ELEMENTS OF COMPUTATIONAL FLUID DYNAMICS

Chapter - 6

2. Thomas algorithm (or tridiagonal matrix algorithm or simply TDMA).

✓ TDM system is a good example of sparse coefficient matrix. If Gaussian elimination is applied to this system, only one of the "a" is eliminated from the column containing the pivot element in each step, since the remaining elements below the diagonal are zero. Only one elimination process is employed at each step.

✓ The number of operations needed for solving a tridiagonal system is of order N, that is, *0(N)* as compared to *0(N-cube*) for a system with a dense coefficient matrix.

✓ Therefore, much smaller number of operations and consequently much lower round-off errors arise in the solution of such systems. Obviously, the computer time is much less for solution by TDMA. Thus, large tridiagonal systems are generally solved by this method.

✓The set of equations can be readily solved by the Gaussian elimination method with a maximum of three variables per equation. The solution can be expressed very concisely.

Tridiagonal matrix algorithm (TDMA).

- ✓The set of equations can be readily solved by the Gaussian elimination method with a maximum of three variables per equation. The solution can be expressed very concisely.
- ✓The prior equation a special form of the system (using N = M – 1)

$$b_1 T_1 + c_1 T_2 = d_1$$

$$a_2 T_1 + b_2 T_2 + c_2 T_3 = d_2$$

$$a_3 T_2 + b_3 T_3 + c_3 T_4 = d_3$$

$$\vdots$$

$$a_i T_{i-1} + b_i T_i + c_i T_{i+1} = d_i$$

$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$

$$a_{N-1} T_{N-2} + b_{N-1} T_{N-1} + c_{N-1} T_N = d_{N-1}$$

$$a_N T_{N-1} + b_N T_N = d_N$$

TDMA considers a recursion solution of the form: $T_i = \gamma_i - \frac{c_i}{\beta_i} T_{i+1}$

in which the constants β_i and γ_i are to be determined.

Substituting the expression for T_i in the generalized equation, one may obtain,

$$a_i T_{i-1} + b_i T_i + c_i T_{i+1} = d_i$$

$$\Rightarrow a_i \left(\gamma_{i-1} - \frac{c_{i-1}}{\beta_{i-1}} T_i \right) + b_i T_i + c_i T_{i+1} = d_i$$

$$\therefore T_i = \frac{d_i - a_i \gamma_{i-1}}{\left(b_i - \frac{a_i c_{i-1}}{\beta_{i-1}} \right)} - \frac{c_i T_{i+1}}{\left(b_i - \frac{a_i c_{i-1}}{\beta_{i-1}} \right)}$$

Also, from the first equation, one may obtain, $b_1 T_1 + c_1 T_2 = d_1$

$$\therefore T_1 = \frac{d_1}{b_1} - \frac{c_1 T_2}{b_1}$$

where β's and γ's are determined from the recursion formulae

$$\beta_1 = b_1, \quad \gamma_1 = \frac{d_1}{\beta_1}$$

$$\beta_i = b_i - \frac{a_i\, c_{i-1}}{\beta_{i-1}}, \qquad i = 2, 3, \ldots, N$$

$$\gamma_i = \frac{d_i - a_i\, \gamma_{i-1}}{\beta_i}, \qquad i = 2, 3, \ldots, N$$

From the last equation, one may obtain,

$$a_N T_{N-1} + b_N T_N = d_N$$

$$\therefore T_N = \frac{d_N - a_N T_{N-1}}{b_N} = \frac{d_N - a_N\left(\gamma_{N-1} - \frac{c_{N-1}}{\beta_{N-1}} T_N\right)}{b_N} = \frac{d_N - a_N \gamma_{N-1}}{b_N - \frac{a_N c_{N-1}}{\beta_{N-1}}} = \gamma_N$$

Therefore, in a nutshell, the TDMA algorithm can be written as,

$$T_N = \gamma_N$$

$$T_i = \gamma_i - \frac{c_i}{\beta_i} T_{i+1}$$

Where, the constant parameters are calculated as,

$$\beta_1 = b_1, \quad \gamma_1 = \frac{d_1}{\beta_1}$$

$$\beta_i = b_i - \frac{a_i\, c_{i-1}}{\beta_{i-1}}, \qquad i = 2,3,\ldots,N$$

$$\gamma_i = \frac{d_i - a_i\, \gamma_{i-1}}{\beta_i}, \qquad i = 2,3,\ldots,N$$

Gauss–Seidel (G–S) Iterative Method

✓ In this method, unlike in direct methods such as Gaussian elimination, the round-off error does not accumulate.

✓ The round-off error after each iteration simply produces a less accurate input for the next iteration.

✓ Therefore, the resulting round-off error in the numerical solution is only what arises in the computation for the final iteration.

✓ However, the solution is not exact but is obtained to an arbitrary, specified, convergence criterion.

Jacobi Method (Predecessor of G–S Method)

Let us consider a systems of equations as follows

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1N}x_N = c_1$$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2N}x_N = c_2$$

$$\vdots$$

$$a_{N1}x_1 + a_{N2}x_2 + \cdots + a_{NN}x_N = c_N$$

The method assumes the solution to the systems of equations as,

$$x_1 = \frac{c_1 - a_{12}x_2 - a_{13}x_3 - .... - a_{1N}x_N}{a_{11}}$$

$$x_2 = \frac{c_2 - a_{21}x_1 - a_{23}x_3 - .... - a_{2N}x_N}{a_{22}}$$

$$\bullet$$

$$\bullet$$

$$x_N = \frac{c_N - a_{N1}x_1 - a_{N2}x_2 - .... - a_{NN-1}x_{N-1}}{a_{NN}}$$

Therefore, in general,

$$x_i = \frac{c_i - \sum_{j=1, j\neq i}^{N} a_{ij}x_j}{a_{ii}} \qquad for\ i = 1, 2, 3, ...N$$

To start the iteration process, initial guess values are assigned to all unknowns. If $x_1^{(0)}$, $x_2^{(0)}$, ….. $x^N_{(0)}$ are initial guess values, then, the value of x_1 after the first iteration, $x_1^{(1)}$,

$$x_1^{(1)} = \frac{c_1 - a_{12}x_2^{(0)} - a_{13}x_3^{(0)} - .... - a_{1N}x_N^{(0)}}{a_{11}}$$

Therefore, in general,

$$x_i^{(1)} = \frac{c_i - \sum_{j=1, j\neq i}^{N} a_{ij}x_j^{(0)}}{a_{ii}} \qquad for\ i = 1, 2, 3, ...N$$

The values obtained in the first iteration is to be used for the next iteration, thus

$$x_i^{(p+1)} = \frac{c_i - \sum_{j=1, j\neq i}^{N} a_{ij} x_j^{(p)}}{a_{ii}} \qquad for\ i = 1, 2, 3, ...N$$

Disadvantages of Jacobi method

- The main disadvantage is that the computer storage is needed for the present iteration as well as for the previous one.
- This is because all the values are computed using previous values before any unknown is updated.

Gauss–Seidel Method (An Improvement Over Jacobi Method)

- ✓A significant improvement in the rate of convergence and in the storage requirements can be obtained by replacing the values from the previous iteration by new ones as soon as they are computed.
- ✓Then the values of only the latest iteration are stored, and each iterative computation of the unknown employs the most recent values of the other unknowns.
- ✓This computational scheme is known as point-by-point Gauss–Seidel method.

Gauss–Seidel Method

$$x_2^{(2)} = \frac{c_2 - a_{21}x_1^{(2)} - a_{23}x_3^{(1)} - .... - a_{2N}x_N^{(1)}}{a_{11}}$$

Therefore, in general,

$$x_i^{(p+1)} = \frac{c_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(p+1)} - \sum_{j=i+1}^{N} a_{ij}x_j^{(p)}}{a_{ii}} \qquad for\ i = 1, 2, 3, ...N$$

Convergence criteria for G–S method

(a) $$\left| x_i^{(p+1)} - x_i^{(p)} \right| \le \varepsilon \qquad for\ i = 1, 2, 3, ...N$$

(b) $$\left| \frac{x_i^{(p+1)} - x_i^{(p)}}{x_i^{(p)}} \right| \le \varepsilon \qquad for\ i = 1, 2, 3, ...N$$

Criteria (b) if an estimate of the magnitude of the unknowns x_i is not available and none of the unknowns is expected to be zero

Condition for convergence in G–S method (Scarborough criterion)

The convergence is guaranteed for linear systems that is, when the system is diagonally dominant. This is also known as Scarborough criterion.

$$\text{if} \qquad |a_{ii}| \geq \sum_{j=1, j\neq i}^{N} |a_{ij}|, \qquad \text{for all } i$$

$$\text{and if} \qquad |a_{ii}| > \sum_{j=1, j\neq i}^{N} |a_{ij}|, \qquad \text{for at least one } i$$

Why does diagonal dominance ensure convergence?

In the simple three-equation system, we see in a particular iteration:

$$x_1 = (x_1)_{\text{exact}} + \varepsilon_1$$

$$x_2 = (x_2)_{\text{exact}} + \varepsilon_2$$

$$x_3 = (x_3)_{\text{exact}} + \varepsilon_3$$

Then the condition of diagonal dominance is forcing ε to become progressively smaller in successive iterations.

Relaxation (over-relaxation and under-relaxation)

One of the problems with G–S method is that it is relatively slow to converge to the solution. The rate of convergence can often be improved by relaxation method

$$x_i^{(p+1)} = \frac{\alpha\left[c_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(p+1)} - \sum_{j=i+1}^{N} a_{ij} x_j^{(p)}\right]}{a_{ii}} + (1-\alpha)\, x_i^{(p)} \qquad for\ i = 1, 2, 3, \ldots\ldots N$$

$0 < \alpha < 1$ Under-relaxation

$1 < \alpha < 2$ Over-relaxation

✓Successive under-relaxation (SUR) or under-relaxation is generally used for non-linear equations and for systems that result in a divergent G–S iteration.

✓Successive over-relaxation (SOR) or over-relaxation is widely used for accelerating convergence in linear systems.

Summary of three methods

G-E	TDMA	G-S
Direct solver	Direct solver	Iterative
Based on foreword elimination and back substitution	Based on recursion formula	Based on initial guess of unknowns and subsequent improvement in each iteration
High round-off error	Low round-off error	Low round-off error
Number of arithmetic operation $O(N^3)$	Number of arithmetic operation O(N)	Number of arithmetic operation O(N)
Can be used low numbers of equations	Can be used high numbers of equations	Can be used for several thousands numbers of equations

Checking for accuracy

The accuracy of a numerical solution is usually checked by one of the following three ways:

1. Comparison with the analytical solution. For most practical problems, analytical solutions do not exist. But, this is a good way of checking accuracy of a new numerical method.

2. Comparison with the limiting case analytical solution. This is possible when the analytical solution for some limiting values of a parameter governing the solution is available.

3. Comparison with experimental results. This is most desirable for complex problems such as turbulence, combustion, non-Newtonian fluid flow, and heat transfer, which require many assumptions for the purpose of modeling.

Convective boundary condition

The dimensionless boundary condition at the fin tip in the changed scenario would be written in the mathematical form as

$$\frac{d\theta}{dX} + \frac{hL}{k}\theta = 0$$

At *i*= *M*, the discretization equation is:

$$\frac{\theta_{M+1} - \theta_{M-1}}{\Delta X} + \frac{hL}{k}\theta_M = 0$$

$$\therefore \theta_{M+1} = \theta_{M-1} - \frac{2hL\Delta X}{k}\theta_M \qquad (1)$$

At *x = L, i.e., at i = M,* we have: $\theta_{M-1} - D\theta_M + \theta_{M+1} = 0$

Substituting the expression of Θ_{M+1} from (1), we have

$$2\theta_{M-1} - D_1\theta_M = 0$$

Where,

$$D_1 = D + \frac{2hL\Delta X}{k}$$