Homework #5

Math 465/565

A few notes about this homework.

- You will be graded on your presentation as well as the content of the work. This means that all of your plots should have axis labels, a title and legends, and any other annotation that makes it clear what you are demonstrating.
- You may submit your homework to BlackBoard. Please submit a PDF of your homework.
- Students in 465 may do the problems for 565 for extra credit.

1. **Lagrange polynomials and the Barycentric formula**. For this problem, assume that you have data points (x_k, y_k), $k = 0, 1, ..., N$.

   (a) Write out the interpolating polynomial $p_2(x)$ for $N = 2$ using the Lagrange Interpolation Formula. You only need to write out each Lagrange polynomial involved in the formula, but do not simplify anything.

   (b) How many operations are required to evaluate $p_2(x)$, using the Lagrange Interpolation Formula?

   (c) The Second Barycentric Form (described in the lecture notes) is given by

   $$p_N(x) = \frac{\sum_{j=0}^{N} \frac{\omega_j}{x-x_j} y_j}{\sum_{j=0}^{N} \frac{\omega_j}{x-x_j}}$$

   where weights are computed as

   $$\omega_j = \frac{1}{\prod_{k=0,k\neq j}^{N}(x_j - x_k)}, j = 0, 1, \ldots N$$

   For $N = 2$, show that the the Barycentric Form produces exactly the same formula as the Lagrange interpolation formula.

   (d) How many operations are required to evaluate the Barycentric Form for $N = 2$? Do not count the computation of the weights.

   (e) Compute the number of operations required for both the Lagrange Interpolation Formula and the Barycentric Form for a general N. Which approach is more efficient?

   (f) Approximate the function $f(x)$ below with a 10^{th} polynomial $p_{10}(x)$.

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

   over the interval $[-1, 1]$ at points $(x_k, f(x_k))$ for $x_k = -1 + kh$, where $h = 2/N = 0.2$. Turn in a plot of your solution.

   - **Math 465.** You may use the Lagrange Interpolation formula.
   - **Math 565.** Implement the Barycentric Form of the Lagrange Interpolation Formula. Turn in your code for this problem.

   (g) Repeat Problem 1f, but this time, use x-values

   $$x_k = -\cos\left(\frac{k\pi}{10}\right), \qquad k = 0, 1, \ldots N$$

   Turn in plot using these x_k (known as "Chebyshev nodes") and compare the results of this problem to the interpolating polynomial you got in Problem 1f. What differences do you see?

2. **Splines.** Piecewise cubic Hermite interpolating polynomials are a good way to avoid the oscillations that can appear with high degree polynomials. For cubic Hermite polynomials, we need to specify both values y_k and derivatives d_k at data points x_k. While we will have the values y_k (from measurements, for example), we won't typically have derivatives. The main goal of this problem is to test different ways to obtain derivatives.

   For this problem, you will be using a spline to approximate the function

$$f(x) = e^{-15(x-0.5)^2} \sin(8\pi x)$$

   (a) **Math 465.** Use the explicit formula to compute "PCHIP" derivatives d_k. See pages 16-17 of the lecture notes on piecewise polynomials.

   (b) **Math 565.** Use the classic spline conditions to compute derivatives at internal nodes. See pages 18-25 of the lecture notes on piecewise polynomials.

      i. Implement both the clamped and natural endpoint conditions to compute values for d_0 and d_N.

      ii. Why does the Newton iteration, described on slide 24, converge in one step?

   Much of the work of the above is done for you in the Matlab codes `math465_build_spline.m` and `math565_build_spline.m`. Python versions may also be made available. The test program `spline_test.m` runs both codes and does all the plotting for you.

   For this problem,

   (a) Complete the missing details (indicated with a `TODO`) in the above code for 465/565.

   (b) Turn in your `build_spline` code, along with your plot of your interpolating spline for $f(x)$ aboove.

   (c) **Math 465.** Use your interpolating polynomial to interpolate the data in `snow.dat`, available on Blackboard. 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. The main part of your grade on this problem will be based on the presentation quality of your plot, so be sure to add titles, axis labels, and a legend. Plot your data points as well as the curve, and be sure the data points are clearly visible.

      - Why are the PCHIP derivatives advantageous for this data set?

   (d) **Math 565.** Load the data `XY_dots.dat` and create a picture by interpolating the data using your spline code. Use the "natural" spline conditions as endpoint conditions. When you create your plot, be sure to include enough points (~ 2000 ?) so that you create a nice smooth curve.

3. (**Trapezoidal rule.**) The arc-length along an ellipse with major axis A and minor axis B, $B \le A$ can be computed by evaluating the integral

$$s(t) = A \int_0^t \sqrt{1 - k^2 \sin(\theta)^2} d\theta$$

   where $k^2 = 1 - (B/A)^2$ and $t \in [0, 2\pi]$. For the following problems, use values $A = 1$ and $B = 0.5$.

   (a) Evaluate the integral above for $t = 1$ using the trapezoidal rule. The exact solution for this integral is $0.886\,625\,123\,536\,706\,948\,2$. Create a table of errors for $N = 8, 16, 32, 64, 128, 256$ and show that the error converges at rate $\mathcal{O}(h^2)$, where $h = 1/N$. **Hint.** Create a loglog plot of the errors vs. N. Determine the slope of the line.

   (b) Repeat Problem 3a using Simpson's rule. Show that the errors converge at a rate $\mathcal{O}(h^4)$.

(c) The trapezoidal method can be corrected to obtain a fourth order method using the following approach.

$$T_c(f) = T(f) - \frac{h^2}{12}(f'(b) - f'(a))$$

where $T(f)$ is the usual composite trapezoidal rule on the interval $[a, b]$. Use this corrected formula to compute $s(1)$. Plot a loglog plot of the errors and show that the errors converge as $\mathcal{O}(h^4)$.

(d) **Math 565.** The trapezoidal rule exhibits a remarkable *exponential convergence* when used to compute the integral over a periodic domain. To see this, compute the circumference of the ellipse above as

$$C = 4A \int_0^{\pi/2} \sqrt{1 - k^2 \sin(\theta)^2} d\theta \tag{1}$$

The solution (to 20 digits) is given by $4.844\,224\,110\,273\,838\,099\,21$. Evaluate the integral using the trapezoidal rule for $N = 4, 5, 6, \ldots, 20$.

i. Create a log-linear (`semilogy`) plot of the error in your circumference calculation vs. N. You should observe that the errors lie on a straight line with slope, until at about $N \geq 14$, when the error reaches machine precision and no longer decreases.

ii. The convergence rate of the error sequence you generated above is $\mathcal{O}(e^{-cN})$. Plot a best-fit line through your data. **Hint.** Assume the error is approximately $E_N \sim \beta e^{-cN}$. Find parameters c and β. Add the plot of βe^{-cN} to your error plot. What is the slope of this line?

4. **Extra credit.** The integral in (1) is an example of a *complete elliptic integral of the second kind.* The elliptic integral functions are examples of special functions that can arise as solutions to differential equations. You are already familiar with two special functions, e^x and $\sin(x)$ that arise as solutions to $y'(x) = y$ and $y''(x) = -y$, respectively. Like $\sin(x)$ and e^x, many special functions are available in Matlab and scientific packages for Python.

Another special function that shows up as the solution to a differential equation is the *first kind Bessel function* of order n, given by $\mathcal{J}_n(x)$. Like the elliptic integral, Bessel functions can be evaluated by computing an integral. The first kind Bessel function $\mathcal{J}_n(x)$ can be computed as

$$\mathcal{J}_n(x) = \frac{1}{\pi} \int_0^{\pi} \cos(nt - x \sin t)\, dt$$

In Matlab, this function can be evaluated using `besselj(n,x)`.

- **Math 465.** Evaluate the $\mathcal{J}_1(5.4)$ using the Trapezoidal rule. Find the smallest possible N so that your error (when compared to either the Matlab function, or an equivalent function in Python) is approximately 10^{-16}.

- **Math 565.** Evaluate $J_1(x)$ at 20 equispaced points $x_i = ih$, $h = 20/N$ in the interval $[0, 20]$. Use the trapezoidal rule and choose the smallest possible N for which all 20 values you compute have an error $\approx 10^{-16}$. Plot the Bessel function $\mathcal{J}_1(x)$ over a range of $x \in [0, 20]$ using the Matlab function `besselj(n,x)` (or equivalent Python function). Use enough points to get a smooth curve. On the same plot, plot the 20 values you found and show clearly that they lie on the same curve. Turn in your plot (with axis labels, title, and legend) along with a table the values and errors in your approximation to $\mathcal{J}_1(x_i)$.

  Can you speculate what the convergence behavior of the Trapezoidal rule is for this integral? Provide a convincing argument.