Assignment 3.

Given February 10, due February 21.

Objective: Software issues for basic numerical computing.

We want to study the function

$$f(x) = \int_0^1 \cos\left(xt^2\right) dt \quad . \tag{1}$$

Step 1: Write a procedure to estimate $f(x)$ using a panel integration method with uniformly spaced points. The procedure should be well documented, robust, and clean. Robust will mean many things in future assignments. Here is means: (i) that it should use the correct number of panels even though the points t_k are computed in inexact floating point arithmetic, and (ii) that the procedure will signal failure and possibly print an error message if one of the calling arguments is out of range (here, probably just $n \leq 0$). It should take as inputs x and $n = 1/\Delta t$ and return the approximate integral with that x and Δt value. This routine should be written so that another person could easily substitute a different panel method or a different integrand by changing a few lines of code.

Step 2: Verify the correctness of this procedure by checking that (i) it gives the right answer with integrand $\cos(xt)$, and (i) it gives the right answer for small x. We can estimate $f(x)$ for small x using a few terms of its Taylor series. This series can be computed by integrating the Taylor series for $\cos(xt^2)$ term by term. This will require you to write a "driver" that calls the integration procedure with some reasonable but not huge values of n and compares the returned values with the Taylor series approximation.

Step 3: With $x = 1$, do a convergence study to verify the second order accuracy of the trapezoid rule and the fourth order accuracy of Simpson's rule. This requires you to write a different driver to call the integration procedure with several values of n and compare the answers in the manner of a convergence study. Once you have done this for the trapezoid rule, it should take less than a minute to redo it for Simpson's rule. This is how you can tell whether you have done Step 1 well.

Step 4: Write a procedure that uses the basic integration procedure from Step 1, together with Richardson error estimation to find an n that gives $f(x)$ to within a specified error tolerance. The procedure should work by repeatedly doubling n until the estimated error, based on comparing approximations, is less than the tolerence given. This routine should be robust enough to quit and report failure if it is unable to achieve the requested accuracy. The input should be x and the desired error bound. The output should be the estimated value of f, the number of points used, and an error flag to report failure. Before applying this procedure to the panel integration procedure, apply it to the fake

procedure `fakeInt.c` or `fakeInt.C`. Note that these testers have options to make the Richardson program fail or succeed. You should try it both ways, to make sure the robustness feature of your Richardson procedure works. Include with your homework, output illustrating the behavior of your Richardson procedure when it fails.

Step 5: Here is the "science" part of the problem, what you have been doing all this coding for. We want to test an approximation to f that is supposed to be valid for large x. The supposed approximation is:

$$f(x) \sim \sqrt{\frac{\pi}{8x}} + \frac{1}{2x}\sin(x) - \frac{1}{16x^2}\cos(x) + \cdots \quad . \tag{2}$$

Make a few plots showing f and its approximations using one, two and all three terms on the right side of (2) for x in the range $1 \leq x \leq 1000$. In all cases we want to evaluate f so accurately that the error in our f value is much less than the error of the approximation (2). Note that even for a fixed level of accuracy, more points are needed for large x. Why?