Faculty of Engineering
Civil Engineering

Numerical Methods

Chapter 2

Special Matrices and Gauss-Siedel

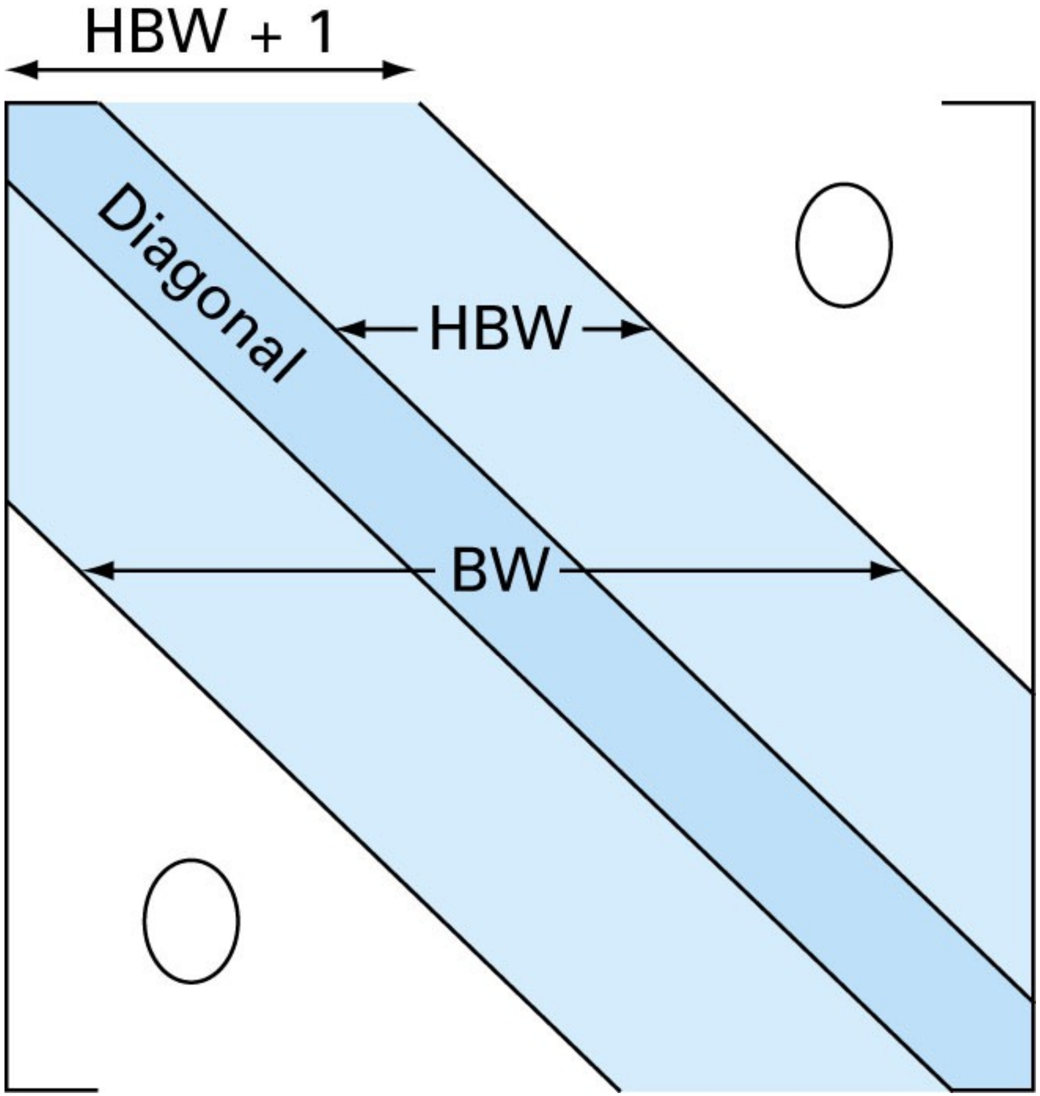
Introduction

- Certain matrices have particular structures that can be exploited to develop efficient solution schemes.

- A ***banded matrix*** is a square matrix that has all elements equal to zero, with the exception of a band centered on the main diagonal. These matrices typically occur in solution of differential equations.

- The dimensions of a banded system can be quantified by two parameters: the band width BW and half-bandwidth HBW. These two values are related by BW=2HBW+1.

Banded matrix



Tridiagonal Systems

- A tridiagonal system has a bandwidth of 3:

$$\begin{bmatrix} f_1 & g_1 & & \\ e_2 & f_2 & g_2 & \\ & e_3 & f_3 & g_3 \\ & & e_4 & f_4 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{Bmatrix}$$

- An efficient LU decomposition method, called *Thomas algorithm*, can be used to solve such an equation. The algorithm consists of three steps: decomposition, forward and back substitution, and has all the advantages of LU decomposition.

(*a*) Decomposition

```
DO k = 2, n
  e_k = e_k/f_{k-1}
  f_k = f_k - e_k · g_{k-1}
END DO
```

(*b*) Forward substitution

```
DO k = 2, n
  r_k = r_k - e_k · r_{k-1}
END DO
```

(*c*) Back substitution

```
x_n = r_n/f_n
DO k = n-1, 1, -1
  x_k = (r_k - g_k · x_{k+1})/f_k
END DO
```

Cholesky Decomposition

- This method is suitable for only symmetric systems where:

$$a_{ij} = a_{ji} \quad and \quad A = A^T$$

$$[L] = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix}$$

$$A = L * L^T$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} * \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{bmatrix}$$

Cholesky Decomposition

$$l_{ki} = \frac{a_{ki} - \sum_{j=1}^{i-1} l_{ij} \cdot l_{kj}}{l_{ii}} \quad for \quad i = 1, 2, \cdots, k-1$$

$$l_{kk} = \sqrt{a_{kk} - \sum_{j=1}^{k-1} l_{kj}^2}$$

Pseudocode for Cholesky's LU Decomposition algorithm (cont'd)

```
DO k = 1, n
  DO i = 1, k − 1
    sum = 0.
    DO j = 1, i − 1
      sum = sum + a_ij · a_kj
    END DO
    a_ki = (a_ki − sum)/a_ii
  END DO
  sum = 0.
  DO j = 1, k − 1
    sum = sum + a²_kj
  END DO
  a_kk = √(a_kk − sum)
END DO
```

Gauss-Siedel

- Iterative or approximate methods provide an alternative to the elimination methods. The Gauss-Seidel method is the most commonly used iterative method.
- The system *[A]{X}={B}* is reshaped by solving the first equation for x_1, the second equation for x_2, and the third for x_3, …and nth equation for x_n. We will limit ourselves to a 3x3 set of equations.

Gauss-Siedel

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2$$

$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$$

$$\Rightarrow$$

$$x_1 = \frac{b_1 - a_{12}x_2 - a_{13}x_3}{a_{11}}$$

$$x_2 = \frac{b_2 - a_{21}x_1 - a_{23}x_3}{a_{22}}$$

$$x_1 = \frac{b_3 - a_{31}x_1 - a_{32}x_2}{a_{33}}$$

Now we can start the solution process by choosing guesses for the x's. A simple way to obtain initial guesses is to assume that they are zero. These zeros can be substituted into x_1 equation to calculate a new $x_1 = b_1 / a_{11}$.

Gauss-Siedel

- New x_1 is substituted to calculate x_2 and x_3. The procedure is repeated until the convergence criterion is satisfied:

$$\left|\varepsilon_{a,i}\right| = \left|\frac{x_i^{new} - x_i^{old}}{x_i^{new}}\right| 100\% < \varepsilon_s$$

Jacobi iteration Method

An alternative approach, called ***Jacobi iteration,*** utilizes a somewhat different technique. This technique includes computing a set of new x's on the basis of a set of old x's. Thus, as the new values are generated, they are not immediately used but are retained for the next iteration.

First Iteration

(a)

$$x_1 = (c_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$$

$$x_2 = (c_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$$

$$x_3 = (c_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$$

(b)

$$x_1 = (c_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$$

$$x_2 = (c_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$$

$$x_3 = (c_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$$

Second Interation

(a)

$$x_1 = (c_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$$

$$x_2 = (c_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$$

$$x_3 = (c_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$$

(b)

$$x_1 = (c_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$$

$$x_2 = (c_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$$

$$x_3 = (c_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$$

(a) The Gauss-Seidel method

(b) The Jacobi iteration method

Convergence Criterion for Gauss-Seidel Method

- The gauss-siedel method is similar to the technique of fixed-point iteration.
- The Gauss-Seidel method has two fundamental problems as any iterative method:
  1. It is sometimes non-convergent, and
  2. If it converges, converges very slowly.
- Sufficient conditions for convergence of two linear equations, *u(x,y)* and *v(x,y)* are:

$$\left|\frac{\partial u}{\partial x}\right| + \left|\frac{\partial u}{\partial y}\right| < 1$$

$$\left|\frac{\partial v}{\partial x}\right| + \left|\frac{\partial v}{\partial y}\right| < 1$$

Convergence Criterion for Gauss-Seidel Method (cont'd)

- Similarly, in case of two simultaneous equations, the Gauss-Seidel algorithm can be expressed as:

$$u(x_1, x_2) = \frac{b_1}{a_{11}} - \frac{a_{12}}{a_{11}} x_2$$

$$v(x_1, x_2) = \frac{b_2}{a_{22}} - \frac{a_{21}}{a_{22}} x_1$$

$$\frac{\partial u}{\partial x_1} = 0 \qquad \frac{\partial u}{\partial x_2} = -\frac{a_{12}}{a_{11}}$$

$$\frac{\partial v}{\partial x_1} = -\frac{a_{21}}{a_{22}} \qquad \frac{\partial v}{\partial x_2} = 0$$

Convergence Criterion for Gauss-Seidel Method (cont'd)

Substitution into convergence criterion of two linear equations yield:

$$\left|\frac{a_{12}}{a_{11}}\right| < 1, \qquad \left|\frac{a_{21}}{a_{22}}\right| < 1$$

In other words, the absolute values of the slopes must be less than unity for convergence:

$$|a_{11}| > |a_{12}|$$
$$|a_{22}| > |a_{21}|$$

For n equations

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{i,j}|$$

That is, the diagonal element must be greater than the off-diagonal element for each row.

Gauss-Siedel Method- Example 1

$$\begin{bmatrix} 3 & -0.1 & 0.2 \\ 0.1 & 7 & -0.3 \\ 0.3 & -0.2 & 10 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 7.85 \\ -19.3 \\ 71.4 \end{Bmatrix} \Rightarrow \begin{aligned} x_1 &= \frac{7.85 + 0.1x_2 + 0.2x_3}{3} \\ x_2 &= \frac{-19.3 - 0.1x_1 + 0.3x_3}{7} \\ x_3 &= \frac{71.4 - 0.3x_1 + 0.2x_2}{10} \end{aligned}$$

- **Guess** x_1, x_2, x_3 = zero for the first guess

Iter.	x_1	x_2	x_3	$\lvert\varepsilon_{a,1}\rvert$(%)	$\lvert\varepsilon_{a,2}\rvert$ (%)	$\lvert\varepsilon_{a,3}\rvert$ (%)
0	0	0	0	-	-	-
1	2.6167	-2.7945	7.005610	100	100	100
2	2.990557	-2.499625	7.000291	12.5	11.8	0.076

Improvement of Convergence Using Relaxation

$$x_i^{new} = \lambda \cdot x_i^{new} + (1 - \lambda) \cdot x_i^{old}$$

- Where λ is a weighting factor that is assigned a value between [0, 2]
- If λ = 1 the method is unmodified.
- If λ is between 0 and 1 (under relaxation) this is employed to make a non convergent system to converge.
- If λ is between 1 and 2 (over relaxation) this is employed to accelerate the convergence.

Gauss-Siedel Method- Example 2

$$-8x_1 + x_2 - 2x_3 = -20$$
$$-3x_1 - x_2 + 7x_3 = -34$$
$$2x_1 - 6x_2 - x_3 = -38$$

Rearrange so that the equations are diagonally dominant

$$-8x_1 + x_2 - 2x_3 = -20$$
$$2x_1 - 6x_2 - x_3 = -38$$
$$-3x_1 - x_2 + 7x_3 = -34$$

$$x_1 = \frac{-20 - x_2 + 2x_3}{-8} \qquad x_2 = \frac{-38 - 2x_1 + x_3}{-6} \qquad x_3 = \frac{-34 + 3x_1 + x_2}{7}$$

Gauss-Siedel Method- Example 2

iteration	unknown	value	ε_a	maximum ε_a
0	x_1	0		
	x_2	0		
	x_3	0		
1	x_1	2.5	100.00%	
	x_2	7.166667	100.00%	
	x_3	-2.7619	100.00%	100.00%
2	x_1	4.08631	38.82%	
	x_2	8.155754	12.13%	
	x_3	-1.94076	42.31%	42.31%
3	x_1	4.004659	2.04%	
	x_2	7.99168	2.05%	
	x_3	-1.99919	2.92%	2.92%

Gauss-Siedel Method- Example 2

The same computation can be developed with relaxation where $\lambda = 1.2$

<u>First iteration:</u>

$$x_1 = \frac{-20 - x_2 + 2x_3}{-8} = \frac{-20 - 0 + 2(0)}{-8} = 2.5$$

Relaxation yields: $x_1 = 1.2(2.5) - 0.2(0) = 3$

$$x_2 = \frac{-38 - 2x_1 + x_3}{-6} = \frac{-38 - 2(3) + 0}{-6} = 7.333333$$

Relaxation yields: $x_2 = 1.2(7.333333) - 0.2(0) = 8.8$

$$x_3 = \frac{-34 + 3x_1 + x_2}{7} = \frac{-34 + 3(3) + 8.8}{7} = -2.3142857$$

Relaxation yields: $x_3 = 1.2(-2.3142857) - 0.2(0) = -2.7771429$

Gauss-Siedel Method- Example 2

Iter.	unknown	value	relaxation	ε_a	maximum ε_a
1	x_1	2.5	3	100.00%	
	x_2	7.3333333	8.8	100.00%	
	x_3	-2.314286	-2.777143	100.00%	100.000%
2	x_1	4.2942857	4.5531429	34.11%	
	x_2	8.3139048	8.2166857	7.10%	
	x_3	-1.731984	-1.522952	82.35%	82.353%
3	x_1	3.9078237	3.7787598	20.49%	
	x_2	7.8467453	7.7727572	5.71%	
	x_3	-2.12728	-2.248146	32.26%	32.257%
4	x_1	4.0336312	4.0846055	7.49%	
	x_2	8.0695595	8.12892	4.38%	
	x_3	-1.945323	-1.884759	19.28%	19.280%

MATLAB M-File for Gauss-Seidel method

```
function x = GaussSeidel(A, b, es, maxit)
% GaussSeidel(A,b): Gauss-Seidel method.
% input:
%    A = Coefficient Matrix
%    b = right hand side vector
%    es = (optional) stop criterion (%) (default = 0.00001)
%    maxit = (optional) maximum iterations (default = 50)
% Output
%    x = solution vector
if nargin < 4, maxit = 50; end
if nargin < 3, es = 0.00001; end
[m,n] = size(A);
if m ~= n, error('Matrix A must be square'); end
C = A;
for i = 1 : n
    C(i,i) = 0;
    x(i) = 0;
end
x = x';
for i = 1 : n
    C(i,1:n) = C(i,1:n) / A(i,i);
end
for i = 1: n
    d(i) = b(i) / A(i,i);
end
```

Continued on next page

MATLAB M-File for Gauss-Seidel method

Continued from previous page

```
disp('        i          x1          x2          x3          x4    ...');
while (1)
    xold = x;
    for i = 1 : n
        x(i) = d(i) - C(i,:) * x;
        if x(i) ~= 0
            ea(i) = abs((x(i) - xold(i)) / x(i)) * 100;
        end
    end
    iter = iter + 1;
    disp([iter  x'])
    if max(ea) <= es | iter >= maxit, break, end
end
if iter >= maxit
disp('Gauss Seidel method did not converge');
disp('results after maximum number of iterations');
else
    disp('Gauss Seidel method has converged');
end
x;
```

Gauss-Seidel Iteration

```
» A = [4 -1 -1; 6 8 0; -5 0 12];
» b = [-2 45 80];
» x=Seidel(A,b,x0,tol,100);
       i          x1          x2          x3      x4 ....
    1.0000    -0.5000     6.0000      6.4583
    2.0000     2.6146     3.6641      7.7561
    3.0000     2.3550     3.8587      7.6479
    4.0000     2.3767     3.8425      7.6569
    5.0000     2.3749     3.8439      7.6562
    6.0000     2.3750     3.8437      7.6563
    7.0000     2.3750     3.8438      7.6562
    8.0000     2.3750     3.8437      7.6563
Gauss-Seidel method converged
```

Converges faster than the Jacobi method shown in next page

Jacobi Iteration

```
» A = [4 -1 -1; 6 8 0; -5 0 12];
» b = [-2 45 80];
» x=Jacobi(A,b,0.0001,100);
       i          x1          x2          x3       x4 ....
    1.0000    -0.5000     5.6250      6.6667
    2.0000     2.5729     6.0000      6.4583
    3.0000     2.6146     3.6953      7.7387
    4.0000     2.3585     3.6641      7.7561
    5.0000     2.3550     3.8561      7.6494
    6.0000     2.3764     3.8587      7.6479
    7.0000     2.3767     3.8427      7.6568
    8.0000     2.3749     3.8425      7.6569
    9.0000     2.3749     3.8438      7.6562
   10.0000     2.3750     3.8439      7.6562
   11.0000     2.3750     3.8437      7.6563
   12.0000     2.3750     3.8437      7.6563
   13.0000     2.3750     3.8438      7.6562
   14.0000     2.3750     3.8438      7.6562

Jacobi method converged
```