MATH2089
Numerical Methods
Lecture 5

Part 2: Systems of Linear Equations

Elimination Methods,

LU Factorization,

Iterative Methods,

Special Linear Systems

Solving Linear Systems

- These linear systems can be solved by either *direct* or *indirect* (*iterative*) methods

<u>Direct methods</u>	<u>Iterative methods</u>
Gauss elimination	*Jacobi*
Gauss-Jordan	*Gauss-Seidel*
LU decomposition	*Relaxation*

LU Decomposition

- A square matrix $[A]$ can be resolved into the product of a lower- and upper-triangular matrix in an infinite number of ways

$$[A]=[L][U]$$

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

- Forward substitution:

$$[L]\{d\}=\{b\} \quad \text{or} \quad \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

LU Decomposition (continue)

- Backward substitution:

$$[U]\{x\}=\{d\} \quad \text{or} \quad \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}$$

- Thus, $[A]\{x\}=[L]\underbrace{[U]\{x\}}_{\{d\}}=\{b\}$

LU Decomposition (continue)

LU Decomposition with Gauss Elimination (**Doolittle Decomposition**)

- $[U]$ is taken to be the upper-triangular matrix from Gauss elimination

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \xrightarrow{\text{elimination}} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} \\ 0 & 0 & a_{33}^{(2)} \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

- $[L]$ has one on its diagonals

$$\begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \left(a_{21}/a_{11}\right) & 1 & 0 \\ \left(a_{31}/a_{11}\right) & \left(a_{32}^{(1)}/a_{22}^{(1)}\right) & 1 \end{bmatrix}$$

LU Decomposition (continue)

Crout Decomposition

- In contrast to LU decomposition with Gauss elimination, $[U]$ has one on its diagonals

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & u_{13} \\ 0 & 1 & u_{23} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} l_{11} & (l_{11}u_{12}) & (l_{11}u_{13}) \\ l_{21} & (l_{21}u_{12}+l_{22}) & (l_{21}u_{13}+l_{22}u_{23}) \\ l_{31} & (l_{31}u_{12}+l_{32}) & (l_{31}u_{13}+l_{32}u_{23}+l_{33}) \end{bmatrix}$$

- Comparing the coefficients

1 $l_{11} = a_{11},\ l_{21} = a_{21},\ l_{31} = a_{31}$

2 $u_{12} = a_{12}/l_{11},\ u_{13} = a_{13}/l_{11}$

3 $l_{22} = a_{22} - l_{21}u_{12},\ l_{32} = a_{32} - l_{31}u_{12}$

4 $u_{23} = (a_{23} - l_{21}u_{13})/l_{22}$

5 $l_{33} = a_{33} - l_{31}u_{13} - l_{32}u_{23}$

$$\begin{array}{ccc} \boxed{\begin{matrix} l_{11} \\ l_{21} \\ l_{31} \end{matrix}} & \begin{matrix} 0 \\ \boxed{\begin{matrix} l_{22} \\ l_{32} \end{matrix}} \end{matrix} & \begin{matrix} 0 \\ 0 \\ \boxed{l_{33}} \end{matrix} \\ 1 & 3 & 5 \end{array} \qquad \begin{array}{ccc|c} 1 & \boxed{u_{12} \quad u_{13}} & & 2 \\ 0 & 1 & \boxed{u_{23}} & 4 \\ 0 & 0 & 1 & \end{array}$$

Example

- Perform the Crout decomposition on

$$[A]=\begin{bmatrix}3 & -1 & 2\\ 1 & 2 & 3\\ 2 & -2 & -1\end{bmatrix}$$

Solution

$$l_{11}=a_{11}=3,\ \ l_{21}=a_{21}=1,\ \ l_{31}=a_{31}=2$$

$$u_{12}=a_{12}/l_{11}=-1/3,\quad u_{13}=a_{13}/l_{11}=2/3$$

$$l_{22}=a_{22}-l_{21}u_{12}=2-(1)(-\tfrac{1}{3})=\tfrac{7}{3},\ \ l_{32}=a_{32}-l_{31}u_{12}=-2-(2)(-\tfrac{1}{3})=-\tfrac{4}{3}$$

$$u_{23}=(a_{23}-l_{21}u_{13})/l_{22}=(3-(1)(\tfrac{2}{3}))/\tfrac{7}{3}=1$$

$$l_{33}=a_{33}-l_{31}u_{13}-l_{32}u_{23}=-1-(2)(\tfrac{2}{3})-(-\tfrac{4}{3})(1)=-1$$

Example (continue)

- Lower-triangular

$$[L]=\begin{bmatrix} 3 & 0 & 0 \\ 1 & \frac{7}{3} & 0 \\ 2 & -\frac{4}{3} & -1 \end{bmatrix}$$

- Upper-triangular

$$[U]=\begin{bmatrix} 1 & -\frac{1}{3} & \frac{2}{3} \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

- Compact form

$$[L][U]=\begin{bmatrix} 3 & -\frac{1}{3} & \frac{2}{3} \\ 1 & \frac{7}{3} & 1 \\ 2 & -\frac{4}{3} & -1 \end{bmatrix}$$

Example (continue)

- For $\vec{b} = \begin{Bmatrix} 12 \\ 11 \\ 2 \end{Bmatrix} \rightarrow [L]\{d\} = \{b\} \rightarrow \begin{bmatrix} 3 & 0 & 0 \\ 1 & \frac{7}{3} & 0 \\ 2 & -\frac{4}{3} & -1 \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} 12 \\ 11 \\ 2 \end{Bmatrix} \rightarrow$

$$\vec{d} = \begin{Bmatrix} 4 \\ 3 \\ 2 \end{Bmatrix}$$

$$[U]\{x\} = \{d\} \rightarrow \begin{bmatrix} 1 & -\frac{1}{3} & \frac{2}{3} \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{Bmatrix} 4 \\ 3 \\ 2 \end{Bmatrix} \rightarrow \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{Bmatrix} 3 \\ 1 \\ 2 \end{Bmatrix}$$

Note!

- The LU decomposition is more efficient than the Gauss elimination method when [A] is the same, but with different {b}
- It computes the determinant of a matrix very efficiently

$$\det(A) = \det\left(L * U\right) = \det\left(L\right) * \det\left(U\right)$$

For Doolittle decomposition: $\det(L)$ = 1 then

$$\det(A) = \det\left(U\right) = u_{11} * u_{22} * u_{33}$$

For Crout decomposition: $\det(U)$ = 1 then

$$\det(A) = \det\left(L\right) = l_{11} * l_{22} * l_{33}$$

Note! (continue)

- This method is popular in programs to save storage space. The matrix $[L]$ and $[U]$ need not to be stored separately

  It can be stored as $\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ l_{21} & u_{22} & u_{23} \\ l_{31} & l_{32} & u_{33} \end{bmatrix} =$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ (a_{21}/a_{11}) & a_{22}^{(1)} & a_{23}^{(1)} \\ (a_{31}/a_{11}) & (a_{32}^{(1)}/a_{22}^{(1)}) & a_{33}^{(2)} \end{bmatrix}$$

Doolittle decomposition (Gauss elimination)

$$\begin{bmatrix} l_{11} & u_{12} & u_{13} \\ l_{21} & l_{22} & u_{23} \\ l_{31} & l_{32} & l_{33} \end{bmatrix}$$

Crout decomposition

Note! (continue)

- Because we can condense the $[L]$ and $[U]$ matrices into one array and store their elements in the space of $[A]$, this method is often called a *compact scheme*
- A special advantage of these LU methods is that we can accumulate the sums in double precision which is not easily done with the Gaussian elimination

Tri-Diagonal Systems

- This matrix consists of nonzero entries only on its diagonal and the positions adjacent to the main diagonal

$$\begin{bmatrix} B_1 & C_1 & 0 & 0 & 0 & 0 & 0 \\ A_2 & B_2 & C_2 & 0 & 0 & 0 & 0 \\ 0 & A_3 & B_3 & C_3 & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & A_{N-2} & B_{N-2} & C_{N-2} & 0 \\ 0 & 0 & 0 & 0 & A_{N-1} & B_{N-1} & C_{N-1} \\ 0 & 0 & 0 & 0 & 0 & A_N & B_N \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{N-2} \\ x_{N-1} \\ x_N \end{bmatrix} = \begin{bmatrix} D_1 \\ D_2 \\ D_3 \\ \vdots \\ D_{N-2} \\ D_{N-1} \\ D_N \end{bmatrix}$$

- Such tri-diagonal systems occur very frequently in applied problems. They are also diagonally dominant in which pivoting is unnecessary

Tri-Diagonal Systems (continue)

- Thomas developed the Gauss elimination technique for rapidly solving tri-diagonal systems that is now called the Thomas algorithm or tri-diagonal matrix algorithm (TDMA)
- The TDMA is actually a direct method for 1-D situations, but it can be applied iteratively, in a line-by-line fashion, to solve multi-dimensional problems and is widely used in computational fluid dynamics (CFD) programs

Tri-Diagonal Systems (continue)

- It consists of two processes:

Forward elimination $\rightarrow$

$$B_1^{(1)} = B_1,$$

$$B_i^{(1)} = B_i - \frac{A_i}{B_{i-1}^{(1)}} C_{i-1},$$

$$D_i^{(1)} = D_i - \frac{A_i}{B_{i-1}^{(1)}} D_{i-1}, \qquad i = 2,3,\ldots,N-1$$

Back substitution $\rightarrow$

$$x_N = \frac{D_N^{(1)}}{B_N^{(1)}},$$

$$x_i = \frac{D_i^{(1)} - C_i\, y_{i+1}}{B_i^{(1)}}, \qquad i = N-1, N-2,\ldots,2,1$$

Example

- Solve the matrix using the tri-diagonal matrix algorithm

$$\begin{bmatrix} -4 & 2 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 0 \\ 0 & 1 & -4 & 1 & 0 \\ 0 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 2 & -4 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{Bmatrix} = \begin{Bmatrix} 0 \\ -4 \\ -11 \\ 5 \\ 6 \end{Bmatrix}$$

Solution

Example (continue)

$$\left[\begin{array}{ccccc|c} -4 & 2 & 0 & 0 & 0 & 0 \\ 0 & -\frac{7}{2} & 1 & 0 & 0 & -4 \\ 0 & 0 & -\frac{26}{7} & 1 & 0 & -\frac{85}{7} \\ 0 & 0 & 0 & -\frac{97}{26} & 1 & \frac{45}{26} \\ 0 & 0 & 0 & 0 & -\frac{336}{97} & \frac{17472}{2522} \end{array}\right]$$

$$B_2^{(1)} = -4 - \tfrac{1}{-4}(2),\ D_2^{(1)} = -4 - \tfrac{1}{-4}(0) = -4$$

$$B_3^{(1)} = -4 - (-\tfrac{2}{7})(1),\ D_3^{(1)} = -11 - (-\tfrac{2}{7})(-4) = -\tfrac{85}{7}$$

$$B_4^{(1)} = -4 - (-\tfrac{7}{26})(1),\ D_4^{(1)} = 5 - (-\tfrac{7}{26})(-\tfrac{85}{7}) = \tfrac{45}{26}$$

$$B_5^{(1)} = -4 - (-\tfrac{26(2)}{97})(1),\ D_5^{(1)} = 6 - (-\tfrac{26(2)}{97})(\tfrac{45}{26}) = \tfrac{17472}{2522}$$

Forward elimination

$$\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 2 \\ 3 \\ -1 \\ -2 \end{Bmatrix}$$

Back substitution

Jacobi Iterative Method

- Iterative methods are preferred over the direct methods - when the coefficient matrices are sparse (have many zeros) they may be more rapid
- They may be more economical in memory requirements of a computer
- They can also be applied to sets of nonlinear equations
- When the system of equations can be ordered so that each diagonal entry of the coefficient matrix is larger in magnitude than the sum of the magnitude of the other coefficients in that row – such a system is called diagonally dominant which is a sufficient condition that the iteration will converge for any starting values

Jacobi Iterative Method (continue)

- Diagonally dominant condition:

$$|a_{ii}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|, \quad i = 1, 2, \ldots, n$$

- The iterative methods depend on the rearrangement of the equations in this manner

$$x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij} x_j \right); \quad i = 1, 2, \ldots, n \qquad \text{or}$$

$$\left\{ \begin{array}{l} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1; \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2; \\ \qquad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n. \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} x_1 = \left(b_1 - a_{12}x_2 - a_{13}x_3 - \cdots - a_{1n}x_n\right)/a_{11}; \\ x_2 = \left(b_2 - a_{21}x_1 - a_{23}x_3 - \cdots - a_{2n}x_n\right)/a_{22}; \\ \qquad \vdots \\ x_n = \left(b_n - a_{n1}x_1 - a_{n2}x_2 - \cdots - a_{n,n-1}x_{n-1}\right)/a_{nn}. \end{array} \right\}$$

Jacobi Iterative Method (continue)

- Begin with some initial approximation to the value of the variables

$$x_i^{(1)} \to \left(x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(1)}\right)$$

- Each component might be taken equal to zero if no better initial estimates are at hand

$$x_i^{(1)} = 0$$

- Substituting these approximations into the right-hand sides of the set of equations generates new approximations that, we hope, are closer to the true value

$$x_i^{(2)} = \frac{1}{a_{ii}}\left(b_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij} x_j^{(1)}\right)$$

Jacobi Iterative Method (continue)

- The new values are substituted in the right-hand sides to generate a second approximation, and the process is repeated until successive values of each of the variables are sufficiently alike

$$x_i^{(k+1`)} = \frac{1}{a_{ii}}\left(b_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij} x_j^{(k)} \right); k = 1, 2, \ldots$$

until

$$\left| x_i^{(k+1)} - x_i^{(k)} \right| \leq \varepsilon \quad \text{or} \quad \left| \frac{x_i^{(k+1)} - x_i^{(k)}}{x_i^{(k+1)}} \right| \leq \varepsilon$$

Example

- Solve the system of equations below using Jacobi iteration method

$$
\begin{aligned}
6x_1 - 2x_2 + x_3 &= 11 \\
x_1 + 2x_2 - 5x_3 &= -1 \\
-2x_1 + 7x_2 + 2x_3 &= 5
\end{aligned}
$$

Solution

Example (continue)

- Reorder the equations so that the coefficient matrix is diagonally dominant

$$\begin{aligned} 6x_1 - 2x_2 + x_3 &= 11 \\ -2x_1 + 7x_2 + 2x_3 &= 5 \\ x_1 + 2x_2 - 5x_3 &= -1 \end{aligned}$$

- Rearrangement of the equations to solve for the variables

$$\begin{aligned} x_1 &= \left(11 + 2x_2 - x_3\right)/6 \\ x_2 &= \left(5 + 2x_1 - 2x_3\right)/7 \\ x_3 &= \left(1 + x_1 + 2x_2\right)/5 \end{aligned}$$

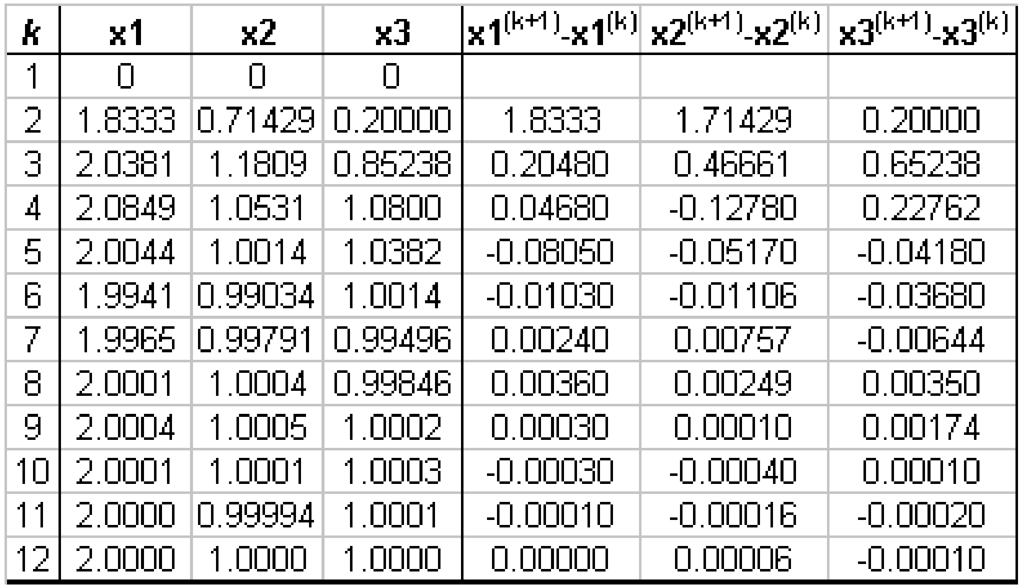
Example (continue)

- By putting superscripts on the variables to indicate successive iterates

$$
\begin{aligned}
x_1^{(k+1)} &= \left(11 + 2x_2^{(k)} - x_3^{(k)}\right)/6 \\
x_2^{(k+1)} &= \left(5 + 2x_1^{(k)} - 2x_3^{(k)}\right)/7 \\
x_3^{(k+1)} &= \left(1 + x_1^{(k)} + 2x_2^{(k)}\right)/5
\end{aligned}
$$

- Starting with an initial value $x_i^{(1)} = 0$

Example (continue)



k	x1	x2	x3	$x1^{(k+1)}-x1^{(k)}$	$x2^{(k+1)}-x2^{(k)}$	$x3^{(k+1)}-x3^{(k)}$
1	0	0	0			
2	1.8333	0.71429	0.20000	1.8333	1.71429	0.20000
3	2.0381	1.1809	0.85238	0.20480	0.46661	0.65238
4	2.0849	1.0531	1.0800	0.04680	-0.12780	0.22762
5	2.0044	1.0014	1.0382	-0.08050	-0.05170	-0.04180
6	1.9941	0.99034	1.0014	-0.01030	-0.01106	-0.03680
7	1.9965	0.99791	0.99496	0.00240	0.00757	-0.00644
8	2.0001	1.0004	0.99846	0.00360	0.00249	0.00350
9	2.0004	1.0005	1.0002	0.00030	0.00010	0.00174
10	2.0001	1.0001	1.0003	-0.00030	-0.00040	0.00010
11	2.0000	0.99994	1.0001	-0.00010	-0.00016	-0.00020
12	2.0000	1.0000	1.0000	0.00000	0.00006	-0.00010

- Note that this method is exactly the same as the method of fixed-point iteration for a single equation

Gauss-Siedel Iterative Method

- In the Jacobi method, even though we have computed the new $x_1^{(k+1)}$, we still do not use this value in computing new $x_2^{(k+1)}$, even though in nearly all cases the new values are better than the old, and should be use in preference to the poorer values
- When this is done, the procedure is known as the Gauss-Seidel iterative method
- To improve each x-value in turn, always use the most recent approximations to the values of the other variables

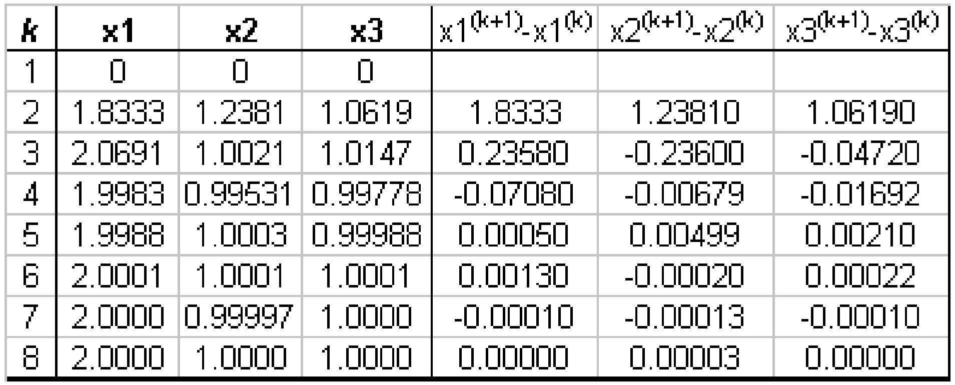
Gauss-Siedel Iterative Method (continue)

- Revisiting the previous example solved through the Jacobi method, more rapid convergence can be achieved:

$$x_1^{(k+1)} = \left(11 + 2x_2^{(k)} - x_3^{(k)}\right)/6$$

$$x_2^{(k+1)} = \left(5 + 2\boxed{x_1^{(k+1)}} - 2x_3^{(k)}\right)/7$$

$$x_3^{(k+1)} = \left(1 + \boxed{x_1^{(k+1)}} + 2\boxed{x_2^{(k+1)}}\right)/5$$



k	x1	x2	x3	$x1^{(k+1)}-x1^{(k)}$	$x2^{(k+1)}-x2^{(k)}$	$x3^{(k+1)}-x3^{(k)}$
1	0	0	0			
2	1.8333	1.2381	1.0619	1.8333	1.23810	1.06190
3	2.0691	1.0021	1.0147	0.23580	-0.23600	-0.04720
4	1.9983	0.99531	0.99778	-0.07080	-0.00679	-0.01692
5	1.9988	1.0003	0.99988	0.00050	0.00499	0.00210
6	2.0001	1.0001	1.0001	0.00130	-0.00020	0.00022
7	2.0000	0.99997	1.0000	-0.00010	-0.00013	-0.00010
8	2.0000	1.0000	1.0000	0.00000	0.00003	0.00000

Gauss-Siedel Iterative Method (continue)

- Standard form of the Gauss-Seidel method is

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}\right)$$

- Note that given row diagonal dominance in the coefficient matrix, the Gauss-Seidel method is often the better choice
- Note also that We may still prefer the Jacobi method if we are running the program on parallel processors because all n equations can be solved simultaneously at each iteration

Relaxation Method

- This is an iterative method that is more rapidly convergent than Gauss-Seidel method and that can be used to our advantage for hand calculations
- We should improve the x-value is most in error, because that variable does not appear on the right of the rearranged form, and hence its own error will not affect the next iterate
- The method of relaxation is a scheme that permits one to select the best equation to be used for maximum rate of convergence

Relaxation Method (continue)

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}\right) + (1-\omega)\, x_i^{(k)}$$

where ω (called relaxation factor) is a constant in the range $0 < \omega < 2$

$0 < \omega < 1 \quad \rightarrow$ Successive Under Relaxation method

$\omega = 1 \quad \rightarrow$ Gauss-Seidel method

$1 < \omega < 2 \quad \rightarrow$ Successive Over Relaxation method

However, the optimum value of ω is not known, it is usually determined by a trial-and-error process

Example

- Find the solution of the following equations using the relaxation method. Also find the optimum value of the relaxation factor, ω.

$$
\begin{aligned}
-5x_1 - x_2 + 2x_3 &= 1 \\
2x_1 + x_2 + 7x_3 &= 32 \\
2x_1 + 6x_2 - 3x_3 &= 2
\end{aligned}
$$

Solution

Example (continue)

- Reorder the equations so that the coefficient matrix is diagonally dominant

$$\begin{aligned} -5x_1 - \;\; x_2 + 2x_3 &= 1 \\ 2x_1 + 6x_2 - 3x_3 &= 2 \\ 2x_1 + \;\; x_2 + 7x_3 &= 32 \end{aligned}$$

- Rearrangement of the equations with the relaxation factor

$$\begin{aligned} x_1^{k+1} &= \omega\left(-1 - x_2^k + 2x_3^k\right)\big/5 + (1-\omega)\,x_1^k \\ x_2^{k+1} &= \omega\left(2 - 2x_1^{k+1} + 3x_3^k\right)\big/6 + (1-\omega)\,x_2^k \\ x_3^{k+1} &= \omega\left(32 - 2x_1^{k+1} - x_2^{k+1}\right)\big/7 + (1-\omega)\,x_3^k \end{aligned}$$

Example (continue)

- Starting with an initial value $x_i^{(1)} = 0$ with

$$\left| \frac{x_i^{(k+1)} - x_i^{(k)}}{x_i^{(k+1)}} \right| \leq 0.01$$

For ω = 0.9

k	x1	x2	x3
1	-0.180000	0.354000	4.115057
2	1.219701	1.821266	3.977991
3	1.046219	1.958357	3.991268
4	1.008974	1.989214	3.998206
5	1.002193	1.997456	3.999584
6	1.000527	1.999400	3.999900
7	1.000125	1.999858	3.999976
8	**1.000030**	**1.999966**	**3.999994**

Example (continue)

		Converged solution obtained		
ω	Iteration	x1	x2	x3
0.1	70	1.001418	1.998299	4.000718
0.2	41	1.000542	1.999447	4.000010
0.3	29	1.000273	1.999676	3.999961
0.4	22	1.000177	1.999768	3.999964
0.5	18	1.000093	1.999868	3.999978
0.6	15	1.000053	1.999924	3.999988
0.7	12	1.000053	1.999927	3.999988
0.8	10	1.000035	1.999955	3.999993
0.9	**8**	**1.000030**	**1.999966**	**3.999994**
1.0	9	0.999986	1.999994	4.000005
1.1	14	1.000020	2.000008	3.999990
1.2	22	1.000029	2.000010	3.999982
1.3	40	1.000038	2.000012	3.999973
1.4	130	1.000046	2.000012	3.999963
1.5	solution diverge			

Optimum under relaxation factor is about ω = 0.9 for this example

Special Systems of Linear Equations

- If the number of equations (m) is larger than the number of unknowns (n), the matrix $[A]$ becomes $m \times n$ matrix where $m > n$ and the system of equations is called over-determined system of equations
- If the number of equations (m) is smaller than the number of unknowns (n), the matrix $[A]$ becomes $m \times n$ matrix where $m < n$ and the system of equations is called underdetermined system of equations
- For over-determined system of equations there is no exact solution so we define an error vector and minimize this error vector by using a suitable criterion

Special Systems of Linear Equations (continue)

- Very common technique is minimizing the sum of squares of the components of the error vector (least squares method)
- We will look at this method as part of curve fitting techniques
- For under-determined system of equations we select basic and independent variables
- For example, one equation with two unknowns

$$5x_1 - x_2 = -1$$

can be rearranged as

$$x_1 = (x_2 - 1)/5$$

Special Systems of Linear Equations (continue)

so that x_2 is independent and x_1 is a basic variable which can be determined for any value of x_1

- In general case for m linearly independent equations with n unknowns ($m < n$) m variables can be selected as basic ones and determined using remaining (n - m) independent variables