Math 365 **Homework 4** **Due Oct. 19, 2016**
Grady Wright

1. (**Polynomial interpolation**, 10pts) Consider the following three samples of some function $f(x)$:

x	$f(x)$
-1	6
0	1
1	2

   (a) Determine the unique second degree polynomial interpolant $p_2(x)$ to the above data using Lagrange's interpolation formula. You do not need to simplify the result. You may either work this out by hand or simply write an anonymous function in MATLAB for the polynomial.

   (b) Repeat part (a) but use the Vandermonde matrix approach. In this case you will be solving for the coefficients a_0, a_1, and a_2 in the polynomial $p_2(x) = a_0 + a_1 x + a_2 x^2$ that interpolates the data. You can simply set up the linear system and solve the linear system in MATLAB . Be sure to print out the coefficients.

   (c) Show that (a) and (b) give the same result. You can do this by numerically comparing the polynomials at several points over the interval $[-1, 1]$.

2. (**Barycentric interpolation** 10pts) As discussed in class, Lagrange's interpolation formula can be rearranged into the magnificent *barycentric formula* (or more specifically the second (true) form of the barycentric formula):

$$p_n(x) = \frac{\sum_{j=0}^{n} \frac{w_j}{x - x_j} f_j}{\sum_{j=0}^{n} \frac{w_j}{x - x_j}}, \tag{1}$$

   where

$$w_j = \frac{1}{\prod_{\substack{i=0 \\ i \neq j}}^{n} (x_j - x_i)}, j = 0, 1, \ldots, n .$$

   Here x_j are the given nodes and f_j the corresponding function values. In this problem you will demonstrate the superiority of this method for high degree polynomial interpolation over the `polyfit` and `polyval` functions in MATLAB .

   (a) Consider the function $f(x) = |x - 0.2| + (1 - x^2)^3$. Generate samples of this function at the following points: $x_j = -\cos(j\pi/n)$, $j = 0, 1, \ldots, n$, where $n = 52$. This gives a total of 53 values $(x_j, f(x_j))$. Download that `baryinterp` MATLAB function from the course webpage, which implements the barycentric polynomial interpolation formula (1). Use this function to construct the 52 degree polynomial interpolant $p_{52}(x)$ to the data $(x_j, f(x_j))$, $j = 0, 1, \ldots, 52$, and evaluate it at the point $u_i = -\cos(\pi i/m)$, $i = 0, 1, \ldots, m$, where $m = 1000$. Produce a plot containing the exact function $f(x)$ and the polynomial interpolant evaluated at u_i.

   (b) Repeat part (a), but use the `polyfit` and `polyval` functions in MATLAB to generate the polynomial interpolant.

Mathematically speaking the results of part (a) and part (b) should be identical. However, because we are using floating point arithmetic they are not. The results from part (a) are much better because the barycentric formula for computing the interpolating polynomial is numerically stable, whereas the algorithm behind `polyfit` (which solves a linear system using Vandermonde matrices) is numerically ill-conditioned.

3. (**Approximation theory**, 20pts) A common practice in numerical computing is to replace complicated functions (ones that involve expensive function evaluations, for example) with simple ones that approximate the function to some specified tolerance (usually machine precision). The simplest functions, both in representation and efficient computational manipulations, are polynomials. Can we approximate a given function with a polynomial to a specified precision? This is one of the central questions of a fundamental area of mathematics called *approximation theory*.

   In Calculus you learned that a function $f(x)$ can be approximated as a Taylor series, assuming it has enough derivatives. The Taylor series of a function about a point c is

$$f(x) = f(c) + (x-c)f'(x) + \frac{(x-c)^2}{2!}f''(c) + \frac{(x-c)^3}{3!}f'''(c) + \cdots .$$

   Truncating this infinite series expansion after $n+1$ terms gives a polynomial approximant of degree n. One disadvantage of this approach is that it may be difficult (or impossible) to generate the needed derivatives of f. Another (as you will see below) is that the approximation may only be good in a limited neighborhood around $x = c$.

   An alternative approach to generating a polynomial approximants of f is to use polynomial interpolation at a suitably chosen set of points over the interval one wants the approximation to be valid.

   In this problem you will experiment with these approaches. In particular, for the polynomial interpolation approach you will experiment with the following three node sets, which go be the names equispaced, Chebyshev, and Legendre points, respectively:

   (i) $x_j = -1 + \dfrac{2j}{14},\ j = 0, 1, \ldots, 14$

   (ii) $x_j = -\cos\left(\dfrac{j\pi}{14}\right),\ j = 0, 1, \ldots, 14$

   (iii)

-0.987992518020485	-0.394151347077563	0.570972172608539
-0.937273392400706	-0.201194093997435	0.724417731360170
-0.848206583410427	0	0.848206583410427
-0.724417731360170	0.201194093997435	0.937273392400706
-0.570972172608539	0.394151347077563	0.987992518020485.

   There are 15 points in each set, so you will be generating polynomial interpolants of degree 14. Note that all of these interpolants must be computed using the `baryinterp` function from the course webpage.

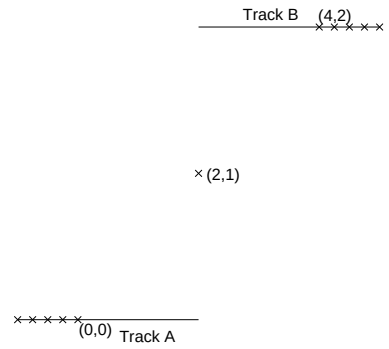
   (a) Consider the function $f(x) = e^{2x}$. Generate the Taylor series polynomial approximation to f of degree 14 about $c = 0$. Evaluate the approximation at 201 equally spaced points between $[-1, 1]$ (generated using `linspace(-1,1,201)`) and compute the error in the approximation at these points. Plot the error vs. these 201 equally spaced points. Report the 1-norm, 2-norm, and ∞-norm of the error in a nice table. (Hint: You may want to use the MATLAB function `polyval` to evaluate the Taylor polynomial once you determine the coefficients.)

   (b) Repeat part (a), but using polynomial interpolation at each of the three point sets (i), (ii), and (iii) given above. You should produce three different plots and three tables with results for the 1-norm, 2-norm, and ∞-norm associated with each point set.

   (c) Which technique produced the best results from (a) and (b)? Explain the criteria you used to determine what "best" means.

(d) **Extra credit** (5 points) Chebfun is a software system for computing with *functions* in MATLAB (see `http://www.chebfun.org`). The key technology employed by Chebfun is interpolation at Chebyshev points. Download Chebfun and install it in MATLAB . Use it to construct a function approximation to $f(x) = e^{2x}$. From this construction determine the degree of the polynomial that can be used to approximate f to machine precision. Use the Chebfun function `remez` to generate the "best" approximation to f of degree 14. Plot the error in this approximation together with the error in the 14th degree polynomial interpolant of f at the Chebyshev points from part (b). The best approximation in this case is the one that minimizes the maximum absolute difference between the function and the polynomial. These are optimal and are used in many libraries for computing special functions.

4. (**Piecewise cubic Hermite interpolation**, 15pts) Union Pacific has decided to reopen the Boise Train Depot for rail travel. They have contracted your computational science consulting company to design a new switching path between two of the rail lines (Track A and B) that enter the Depot. The requirements for the path are that it pass through the points $(0, 0)$, $(2, 1)$, and $(4, 2)$ (see Figure below). Furthermore, the path should be tangent to the line $y = 0$ at $(0, 0)$, tangent to the line $y = 2$ at $(4, 2)$, *and* have a slope of $19/22$ at $x = 2$.



(a) Since you are given both the function and derivative values at each of the node points, you decide to use a piecewise cubic Hermite polynomial to model the switching path. Analytically compute the piecewise polynomial for the intervals $[-1, 0]$,$[0, 2]$,$[2, 4]$,$[4, 5]$ (see Section 3.3 of the NCM book).

(b) Use the `mkpp` function in MATLAB to make a piecewise polynomial representation for the train track.

(c) Use `ppval` with the piecewise polynomial function you created from (b) to make a nice smooth plot the solution from part (a) for Union Pacific. If you did everything correctly, the value of the piecewise polynomial at $x = 2.5$ should be $985/704$.

5. (**Closed curve interpolation**, 20pts)

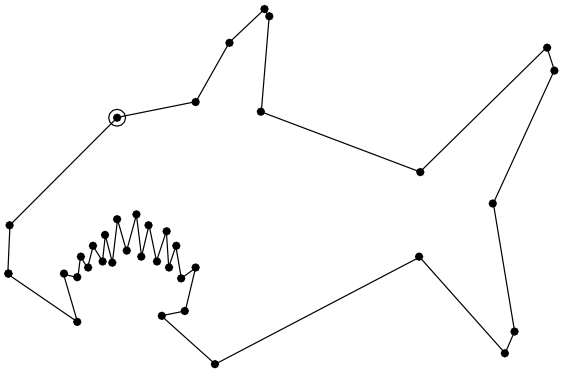
(a) Download the `shark.mat` file from the course web page to your working MATLAB directory. Execute the following commands:

```
load shark;
plot(x,y,'k.-','MarkerSize',10);
```

This should load into MATLAB two vectors `x` and `y` that represent the coordinates of the cartoon shark shown below.

(b) For the shark data in part (a), generate a smooth parametric curve for drawing the shark using the `pchip` and `spline`. Make a plot of the shark for each of these functions.

(c) One problem with the plots from part (b) is that the parametric curves have awkward kinks where the last point of the shark joins the first point. Come up with a way to fix this kink by altering the input data to `pchip` and `spline`. Make plots of the shark using the altered input data.



(d) Which representation of the shark do you like best from (b) and (c). Explain your choice.

(e) Compute the tangent vector to the shark using both `pchip` and `spline` at the point marked with a dotted circle $\odot$ in the the above figure. Produce parametric plots of the shark with these tangent vectors included. You should produce one plot for the `pchip` curve with the tangent vector, and one plot for the `spline` curve with the tangent vector.

6. (**Scattered data interpolation**, 15pts) In this problem you will use radial basis function (RBF) interpolation with thin plate splines to reconstruct the topograhy of a region from scattered topographic samples. Download the MATLAB `m` files `rbffit` and `rbfval`, and the data file `topodata.mat` from the course web page and save them to your working directory.

   The data file contains 1000 topographic samples of a two-ravine drainage from Upper State New York. To view the raw data issue the following commands:

```
load topodata;
scatter3(x,y,z,40,z,'.');
colorbar
```

   Assuming that x and y are the independent variables and z (the elevation) is the dependent variable, use the `rbffit` function to construct an interpolant to the data. Next, use the MATLAB function `meshgrid` to generate a rectangular grid of points for evaluating the interpolant. In the x-direction the grid should contain 100 equally spaced points in the interval $[0, 261]$, and in the y-direction the grid should contain 153 equally spaced points in the interval $[0, 399]$. The function `rbfval` should be used to do the evaluation. Produce a surface plot of the interpolant from these grided data values using the `surf` command. Include the original scattered data values in the plot and make sure you include labels on the coordinate axes and a title. Once the surface plot has been generated, experiment with the functions `shading`, `colormap`, `lighting`, and `camlight` to change the appearance of your plot. The following sequence of those commands should generate a very sharp looking image

```
shading interp;
colormap(autumn);
lighting phong;
camlight right;
```

   Finally, generate a contour plot of the surface using the command `contour`. Make your plot contain 14 contours from 0 to 260 in increments of 20. Include the original (x, y) sample locations on your plot, and labels.