Lecture 25

DIRECT AND BASIC ITERATIVE METHODS FOR LINEAR SYSTEMS

25.1 INTRODUCTION

The methods for solution of linear algebraic equations can be broadly put into two categories:

1. Direct solvers (e.g. Gauss elimination, LU decomposition etc.)
2. Iterative solvers (e.g. Gauss-Seidel, SOR, PCG etc.)

Choice of a particular solver depends on the size and nature of the system matrix. There are specialized versions of direct as well as iterative solvers for symmetric systems. For example,

- If the system matrix is symmetric (and positive definite)
  - Direct solvers: Cholesky decomposition
  - Iterative solvers: SSOR, preconditioned conjugate method
- Otherwise, for general systems
  - LU decomposition
  - Iterative solvers: Gauss-Seidel, SOR, GMRES, bi-conjugate gradient method

In this lecture, we provide brief outline/algorithms of TDMA and basic iterative methods. For a detailed account of iterative techniques, see Saad (2003) and for multi-grid methods, see Trottenberg et al. (2001). In all the discussions, we assume that the linear system is expressed as

$$\mathbf{Ax} = \mathbf{b} \qquad (25.1)$$

25.2 EFFECT OF COMPUTATIONAL COMPLEXITY

Number of arithmetic operations involved in numerical solution of an algebraic system is commonly referred to as the computational complexity. It depends on the order of the system (i.e. the number of unknowns, N), the nature and structure of the system matrix and the solution algorithm. Normally,

- Computational complexity of direct solvers grows as $O(N^3)$ (specialized solvers such as TDMA are exceptions to this rule).
- In contrast, many fast iterative methods aim at a computational complexity $O(N)$ or $O(N \log N)$.

Therefore, a direct solver is normally preferred for smaller systems, whereas iterative solvers are preferred for larger systems.

25.3 DIRECT SOLVERS FOR DISCRETE ALGEBRAIC SYSTEMS

The system matrix obtained from FDM/FVM/FEM discretization in CFD is sparse and order of system is usually very large (of order of a few millions to a few billions in simulation of LES/DNS of turbulent flows). Thus, direct methods such as Gaussian elimination or LU-decomposition are not suitable for these systems. There are special variants of these methods which have been devised for specialized situations:

- Use of central difference scheme (or linear finite elements) to one dimensional elliptic problem leads to a tri-diagonal system. For such systems, Gaussian elimination takes a rather simple and efficient form which is popularly known TDMA.

- On structured grids, efficient iterative procedures such as ADI based on TDMA can be devised for multi-dimensional problems (Ferziger and Peric, 2003).
- For systems obtained on unstructured grids which lead to a band diagonal system, specialized version of LU decomposition can be used (Press et al., 2002).

Note that incomplete factorization schemes (such incomplete LU decomposition) are sometimes used to construct pre-conditioners for iterative solvers.

25.4 TRI-DIAGONAL MATRIX ALGORITHM (TDMA)

The system matrix arising from CDS or linear finite element approximation of one dimensional elliptic equation has a special structure: all entries except those on the main diagonal and two adjacent diagonals are zero. Let us represent the entries on main diagonal by A_P, diagonal immediately above as A_E and one immediately below as A_W. With this notation, discrete system (25.1) can be re-written as

$$A_W^i x_{i-1} + A_P^i x_i + A_E^i x_{i+1} = b_i, \qquad i = 1, \dots, N \tag{25.2}$$

Solution method based on Gaussian elimination and back-substitution takes a rather simple form for this system. The resulting algorithm, called TDMA, consists of the following steps:

- In forward elimination process, modify the main diagonal term A_P^i and the source term using the following formulae:

$$A_P^i \leftarrow A_P^i - \frac{A_W^i A_E^{i-1}}{A_P^{i-1}} \tag{25.3}$$

$$b_i^* = b_i - \frac{A_W^i b_{i-1}^*}{A_P^{i-1}} \tag{25.4}$$

- Back-substitution process, evaluate x_i using the formula

$$x_i = \frac{b_i^* - A_E^i x_{i+1}}{A_P^i} \tag{25.5}$$

25.5 ITERATIVE METHODS FOR LINEAR SYSTEMS

Iterative methods for linear systems can be broadly put into two categories:

(a) **Basic iterative methods** such as Jacobi, Gauss-Seidel and SOR: These are very easy to program, require very little computation time per iteration, but have *very slow convergence*. Thus, with exception of SOR, these methods are not used as stand-alone solvers. These are primarily used as smoothing/relaxation procedures in conjunction with multi-grid techniques, and also occasionally as pre-conditioners with Krylov subspace methods (such as PCG and GMRES).

(b) **Accelerated iterative methods** such as conjugate gradient, GMRES, and multi-grid techniques. Most of the commercial FEM or CFD packages employ some form of multi-grid method or Krylov subspace method for solution of algebraic equations.

25.6 BASIC ITERATIVE METHODS

Let k denote the iteration counter, and the i_{th} component of the new approximation (at iteration level k+1) be denoted by x_i^{k+1}.

Jacobi's Method

The i_{th} component of the new approximation is determined so as to annihilate the i_{th} component of the residual vector, i.e.

$$x_i^{k+1} = \frac{1}{A_{ii}}\left(b_i - \sum_{\substack{j=1 \\ j\neq i}}^{N} A_{ij} x_j^k \right) \tag{25.6}$$

Gauss-Seidel Method

The i_{th} component of the new approximation is determined so as to annihilate the i_{th} component of the residual vector using the most recent values of available components, i.e.

$$x_i^{k+1} = \frac{1}{A_{ii}}\left(b_i - \sum_{j=1}^{i-1} A_{ij} x_j^{k+1} - \sum_{j=i+1}^{N} A_{ij} x_j^k \right) \tag{25.7}$$

Successive Over-Relaxation (SOR)

To improve the convergence of Gauss-Seidel iteration, a relaxation strategy given by the following equation is used:

$$x_i^{k+1} = \frac{\omega}{A_{ii}}\left(b_i - \sum_{j=1}^{i-1} A_{ij} x_j^{k+1} - \sum_{j=i+1}^{N} A_{ij} x_j^k \right) + (1-\omega) x_i^k \tag{25.8}$$

where $1 < \omega < 2$ is the over-relaxation factor. An optimum value of ω can lead to fast convergence. There is a procedure to determine ω on structured grids (Press et al., 2002), but little help is available for general problems. In such cases, it is better to choose a larger value (say around 1.7).

REFERENCES/FURTHER READING

Chung, T. J. (2010). *Computational Fluid Dynamics*. 2nd Ed., Cambridge University Press, Cambridge, UK.

Ferziger, J. H. And Perić, M. (2003). *Computational Methods for Fluid Dynamics*. Springer.

Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. (2002). *Numerical Recipes in C++*. Cambridge University Press, Cambridge.

Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia.