Chapter 7 Methods of Solving Sets of Algebraic Equations

OUTLINE

7-1 General Remarks (1)

- The transport problem governed by a single or a set of differential equations and the boundary conditions can be approximated by a system of algebraic equations.
- If the resulting system is linear and the algebraic equations are not so many, they can readily be solved by using any one of the standard computer subroutines for solving system of algebraic equations.
- If the number of equations to be solved is very large and/or the equations are nonlinear, one needs to examine the nature of the resulting system of equations.

7-1 General Remarks (2)

- The proper choice of the computer subroutine for solving sets of algebraic equations is strongly affected by the following considerations:

  (i) Whether the problem is linear or nonlinear,

  (ii) Whether the coefficient matrix is tridiagonal, full or sparse(i.e., large percentage of entries are zero),

  (iii) Whether the number of operations involved in the algorithm is so large as to give rise to excessive accumulation of round-off errors,

  (iv) Whether the coefficient matrix is "diagonally dominant",

  (v) Whether the coefficient matrix is ill-conditioned (i.e., small changes in the coefficients, such as those introduced by round-off errors produce large changes in the solution).

7-2 Reduction to Algebraic Equations (1)

- A large variety of finite difference schemes is available for discretizing the derivatives in differential equations; the choice depends on the nature of the governing differential equation and its boundary conditions.
- Here our objective is to illustrate the basic steps in the transformation of a differential equation and its boundary condition into a set of algebraic equations.
- Consider the following simple example:

  (1) Energy is generated in a slab of thickness L at a rate of g(x) W/m^3, while it is dissipated from the boundary surfaces at x=0 and x=L by convection into ambients at temperatures $T_{\infty,0}$ and $T_{\infty,L}$, with heat transfer coefficient h_0 and h_L, respectively.

7-2 Reduction to Algebraic Equations (2)

(2) The mathematical formulation of this problem for the steady state is given by

$$\frac{d^2T(x)}{dx^2} + \frac{1}{k}g(x) = 0 \qquad in \qquad 0 < x < L \ldots \quad (4-1a)$$

$$-k\frac{dT(x)}{dx} + h_0T(x) = h_0T_{\infty,0} \qquad at \qquad x = 0 \ldots \quad (4-1b)$$

$$k\frac{dT(x)}{dx} + h_LT(x) = h_LT_{\infty,L} \qquad at \qquad x = L \ldots \quad (4-1c)$$

(3) The basic steps in the transformation of this problem by finite differences into sets of algebraic equations for the temperatures T_i at a finite number of grid points i=1,2,…,M, chosen over the solution domain of the problem as follows:

7-2 Reduction to Algebraic Equations (3)

(i) The domain $0 \le x \le L$ is divided into M equal subregions each of thickness $\Delta x = L/M$.(Grids of unequal size can also be used)

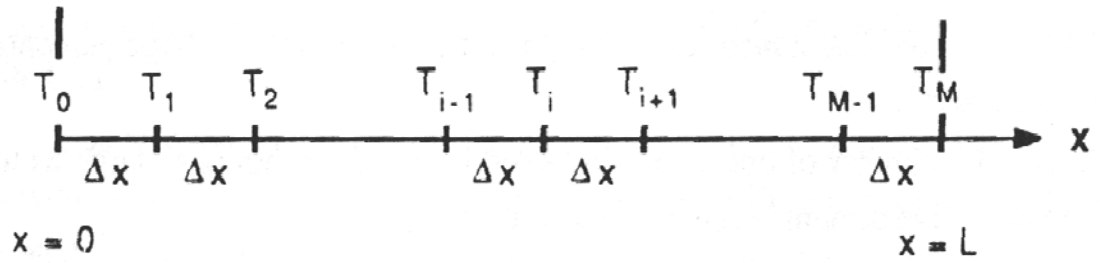


Figure 3-1. One-dimensional finite-difference grid

7-2 Reduction to Algebraic Equations (4)

(ii) (a) The differential equation (4-1a) is discretized by a suitable finite difference scheme at the internal grid points i=1,2,…,M-1. By using the classical second-order accurate central-difference formula to discretize the second derivative, eq. (4-1a) reduces to

$$\frac{T_{i-1}-2T_i+T_{i+1}}{(\Delta x)^2}+\frac{1}{k}g_i=0$$

with a truncation error $o\left[(\Delta x)^2\right]$

This result is rearranged in the form

$$T_{i-1}-2T_i+T_{i+1}+G_i=0 \qquad i=1,2,3,\ldots,(M-1)\ldots \qquad (4-2a)$$

$$G_i=\frac{(\Delta x)^2 g_i}{k}\ldots \qquad (4-2b)$$

7-2 Reduction to Algebraic Equations (5)

(b) The system provides M-1 algebraic equations, but it contains M+1 unknown grid-point temperatures $T_0, T_1,\ldots,T_M$. Two additional relations needed for making the number of equations equal to the number of unknowns are obtained by discretizing the boundary conditions.

(iii) (a) The boundary conditions given by eqs. (4-1b) and (4-1c) need to be discretized because they contain the first derivative of temperature. If the forward and backward differencing formula are used, the results are first-order accurate. It is desirable to use a second-order accurate formula in order to be consistent with the second-order accuracy of the discretized differential equation.

7-2 Reduction to Algebraic Equations (6)

(b) To obtain a second-order accurate formula for the first-derivative, it is need to implement this formula at the boundary grid points i=0 and i=M. Additional grid-points are needed to the left and to the right of the boundary nodes i=0 and i=M, respectively. Therefore, fictitious nodes located at a distance Δx to the left and right of the boundaries at x=0 and x=L, at fictitious temperatures T_{-1} and T_{M+1}, respectively, are considered.

Figure 3-2 Fictitious nodes at fictitious temperatures T_{-1} and T_{M+1}.

7-2 Reduction to Algebraic Equations (7)

(c) Applying the central-difference formula to eqs. (4-1b) and (4-1c), we can get

$$-k\frac{T_1 - T_{-1}}{2\Delta x} + h_0 T_0 = h_0 T_{\infty,0} \ldots \quad (4-3a)$$

$$k\frac{T_{M+1} - T_{M-1}}{2\Delta x} + h_L T_M = h_L T_{\infty,L} \ldots \quad (4-3b)$$

To eliminate the fictitious temperatures T_{-1} and T_{M+1}, two additional relations obtained by evaluating eq. (4-2a) for i=0 and i=M, to give, respectively,

$$T_{-1} - 2T_0 + T_1 + \frac{(\Delta x)^2 g_0}{k} = 0 \ldots \quad (4-4a)$$

$$T_{M-1} - 2T_M + T_{M+1} + \frac{(\Delta x)^2 g_M}{k} = 0 \ldots \quad (4-4b)$$

7-2 Reduction to Algebraic Equations (8)

(d) Combing eqs. (4-3) with eqs. (4-4), we have

$$2T_{-1} - 2\beta_0 T_0 + (2\gamma_0 + G_0) = 0 \quad at \quad x = 0 \quad (i = 0)... \quad (4-5a)$$

$$2T_{M-1} - 2\beta_L T_M + (2\gamma_L + G_M) = 0 \quad at \quad x = L \quad (i = M)... \quad (4-5b)$$

where

$$\beta_0 = 1 + \frac{\Delta x h_0}{k}, \quad \gamma_0 = \frac{\Delta x (h_0 T_{\infty,0})}{k}$$

$$\beta_L = 1 + \frac{\Delta x h_L}{k}, \quad \gamma_L = \frac{\Delta x (h_L T_{\infty,L})}{k}$$

$$G_0 = \frac{(\Delta x)^2 g_0}{k}, \quad G_M = \frac{(\Delta x)^2 g_M}{k}$$

7-2 Reduction to Algebraic Equations (9)

(e) Equations (6-2)~(6-5) provide M+1 algebraic equations for the determination of M+1 unknown node temperatures T_i, (i=0,1,2,…,M). These equations are summarized below

$$2T_1 - 2\beta_0 T_0 = -(2\gamma_0 + G_0), \qquad i = 0 \qquad (4-6a)$$

$$T_{i-1} - 2T_i + T_{i+1} = -G_i, \qquad i = 1,2,\ldots,M-1 \qquad (4-6b)$$

$$2T_{M-1} - 2\beta_L T_L = -(2\gamma_L + G_M), \qquad i = M \qquad (4-6c)$$

7-2 Reduction to Algebraic Equations (10)

(iv) (a) The sets of equations (4-6a)~(4-6c) are expressed in the matrix form [A] {T}={B}, where

$$[A]=\begin{bmatrix} -2\beta_0 & 2 & 0 & \cdots & 0 & 0 & 0 \\ 1 & -2 & 1 & & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & 0 \\ \cdot & & & & & & \\ 0 & & & & 1 & -2 & 1 \\ 0 & 0 & \cdots & \cdots & 0 & 2 & -2\beta_L \end{bmatrix} = \text{known coefficient matrix}$$

$$\{T\}=\begin{Bmatrix} T_0 \\ T_1 \\ \cdot \\ \cdot \\ T_M \end{Bmatrix} = unknown \quad vector, \quad \{B\}=\begin{Bmatrix} -(G_0+2\gamma_0) \\ -G_1 \\ \cdot \\ \cdot \\ -(G_M+2\gamma_L) \end{Bmatrix} = known \quad vector$$

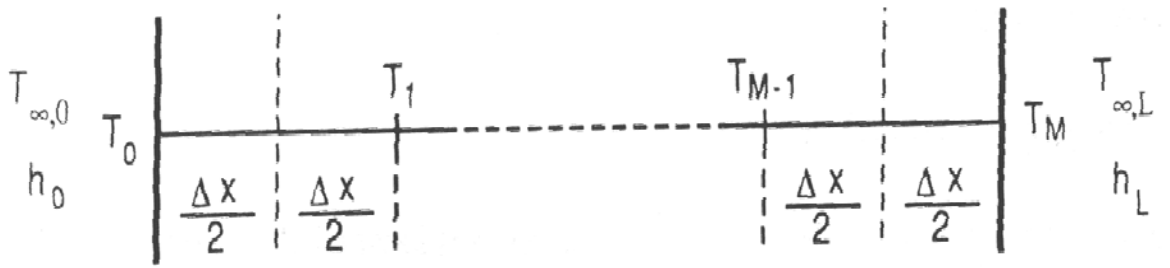
7-2 Reduction to Algebraic Equations (11)

(b) For the one-dimensional problem considered here, the coefficient matrix [A] is a tridiagonal matrix. Depending on the nature of the problem, the dimensions and the finite-difference scheme used, a multidiagonal, a full or a sparse matrix may result.

7-2 Reduction to Algebraic Equations (12)

(4) Control-volume approach:

(a) In the above illustration we used fictitious nodes in order to develop a second-order accurate finite difference scheme to discretize the boundary conditions. The same equation can also be developed by applying the control-volume approach for a volume element about the boundary node.



7-2 Reduction to Algebraic Equations (13)

(b) Consider control volumes of thickness $\Delta x/2$ next to the boundary surfaces at x=0 and x=L. The steady-state energy conservation principle for each of these control volumes is stated as

(Rate of heat gain by convection)+

(Rate of heat gain by conduction)+

(Rate of energy generation)=0

7-2 Reduction to Algebraic Equations (14)

(c) The application of this conservation equation for the boundary nodes about i=0 and i=M, respectively, gives

$$h_0\left(T_{\infty,0}-T_0\right)+k\frac{T_1-T_0}{\Delta x}+\frac{\Delta x}{2}g_0=0 \quad at \quad x=0 \quad (i.e.,i=0)$$

$$h_L\left(T_{\infty,L}-T_M\right)+k\frac{T_{M-1}-T_M}{\Delta x}+\frac{\Delta x}{2}g_M=0 \quad at \quad x=L \quad (i.e.,i=M)$$

These results are rearranged as

$$2T_1-\left(2+\frac{2\Delta x h_0}{k}\right)T_0+\left[\frac{2\Delta x h_0}{k}T_{\infty,0}+\frac{(\Delta x)^2 g_0}{k}\right]=0$$

$$2T_{M-1}-\left(2+\frac{2\Delta x h_L}{k}\right)T_M+\left[\frac{2\Delta x h_L}{k}T_{\infty,L}+\frac{(\Delta x)^2 g_M}{k}\right]=0$$

7-2 Reduction to Algebraic Equations (15)

(5) So far we illustrated the basic steps in the transformation of a PDE and its B.C.s into a system of algebraic equations. The methods of solving such a system of algebraic equations can be put into one of the two categories:

(i) The direct methods.

(ii) The iterative techniques.

7-3 Direct Methods (1)

- Generally, the direct methods are preferred for systems having banded matrix coefficients and for problems involving relatively simple geometries and B.C.s. They are very efficient, but require large computer storage and give rise to the accumulation of round-off error if the number of equations is large.
- Gauss Elimination Method:

(1) This is a direct method commonly used for solving simultaneous algebraic equations. In this method, the coefficient matrix is transformed into an upper triangular matrix by systematic application of some algebraic operations under which the solution to the system of equations remains invariant. Two principal operations applied include:

(i) Multiplication or division of any equations by a constant.

(ii) Replacement of any equation by the sum (or difference) of that equation with any other equation.

7-3 Direct Methods (2)

(2) Once the system is transformed into upper diagonal form, the solution starts from the last equation and proceeds upwards by back substitutions.

(3) Example:

(i) Consider a simple example involving three unknowns T_1, T_2, and T_3.

$$a_{11}T_1+a_{12}T_2+a_{13}T_3=d_1$$
$$a_{21}T_1+a_{22}T_2+a_{23}T_3=d_2$$
$$a_{31}T_1+a_{32}T_2+a_{33}T_3=d_3$$

7-3 Direct Methods (3)

(ii) We choose the first equation as the "pivot" equation and use it to eliminate T_1 from the second and third equations. We obtain

$$a_{11}T_1 + a_{12}T_2 + a_{13}T_3 = d_1$$
$$0 + a_{22}^* T_2 + a_{23}^* T_3 = d_2^*$$
$$0 + a_{32}^* T_2 + a_{33}^* T_3 = d_3^*$$

To eliminate T_2 from the third equation, the second equation is used as the "pivot" equation. The result is

$$a_{11}T_1 + a_{12}T_2 + a_{13}T_3 = d_1$$
$$0 + a_{22}^{'} T_2 + a_{23}^{'} T_3 = d_2^{'}$$
$$0 + a_{33}^{'} T_3 = d_3^{'}$$

7-3 Direct Methods (4)

(iii) The unknowns T_i are immediately determined from this system by starting from the last equation and by back substitution. We obtain

$$T_3 = \frac{d_3'}{a_{33}'} \qquad (4-3a)$$

$$T_2 = \frac{\left(d_2' - a_{23}' T_3\right)}{a_{22}'} \qquad (4-3b)$$

$$T_1 = \left(d_1 - a_{13} T_3 - a_{12} T_2\right) / a_{11} \qquad (4-3c)$$

The above procedure can be readily generalized to a system of N equations. The number of multiplications involved in the solution of a system of N algebraic equations with a full matrix by using Gauss elimination varies as N^3.

7-4 Thomas Algorithm (1)

- In the case of a tridiagonal system of algebraic equations, such as the one encountered in the solution of 1-D heat conduction problems, the Gauss elimination method can be further simplified by taking advantage of the zeros of the tridiagonal coefficient matrix. This modified procedure, generally referred to as Thomas Algorithm, is an extremely efficient method for solving large number of such equations
- Consider a system of N algebraic equations having a tridiagonal coefficient matrix given as follows:

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & \cdots & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & & 0 & 0 \\ \cdot & & & & & & \cdot \\ \cdot & & & & & & \cdot \\ \cdot & & & & & & \cdot \\ 0 & 0 & & & a_{N-1} & b_{N-1} & c_{N-1} \\ 0 & 0 & 0 & \cdots & 0 & a_N & b_N \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ \cdot \\ \cdot \\ \cdot \\ T_{N-1} \\ T_N \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \cdot \\ \cdot \\ \cdot \\ d_{N-1} \\ d_N \end{bmatrix} \quad \ldots(4\text{-}4)$$

7-4 Thomas Algorithm (2)

- Procedure of Thomas Algorithm:

(1) The first row is chosen as the "pivot", multiplied by "a_2/b_1" and substracted from the second row to eliminate a_2. The resulting second equation is equivalent to

replacing "b_2" by (b_2-a_2/b_1*c_1)

replacing "d_2" by (d_2-a_2/b_1*d_1)

(2) The modified second equation is chosen as the "pivot", a similar approach is followed to eliminate a_3. The resulting third equation is equivalent to

replacing "b_3" by (b_2-a_3/b_2*c_2)

replacing "d_3" by (d_3-a_3/b_2*d_2)

7-4 Thomas Algorithm (3)

(3) The procedure is continued until a_N is eliminated from the last equation. Thus the general procedure for upper diagonalizing eq. (4-4) is stated as

replacing "b_i" by (b_i-a_i/b_{i-1}*c_{i-1}) for I=2,3,…,N

replacing "d_i" by (d_i-a_i/b_{i-1}*d_{i-1}) for I=2,3,…,N

(4) Once the tridiagonal form is achieved by the above procedure, the unknown T_i's are determined by back substitution, starting from the last equation and working backwards

$$T_N = \frac{d_N}{b_N}$$

$$T_i = \frac{d_i - c_i T_{i+1}}{b_i}, \quad i = N-1, N-2, ..., 1$$

7-4 Thomas Algorithm (4)

(5) Using the Thomas algorithm, the number of basic arithmetic operations for solving a tridiagonal set of the order N, in contrast to $O(N^3)$ operations required for solving with Gauss elimination. Therefore, not only are the computation times much shorter, but the round-off errors also are significantly reduced.

(6) Example:

$$\begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} T_0 \\ T_1 \\ T_2 \\ T_3 \end{bmatrix} = \begin{bmatrix} -40 \\ -30 \\ -30 \\ -30 \end{bmatrix}$$

7-4 Thomas Algorithm (5)

Solution. The upper diagonalization procedure defined by Eqs. (3-21a,b) gives

$$b_i\text{'s:}\begin{cases} b_1 = -1 \\ b_2 \equiv b_2 - \dfrac{a_2}{b_1}c_1 = -2 - \dfrac{(1)}{(-1)}(1) = -1 \\ b_3 \equiv b_3 - \dfrac{a_3}{b_2}c_2 = -2 - \dfrac{(1)}{(-1)}(1) = -1 \\ b_4 \equiv b_4 - \dfrac{a_4}{b_3}c_3 = -2 - \dfrac{(1)}{(-1)}(1) = -1 \end{cases}$$

and

$$d_i\text{'s:}\begin{cases} d_1 = -40 \\ d_2 \equiv d_2 - \dfrac{a_2}{b_1}d_1 = -30 - \dfrac{(1)}{(-1)}(-40) = -70 \\ d_3 \equiv d_3 - \dfrac{a_3}{b_2}d_2 = -30 - \dfrac{(1)}{(-1)}(-70) = -100 \\ d_4 \equiv d_4 - \dfrac{a_4}{b_3}d_3 = -30 - \dfrac{(1)}{(-1)}(-100) = -130 \end{cases}$$

The back substitution defined by Eqs. (3-21) and (3-22), give the four temperatures as

$$T_i\text{'s:}\begin{cases} T_0 = \dfrac{d_4}{b_4} = \dfrac{-130}{-1} = 130 \\ T_1 = \dfrac{d_3 - c_3T_4}{b_3} = \dfrac{-100 - (1)(130)}{(-1)} = 230 \\ T_2 = \dfrac{d_2 - c_2T_3}{b_2} = \dfrac{-70 - (1)(230)}{(-1)} = 300 \\ T_3 = \dfrac{d_1 - c_1T_2}{b_1} = \dfrac{-40 - (1)(300)}{(-1)} = 340 \end{cases}$$

7-5 Iterative Methods (1)

- When the number of equations is very large, the coefficient matrix is sparse but not banded and the computer storage is critical, an iterative method is preferred to the direct method of solution.
- If the iterative process is convergent, the solution is obtained within a specified accuracy of the exact answer in a finite but not predeterminable number of operations. The method is certain to convergence for a system having diagonal dominance.
- Iterative methods have rather simple algorithms, are easy to apply and are not restricted for use with simple geometries and B.C.s. They are also preferred when the number of operations in the calculations is so large that the direct methods may prove inadequate because of the accumulation of round-off errors.

7-5 Iterative Methods (2)

- **Gauss-Seidel iteration:**

  (1) This is a very simple, efficient point-iterative procedure for solving large, sparse systems of algebraic equations. Basic steps are as follows:

  (i) Solve each equation for the main diagonal unknowns.

  (ii) Make an initial guess for all unknowns.

  (iii) Computations begin with the use of the guessed values to compute a first approximation for each of the main diagonal unknowns solved successively in step (i). In each computation, whenever possible, the most recently computed values are used and the first-round of iterations are completed.

  (iv) The value determined from the first-round of iterations and, whenever possible, the most recently computed values are used to complete the second-round of iterations.

  (v) The procedure is continued until a specified convergence criteria is satisfied for all unknowns.

7-5 Iterative Methods (3)

(2) Consider the following three equations:

$$a_{11}T_1+a_{12}T_2+a_{13}T_3=d_1 \quad \text{...(4-5a)}$$

$$a_{21}T_1+a_{22}T_2+a_{23}T_3=d_2 \quad \text{...(4-5b)}$$

$$a_{31}T_1+a_{32}T_2+a_{33}T_3=d_3 \quad \text{...(4-5c)}$$

where

$$a_{ii} \neq 0 \quad for \quad i=1 \quad to \quad 3$$

Equations are successively solved for the main diagonal unknowns

$$T_1=(d_1-a_{12}T_2-a_{13}T_3)/a_{11}$$

$$T_2=(d_2-a_{21}T_2-a_{23}T_3)/a_{22}$$

$$T_3=(d_3-a_{31}T_2-a_{32}T_3)/a_{33}$$

Initial guess are chosen as

$$T_1^{(0)}, T_2^{(0)}, T_3^{(0)}$$

7-5 Iterative Methods (4)

(3) These guessed values are used together with the most recently computed values to complete the first-round of iterations as

$$T_1^{(1)} = \frac{1}{a_{11}}\left(d_1 - a_{12}T_2^{(0)} - a_{13}T_3^{(0)}\right)$$

$$T_2^{(1)} = \frac{1}{a_{22}}\left(d_2 - a_{21}T_1^{(1)} - a_{23}T_3^{(0)}\right)$$

$$T_3^{(1)} = \frac{1}{a_{33}}\left(d_3 - a_{31}T_1^{(1)} - a_{32}T_2^{(1)}\right)$$

7-5 Iterative Methods (5)

(4) These first-approximations are used together with the most recently computed values to complete the second-round of iterations as

$$T_1^{(2)} = \frac{1}{a_{11}}\left(d_1 - a_{12}T_2^{(1)} - a_{13}T_3^{(1)}\right)$$

$$T_2^{(2)} = \frac{1}{a_{22}}\left(d_2 - a_{21}T_1^{(2)} - a_{23}T_3^{(1)}\right)$$

$$T_3^{(2)} = \frac{1}{a_{33}}\left(d_3 - a_{31}T_1^{(2)} - a_{32}T_2^{(2)}\right)$$

The iteration procedure is continued in a similar manner.

7-5 Iterative Methods (6)

(5) A general expression for the "n+1" th-round of iterations of the above system is written as

$$T_1^{(n+1)} = \frac{1}{a_{11}}\left(d_1 - a_{12}T_2^{(n)} - a_{13}T_3^{(n)}\right)$$

$$T_2^{(n+1)} = \frac{1}{a_{22}}\left(d_2 - a_{21}T_1^{(n+1)} - a_{23}T_3^{(n)}\right)$$

$$T_3^{(n+1)} = \frac{1}{a_{33}}\left(d_3 - a_{31}T_1^{(n+1)} - a_{32}T_2^{(n+1)}\right)$$

(6) In the general case of M equations, the "n+1" th-round of iterations can be written as

$$T_i^{(n+1)} = \frac{1}{a_{ii}}\left\{d_i - \sum_{j=1}^{i-1} a_{ij}T_j^{(n+1)} - \sum_{j=i+1}^{M} a_{ij}T_j^{(n)}\right\} \quad for \quad i=1 \quad to \quad M$$

7-5 Iterative Methods (7)

(7) The criterion for convergence can be specified either as the absolute convergence criterion in the form

$$\left| T_i^{(n+1)} - T_i^{(n)} \right| \leq \varepsilon$$

or as the relative convergence criterion in the form

$$\left| \frac{T_i^{(n+1)} - T_i^{(n)}}{T_i^{(n+1)}} \right| \leq \varepsilon$$

which should be satisfied for all T_i.

7-5 Iterative Methods (8)

(8) Convergence with iterative method:

(a) The convergence with iterative method does not depend on the initial guess for the unknowns, but it depends on the character of the coefficient matrix.

(b) For a convergent system, a good first guess for the unknowns significantly reduces the number of iterations for the specified convergence criterion to be satisfied.

(c) The system of equations in which the diagonal elements are the largest elements (in magnitude) in each row are the best suited for iterative solution.

(d) In situations when this is not the case, equations may be rearranged in order to bring the largest element in each row on the diagonal, if possible.

7-5 Iterative Methods (9)

(e) In most heat transfer problems, the diagonal elements of the difference equations happen to be the largest element in each row.

(f) A sufficient condition for convergence is given by

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ i \neq j}}^{M} |a_{ij}| \qquad for \qquad i = 1,2,..., n$$

$$\text{or } |a_{ii}| \geq \sum_{\substack{j=1 \\ i \neq j}}^{M} |a_{ij}| \qquad for \qquad i = 1,2,..., n \quad \text{for at least one (i.e., row)}$$

However, in practice, convergence may be obtained when this condition is not fully met.

7-5 Iterative Methods (10)

(9) Example:

$$6T_1 + T_2 + 3T_3 = 17$$
$$T_1 - 10T_2 + 4T_3 = -7$$
$$T_1 + T_2 + 3T_3 = 12$$

Solution. We note that in each equation the largest element (in magnitude) is in the diagonal. These equations are solved for the main-diagonal unknowns as

$$T_1 = \frac{1}{6}(17 - T_2 - 3T_3)$$
$$T_2 = \frac{1}{10}(7 + T_1 + 4T_3)$$
$$T_3 = \frac{1}{3}(12 - T_1 - T_2)$$

and the initial guess values are arbitrarily chosen as

$$T_1^{(0)} = T_2^{(0)} = T_3^{(0)} = 1$$

The *first-round* of iterations are determined as

$$T_1^{(1)} = \frac{1}{6}[17 - T_2^{(0)} - 3T_3^{(0)}] = 2.167$$
$$T_2^{(1)} = \frac{1}{10}[7 + T_1^{(1)} + 4T_3^{(0)}] = 1.317$$
$$T_3^{(1)} = \frac{1}{3}[12 - T_1^{(1)} - T_2^{(1)}] = 2.839$$

The *second-round* of iterations become

$$T_1^{(2)} = \frac{1}{6}[17 - T_2^{(1)} - 3T_3^{(1)}] = 1.194$$
$$T_2^{(2)} = \frac{1}{10}[7 + T_1^{(2)} + 4T_3^{(1)}] = 1.955$$
$$T_3^{(2)} = \frac{1}{3}[12 - T_1^{(2)} - T_2^{(2)}] = 2.950$$

and the *third-round* of iterations give

$$T_1^{(3)} = \frac{1}{6}[17 - T_2^{(2)} - 3T_3^{(2)}] = 1.032$$
$$T_2^{(3)} = \frac{1}{10}[7 + T_1^{(3)} + 4T_3^{(2)}] = 1.999$$
$$T_3^{(3)} = \frac{1}{3}[12 - T_1^{(3)} - T_2^{(3)}] = 2.989$$

The values obtained with three iterations are sufficiently close to the exact answer $T_1 = 1$, $T_2 = 2$ and $T_3 = 3$. The solution is converging to the exact results.

7-5 Iterative Methods (11)

- Successive Over-Relaxation (SOR):

(1) The Gauss-Seidel method generally does not converge sufficiently fast. Successive over-relaxation is a method that can accelerate the convergence.

(2) The basic idea in this approach is

$$T_1^{(n+1)} = T_1^{(n)} + \left[\frac{1}{a_{11}}\left(d_1 - a_{11}T_1^{(n)} - a_{12}T_2^{(n)} - a_{13}T_3^{(n)}\right)\right]$$

$$T_2^{(n+1)} = T_2^{n} + \left[\frac{1}{a_{22}}\left(d_2 - a_{21}T_1^{(n+1)} - a_{22}T_2^{(n)} - a_{23}T_3^{(n)}\right)\right]$$

$$T_3^{(n+1)} = T_3^{(n)} + \left[\frac{1}{a_{33}}\left(d_3 - a_{31}T_1^{(n+1)} - a_{32}T_2^{(n+1)} - a_{33}T_3^{(n)}\right)\right]$$

7-5 Iterative Methods (12)

(3) As the exact solution is approached, $T_i^{(n+1)}$ approaches $T_i^{(n)}$ and the terms inside the brackets become zero identically. Therefore, the terms inside the square brackets can be regarded as correction terms to $T_i^{(n)}$ (i=1,2,3), for each iteration.

(4) In the SOR method the bracketed terms are multiplied by a factor ω, called the relaxation parameter and the equations are rewritten as:

$$T_1^{(n+1)} = T_1^{(n)} + \frac{\omega}{a_{11}}\left(d_1 - a_{11}T_1^{(n)} - a_{12}T_2^{(n)} - a_{13}T_3^{(n)}\right)$$

$$T_2^{(n+1)} = T_2^{n} + \frac{\omega}{a_{22}}\left(d_2 - a_{21}T_1^{(n+1)} - a_{22}T_2^{(n)} - a_{23}T_3^{(n)}\right)$$

$$T_3^{(n+1)} = T_3^{(n)} + \frac{\omega}{a_{33}}\left(d_3 - a_{31}T_1^{(n+1)} - a_{32}T_2^{(n+1)} - a_{33}T_3^{(n)}\right)$$

7-5 Iterative Methods (13)

(5) The values of the relaxation parameter ω must lie in the range $0< \omega <2$ over-relaxation and $\omega=1$ to Gauss-Seidel iteration.

(6) The above procedure for SOR can be generalized for the case of M equations as

$$T_i^{(n+1)} = T_i^{(n)} + \frac{\omega}{a_{ii}}\left\{ d_i - \sum_{j=1}^{i-1} a_{ij} T_j^{(n+1)} - \sum_{j=i}^{M} a_{ij} T_j^{(n)} \right\} \qquad for \quad i=1 \quad to \quad M$$

which is rearranged as

$$T_i^{(n+1)} = (1-\omega) T_i^{(n)} + \frac{\omega}{a_{ii}}\left\{ d_i - \sum_{j=1}^{i-1} a_{ij} T_j^{(n+1)} - \sum_{j=i+1}^{M} a_{ij} T_j^{(n)} \right\} \qquad for \quad i=1 \quad to \quad M$$

7-5 Iterative Methods (14)

- Residue

$$R_i^{(n+1)} = \left\{ d_i - \sum_{j=1}^{i-1} a_{ij} T_j^{(n+1)} - \sum_{j=i}^{M} a_{ij} T_j^{(n)} \right\} \qquad i = 1 \sim M$$

- L_2 norm

$$L_2 = \sqrt{\frac{\sum_{i=1}^{M} (R_i^{n+1})^2}{M}} \qquad \text{or} \qquad \sqrt{\sum_{i}^{M} (R_i^{n+1})^2}$$

- Convergence Criterion:

$$L_2 \le \varepsilon \qquad \varepsilon \text{ : very small number}$$

7-6 Example (1)

Dealing with Nonlinearity

The momentum conservation equation for a fluid is nonlinear due to the convection term $(\vec{V} \cdot \nabla)\vec{V}$. Phenomena such as turbulence and chemical reaction introduce additional nonlinearities. The highly nonlinear nature of the governing equations for a fluid makes it challenging to obtain accurate numerical solutions for complex flows of practical interest.

We will demonstrate the effect of nonlinearity by setting $m = 2$ in our simple 1D example (1):

$$\frac{du}{dx} + u^2 = 0; \quad 0 \le x \le 1; \quad u(0) = 1$$

A first-order finite-difference approximation to this equation, analogous to that in (4) for $m = 1$, is

$$\frac{u_i - u_{i-1}}{\Delta x} + u_i^2 = 0 \tag{11}$$

This is a nonlinear algebraic equation with the u_i^2 term being the source of the nonlinearity.

7-6 Example (2)

The strategy that is adopted to deal with nonlinearity is to linearize the equations about a *guess value* of the solution and to iterate until the guess agrees with the solution to a specified tolerance level. We'll illustrate this on the above example. Let u_{g_i} be the guess for u_i. Define

$$\Delta u_i = u_i - u_{g_i}$$

Rearranging and squaring this equation gives

$$u_i^2 = u_{g_i}^2 + 2u_{g_i}\Delta u_i + (\Delta u_i)^2$$

Assuming that $\Delta u_i \ll u_{g_i}$, we can neglect the Δu_i^2 term to get

$$u_i^2 \simeq u_{g_i}^2 + 2u_{g_i}\Delta u_i = u_{g_i}^2 + 2u_{g_i}(u_i - u_{g_i})$$

Thus,

$$u_i^2 \simeq 2u_{g_i}u_i - u_{g_i}^2$$

The finite-difference approximation (11) after linearization becomes

$$\frac{u_i - u_{i-1}}{\Delta x} + 2u_{g_i}u_i - u_{g_i}^2 = 0 \qquad (12)$$

Since the error due to linearization is $O(\Delta u^2)$, it tends to zero as $u_g \to u$.

7-6 Example (3)

In order to calculate the finite-difference approximation (12), we need guess values u_g at the grid points. We start with an initial guess value in the first iteration. For each subsequent iteration, the u value obtained in the previous iteration is used as the guess value.

Iteration 1: $u_g^{(1)} =$ Initial guess

Iteration 2: $u_g^{(2)} = u^{(1)}$

$\vdots$

Iteration l: $u_g^{(l)} = u^{(l-1)}$

The superscript indicates the iteration level. We continue the iterations until they converge. We'll defer the discussion on how to evaluate convergence until a little later.

This is essentially the process used in CFD codes to linearize the nonlinear terms in the conservations equations, with the details varying depending on the code. The important points to remember are that the linearization is performed about a guess and that it is necessary to iterate through successive approximations until the iterations converge.

7-6 Example (4)

Direct and Iterative Solvers

We saw that we need to perform iterations to deal with the nonlinear terms in the governing equations. We next discuss another factor that makes it necessary to carry out iterations in practical CFD problems.

Verify that the discrete equation system resulting from the finite-difference approximation (12) on our four-point grid is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1+2\Delta x\, u_{g_2} & 0 & 0 \\ 0 & -1 & 1+2\Delta x\, u_{g_3} & 0 \\ 0 & 0 & -1 & 1+2\Delta x\, u_{g_4} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 1 \\ \Delta x\, u_{g_2}^2 \\ \Delta x\, u_{g_3}^2 \\ \Delta x\, u_{g_4}^2 \end{bmatrix} \tag{13}$$

In a practical problem, one would usually have thousands to millions of grid points or cells so that each dimension of the above matrix would be of the order of a million (with most of the elements being zeros). Inverting such a matrix directly would take a prohibitively large amount of memory. So instead, the matrix is inverted using an iterative scheme as discussed below.

7-6 Example (5)

Rearrange the finite-difference approximation (12) at grid point i so that u_i is expressed in terms of the values at the neighboring grid points and the guess values:

$$u_i = \frac{u_{i-1} + \Delta x\, u_{g_i}^2}{1 + 2\,\Delta x\, u_{g_i}}$$

If a neighboring value at the current iteration level is not available, we use the guess value for it. Let's say that we sweep from right to left on our grid i.e. we update u_4, then u_3 and finally u_2 in each iteration. In the m^{th} iteration, $u_{i-1}^{(l)}$ is not available while updating u_i^m and so we use the guess value $u_{g_{i-1}}^{(l)}$ for it instead:

$$u_i^{(l)} = \frac{u_{g_{i-1}}^{(l)} + \Delta x\, u_{g_i}^{(l)^2}}{1 + 2\,\Delta x\, u_{g_i}^{(l)}} \tag{14}$$

7-6 Example (6)

Since we are using the guess values at neighboring points, we are effectively obtaining only an approximate solution for the matrix inversion in (13) during each iteration but in the process have greatly reduced the memory required for the inversion. This tradeoff is good strategy since it doesn't make sense to expend a great deal of resources to do an exact matrix inversion when the matrix elements depend on guess values which are continuously being refined. In an act of cleverness, we have combined the iteration to handle nonlinear terms with the iteration for matrix inversion into a single iteration process. Most importantly, as the iterations converge and $u_g \rightarrow u$, the approximate solution for the matrix inversion tends towards the exact solution for the inversion since the error introduced by using u_g instead of u in (14) also tends to zero.

Thus, iteration serves two purposes:

1. It allows for efficient matrix inversion with greatly reduced memory requirements.

2. It is necessary to solve nonlinear equations.

In steady problems, a common and effective strategy used in CFD codes is to solve the unsteady form of the governing equations and "march" the solution in time until the solution converges to a steady value. In this case, each time step is effectively an iteration, with the the guess value at any time level being given by the solution at the previous time level.

7-6 Example (7)

Iterative Convergence

Recall that as $u_g \rightarrow u$, the linearization and matrix inversion errors tends to zero. So we continue the iteration process until some selected measure of the difference between u^g and u, refered to as the residual, is "small enough". We could, for instance, define the residual R as the RMS value of the difference between u and u_g on the grid:

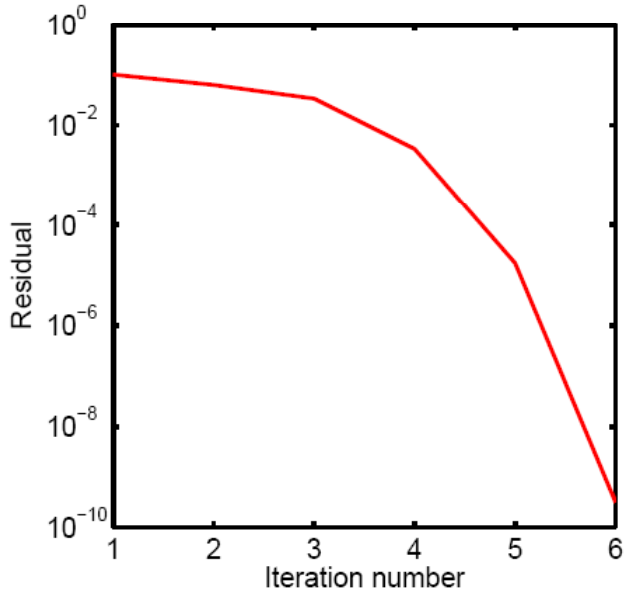
$$R \equiv \sqrt{\frac{\sum_{i=1}^{N} (u_i - u_{g_i})^2}{N}}$$

It's useful to scale this residual with the average value of u in the domain. An unscaled residual of, say, 0.01 would be relatively small if the average value of u in the domain is 5000 but would be relatively large if the average value is 0.1. Scaling ensures that the residual is a relative rather than an absolute measure. Scaling the above residual by dividing by the average value of u gives

$$R = \left(\sqrt{\frac{\sum_{i=1}^{N} (u_i - u_{g_i})^2}{N}} \right) \left(\frac{N}{\sum_{i=1}^{N} u_i} \right) = \frac{\sqrt{N \sum_{i=1}^{N} (u_i - u_{g_i})^2}}{\sum_{i=1}^{N} u_i} \qquad (15)$$

7-6 Example (8)

For the nonlinear 1D example, we'll take the initial guess at all grid points to be equal to the value at the left boundary i.e. $u_g^{(1)} = 1$. In each iteration, we update u_g, sweep from right to left on the grid updating, in turn, u_4, u_3 and u_2 using (14) and calculate the residual using (15). We'll terminate the iterations when the residual falls below 10^{-9} (which is referred to as the convergence criterion). Take a few minutes to implement this procedure in MATLAB which will help you gain some familiarity with the mechanics of the implementation. The variation of the residual with iterations obtained from MATLAB is shown below. Note that logarithmic scale is used for the ordinate. The iterative process converges to a level smaller than 10^{-9} in just 6 iterations. In more complex problems, a lot more iterations would be necessary for achieving convergence.

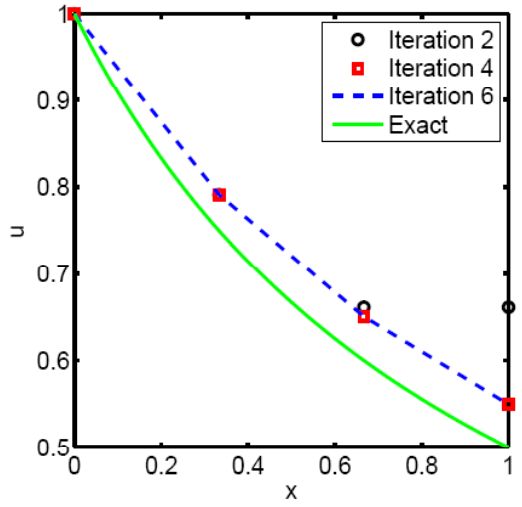


7-6 Example (9)

The solution after 2,4 and 6 iterations and the exact solution are shown below. It can easily be verified that the exact solution is given by

$$u_{exact} = \frac{1}{x+1}$$

The solutions for iterations 4 and 6 are indistinguishable on the graph. This is another indication that the solution has converged. The converged solution doesn't agree well with the exact solution because we are using a coarse grid for which the truncation error is relatively large. The iterative convergence error, which is of order 10^{-9}, is swamped out by the truncation error of order 10^{-1}. So driving the residual down to 10^{-9} when the truncation error is of order 10^{-1} is a waste of computing resources. In a good calculation, both errors would be of comparable level and less than a tolerance level chosen by the user. The agreement between the numerical and exact solutions should get much better on refining the grid as was the case for $m = 1$.



7-6 Example (10)

Some points to note:

1. Different codes use slightly different definitions for the residual. Read the documentation to understand how the residual is calculated.

2. In the FLUENT code, residuals are reported for each conservation equation. A discrete conservation equation at any cell can be written in the form LHS $= 0$. For any iteration, if one uses the current solution to compute the LHS, it won't be exactly equal to zero, with the deviation from zero being a mesaure of how far one is from achieving convergence. So FLUENT calculates the residual as the (scaled) mean of the absolute value of the LHS over all cells.

3. The convergence criterion you choose for each conservation equation is problem- and code-dependent. It's a good idea to start with the default values in the code. One may then have to tweak these values.