CS 322: Problem Set 2

Due: Wednesday, July 3, 2002 (In Lecture)

The policies for this homework assignment are as follows:

- You must work individually on Question 1. You may work with at most one other person on Questions 2 and 3. Consult the course website for the Academic Integrity rules.
- Problem sets will be weighted equally in determining your final grade. The number of questions or total number of points on a given problem set is irrelevant.
- Writing will be graded on content, as well as on grammar, spelling, punctuation, etc.
- Mathematical exercises will be graded on the overall set-up of the problem as well as correctness.
- Points will be deducted on the Matlab questions for poorly commented code and inefficient code/redundant computations.
- For the Matlab questions, you should hand-in the appropriate plot(s) and the script/function(s) necessary to generate the plot(s).
- Submit your assignment in two different parts (Part 1 for Question 1 and Part 2 for Questions 2 and 3). Every student should submit Part 1. At the top of the first page for Part 1, write your name, the course number, the problem set number, Part 1, your e-mail address, your student ID number, and the date. Each team of up to two students should submit Part 2 jointly. The information at the top of Part 2 should be the same, except there should be up to two names at the top of the paper, and Part 1 should be replaced with Part 2.

1. (10 points) **Determining the Best Interpolant:** We have seen in lecture that there are many factors which determine whether or not a particular interpolant is a good approximation to the function being approximated. In this exercise, you will compute various interpolants and analyze their abilities to approximate a specific function. Some new types of interpolants are included here to give you a better idea of a wider range of interpolants. First, download the specified Matlab functions from the Homework section of the course website that perform the following types of interpolation:

   - Taylor polynomial (** - use Taylor.m in Matlab)
   - Vandermonde polynomial (InterpV.m)
   - Newton polynomial (InterpN.m)
   - Trigonometric interpolation (*** - CSInterp.m)
   - Piecewise Cubic Hermite (pwC.m)
   - Cubic spline - natural (CubicSpline.m)
   - Cubic spline - complete (CubicSpline.m)
   - Cubic spline - not-a-knot (use spline.m in Spline Toolbox)

   Note that for Taylor polynomials and not-a-knot cubic splines, you will use built-in Matlab functions identified above.

   Write a script that asks the user for the number of data points, creates equally spaced data on the interval, performs the necessary interpolation, and plots the function, interpolant, and data points in one plot for each case.

   Test the abilities of the interpolants to approximate the function given by

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

on $[-1, 1]$ with 8 equally-spaced data points. In the case of Taylor approximation, you should compute the Taylor polynomial of degree 3 centered at 0 instead. Make your code general enough to ask the user which degree of Taylor polynomial they want and where to center the polynomial. In the case of the trigonometric interpolation, use a period of 0.1.

In each case, plot the function and the interpolant on the same graph. Use a black line for the function and a red dashed line for the interpolant. In addition, plot the actual data points with a '*' in the same plot. Remember to use hold on for plotting 2 or more functions in the same plot. Put appropriate labels and titles on your plot. Use the legend command to distinguish between the interpolant, the actual function, and the data points. Use the subplot command to arrange the 8 plots in groups of 4 (2 rows and 2 columns in each figure for a total of 2 figures).

Next, describe the abilities of each interpolant to approximate the function and the advantages and disadvantages of using each method. You should discuss accuracy, efficiency, and any other relevant features. Your write-up should include the necessary details but should also be concise and well-organized. Here is the recommended order of topics for your write-up:

- Accuracy of the non-piecewise interpolant methods.
- Accuracy of the piecewise interpolant methods.
- Advantages and disadvantages (including efficiency) of non-piecewise interpolants.
- Advantages and disadvantages (including efficiency) of piecewise interpolants.

(**) Taylor approximations: Recall from calculus that we can derive a Taylor Series approximation for a function f if it can be differentiated the appropriate number of times. These approximations are of high quality at or near the point used to compute the Taylor Series. For more information on Taylor approximations, refer to any calculus book.

(***) Trigonometric Interpolation: This is simply interpolation using several sin and cos functions as the basis functions. See 2.4.4 in the book for more details.

2. (10 points) **Parametric Curves and Interpolation:** None of the interpolation techniques we have learned about can be used to generate cuves which are not themselves functions. For example, they cannot be used to construct the letter 'S'. A technique to deal with this problem is to determine a polynomial or piecewise polynomial that connects the points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$ in the specified order on an interval $[t_0, t_n]$ and constructs approximation functions with $x_i = x(t_i)$ and $y_i = y(t_i)$ for each $i = 0, 1, \ldots, n$. In order to generate the plot of interest, the curves x_i and y_i are plotted for $i = 0, 1, \ldots, n$.

   Computer graphics is an application area of scientific computing that requires the use of parametric curves. Problems in this field require that smooth be generated and modified rather easily and quickly. For this reason, computer scientists often choose to use a form of the piecewise cubic Hermite polynomial for the curves. The endpoints and the derivatives at the endpoints completely specify each portion of the cubic Hermite polynomial. This is ideal because it allows one portion of the curve to be modified while leaving the rest of it intact.

   Read pages $158 - 163$ in the attached handout to learn how parametric curves (and Bézier curves) are constructed. Invest the most effort in understanding pages $160 - 161$.

   After having read the handout, you should implement the algorithm for computing a Bézier curve which is given on pages $163 - 164$. Implement the algorithm in a function called **bezier.m.** Note that in the pseudocode given, the indexing starts at 0. Because Matlab does indexing starting at 1, you will need to keep this mind when implementing their algorithm. This function should be fully vectorized.

   Finally, test your Bézier curve code by writing a script, **question2.m**, that uses the data in the table for Problem 4 of Exercise Set 3.5 on page 164 of the handout, to approximate the letter η. Plot the resulting parametric curve. In order to do this, use **linspace** to create a vector of t values on $[0, 1]$. Use Horner's Method to compute $x_i(t)$ and $y_i(t)$. Then plot $(x_i(t), y_i(t))$. Do this for all appropriate values of i. Make your code as efficient as possible.

3. (10 points) **The Mathematics of Interpolation:**

(a) Consider the following interpolation problem. Given two data points (x_1, y_1) and (x_2, y_2) such that $x_1 < x_2$ and a scalar a, find a quadratic function $p(x)$ that interpolates the two data points and satsfies the following additional condition:

$$\int_{x_1}^{x_2} p(x)dx = a.$$

Write down the linear equations that must hold for the coefficients of p to satisfy these conditions. Finally, solve for the coefficients.

(b) Consider a sequence of data points $(x_1, y_1), \ldots, (x_n, y_n)$ such that $x_1 < x_2 < \ldots < x_n$ and $y_1 < y_2 < \ldots < y_n$. These data points appear to be describing an increasing function. Nonetheless, show by means of an explicit example that the polynomial interpolant for this data (a polynomial of degree $n-1$ or less) is not necessarily an increasing function over the whole interval $[x_1, x_n]$.

(c) Let f, g be two cubic spline functions defined over the interval $[0, 1]$. Suppose the breakpoints of f and g are different. In particular, suppose the breakpoints (knots) of f are at $0, .1, .2, \ldots, .9, 1$ whereas the breakpoints of g are at $0, 1/3, 2/3, 1$. Is it true that $f+g$ is necessarily a cubic spline function? Explain why or why not.