AMSC/CMSC 466 Problem set 3

1. Problem 1 of KC, p180, parts (a), (b) and (c). Do part (a) by hand, with and without pivoting. Use MATLAB to check your answer. Use the command `A\b` to get the solution, and the command `[L,U,P] = lu(A)` to verify the factorization. Then do parts (b) and (c) using MATLAB to find the solution x and to find the factoriztion.

2. Consider the linear system $Ax = b$ where A is the matrix

$$A = \begin{bmatrix} 6 & 2 & 2 \\ 2 & 2/3 & 1/3 \\ 1 & 2 & -1 \end{bmatrix}$$

and $b = (-2, 1, 0)$.

a) Verify that the exact solution is $x_1 = 2.6,\ x_2 = -3.8,\ x_3 = -5.0$.

b) Using four digit floating-point decimal arithmetic with rounding, solve this system by Gaussian elimination without pivoting. In performing the arithmetic operations, remember to round to four digits after each operation. In particular, use this rounding when entering the elements of the matrix, and in computing the multipliers. Call the computed solution y.

c) Repeat part b) using partial pivoting. Call the computed solution z. Compare the solutions y and z with the exact solution x.

3. Problem 32, page 160 of KC. Use the algorithm given in class for the Cholesky factorization. You may check your answer with MATLAB.

4. The $n \times n$ Hilbert matrix is

$$H_n = [h_{i,j}] \quad \text{where } h_{i,j} = \frac{1}{i+j-1}, \quad 1 \le i, j \le n.$$

This matrix is nonsingular and the inverse may be computed explicitly, The inverse $H_n^{-1} = a_{ij}$ where

$$a_{ij} = \frac{(-1)^{i+j}(n+i-1)!(n+j-1)!}{(i+j-1)[(i-1)!(j-1)!]^2(n-i)!(n-j)!}.$$

The MATLAB functions `hilb(n)` and `invhilb(n)` give H_n and H_n^{-1} respectively, using the formulas. If we set $b_n = (1, 0, \ldots, 0)$, then the exact solution x_n of $H_n x_n = b_n$ is the first column of H_n^{-1}.

(a) Using the MATLAB back slash command, solve for x_n for the cases $n = 5$ and $n = 10$. Call this computed result y_n.

(b) Let $e_n = x_n - y_n$ where x_n is the exact solution as given by the formula above and let $r_n = b_n - H_n y_n$ be the residual. Find the condition number of H_n in the 2-norm using the MATLAB command `cond` for $n = 5$ and $n = 10$. Verify that

$$\frac{\|e_n\|}{\|x_n\|} \leq \text{cond}(H_n)u$$

where $u = \text{eps}/2$ is the unit roundoff, and $\|\ \ \|$ is the 2-norm. The MATLAB command `norm(v)` computes the 2-norm of a vector v.

(c) How many accurate digits are there in the computed solutions in each case?

(d) Verify in each case that

$$\frac{\|r_n\|}{\|H_n\|\|y_n\|} \leq u.$$

5. Determine (by hand) the condition number, relative to the 1-norm, of the matrix

$$A = \begin{bmatrix} 1 & a \\ a & 1 \end{bmatrix}$$

as a function of the parameter a. For what values of a does A become ill conditionned, assuming the unit roundoff $u = 10^{-7}$ and at least three significant digits are desired ?

6. Let A be an $n \times n$ matrix, and let a_j be the jth column of A. Prove that

$$\text{cond}(A) \geq \frac{\max \|a_j\|}{\min \|a_j\|}.$$

Hint: Use the fact that $Ae_j = a_j$, where e_j is the jth column of the identity matrix, to derive bounds for $\|A\|$ and $\|A^{-1}\|$. Here $\|\ \ \|$ can be the 1-norm, the 2-norm or the infinity-norm.

7. Consider the problem of finding a solution of the boundary value problem for given $f(x)$

$$-u''(x) = f(x) \qquad \text{for } 0 < x < 1 \tag{1}$$

$$u(0) = u(1) = 0.$$

The solution u can be found analytically in the form of an integral, but we shall use a finite difference approximation to calculate some numerical solutions of this problem.

First we approximate the second derivative with a difference quotient:

$$u''(x) \approx \frac{u(x + \Delta x)) - 2u(x) + u(x - \Delta x)}{(\Delta x)^2}$$

so that the DE becomes

$$-\frac{u(x + \Delta x) - 2u(x) + u(x - \Delta x)}{(\Delta x)^2} \approx f(x).$$

Now introduce grid points in the interval $[0, 1]$. Let the number of subintervals be n, setting $\Delta x = 1/n$ and $x_j = j\Delta x$, $j = 0, \ldots, n$. We let u_j be the approximate value we wish to compute for $u(x_j)$. The u_j are chosen to satisfy the system of linear equations

$$-u_{j+1} + 2u_j - u_{j-1} = (\Delta x)^2 f(x_j), \qquad j = 1, \ldots, J$$

where $J = n - 1$ is the number of interior mesh points. The boundary conditions say that $u_0 = u_{J+1} = 0$ so we get the $J \times J$ system

$$T\mathbf{u} = (\Delta x)^{-2} S\mathbf{u} = \mathbf{f}$$

where S is the the $J \times J$ tridiagonal matrix

$$S = \begin{bmatrix} 2 & -1 & 0 & \ldots & 0 \\ -1 & 2 & -1 & \ldots & 0 \\ 0 & -1 & 2 & \ldots & 0 \\ . & . & . & \ldots & 0 \\ . & . & . & \ldots & 0 \\ . & . & \ldots & 2 & -1 \\ 0 & 0 & \ldots & -1 & 2 \end{bmatrix}$$

$\mathbf{u} = (u_1, \ldots, u_J)$ and $\mathbf{f} = (f(x_1), \ldots, f(x_J))$.

Write a short MATLAB program (script mfile) to solve this system with the given function f as input. You can write f as in inline function. The

program should have n as an input parameter and should plot the solution u along with the function f on the interval $[0, 1]$. The program should be written to accomodate values of n as large as 200 ($J = 199$). Remember that in MATLAB an index in a vector always starts with one.

We shall take advantage of the fact that S is a sparse matrix to save on storage space. You will need to enter the matrix S as a sparse matrix using the `sparse` commands of MATLAB. Enter `help sparse` at the MATLAB prompt to get information.

Note that S can be written $S = E + D + E'$ where $D = 2I$ and E has -1 on the super diagonal and zeros elsewhere. D and E can each be entered as sparse matrices.

You can use the MATLAB command `u = T\f` to solve the sytem. You can also use the sequence of commands

```
[L,U] = lu(T)
v = L\f;
u = U\v;
```

For this size problem, the LU factorization does not save much time.

a) For starters take $f(x) = 1$. The exact solution is $u(x) = x(1 - x)/2$. Note that the values of u at the points x_j satisfy the difference equation exactly. You can use this solution to see if your program is working correctly.

b) Now use the function $f(x) = \sin(\pi x)$ with exact solution $u(x) = \sin(\pi x)/\pi^2$. Run your program with values of $n = 10, 50, 100, 200$. To see convergence of the numerical solutions to the exact solution, plot the numerical solutions for these values of n together on the same graph using the `hold on` command.

c) For each n of part c) compute the error $e_n = \max |u_j - \sin(\pi x_j)/\pi^2|$. According to what power of $\Delta x = 1/n$ is e_n tending to zero as n increases?

d) Now try a function f which is continuous but not C^1, such as

$$f(x) = \begin{cases} x/a, & 0 \le x \le a \\ (1-x)/(1-a), & a \le x \le 1 \end{cases} .$$

How does the solution curve compare with f? Is it smoother, or does it have the same jump in the derivative at $x = a$? Again try $n = 20, 50, 100, 200$ and observe the convergence of the computed solutions.

8. Now consider the heat equation for $u(x,t)$

$$u_t(x,t) - u_{xx}(x,t) = 0, \qquad 0 < x < 1, t > 0$$

$$u(0,t) = u(1,t) = 0, \quad t \geq 0$$

with the initial condition

$$u(x,0) = f(x), \quad 0 \leq x \leq 1.$$

Note that when u does not depend on t, the problem reduces to the one considered in problem 7. We will see that the time dependent problem can be solved numerically by the same methods.

We introduce a grid of mesh points in the x,t plane, setting $x_j = j\Delta x$ and $t_k = k\Delta t$. Here $\Delta x = 1/n$, with n being the number of subintervals. $J = n - 1$ is the number of interior meshpoints. We approximate the time derivative u_t with a difference quotient

$$u_t(x,t) \approx \frac{u(x,t+\Delta t) - u(x,t)}{\Delta t}$$

and u_{xx} with the difference quotient

$$u_{xx}(x,t) \approx \frac{u(x+\Delta x,t) - 2u(x,t) + u(x-\Delta x,t)}{(\Delta x)^2}.$$

Then we use $u_{j,n}$ to denote the values we will compute to approximate $u(x_j,t_k)$. We will use an *implicit* scheme to calculate the $u_{j,k}$ because it is more stable (resistent to error). We center the second difference quotient at the point (x_j, t_{k+1}) which yields the system of equations

$$\frac{u_{j,k+1} - u_{j,k}}{\Delta t} - \frac{u_{j+1,k+1} - 2u_{j,k} + u_{j-1,k+1}}{(\Delta x)^2} = 0$$

for $j = 1, \ldots, J$, $k = 0, 1, 2, \ldots$. The boundary condition $u(0,t) = u(1,t) = 0$ says that we must have $u_{0,k} = u_{n,k} = u_{J+1,k} = 0$ for all k. Let the vector $\mathbf{u}_k = (u_{1,k}, \ldots, u_{J,k})$. Then the system can be written compactly as

$$S\mathbf{u}_{k+1} = \mathbf{u}_k \tag{2}$$

where now the $J \times J$ matrix S is again tridiagonal,

$$S = \begin{bmatrix} 1+2\rho & -\rho & 0 & \dots & 0 \\ -\rho & 1+2\rho & -\rho \dots & 0 & 0 \\ 0 & -\rho & 1+2\rho & \dots & 0 \\ . & . & . & \dots & 0 \\ . & . & \dots & 1+2\rho & -\rho \\ . & . & \dots & -\rho & 1+2\rho \end{bmatrix}.$$

Here $\rho = \Delta t/(\Delta x)^2$. The initial condition $u(x,0) = f(x)$ tells us that the first vector in the sequence is $\mathbf{u}_0 = (f(x_1), \dots, f(x_J))$. Then we find the vectors $\mathbf{u}_k$ by repeatedly solving the system (2). In fact we have

$$\mathbf{u}_k = S^{-k}\mathbf{u}_0$$

but this is not a very fast nor accurate way of calculating the $\mathbf{u}_k$.

a) Write a MATLAB program which solves the system (2) for K time steps and plots out the snapshot of the solution at time $t = K\Delta t$. n, the number of spatial subintervals and and K, the number of time steps, should be input parameters. The initial data $f(x)$ can be written as an inline function. Enter the sparse matrix S using the techniques of problem 7. You will need to write a for loop, $k = 1, K$. You could use the short loop

```
for k = 1:K
   u = S\u;
end
```

However, this means you are making the LU factorization over and over again, at each iteration. Now we can take advantage of the LU factorization of S. Factor S once before the loop and then use the iteration

```
[L,U] = lu(S)
for k = 1:K
   v = L\u;
   u = U\v;
end
```

b) To begin use $f(x) = \sin(\pi x)$. The exact solution is

$$u(x,t) = \exp(-\pi^2 t)\sin(\pi x).$$

Check your code to see if it is working correctly with this example.

c) Now watch the convergence of computed solutions to the exact solution. Fix $n = 100$. First use $\Delta t = .01$ and $K = 10$ which computes the solution at $t = .1$. Next use $\Delta t = .005$ and $K = 20$. Superimpose the graphs of the two runs using the `hold on` command. Finally use $\Delta t = .0025$ and $K = 40$. What happens to the graphs? They all represent the computed solution at time $t = .1$. For each run, compute

$$\max_{1 \le j \le J} |u_{x_j,K} - u(x_j, .1)|.$$

How is the error converging to zero?

c) Now use initial data $f(x) = x(.33 - x)(x - 1)$. Plot f with a vector of points `x = 0:.01:1` in red with the command `plot(x,f(x), 'r')`. Then enter `hold on`. Next run your program with a time step $\Delta t = .01$, and $K = 1, 2, 3, 4, 5, 10$, plotting all the curves on the same graph. How does the solution evolve? What happens to the hot spot, what happens to the cool spot?

Hand in programs (script files and function files), Diaries, and plots. Be sure to label plots carefully. You can put in comment lines in the script and mfiles to indicated what is being done. You can edit the diaries to make additional comments. Above all, make sure your work is well organized and understandable.