Homework Project #5

Math 365, Fall 2017

You will be graded on correctness of your code, coding style, and plots. Be sure to

- Add titles, axis labels and a legend to your plots (unless otherwise stated).
- Use the auto-indent features in MATLAB to indent your code to make it more readable.
- Include comments in your code to indicate different parts of the code, i.e. "Problem definition", "Numerical method", "Plotting".

1. (**Barycentric Formula**) The Lagrange Formula is a convenient way to write down an explicit formula for a polynomial interpolant. The fastest and most stable way to evaluate a polynomial using this form, however, is to use the *Barycentric Formula*.

   In this problem, you are asked to approximate the function

   $$f(x) = |x| + \frac{x}{2} - x^2, \qquad x = x(t) = -\cos(t), \qquad t \in [0, \pi]$$

   with a degree $d = 54$ polynomial.

   (a) Plot $f(x)$ over the interval $[-1, 1]$. Use enough resolution to see the smooth behavior of the function.

   (b) Sample $f(x)$ at $N = d + 1$ Chebyshev nodes to get sample points (x_j, y_j), where

   $$y_j = f(x_j), \qquad x_j = -\cos(t_j), \qquad j = 1, 2, \ldots N$$

   where t_j is a set of N equally spaced points in interval $[0, \pi]$.

   (c) Write a function `baryweights` that computes the barycentric weights

   $$\omega_j = \frac{1}{\Pi_{k=0, k\neq j}^{n}(x_j - x_k)}, \qquad j = 1, 2, \ldots, N.$$

   Your function should take the x-coordinates x_j and return ω_j.

   (d) Write a function `baryval` that evaluates the Barycentric Formula, given by

   $$P(x) = \frac{\sum_{j=1}^{N} \frac{w_j}{x - x_j} y_j}{\sum_{j=1}^{N} \frac{w_j}{x - x_j}}$$

   Your function `baryval` should take a vector $\mathbf{x}$ and returns a vector $\mathbf{y} = P(\mathbf{x})$.

   (e) Plot the approximating polynomial $P(x)$.

   (f) Add axis labels and a legend to your plot. Show clearly that your interpolant $P(x)$ agrees with the exact solution in the "eye-ball" norm. (i.e. your solution looks correct!).

   Include the functions `baryweights` and `baryval`. in your main homework script. You will use these for other problems in this assignment.

2. **Runge-phenomena.** A problem with polynomial interpolation is that near the endpoints of the domain, the polynomial interpolant can take on extreme values. A classic example of this phenomena, called the Runge Phenomena, can be seen when trying to approximate the function

   $$R(x) = \frac{1}{1 + 25x^2}$$

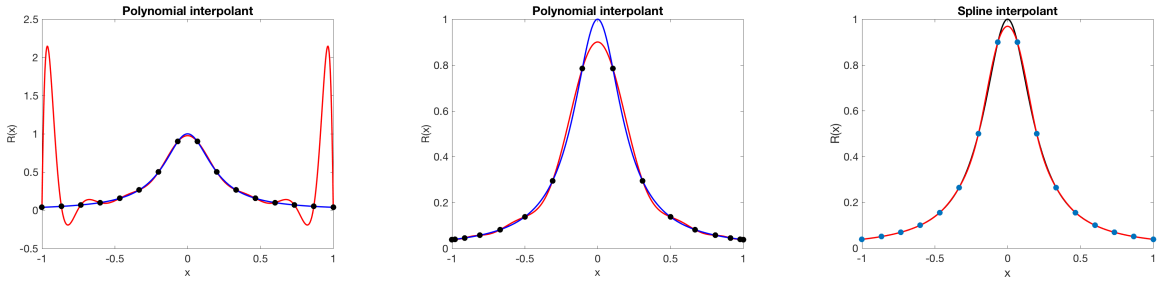

Figure 1: (1) High degree polynomial approximation at equally spaced points; (2) High degree polynomial at Chebyshev nodes; (3) Spline interpolant at equally spaced points.

over the interval $[-1, 1]$.

While equally spaced points seem like a reasonable choice of sampling points, the *optimal* set of points (in mathematical sense) are the Chebyshev nodes.

In this problem, you will create two polynomial approximations to the function $R(x)$. The first approximation will be created using equally spaced nodes, and the second is creating using Chebyshev nodes.

(a) Sample $R(x)$ at $N = 16$ equally spaced points in the interval $[-1, 1]$. Use your function `baryweights` to compute the barycentric weights for the points (x_j, y_j) you obtain. Use `baryval` to evaluate the approximating polynomial at enough points to plot a smooth curve. Plot the polynomial approximation.

(b) Repeat the above steps using sampled data at Chebyshev nodes. Use the Barycentric Formula to plot the resulting polynomial approximation in a second plot.

(c) Add a title, axis labels and legend to each of your two plots.

With these two plots, you should see that the Chebyshev nodes produce a much better interpolating polynomial.

3. (**Derivative approximation.**) Computing the derivative of a polynomial is easy if we have the polynomial in the form

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

One disadvantage of the Barycentric Formula (and of the Lagrange Interpolation Formula) is that it does not give us the approximating polynomial in this form.

With the following strategy, however, can still approximate the derivative of a function using the Barycentric Formula.

- Sample $f(x)$ points x_j, $j = 1, \ldots N$ to obtain $y_j = f(x_j)$.
- Use `polyfit` to obtain coefficients $a_0, a_1, \ldots, a_n$ of the approximating polynomial $P(x)$.
- Use the function `polyder` to get a new set of coefficients $\mathbf{b}$ representing the coefficients of $P'(x)$ of the polynomial approximation. The coefficients b_j are related to the a_j by the relation

$$b_0 = a_1, \quad b_1 = 2a_2, \quad b_2 = 3a_3, \ldots, b_{n-1} = na_n.$$

- Use `polyval` and coefficients $\mathbf{b}$ to sample the derivative $P'(x)$ at the nodes x_j to get $w_j = P'(x_j)$.
- Use the Barycentric Formula to evaluate the approximation $P'(x)$.

For this problem, you will use the above strategy to approximate the derivative of the function

$$h(x) = \sin(2\pi \cos(x))$$

(a) Sample the function at $N = 16$ Chebyshev nodes x_j in the interval $[-1, 1]$.

(b) Use the strategy above to obtain sampled points (x_j, w_j).

(c) Plot the approximate derivative $P'(x)$. On the same graph, plot the exact derivative $f'(x)$ (which you can compute analytically, or by some other means). The two curves should agree very closely in the eyeball norm.

(d) Add a title, axis labels and a legend to your plot.

This idea of using polynomial approximations to compute the derivative of a function is used throughout scientific computing. For example, the formula used to approximate the second derivative in the temperature problem from an earlier homework is derived by twice differentiating a second degree polynomial interpolating three equally spaced sample points.

4. (**Spline approximation.**) While the Chebyshev nodes are an optimal choice of nodes for polynomial approximation, often we do not have a choice of nodes and may have to work with what we are given. To obtain an approximate that avoids the Runge Phenomena, we can use the spline interpolant.

   For this problem, you will create a spline interpolant to the Runge function in previous problems

$$R(x) = \frac{1}{1 + 25x^2}$$

   For this problem, create a spline interplant at $N = 16$ equally spaced points on the interval $[-1, 1]$. Add axes labels and a title to your plot.

   See Figure 1 for plots of all three Runge function approximations and interpolants.

5. (**Don Quixote**) For this problem, you will use the Matlab routine `spline` to create a spline interpolant for the two data sets `dq1.dat` and `dq2.dat` from the course website.

   Load the data files and extract the two sets of (x_j, y_j) data points from each of the two data sets. Plot a spline through each of the two data sets. Use enough resolution so that you have a nice curve through the data points. Don't just connect the dots!

   Here are some hints you might find useful.

   - Use

     ```
     set(gca,'ydir','reverse');
     ```

     after your plot command so that your image is oriented correctly.

   - Turn off the axes. For this problem, you do not need to show axes, titles or labels.

     ```
     axis off
     ```

     to suppress the plot axis.

   - You should only have one graph for this problem.

   For this problem, you will not be able to simply construct a spline interpolant for x as a function of y. Instead, create two interpolants for functions $x(t)$ and $y(t)$, where t is the independent variable.

   The data for the curves above were digitized from Pablo Picasso's famous "one-line drawings". These drawings were made with one or two continuous lines. For more of these drawings, do a Google search on "Picasso one-line drawings".

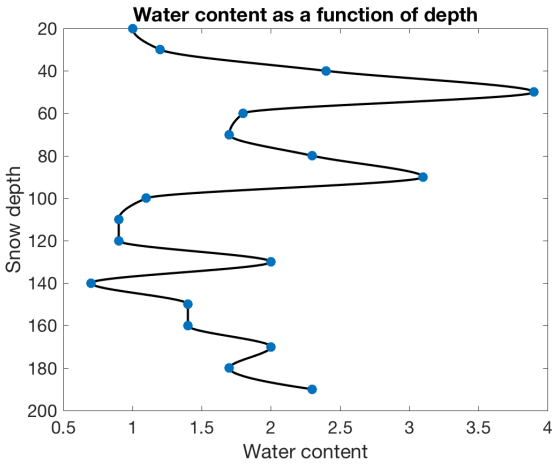

Figure 2: Plot of snow data, used in Problem 6.

6. (**Interpolating snow data**) The MATLAB function `pchip` is a piecewise polynomial that has the feature that its values do not produce values larger than those that appear in the data. This is particularly useful for data that should physically remain within, for example, $[0, 1]$ (for fractions) or $[0, 100]$ (for percentages).

   In this problem, you will use `pchip` to interpolate measurements of fractional water content measured from various depths in a snow pack. The data file, `snow.dat`, is available on the course website. The first column of this file is a measurement of snow depth (cm), and second column is the percentage of liquid water content p, $0 < p < 100$ found at that depth. Plot your interpolating polynomial and include the following :

   - Plot the water content along the x-axis and depth along the y-axis. Your y-axis should have zero depth at the top of the plot and 200 cm at the bottom. **Hint:** Use `set(gca,'ydir','reverse')`.
   - Include your data points in your plot (use `markersize`).
   - Use your interpolation routine to approximate the snow water content at a depth of 106.3 centimeters. Write your results to the file `watercontent.out` (using `write_file`).
   - Add a title and axis labels.

   See Figure 2 for an example of what your final plot should look like.

   (*Thanks to Santiago Rodriguez, Department of Geophysics (BSU) and former Math 365 student, for the snow data.*)