Numerical Computing Homework 8
Quadrature Programming Assignment
Due: Thursday, April 7

This assignment will look at several methods used to compute integrals, including an adaptive method that you will design according to the specs below. Your program should be tested on different functions whose answer you know. (Your writeup should include how you tested your program, for example, you tested it on functions it computes exactly, you can test it using functions that you know the error behavior, to see that it is converging properly, etc.) If your program is written modularly, all the parts below can be included using one driver routine.

1. Write a routine to compute an integral using the trapezoid rule with n uniformly spaced points. You should call it with a sequence of values for n, $2n$, $4n$, and estimate both the convergence rate and improved approximations using Romberg integration (i.e. Richardon extrapolation). This should be applied to the functions listed below. Your writeup should examine questions such as: does the Romberg procedure always help? When should you stop doing it? Try some other interesting functions you may think of.

2. Next we will develop an adaptive quadrature method (read about this in the Heath text). The idea is that if an approximation has too large an error estimate over an interval (e.g. the estimate exceeds your tolerance), it will be recursively divided, and each half of the interval should contribute no more than half the tolerance to the total. You can estimate the error in several ways, for example, use the midpoint rule and the trapezoid rule, and compare the differences, or use the trapezoid rule with 1 point and 2 points and use Richardson to estimate the error. The main thing about this part is that you should not evaluate the function more than one time at a given point. Design a data structure to save the function values and reuse them as needed. A tree works here, where the root node corresponds to the whole interval, and if the interval is subdivided the node gains a left and right child. Other data structures work nicely as well.

   Run your adaptive routine to compute

   $$F(t) = \int_0^1 cos(tx^2)dx$$

   For $t = 1$, how many function evaluations does your adaptive routine take, and how many does it take for the composite trapezoid rule with uniform points to get the same accuracy (just repeatedly double n until you reach the accuracy you want, you don't have to find the *exact* value of n). Plot the points where your function was evaluated, along with the total number of evaluations for the given examples. Is this what you would expect? What happens if you try to evaluate $F(t)$ above for large t, say $t = 1000$? Comment on any changes you had to make to your code for this case.