CS 222 - Homework 2
Due in lecture Wednesday, July 18, 2001

The policies for this (and other problem sets) are as follows:

- The policies for turning in late HW assignments are outlined on the course information sheet passed out in class and on the class website.
- Problem sets may be done individually or in teams of two. Put your name or names on the front page. Re-read the academic integrity statement on the web (cited on the course information sheet).
- When MATLAB code is part of an assigned problem, you must turn in your code and all output required to receive credit. Points will be deducted for poorly commented code, redundant calculations, and inefficient code. All code should be vectorized as much as possible.

1. (10 points) You saw in lecture that Monte Carlo estimation could be used to estimate the value of π (see also section 1.3 - page 36). Using the same idea, we can also use Monte Carlo estimation to approximate the value of integrals that cannot be evaluated analytically.

   In particular, let $f(x, y) = e^{x^2 y^2}$. In this problem, you will devise a way to approximate the value of

   $$\int_0^1 \int_0^{\sqrt{1-x^2}} f(x, y) dy dx$$

   using *Monte Carlo* estimation. You are to write a MATLAB function `DoubleIntEst` that implements your method. `DoubleIntEst` should take as a parameter n (the number of trials) and should output the approximate value of the integral. Use the command `rand('seed',.12345)` at the top of `DoubleIntEst`.

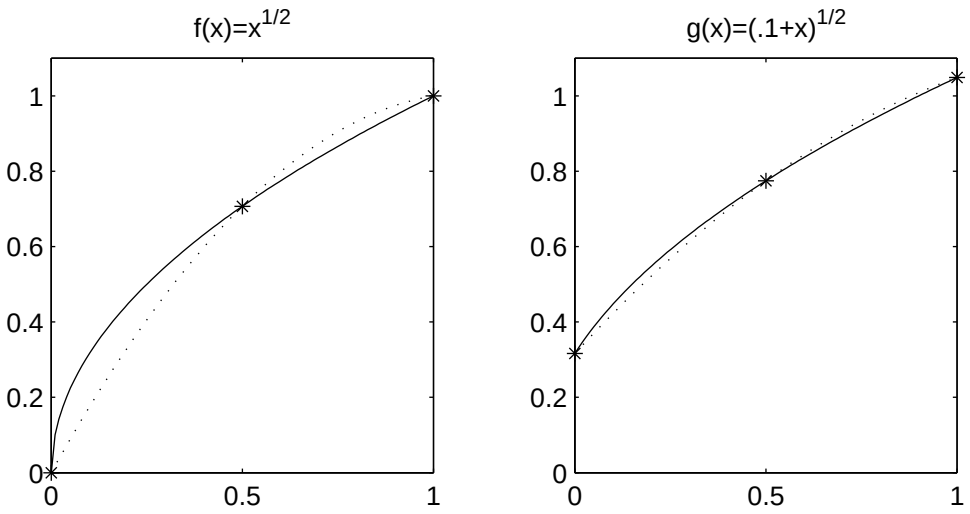
   Download `hw2p1test.m` from the course web site. Run this script and turn in a printout of its output. In addition, turn in a printout of your function `DoubleIntEst`. Your code should be fully vectorized and there should be *no* loops. It should also be flop efficient. Since there are multiple ways of approaching this problem, it is especially important that you comment your code well. Make sure all variable names are specified in the comments of your code (similar to the blocks of comments given in HW1).

2. (5 points) Consider the following functions

   $$f(x) = \sqrt{x} \qquad g(x) = \sqrt{0.1 + x}$$

   The functions $f(x)$ and $g(x)$ were both interpolated with equally spaced points using a second-degree polynomial (`InterpN.m` was used to generate the interpolant) on the interval $[0, 1]$.

   Consider the following two graphs that contain the function and its second-degree polynomial interpolant:



Clearly, the interpolant over f is much less accurate than the interpolant over g. Explain why this is the case using the error analysis presented in Section 2.3.

3. (15 points) The following functions (specified in your text) are available on the course website: `InterpN.m`, `HornerN.m`, `CSInterp.m`, and `CSEval.m`. You will need to use these functions for this problem.

   Many engineering problems involve functions which are periodic, but not continuous. One ubiquitous such function is the squarewave. This waveform is given explicitly by

$$sqwv(x) = \begin{cases} 0 & 0 \le x < 0.5 \\ 1 & 0.5 \le x < 1 \\ 0 & 1 \le x < 1.5 \\ 1 & 1.5 \le x < 2 \\ 0 & x = 2 \end{cases}$$

   Since noncontinuous functions are difficult to deal with, a common technique in analyzing them is to approximate them with a continuous function. However, the question then becomes, "how good is the approximation?" To answer this question, we need to have a "distance" function which measures how "close" one function is to the other. For any functions f and g defined on $[0, 2]$, the distance function we shall use is given by

$$dist(f, g) = \int_0^2 (f(x) - g(x))^2 \, dx.$$

   (This is called the L^2-distance between f and g, and it is fundamental to Fourier Analysis.) In MATLAB, this can be implemented with the following code:

```
function d=dist(x,f,g)
% x is an m-vector
% f and g are m-vectors containing the
% values of f(x) and g(x), respectively
diff=(f-g);
diffsq=diff.*diff;
d=trapz(x,diffsq);
```

   (a) Set $m = 200$. Let n be a positive integer. Choose n equally spaced points in $[0, 2]$ with

$$0 = x_1 < x_2 < \ldots < x_n = 2$$

Let p be the degree $(n-1)$ polynomial which interpolates $sqwv$. I.e., $p(x_i) = sqwv(x_i)$ for $i = 1:n$. Also, let

$$polyerror = dist(p, sqwv).$$

Write a MATLAB script which will compute p and $polyerror$ given n. Experiment with different values of n and use the value of n that minimizes $polyerror$.

(b) Let k be an even positive integer (k must be even to use `CSInterp.m`). Choose k equally spaced points in $[0, 2]$ with

$$0 = x_1 < x_2 < \ldots < x_k = 2$$

Let t be the trigonometric interpolant of $sqwv$. I.e., $t(x_i) = sqwv(x_i)$ for $i = 1:n$. Also, let

$$trigerror = dist(t, sqwv).$$

Write a MATLAB script to compute t and $trigerror$ given k. Experiment to find the smallest value of k such that $trigerror$ is less than the minimal value of $polyerror$ you found above. You may include code to (a) and (b) in the same MATLAB script if you wish.

In a single figure, plot $sqwv$ (solid line), its polynomial interpolant, p (dotted line), and its trigonometric interpolant, t (dashdot line), for the values of k and n you found above. Use m points in your plots. Label your axes and in the title of your plot include the value of k and n that you used. Also printout the associated values for $polyerror$ and $trigerror$.

Turn in your script(s) and the output (the figure, polyerror, and trigerror). If you create any functions (other than `dist`), turn in a printout of those functions as well.

4. (10 points) The following functions (specified in your text) are available on the course website: `pwL.m`, `pwC.m`, `pwCEval.m`, and `Locate.m`. You may find these useful in this problem (material in this problem involves Section 3.1 and 3.2 which will be covered in class on Friday and Monday).

   We are covering piecewise linear and piecewise cubic interpolation. You may (or may not...) have wondered why piecewise quadratic interpolation is not covered in the text. This problem should help to answer this question.

   In piecewise quadratic interpolation, a function $g(z)$ is interpolated with a quadratic on each of the subintervals. For this problem, define

$$g(z) = \frac{e^z - (\cos(3\pi z))^2}{3 + z}$$

   Analogous to Section 3.2.2 we assume $x_1 < x_2 < \ldots < x_n$ and that $Q(z)$ interpolates the data $(x_1, y_1), \ldots (x_n, y_n)$. We define ith local quadratic by

$$q_i(z) = a_i + b_i(z - x_i) + c_i(z - x_i)(z - x_{i+1})$$

   Therefore the piecewise quadratic polynomial is defined by:

$$Q(z) = \begin{cases} q_1(z) & \text{if } x_1 \le z < x_2 \\ q_2(z) & \text{if } x_2 \le z < x_3 \\ \vdots & \\ q_{n-1}(z) & \text{if } x_{n-1} \le z \le x_n \end{cases}$$

   (a) Write a function, `g` in MATLAB that takes as input a vector x and outputs a vector with the values of $g(x)$.

(b) Using the setup in Section 3.2.2 as an example, derive the equations to calculate a_i, b_i where we require that $Q(x_i) = y_i$. Show all work.

(c) We will require that $Q(z)$ has one continuous derivative. In order to do this, we can define c_i in the following way (you do not need to show why this works):

$$c_i = \frac{1}{x_i - x_{i+1}} \left[c_{i-1}(x_i - x_{i-1}) + \frac{y_i - y_{i-1}}{x_i - x_{i-1}} - \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \right] \qquad \text{for } i = 2 : n-1$$

Notice that c_i depends on c_{i-1}. Therefore, once c_1 is specified, we can easily calculate all of the c_i, $i = 2 : n-1$.

Create a function `pwQ` that computes the piecewise quadratic interpolant of $g(z)$, and a function `pwQEval` that evaluates the polynomial calculated in `pwQ` at a vector of values z.

(d) Let `n=8`, `x=linspace(-.5,.5,n)'`, `y=g(x)`. Using the functions on the course web site and your functions created in parts (a) and (c), write a MATLAB script that computes the piecewise cubic Hermite interpolant and the piecewise quadratic interpolant of g where

$$C(x_i) = Q(x_i) = y_i \qquad \text{for } i = 1 : n$$

(C is the cubic Hermite interpolant).

Evaluate each of these interpolants at $m = 50$ evenly spaced points in $[-0.5, 0.5]$. Output the following:

- the vectors a, b, c, d that specify the cubic Hermite interpolant and the vectors aq, bq, cq that specify the quadratic interpolant (set $cq(1) = 5$).
- two figures; one figure should be a plot of the function $g(z)$ (solid line) and its cubic Hermite interpolant (dotted line), the second figure should be a graph of the function $g(z)$ (solid line) and its quadratic interpolant (dotted line). Label your axes and give each plot a title. Use `legend` to label each graph on your figures.

(e) Now, add 1 to the value of $y(1)$. Re-do part (d) with this "perturbed" data. Using all of your plots from parts (d) and (e), give a couple of reasons why it may be more beneficial to use piecewise cubic Hermite interpolation rather than piecewise quadratic interpolation.

Turn in printouts of all your functions (`g`, `pwQ`, `pwQEval`, and any others that you create). In addition, turn in your scripts from parts (d) and (e), corresponding output, and all of your figures (there should be four total figures).