Homework 5
MA/CS 375, Fall 2005
Due December 2

This homework will count as part of your grade so you must work independently. It is permissible to discuss it with your instructor, the TA, fellow students, and friends. However, the programs/scripts and report must be done only by the student doing the project. Please follow the guidelines in the syllabus when preparing your solutions.

1. (20 pts.) In this problem we explore the volume of a random parallelepiped in 4-dimensional space whose sides are unit vectors and its connection to the condition number. Using a well-known result of analytic geometry, the volume of a parallelepiped is found as the determinant of the matrix formed by the vectors describing all of its sides originating at a given (any) vertex. Your program should:

   (a) Generate 4 random column vectors of size 4×1
   (b) Normalize these veectors so their length is unity.
   (c) Compute the volume of the parallelepiped they define. You must use the LU factorization for computing the determinant of the matrix A whose columns are the 4 random vectors (that is DO NOT use the command $det(\mathbf{A})$ here!).
   (d) compute the condition number of $\mathbf{A}$, $\kappa(\mathbf{A})$.
   (e) Repeat the process 1000 times and produce a scatter plot of $det(\mathbf{A})$ vs $\frac{1}{\kappa(\mathbf{A})}$.

   What are the maximum and minimum values of $det(\mathbf{A})$? What is the condition number for these values? How can you explain the extreme values of the condition number and the corresponding values of the determinant?

2. (20 pts.) Given an n-vector, x, of distinct elements, $x_i \neq x_j$ for $i \neq j$, the Cauchy matrix, $C(x)$, is the $n \times n$ matrix whose ijth entry is:

$$c_{ij} = \frac{1}{x_i + x_j}.$$

   Write a program which generates Cauchy matrices of size $n = [4 \quad 8 \quad 12 \quad 16]$ with random input vectors. Compute the condition numbers of these matrices. In each case set $z = \text{ones}(n, 1)$ and compute $b = Cz$. Use the built-in Matlab functions to solve $Cz = b$. Denote the computed solution by $\hat{z}$. Compute the relative errors and the relative residuals:

$$\frac{\|z - \hat{z}\|}{\|z\|}, \quad \frac{\|b - C\hat{z}\|}{\|b\|}.$$

   Comment on the results. Can you make sense of them given the condition numbers you found?

3. (30 pts.) In this problem we solve the heat equation using a second-order accurate method for timestepping. Consider the heat equation

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

for $-1 \le x \le 1$ and $0 \le t \le T$ with boundary conditions

$$u(x = -1, t) = u(x = 1, t) = 0 \ ,$$

and initial condition

$$u(x, t = 0) = \sin \pi r x, -1 \le x \le 1 \ .$$

The *Crank-Nicolson* scheme is a popular scheme for solving the heat equation because it is accurate of order 2 both in the timestep, $dt := k$ and in the spatial discretization interval $dx := h$. It is defined by the finite difference equations:

$$\frac{u_m^{n+1} - u_m^n}{k} = \frac{1}{2}\frac{u_{m+1}^{n+1} - 2u_m^{n+1} + u_{m-1}^{n+1}}{h^2} + \frac{1}{2}\frac{u_{m+1}^n - 2u_m^n + u_{m-1}^n}{h^2} \ , \ n = 0, \ldots, N-1 \ , \ m = 1, \ldots, M-1$$

where we have introduced the discretization:

$$u_m^n := u(x_m, t_n) \ , \ x_m = -1 + h * m \ , \ m = 0, \ldots, M \ ; \ t_n = k * n \ , \ n = 0, \ldots, N$$

with $h = 2/M$ and $k = T/N$. The reason for the increase in accuracy is that the above expressions give the derivatives of the function $u(x, t)$ with respect to x and t at the point x_m and the instant $t = (t_n + t_{n+1})/2$ so we can think of the time derivative formula as a centered (rather than a forward) difference. You must demonstrate the accuracy of the scheme by carrying out computations for different timesteps k, and comparing your results with the exact solution given by the expression:

$$u_{exact}(x, t) = e^{-\pi^2 r^2 t} \sin \pi r x \ .$$

Specifically:

(a) Write the system of equations in the form

$$(I - \delta D)\, \mathbf{u}^{n+1} = (I + \delta D)\, \mathbf{u}^n \ , \ n = 0, \ldots, N-1$$

where

$$\delta = \frac{k}{2h^2} \ , \ \mathbf{u}^n = (u_1^n, \ldots, u_{M-1}^n)^T \ (u_0^n = u_M^n = 0)$$

(b) Start the computation off by setting

$$u_m^0 = \sin \pi k x_m = \sin \pi k(-1 + h * m) \ , \ m = 1, \ldots, M-1$$

(c) perform the LU factorization of the left-hand matrix $LU = D^- := (I - \delta D)$. You must define D^- (and D^+) using the command *spdiags* to take advantage of its sparsity when solving this system.

(d) Iterate until the final time, $t = T$ is reached.

(e) Run with $T = 1$ and $r = 1$. Use $M = 20, 40, 80, 160$ and $N = 10, 20, 40, 80$.

(f) Compute the relative error

$$e(M, N) = \frac{\sum_{m=1}^{M-1} |u_m^N - u_{exact}(x_m, T)|}{\sum_{m=1}^{M-1} |u_{exact}(x_m, T)|}$$

for each pair (M, N) and show its value in a table.

(g) Determine which pairs work best together; that is determine, for each value of M (and grid-spacing h), which value of N (and timestep k) gives a time-discretization error that is of the same order as the space-discretization error. For these optimal (M, N) pairs, show that the error decreases like either k^2 or like h^2.

(h) For $M = 100$ and the corresponding optimal value for N, solve the heat equation and plot the solutions for $r = 1, 2, 3$; for each value of r show on the same plot solutions for 5 equally spaced time instants (your plots must extend over the entire computation interval, including the endpoints). How do the solutions behave as r increases?

4. (30 pts.) Consider the nonlinear boundary value problem:

$$\frac{d^2u}{dx^2} + \lambda^2 e^{2u} = 0, \quad x \in (-1, 1),$$

with $\lambda = \frac{1}{\cosh 1}$ and with boundary conditions $u(-1) = u(1) = 0$.

i. Verify that:

$$u(x) = -\ln(\lambda \cosh x),$$

solves the problem.

ii. Introducing a uniform grid, $x_j = -1 + jh$, $j = 0, \ldots, n$, $h = \frac{2}{n}$, approximate $u(x_j)$ by U_j and approximate $\frac{d^2u}{dx^2}$ by the second order central difference formula. This leads to the system of $n-1$ nonlinear equations:

$$F_j(u) \equiv \frac{U_{j-1} - 2U_j + U_{j+1}}{h^2} + \lambda^2 e^{2U_j} = 0, \quad j = 1, \ldots, n-1,$$

where we set $U_0 = U_n = 0$. The Jacobian derivative of a collection of $n-1$ functions, F_j of $n-1$ variables, U, is the $(n-1) \times (n-1)$ matrix $J(\mathbf{U})$ whose jkth entry is the derivative of F_j with respect to U_k, $J_{jk} = \frac{\partial F_j}{\partial U_k}$. (We denote the $n-1$-vectors whose entries are F_j and U_k by $\mathbf{F}$ and $\mathbf{U}$.) Show that the Jacobian derivative of the function given above is a tridiagonal matrix whose entries are:

$$J_{j,j+1} = J_{j+1,j} = \frac{1}{h^2}, \quad J_{jj} = -\frac{2}{h^2} + 2\lambda^2 e^{2U_j}.$$

iii. Newton's method for solving a system of $n-1$ equations in $n-1$ unknowns is the direct generalization of the Newton's method for a single equation which we studied in Chapter 2. Precisely, given an initial approximation, $\mathbf{U}^{(0)}$, we generate subsequent approximations by solving:

$$J(\mathbf{U}^{(k)})\, \mathbf{d}^{(k)} = -\mathbf{F}(\mathbf{U}^{(k)}),$$

and setting:

$$\mathbf{U}^{(k+1)} = \mathbf{U}^{(k)} + \mathbf{d}^{(k)}.$$

Write a script implementing Newton's method for the difference approximation described above. Terminate the iteration when:

$$\|\mathbf{d}^{(k)}\| \leq 10^{-8}.$$

Use $\mathbf{U}^{(0)} = \mathbf{0}$. Be sure to treat J as a sparse matrix when solving. Record the number of iterations required and print out $\|\mathbf{d}^{(k)}\|$ for each k. Does the convergence appear to be quadratic? Carry out the computations for $n = 10, 20, 40$. Compute the maximum error, that is the maximum absolute difference between U_j and $u(x_j)$, in each case. Do you observe second order convergence with decreasing h? Plot the solutions in each case.