Homework set 4: Due October 23 at 4:30 pm.

Homework should be given to me in class or put under my office door. Do not put it in my mail box. Homework is accepted up to one week late with a 20% penalty. For general rules, read the course home page.

Please use plots to demonstrate the behavior of the methods, the error, the distribution of points, whenever appropriate.
In all cases, discuss your findings.

1. Write and test a program for the adaptive Simpson quadrature method.
   (This topic is covered on pp. 328–331 in the handout.) The integrand should be defined by a matlab function which provides values of the integrand \( f(x) \) for any given input value \( x \).

   The adaptive Simpson quadrature algorithm is recursive and if your program is properly designed the value of \( f(x) \) should never be computed more than once for any particular value of \( x \). You can check if you do it right by counting the number of quadrature nodes and the number of function calls.

   When the program is running, there is an active interval the contribution of which to the final approximate value of the integral, we are trying to compute accurately enough. The overall tolerance, a positive number, \( \epsilon \), is provided as input. The contribution of each subinterval, to the overall error should not exceed \( \epsilon \) times the length of the subinterval measured as a fraction of the entire, given interval.

   The first active interval is the entire interval. If we decide that the quadrature rule is not accurate enough, the left half of the active interval becomes the active interval. Once the contribution from this interval has been computed accurately enough, we compute the contribution of the right half of the interval. This is a recursive procedure.

   You can either use recursive calls or construct stacks to accomplish the savings of the function values, already computed, that you need
later. A stack can easily be implemented using a vector and an index that serves as a pointer.

2. Test your program by finding the approximate value, for several values of the tolerance, of

\[ \int_0^1 x^{1/2} dx, \]

\[ \int_0^1 (1 - x^2)^{3/2} dx \]

and

\[ \int_0^1 \frac{\sin(x)}{x^{3/2}} dx \]

and compare the results with the exact values of the integrals if they are available.

Note that the third integrand takes on infinite values at one end point. Find an appropriate device to deal with this so as to obtain accurate values of the integral.

3. Find similar algorithms available in matlab and evaluate the same integrals. Compare the results.

4. Find a cubic spline program available in matlab and test it for the classical function due to Runge:

\[ f(x) = \frac{1}{1 + 25x^2}, \quad x \in [-1, +1], \]

and for

\[ f(x) = \sin(4\pi x), \quad x \in [-1, +1]. \]

Work with 5, 10, 20, and 40 equidistant points and try to estimate the error, i.e., the maximum difference of \( f(x) \) and its spline interpolant.

5. In view of the fact that the second function given above is periodic, redevelop the cubic spline method given in the recent handout to provide a periodic cubic spline interpolation. The most important difference is that there are no longer any end points that needs to be treated differently. Write and test a matlab program for equidistant points and compare the resulting spline with the one obtained, ignoring periodicity, by using the program provided by matlab. In particular, make a convergence study using 5, 10, 20, and 40 points.
6. Learn about *polyfit* and *polyval* in matlab and test it on Runge’s example using equidistant points, including the end points. Then, do the experiments over again with

\[ x_i = -\cos((2i - 1)\pi/2n), \quad i = 1, \ldots, n. \]