

MATH 335: Numerical Analysis

Problem Set 9, Solutions

- 4.3.12 (a) The matrix has this appearance, where the nonzero entries occur only in the locations not marked as zero:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} & a_{17} & a_{18} & a_{19} & a_{1,10} \\ a_{21} & a_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_{31} & 0 & a_{33} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_{41} & 0 & 0 & a_{44} & 0 & 0 & 0 & 0 & 0 & 0 \\ a_{51} & 0 & 0 & 0 & a_{55} & 0 & 0 & 0 & 0 & 0 \\ a_{61} & 0 & 0 & 0 & 0 & a_{66} & 0 & 0 & 0 & 0 \\ a_{71} & 0 & 0 & 0 & 0 & 0 & a_{77} & 0 & 0 & 0 \\ a_{81} & 0 & 0 & 0 & 0 & 0 & 0 & a_{88} & 0 & 0 \\ a_{91} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{99} & 0 \\ a_{10,1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{10,10} \end{bmatrix}.$$

- (b) After just one GE step (i.e., after zeroing under just the first pivot), the 10th row looks like

$$-m_{10,1} \begin{bmatrix} 0 & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} & a_{17} & a_{18} & a_{19} & a_{10,10} + a_{1,10} \end{bmatrix}$$

The middle eight of these entries had previously been zero.

- (c) If we exchange the first and last rows, and then follow this up by exchanging the first and last columns, we arrive at the matrix

$$\tilde{\mathbf{A}} = \begin{bmatrix} a_{10,10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{10,1} \\ 0 & a_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{21} \\ 0 & 0 & a_{33} & 0 & 0 & 0 & 0 & 0 & 0 & a_{31} \\ 0 & 0 & 0 & a_{44} & 0 & 0 & 0 & 0 & 0 & a_{41} \\ 0 & 0 & 0 & 0 & a_{55} & 0 & 0 & 0 & 0 & a_{51} \\ 0 & 0 & 0 & 0 & 0 & a_{66} & 0 & 0 & 0 & a_{61} \\ 0 & 0 & 0 & 0 & 0 & 0 & a_{77} & 0 & 0 & a_{71} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{88} & 0 & a_{81} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{99} & a_{91} \\ a_{1,10} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} & a_{17} & a_{18} & a_{19} & a_{11} \end{bmatrix}.$$

In the case of $\tilde{\mathbf{A}}$, no required elementary row operation (used by naive Gaussian elimination) to arrive at echelon form will turn an already-zero entry into a nonzero one.

★14 (a) Here is my routine. When we need $\mathbf{B} = \mathbf{M}_\omega^{-1}\mathbf{N}_\omega$, notice that I use the command

$$\mathbf{M} \setminus \mathbf{N}$$

which I rely upon to carry out this computation as efficiently as possible.

```
function [omega, B, rho] = sorParamPick ( A )
% function omega = sorParamPick ( A )
%
% input:    a square, nonsingular matrix A
% outputs:
%   omega   value in the interval (0, 2) that nearly minimizes the
%           spectral radius (i.e., maximum absolute eigenvalue) of
%           B = inv(D + w*L) * ((1-w)*D - w*U)
%           where
%           D = diag(diag(A))
%           L = tril(A, -1)
%           U = triu(A, 1)
%   rho     the spectral radius of the matrix B
omega = 0;
if ( issquare(A) & det(A) != 0 )
    D = diag(diag(A));
    L = tril(A, -1);
    U = triu(A, 1);

    firstTime = 1;
    for w = 0.005:.005:1.995
        B = (D + w*L) \ ((1 - w)*D - w*U);
        if (firstTime)
            firstTime = 0;
            minSR = max(abs(eig(B)));
            best_w = w;
        else
            currSR = max(abs(eig(B)));
            if (currSR < minSR)
                minSR = currSR;
                best_w = w;
            end
        end
    end
    omega = best_w;
```

```

    rho = minSR;
else
    disp('Input matrix must be nonsingular.')

```

(b) My program is as follows:

```

function [itSoln, nIters, omega, rho] = sor (A, b, x0, tol = 10-6, m = 100)
% function [itSoln, nIters] = sor (A, b, x0, tol, m)
%
% inputs:
%   A      a nonsingular matrix
%   b      a vector for which the solution of Ax = b is desired
%   x0     an initial guess for the solution of Ax = b
%   tol    error bound
%   m      maximum number of iterations
%
% output:
%   itSoln an approximate solution, employing the SOR method, to Ax = b
%   nIters number of iterations
if ( issquare(A) & det(A) != 0 )           % check A is NS
    if ( prod (size(b) == [rows(A) 1]) ) % check b has correct dimensions
        [omega, rho] = sorParamPick (A);
        if (rho < 1)
            D = diag(diag(A));
            L = tril(A, -1);
            U = triu(A, 1);
            M = D/omega + L;
            N = (1 - omega)*D/omega - U;

            curr_x = x0;
            prev_x = zeros(size(x0));
            nIters = 0;
            while (norm(A*curr_x-b)>=tol & (nIters==0 | norm(curr_x-prev_x)>=tol))
                nIters += 1;
                prev_x = curr_x;
                curr_x = M \ (N*prev_x + b);
            end
            itSoln = curr_x;

```

```

else
    disp ('No parameter has been found to make the method converge.')
```

end

```

else
    disp ('Input vector dimensions must correspond to those of matrix.')
```

end

```

else
    disp ('Input matrix must be nonsingular.')
```

end

★15 (a) If we exchange rows 1 and 5, we then have the matrix problem

$$\begin{bmatrix} 42 & -2 & -9 & 0 & 3 & 0 \\ -8 & 36 & -1 & 0 & 1 & 8 \\ 11 & -4 & 25 & 1 & -4 & 4 \\ 1 & 0 & -3 & 19 & 2 & 1 \\ 2 & -5 & 2 & 5 & 30 & -8 \\ -3 & 7 & -7 & 4 & 5 & -32 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 54 \\ 36 \\ 42 \\ 18 \\ 27 \\ 60 \end{bmatrix},$$

whose coefficient matrix is diagonally dominant.

(b) Here are results from the Jacobi method, making use of the program `jacobi.m` supplied on the textbook CD.

```

octave:105> Ay = [42 -2 -9 0 3 0; -8 36 -1 0 1 8; 11 -4 25 1 -4 4; ...
                1 0 -3 19 2 1; 2 -5 2 5 30 -8; -3 7 -7 4 5 -32];
octave:106> be = [54;36;42;18;27;60];
octave:111> jacobi(Ay, be, ones(6,1), 10^(-6), 100)
```

The solution vectors are:

iter #	0	1	...	10	11	12	13
x1 =	1.000000	1.476190		1.674341	1.674337	1.674335	1.674335
x2 =	1.000000	1.000000		1.800051	1.800046	1.800043	1.800042
x3 =	1.000000	1.360000		1.520320	1.520319	1.520321	1.520321
x4 =	1.000000	0.894737		1.159053	1.159048	1.159047	1.159047
x5 =	1.000000	1.033333		0.320283	0.320293	0.320295	0.320295
x6 =	1.000000	-1.687500		-1.775859	-1.775852	-1.775852	-1.775853

The method converges after 13 iterations.

Carrying out Gauss-Seidel iterations on the same system (and using the program `seidel.m` supplied with the text), we get

```
octave:117> seidel(Ay, be, ones(6,1), 10^(-6), 100)
```

The solution vectors are:

iter #	0	...	5	6	7	8	9
x1 =	1.000000		1.673860	1.674331	1.674342	1.674335	1.674335
x2 =	1.000000		1.799748	1.800045	1.800046	1.800042	1.800043
x3 =	1.000000		1.520406	1.520347	1.520319	1.520321	1.520321
x4 =	1.000000		1.159033	1.159039	1.159047	1.159047	1.159047
x5 =	1.000000		0.320420	0.320289	0.320293	0.320295	0.320295
x6 =	1.000000		-1.775874	-1.775860	-1.775853	-1.775853	-1.775853

The method converges after 9 iterations.

Employing the SOR method (and routine from the previous problem),

```
octave:107> [x, n, omega, rho] = sor(Ay, be, ones(6,1), 10^(-6))
```

```
x =
  1.67434
  1.80004
  1.52032
  1.15905
  0.32029
 -1.77585
```

```
n = 9
```

```
omega = 1.0050
```

```
rho = 0.11253
```

Both SOR and Gauss-Seidel arrive at an acceptable answer in 9 iterations, 4 faster than Jacobi.

AP 4.6 (a) First, since $x_0 = 0$ and $x_{20} = 1$, the boundary conditions give that $y_0 = 0$ and $y_{20} = 0$. For $j = 1, \dots, 19$, we have

$$y_{j-1} - 2y_j + y_{j+1} = -\frac{Sh^2}{D} = -\frac{(.05)^2}{.01} = -0.25,$$

which is a system of equations that leads directly to the matrix problem $\mathbf{A}\mathbf{y} = \mathbf{b}$ with \mathbf{A} , \mathbf{y} and \mathbf{b} as specified in the problem.

- (b) If one attempts to use either the *jocobi.m* or *seidel.m* routines supplied with the book, one gets messages that neither converges. Since the coefficient matrix is not strictly diagonally dominant, there was no guarantee going in that either method would work. (A look at the spectral radius $\rho(\mathbf{M}^{-1}\mathbf{N})$ for the two cases shows that $\rho(\mathbf{M}^{-1}\mathbf{N}) = 0.98769$ for the Jacobi method—so it ought to converge very slowly (the ‘No convergence’ message is a consequence of reaching the maximum number of iterations without a satisfactory answer, and I set this maximum at 100)—while $\rho(\mathbf{M}^{-1}\mathbf{N}) = 1$ for Gauss-Seidel, suggesting it unlikely to ever converge.)

In what follows I have used the SOR method as in the routine written for Problem ★14.

```
octave:100> A = diag(ones(18,1), 1) + diag(ones(18,1), -1) - 2*diag(ones(19,1))
octave:101> b = -ones(19,1)/4
octave:103> [x, nIters, omega, rho] = sor(A, b, ones(19,1), 10^(-6));
octave:104> [0 x' 0], nIters, omega, rho
ans =
    0.0000    2.3750    4.5000    6.3750    8.0000    9.3750   10.5000
   11.3750   12.0000   12.3750   12.5000   12.3750   12.0000   11.3750
   10.5000    9.3750    8.0000    6.3750    4.5000    2.3750    0.0000

nIters = 58
omega = 1.7300
rho = 0.73000
```