CS 222 - Homework 3
Due in lecture Wednesday, July 25, 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. (15 points)

   Description:

   Inverse interpolation is a type of interpolation frequently used to find the roots of a given equation. The approach is as follows. Given a monotone invertible function $f(x)$ defined on $[a, b]$, we can find a cubic spline interpolate of f using $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = f(x_i)$ (*monotone* means that the function is strictly increasing or strictly decreasing). Assuming that

   $$a = x_1 < x_2 < \ldots < x_n = b$$

   then the cubic spline interpolant can be found by issuing the command `S=spline(x,y)`.

   Inverse interpolation involves sorting these points with respect to y_i and then interpolating the data (y_i, x_i), $i = 1 : n$. Note that to preserve the initial data, when the y-data is sorted, the x-data should be switched around accordingly (you may use the `sort` function in MATLAB).

   Once this reordering of data is performed, we should have (x_i, y_i) pairs $(i = 1 : n)$ where

   $$y_1 < y_2 < \ldots < y_n$$

   We can now find a cubic spline interpolant of this new data using `S=spline(y,x)`. The polynomial that interpolates the data (y_i, x_i), $i = 1 : n$ is an interpolant of f's inverse.

   Now, suppose $f(x)$ has one root in $[a, b]$. Let $q(y)$ be f's inverse interpolant. Then $q(0)$ can be thought of as an approximation to this root.

   Problem: Define

   $$g(x) = e^x \sin(4\pi x^2) + \frac{x^3}{3} - 0.25$$

   Write a MATLAB script using inverse interpolation with cubic splines and the root-finding method described above to find the maximum of $g(x)$ on $[0, 1]$.

   Turn in your script, any functions that you create, and the maximum value of $g(x)$ on $[0, 1]$. It is especially important that you comment your code well.

2. (10 points) Let $C(x)$ be the cubic Hermite interpolant of $f(x)$ at $x = a$ and $x = b$.

   (a) Show that

   $$\int_a^b C(x)dx = \frac{h}{2}(f(a) + f(b)) + \frac{h^2}{12}(f'(a) - f'(b))$$

   You must show all of your work to receive credit.

   (b) Develop a uniformly spaced composite rule based on the quadrature rule specified in part (a). Simplify your answer to avoid redundant calculations.

   (c) Implement your composite rule in MATLAB by completing the following function `CompQH.m` (the block of comments below should appear at the top of your function).

```
function q=CompQH(fname,fpname,a,b,n)
% q=CompQH(fname,fpname,a,b)
%
% Integrates a function using a composite Hermite rule
%
% fname  is a string containing the name of the function to be integrated
% fpname is a string containing the name of the function that implements
%   the derivative of fname
% a,b are the lower and upper limits of integration (a<b)
% n is the number of intervals used for the composite rule (n>=1)
% q is the approximation to the integral
```

   Now, write a script to test your function by computing the approximate value of

   $$\int_6^7 e^x \sin(x)dx$$

   Use $n = 1 : 5 : 31$ and compute the relative error in the integral. Your script should output the following:

   - Estimated value of the integral for each n above
   - Log-linear plot of the relative error vs. n (see `semilogy`)

   Turn in all your functions, your script, and your output. It may be helpful to know that $\int e^x \sin(x)dx = \frac{1}{2}e^x(\sin(x) - \cos(x))$.

3. (5 points) The following functions are available on the course website for use in this problem: `NCWeights.m`, `CompQNC.m`, and `AdaptQNC.m`. Feel free to modify these functions to fit this problem (however, it's in your best interest to not modify the function that contains the weights!).

   The gamma function crops up in electrical engineering problems in measuring interference in electrical signals. The gamma function is defined by

   $$\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt, \qquad x > 0$$

   One of the nice properties about the gamma function is that if k is a positive integer, then $\Gamma(k) = (k-1)!$.

   Write a MATLAB script that does the following:

   (a) Use the composite Simpson's rule with $n = 10$ to compute the approximate value of the above integral for $x = 1 : 10$. Since the upper limit is ∞, you will need to truncate the upper limit so it is finite. Use a "large" number for this upper limit (by large I mean numbers greater than 50).

(b) Use the adaptive Simpson's rule to compute the value of the integral. Again, use $x = 1 : 10$ and the same upper limit you used in part (a). Use a default tolerance of 0.01.

(c) Compute the relative error in all of your estimates obtained in parts (a) and (b). Print out a table containing the value of x, the estimates of the integral computed in (a) and (b) and the corresponding relative error. Comment on your observations, especially with regard to the error. Turn in your script, output, explanation, and any functions that you create.

4. (10 points) Complete the following function. Make sure that you exploit the structure of A for efficiency. Try to vectorize your code and use as few loops as possible. The following block of comments should appear at the top of your function.

```
function C=multAB(A,sup,d,sub)
% function C=multAB(A,sup,d,sub)
%
% A is an mxn matrix
% sup is a column (n-1)-vector
% d is a column n-vector
% sup is a column (n-1) vector
%
% C=A*B, where B is the tridiagonal matrix with d as its
% diagonal, sup as its superdiagonal, and sub as its subdiagonal
```

Download `hw3p4test.m` from the course website. Run this script and turn in your output as well as a printout of `multAB.m`.