7 points) Make a log-log and a lin-lin plot that displays both the input data and the function $E = kh^p$ using both your and scipy's k and p. Comment on the differences between the two approximations. (Checkout Python's matplotlib.pyplt.loglog command.)

Solution:

(a) Take the log of each side and write in matrix form:

$$\ln(E) = \ln(k) + p\ln(h)$$
$$\mathbf{A}\vec{x} = \vec{b} \qquad \text{where}$$
$$\mathbf{A} = \begin{pmatrix} 1 & \ln(h_1) \\ 1 & \ln(h_2) \\ \vdots & \vdots \\ 1 & \ln(h_8) \end{pmatrix} \qquad \vec{x} = \begin{pmatrix} \ln(k) \\ p \end{pmatrix} \qquad \vec{b} = \begin{pmatrix} \ln(E_1) \\ \ln(E_2) \\ \vdots \\ \ln(E_8) \end{pmatrix}.$$

(b) The normal equations are formed from $\mathbf{A}^T\mathbf{A}\vec{x} = \mathbf{A}^T\vec{b}$, giving

$$\begin{pmatrix} 8 & \sum_{i=0}^{8}\ln(h_i) \\ \sum_{i=0}^{8}\ln(h_i) & \sum_{i=0}^{8}[\ln(h_i)]^2 \end{pmatrix}\begin{pmatrix} \ln(k) \\ p \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^{8}\ln(E_i) \\ \sum_{i=0}^{8}\ln(E_i)\ln(h_i) \end{pmatrix}$$

(c) See the iPython notebook.

(d) See the iPython notebook.

(e) See the iPython notebook.

BONUS (5 points): submit your code by providing read/clone access to an online version control repository where your code is stored (e.g. github or bitbucket). If you don't know what that means and want to learn about it, come talk to me or check out resources here: `http://software-carpentry.org/lessons.html` For Windows, you want to setup the Git GUI. NOTE: You will not be able to do this from AFIT's systems.