**Homework #1, due February 19**

**Objectives:** (1) To warm up to programming with good habits, (2) To illustrate basic numerical ideas.

**(1)** Explore truncation error and rounding error, and resolution and accuracy we seek to solve the following integral equation:

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

This equation is related to the Fresnel integral (Abramowitz and Stegun, Section 7.3):

$$C(z) = \int_0^z \cos\left(\frac{\pi}{2}t^2\right) dt \quad . \tag{2}$$

The equation $f(x)$ can be expressed in terms of the Fresnel integral:

$$f(x) = \sqrt{\frac{\pi}{2x}}\, C\left(\frac{2x}{\pi}\right) . \tag{3}$$

**(a)** Write a program (in matlab or other language) to estimate the integral using a panel integration method (such as trapezoid rule and Simpson's rule), with the number of panels $n = 1/\Delta x$ as input. The program should be modular, well documented, This routine should be written so that another person could easily substitute a different quadrature by changing a few lines of code. (You can use standard subroutines such as these in NAG, Numerical Recipes, Netlib: http://www.netlib.org, MATLAB: type help funfun.) Turn in a listing of this code.

**(b)** Verify the correctness of the program by checking that it gives the right answer for small $x$. We can estimate the integral 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. Turn in a table showing your Taylor series approximation and the value returned by your code for a few small values of $x$, say, $x = 0.1, 0.2, 0.3$.

**(c)** With $x = 1$, do a convergence study to verify the second order accuracy of the trapezoid rule (trapzd.f routine of Numerical Recipes, Section 4.2) and the fourth order accuracy of Simpson's rule (qsimp.f routine of Numerical Recipes, Section 4.2). Turn in a table showing the results for different step sizes $\Delta t$, or a log-log plot of the error, and explain how your results demonstrate the correct order of accuracy. Try different ways if checking accuracy such as comparing with "correct" answer computed numerically by a very fine grid; using relation of

$$(A(4h) - A(2h))/(A(2h) - A(h)) \tag{4}$$

.

**(d)** As in **(c)**, what happens when $n$ is very large for a given precision (single or double)? Find the value $n_0$ such that the accuracy check in **(c)** falls when $n \geq n_0$. What is the connection with rounding error? You will also find that accuracy check in **(c)** falls when $n$ is very small. This is related to the resolution and will be explained in **(e)**.

**(e)** For large values of $x$, the integrand will undergo many oscillations within the limits of the integration. Therefore, in order to achieve accuracy, the value of $h$ must decrease in order to resolve the features of the integrand. Perform a convergence study for large $x$, say $x = 100, 1000$, and find out how many grid points per-wave structure in the integrand is needed in order to obtain a clean accuracy check. You shall use relative error in the convergence study.

**(f)** Write a routine that uses the the trapezoidal rule and Richardson extrapolation to compute the integral to within a specified (absolute) error tolerance. The desired error bound should be input. The output should be the estimated value of the integral and the number of function evaluations used. You can use or modify the routine qtrap.f from Numerical Recipes, Section 4.2. This routine should be robust enough to quit and report failure if it is unable to achieve the requested accuracy. Turn in a few examples with your input and output from the routine. Try to find an example where your code reports failure.

**(g)** For large $x$, we have the following asymptotic approximation for the integration (See Abramowitz and Stegun, Eqn 7.3.9)

$$f(x) \sim \sqrt{\frac{\pi}{8x}} + \frac{1}{2x}\sin(x) - \frac{1}{4x^2}\cos(x) + \cdots \quad . \tag{5}$$

Make a few plots showing $f$ and its approximations using one, two and all three terms on the right side of (5) for $x$ in the range $1 \le x \le 1000$. 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 (5). Note that even for a fixed level of accuracy, more points are needed for large $x$ as explained in **(e)**.

**(2)** Explore finite difference method and Gaussian Elimination.

Consider the following two-point boundary value problem:

$$-(k(x)u')' + q(x)u = f(x), \quad \text{for } 0 \le x \le \pi \tag{6}$$

with the Dirichlet boundary condition

$$u(0) = \alpha, \quad u(\pi) = \beta$$

where $k(x), q(x) > 0$.

To solve the problem numerically, first we set up the grid as

$$x_i = ih, \quad i = 0, 1, \cdots, N, \quad h = \pi/N. \tag{7}$$

**(a)** Derive a (second order) centered difference approximation to the above differential equation and perform a truncation error analysis. Write down the difference approximation as a symmetric tridiagonal linear system with a coefficient matrix as $A = \text{diag}\ (a_i, b_i, b_i)$ .

**(b)** Show that the tridiagonal matrix is diagonal dominant and positive definite.

**(c)** Write a program to perform Gaussian Elimination.

**(d)** When

$$k(x) = q(x) = 1, \quad \alpha = \beta = 0 \tag{8}$$

and $f(x) = (1 + m^2)\sin(mx)$, show that $u(x) = \sin(mx)$ is an exact solution to the two-point value problem (6), where $m$ is an integer. Take $m = 1$ and compute the errors

$$E(h) = \sum_{i=0}^{N} |u_i - u(x_i)|h \tag{9}$$

for $N = 8, 16, 32, 64, 128$ and to show that $E(h)/h^2$ is almost a constant (This is another way to check your code). What happens when $m = 20$ and $N = 8, 16, 32, 64, 128, 256, 512$?

**(2)** Explore the condition number and rate of convergence.

Consider the specially case (8) of the two-point boundary value problem (6) with finite difference approximation:

$$-\frac{u(x_{j+1}) - 2u(x_j) + u(x_{j-1})}{h^2} + u(x_j) = f(x_j), \tag{13}$$

and the boundary conditions:

$$u(x_0) = u(x_N) = 0 \tag{14}$$

- **(a)** For $k = 1, 2, \cdots N-1$, show that vector $u_k = \{\sin(jkh)\}_{j=1}^{N-1}$ is an eigenvector of the tridiagonal matrix and compute the corresponding eigenvalue.
- **(b)** Show that the condition number for the tridiagonal matrix is of order $O(1/h^2)$.
- **(c)** Write programs to perform iteration methods: Jacobi, Gauss-Seidel, and SOR and examine the rate of convergence. Find out the relation between the condition number and the rate of convergence.

**(4)** Explore fast Poisson solver.

Consider the finite difference approximation (10)-(11) to the two-point value problem (6)-(8).

- **(a)** We represent the solution in following Sine series:

$$u_j = \sum_{k=1}^{N-1} \hat{u}_k \sin\left(\frac{jk\pi}{N}\right), \quad j = 1, 2, \cdots, N-1 \tag{12}$$

The Fourier coefficients in (12) can be expressed in the following inverse transformation:

$$\hat{u}_k = \frac{2}{N} \sum_{j=1}^{N-1} u_j \sin\left(\frac{jk\pi}{N}\right), \quad k = 1, 2, \cdots, N-1 \tag{13}$$

Be aware that the exact formulation in (12)-(13), especially the normalizing constants, may be different in FFT packages. You shall read the documentations of the package carefully.

(i) The choice of the Fourier expansion is based on the boundary conditions. Verify that the solution in the form of (12) satisfying the boundary conditions (11).

(ii) Express the right hand side $f$ in (10) in a similar transformation and find the Fourier multiplier $\lambda_k$ such that $\lambda_k \hat{u}_k = \hat{f}_k$.

(iii) Derive and implement a fast Poisson solver for (10)-(11).

- **(b)** For the exact solution in **2(g)**, use your code of fast Poisson solver to check the accuracy for the case $k = 1$ and $k = 20$.
- **(c)** Extend your faster Poisson solver to 2-d domain $[0, 1]^2$.