8. Introduction to Computational Fluid Dynamics

We have been using the idea of distributions of singularities on surfaces to study the aerodynamics of airfoils and wings. This approach was very powerful, and provided us with methods which could be used easily on PCs to solve real problems. Considerable insight into aerodynamics was obtained using these methods. However, the class of effects that could be examined was somewhat restricted. In particular, practical methods for computing fundamentally nonlinear flow effects were excluded. This includes both inviscid transonic and boundary layer flows.

In this chapter we examine the basic ideas behind the direct numerical solution of differential equations. This approach leads to methods that can handle nonlinear equations. The simplest methods to understand are developed using numerical approximations to the derivative terms in the partial differential equation (PDE) form of the governing equations. Direct numerical solutions of the partial differential equations of fluid mechanics constitute the field of computational fluid dynamics (CFD). Although the field is still developing, a number of books have been written.[1,2,3,4,5,6] In particular, the book by Tannehill *et al*,[1] which appeared in 1997 as a revision of the original 1984 text, covers most of the aspects of CFD theory used in current codes and reviewed here in Chapter 14. Fundamental concepts for solving partial differential equations in general using numerical methods are presented in a number of basic texts. Smith[7] and Ames[8] are good references.

The basic idea is to model the derivatives by finite differences. When this approach is used the entire flowfield must be discretized, with the field around the vehicle defined in terms of a mesh of grid points. We need to find the flowfield values at every mesh (or grid) point by writing down the discretized form of the governing equation at each mesh point. Discretizing the equations leads to a system of simultaneous algebraic equations. A large number of mesh points is usually required to accurately obtain the details of the flowfield, and this leads to a very large system of equations. Especially in three dimensions, this generates demanding requirements for computational resources. To obtain the solution over a complete three dimensional aerodynamic configuration millions of grid points are required!

In contrast to the finite difference idea, approximations to the integral form of the governing equations result in the *finite volume* approach. A book has been written recently devoted solely to this approach,[9] and we will cover this approach briefly here.

Thus CFD is usually associated with computers with large memories and high processing speeds. In addition, massive data storage systems must be available to store computed results, and ways to transmit and examine the massive amounts of data associated with a computed result must be available. Before the computation of the solution is started, the mesh of grid points must be established. Thus the broad area of CFD leads to many different closely related but nevertheless specialized technology areas. These include:

- grid generation
- flowfield discretization algorithms
- efficient solution of large systems of equations
- massive data storage and transmission technology methods
- computational flow visualization

Originally, CFD was only associated with the 2nd and 3rd items listed above. Then the problem with establishing a suitable mesh for arbitrary geometry became apparent, and the specialization of grid generation emerged. Finally, the availability of large computers and remote processing led to the need for work in the last two items cited. Not generally included in CFD per se, a current limiting factor in the further improvement in CFD capability is development of accurate turbulence models, discussed in Chapter 10.

This chapter provides an introduction to the concepts required for developing discretized forms of the governing equations and a discussion of the solution of the resulting algebraic equations. For the most part, we adopt the viewpoint of solving equilibrium (elliptic) problems. This in contrast to the more frequent emphasis on solving hyperbolic systems. Although the basic idea of CFD appears straightforward, once again we find that a successful numerical method depends on considerable analysis to formulate an accurate, robust, and efficient solution method. We will see that the classification of the mathematical type of the governing equations (Sec. 2.8) plays an important role in the development of the numerical methods. Although we adopt finite difference/finite volume methods to solve nonlinear equations, to establish the basic ideas we consider only linear equations. Application to nonlinear equations is addressed in Chapters 10, 11 and 12, where additional concepts are introduced and applied to the solution of nonlinear equations. Chapter 13 describes the most advanced approaches currently in use.

8.1 Approximations to partial derivatives

There are many ways to obtain finite difference representations of derivatives. Figure 8-1 illustrates the approach intuitively. Suppose that we use the values of f at a point x_0 and a point a

distance Δx away. Then we can approximate the slope at x_0 by taking the slope between these points. The sketch illustrates the difference between this simple slope approximation and the actual slope at the point x_0. Clearly, accurate slope estimation dependents on the method used to estimate the slope and the use of suitably small values of Δx.

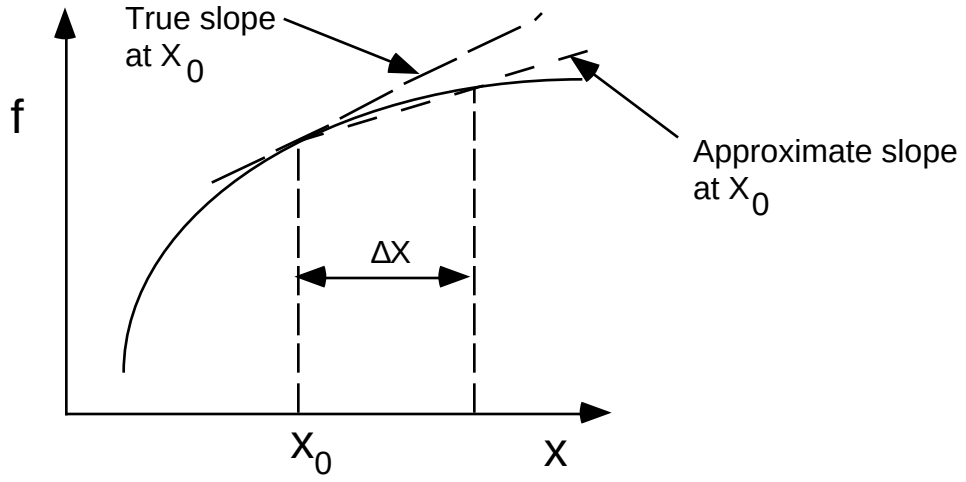


Figure 8-1. Example of slope approximation using two values of the function.

Approximations for derivatives can be derived systematically using Taylor series expansions. The simplest approach is to find an estimate of the derivative from a single series. Consider the following Taylor series:

$$f(x_0 + \Delta x) = f(x_0) + \Delta x \left.\frac{df}{dx}\right|_{x_0} + \frac{(\Delta x)^2}{2}\left.\frac{d^2 f}{dx^2}\right|_{x_0} + \frac{(\Delta x)^3}{6}\left.\frac{d^3 f}{dx^3}\right|_{x_0} + \ldots \tag{8-1}$$

and rewrite it to solve for $\left.\frac{df}{dx}\right|_{x_0}$:

$$\left.\frac{df}{dx}\right|_{x_0} = \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} - \Delta x \frac{1}{2}\left.\frac{d^2 f}{dx^2}\right|_{x_0} - \ldots$$

or:

$$\left.\frac{df}{dx}\right|_{x_0} = \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} + \underbrace{O(\Delta x)}_{\text{Truncation Error}} \tag{8-2}$$

where the last term is neglected and called the truncation error. In this case it is O(Δx). The term "truncation error" means that the error of the approximation vanishes as Δx goes to zero.* The

* This assumes that the numerical results are exactly accurate. There is a lower limit to the size of the difference step in Δx due to the use of finite length arithmetic. Below that step size, roundoff error becomes important. In most

form of the truncation error term is frequently important in developing numerical methods. When the order of the truncation error is O(Δx), the approximation is described as a "first order accurate" approximation, and the error is directly proportional to Δx. The other characteristic of this representation is that it uses only the information on one side of x_0, and is thus known as a one-sided difference approximation. Finally, because it uses information ahead of x_0, it's known as a forward difference. Thus, Eq.(8-2) is a first order, one sided, forward difference approximation to the derivative.

We could also write the approximation to the derivative using information prior to the point of interest. The corresponding first order accurate one sided backward difference approximation is obtained by expanding the Taylor series to a point prior to the point about which the expansion is carried out. The resulting expansion is:

$$f(x_0 - \Delta x) = f(x_0) - \Delta x \left.\frac{df}{dx}\right|_{x_0} + \frac{(\Delta x)^2}{2}\left.\frac{d^2f}{dx^2}\right|_{x_0} - \frac{(\Delta x)^3}{6}\left.\frac{d^3f}{dx^3}\right|_{x_0} + \ldots. \tag{8-3}$$

Solving for the first derivative in the same manner we used above, we obtain:

$$\left.\frac{df}{dx}\right|_{x_0} = \frac{f(x_0) - f(x_0 - \Delta x)}{\Delta x} + O(\Delta x), \tag{8-4}$$

the first order accurate, one sided, backward difference approximation.

Note from Fig. 8-1 above that one sided differences can lead to a fairly large truncation error. In many cases a more accurate finite difference representation would be useful. To obtain a specified level of accuracy, the step size Δx must be made small. If a formula with a truncation term of $O(\Delta x)^2$ is used,* the required accuracy can be obtained with significantly fewer grid points. A second order, $O(\Delta x)^2$, approximation can be obtained by subtracting the Taylor series expansions, Eq.(8-3) from Eq.(8-1):

$$f(x_0 + \Delta x) - f(x_0 - \Delta x) = +2\Delta x \left.\frac{df}{dx}\right|_{x_0} + \frac{(\Delta x)^3}{3}\left.\frac{d^3f}{dx^3}\right|_{x_0} + \ldots.$$

Here the O(Δx) terms cancel in the subtraction. When we divide by $2\Delta x$ and solve for the first derivative, we get an expression with a truncation error of $O(\Delta x)^2$. The resulting expression for the derivative is:

---

cases the stepsize used for practical finite difference calculations is larger than the limit imposed by roundoff errors. We can't afford to compute using grids so finely spaced that roundoff becomes a problem.

* With Δx small, Δx^2 is much smaller than Δx

$$\left.\frac{df}{dx}\right|_{x_0} = \frac{f(x_0+\Delta x)-f(x_0-\Delta x)}{2\Delta x} + O(\Delta x)^2 . \tag{8-5}$$

This is a second order accurate *central* difference formula since information comes from both sides of x_0. Numerous other approximations can be constructed using this approach. It's also possible to write down second order accurate forward and backward difference approximations.

We also need the finite difference approximation to the second derivative. Adding the Taylor series expressions for the forward and backward expansions, Eq.(8-1) and Eq.(8-3), results in the following expression, where the odd order terms cancel:

$$f(x_0+\Delta x) + f(x_0-\Delta x) = 2f(x_0) + (\Delta x)^2 \left.\frac{d^2 f}{dx^2}\right|_{x_0} + O(\Delta x)^4$$

Solving for the second derivative yields:

$$\left.\frac{d^2 f}{dx^2}\right|_{x_0} = \frac{f(x_0+\Delta x)-2f(x_0)+f(x_0-\Delta x)}{(\Delta x)^2} + O(\Delta x)^2 . \tag{8-6}$$

The formulas given above are the most frequently used approximations to the derivatives using finite difference representations. Other methods can be used to develop finite difference approximations. In most cases we want to use no more than two or three function values to approximate derivatives.

Forward and backward finite difference approximations for the second derivative can also be derived. Note that formally these expressions are only first order accurate. They are:

• a forward difference expression:

$$\left.\frac{d^2 f}{dx^2}\right|_{x_0} = \frac{f(x_0)-2f(x_0+\Delta x)+f(x_0+2\Delta x)}{(\Delta x)^2} + O(\Delta x) \tag{8-7}$$

• a backward difference expression:

$$\left.\frac{d^2 f}{dx^2}\right|_{x_0} = \frac{f(x_0)-2f(x_0-\Delta x)+f(x_0-2\Delta x)}{(\Delta x)^2} + O(\Delta x) . \tag{8-8}$$

In addition, expressions can be derived for cases where the points are not evenly distributed. In general the formal truncation error for unevenly spaced points is not as high as for the evenly spaced point distribution. In practice, for reasonable variations in grid spacing, this may not be a serious problem. We present the derivation of these expressions here. A better way of handling non-uniform grid points is presented in the next chapter. The one sided first derivative expressions Eq.(8-2) and Eq.(8-4) are already suitable for use in unevenly spaced situations. We need to obtain a central difference formula for the first derivative, and an expression for the

second derivative. First consider the Taylor expansion as given in Eqs. (8-1) and (8-3). However, the spacing will be different in the two directions. Use Δx^+ and Δx^- to distinguish between the two directions. Eqs. (8-1) and (8-3) can then be rewritten as:

$$f(x_0 + \Delta x^+) = f(x_0) + \Delta x^+ \left.\frac{df}{dx}\right|_{x_0} + \frac{\left(\Delta x^+\right)^2}{2}\left.\frac{d^2f}{dx^2}\right|_{x_0} + \frac{\left(\Delta x^+\right)^3}{6}\left.\frac{d^3f}{dx^3}\right|_{x_0} + \ldots \tag{8-9}$$

$$f(x_0 - \Delta x^-) = f(x_0) - \Delta x^- \left.\frac{df}{dx}\right|_{x_0} + \frac{\left(\Delta x^-\right)^2}{2}\left.\frac{d^2f}{dx^2}\right|_{x_0} - \frac{\left(\Delta x^-\right)^3}{6}\left.\frac{d^3f}{dx^3}\right|_{x_0} + \ldots \tag{8-10}$$

Define $\Delta x^+ = \alpha\ \Delta x^-$. To obtain the forms suitable for derivation of the desired expressions, replace Δx^+ in Eq. (8-9) with $\alpha\ \Delta x^-$, and multiply Eq. (8-10) by α. The resulting expressions are:

$$f(x_0 + \Delta x^+) = f(x_0) + \alpha\Delta x^- \left.\frac{df}{dx}\right|_{x_0} + \frac{\left(\alpha\Delta x^-\right)^2}{2}\left.\frac{d^2f}{dx^2}\right|_{x_0} + \frac{\left(\alpha\Delta x^-\right)^3}{6}\left.\frac{d^3f}{dx^3}\right|_{x_0} + \ldots \tag{8-11}$$

$$\alpha f(x_0 - \Delta x^-) = \alpha f(x_0) - \alpha\Delta x^- \left.\frac{df}{dx}\right|_{x_0} + \alpha\frac{\left(\Delta x^-\right)^2}{2}\left.\frac{d^2f}{dx^2}\right|_{x_0} - \alpha\frac{\left(\Delta x^-\right)^3}{6}\left.\frac{d^3f}{dx^3}\right|_{x_0} + \ldots \tag{8-12}$$

To obtain the expression for the first derivative, subtract Eq (8-12) from Eq. (8-11).

$$\begin{aligned} f(x_0 + \Delta x^+) - \alpha f(x_0 - \Delta x^-) = {} & f(x_0) - \alpha f(x_0) + 2\alpha\Delta x^- \left.\frac{df}{dx}\right|_{x_0} \\ & + \left[\frac{\left(\alpha\Delta x^-\right)^2}{2} - \alpha\frac{\left(\Delta x^-\right)^2}{2}\right]\left.\frac{d^2f}{dx^2}\right|_{x_0} + \ldots \end{aligned} \tag{8-13}$$

and rearrange to solve for *df/dx*:

$$\left.\frac{df}{dx}\right|_{x_0} = \frac{f(x_0 + \Delta x^+) + (\alpha - 1) f(x_0) - \alpha f(x_0 - \Delta x^-)}{2\alpha\Delta x^-} + O(\Delta x^-) \tag{8-14}$$

To obtain the expression for the second derivative, add (8-11) and (8-12):

$$f(x_0 + \Delta x^+) + \alpha f(x_0 - \Delta x^-) = f(x_0) + \alpha f(x_0) + \left[\frac{\left(\alpha\Delta x^-\right)^2}{2} + \alpha\frac{\left(\Delta x^-\right)^2}{2}\right]\left.\frac{d^2f}{dx^2}\right|_{x_0} + O(\Delta x)^3 \ldots$$

which is then solved for d^2f/dx^2:

$$\left.\frac{d^2f}{dx^2}\right|_{x_0} = \frac{f(x_0+\Delta x^+)-(1+\alpha)f(x_0)+\alpha f(x_0-\Delta x^-)}{\frac{\alpha}{2}(1+\alpha)\left(\Delta x^-\right)^2} + O(\Delta x^-) \qquad (8\text{-}15)$$

Note that both Eqs. (8-14) and (8-15) reduce to the forms given in Eq.(8-5) and Eq.(8-6) when the grid spacing is uniform.

Finally, note that a slightly more sophisticated analysis (Tannehill, *et al*,[1] pages 61-63) will lead to a second order expression for the first derivative on unevenly spaced points:

$$\left.\frac{df}{dx}\right|_{x_0} = \frac{f(x_0+\Delta x^+)+\left(\alpha^2-1\right)f(x_0)-\alpha^2 f(x_0-\Delta x^-)}{\alpha(\alpha+1)\Delta x^-} + O(\Delta x^-)^2 \qquad (8\text{-}16)$$

Tannehill, *et al*,[1] give additional details and a collection of difference approximations using more than three points and difference approximations for mixed partial derivatives (Tables 3-1 and 3-2 on their pages 52 and 53). Numerous other methods of obtaining approximations for the derivatives are possible. The most natural one is the use of a polynomial fit through the points. Polynomials are frequently used to obtain derivative expressions on non-uniformly spaced grid points.

These formulas can also be used to represent partial derivatives. To simplify the notation, we introduce a grid and a notation common in finite difference formulations. Figure 8-2 illustrates this notation using $\Delta x = \Delta y = const$ for these examples.

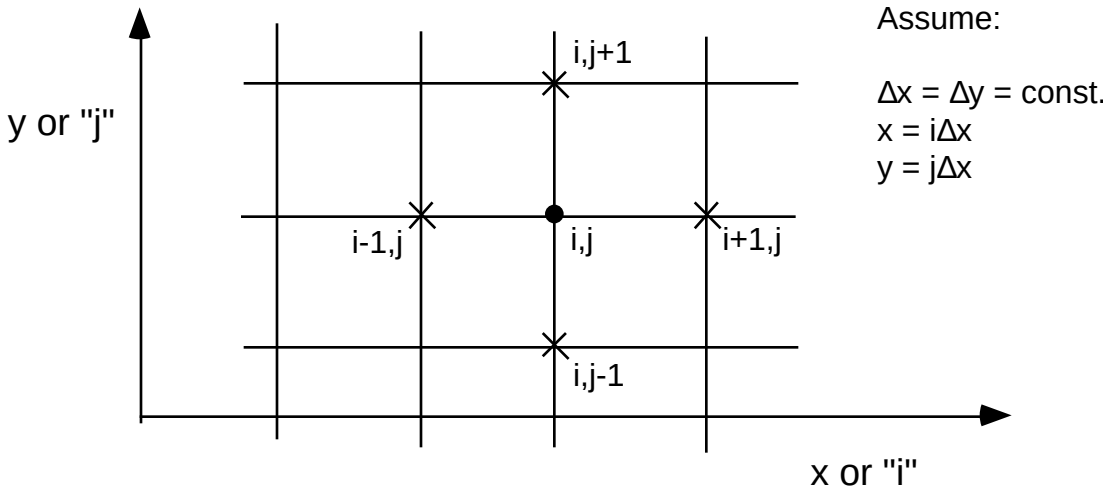


Figure 8-2. Nomenclature for use in partial differential equation difference expressions.

In this notation the following finite difference approximations for the first derivatives are:

$$\frac{\partial f}{\partial x} = \frac{f_{i+1,j} - f_{i,j}}{\Delta x} + O(\Delta x) \qquad \text{1st order forward difference} \qquad (8\text{-}17)$$

$$\frac{\partial f}{\partial x} = \frac{f_{i,j} - f_{i-1,j}}{\Delta x} + O(\Delta x) \qquad \text{1st order backward difference} \qquad (8\text{-}18)$$

$$\frac{\partial f}{\partial x} = \frac{f_{i+1,j} - f_{i-1,j}}{2\Delta x} + O(\Delta x)^2 \qquad \text{2nd order central difference} \qquad (8\text{-}19)$$

and the second derivative is:

$$\frac{\partial^2 f}{\partial x^2} = \frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{(\Delta x)^2} + O(\Delta x)^2 \qquad \text{2nd order central difference} \qquad (8\text{-}20)$$

Similar expressions can be written for the *y* derivatives. To shorten the expressions, various researchers have introduced different shorthand notations to replace these expressions. The shorthand notation is then used in further operations on the difference expressions.

8.2 Finite difference representation of Partial Differential Equations (PDE's)

We can use the approximations to the derivatives obtained above to replace the individual terms in partial differential equations. The following figure provides a schematic of the steps required, and some of the key terms used to ensure that the results obtained are in fact the solution of the original partial differential equation. We will define each of these new terms below.

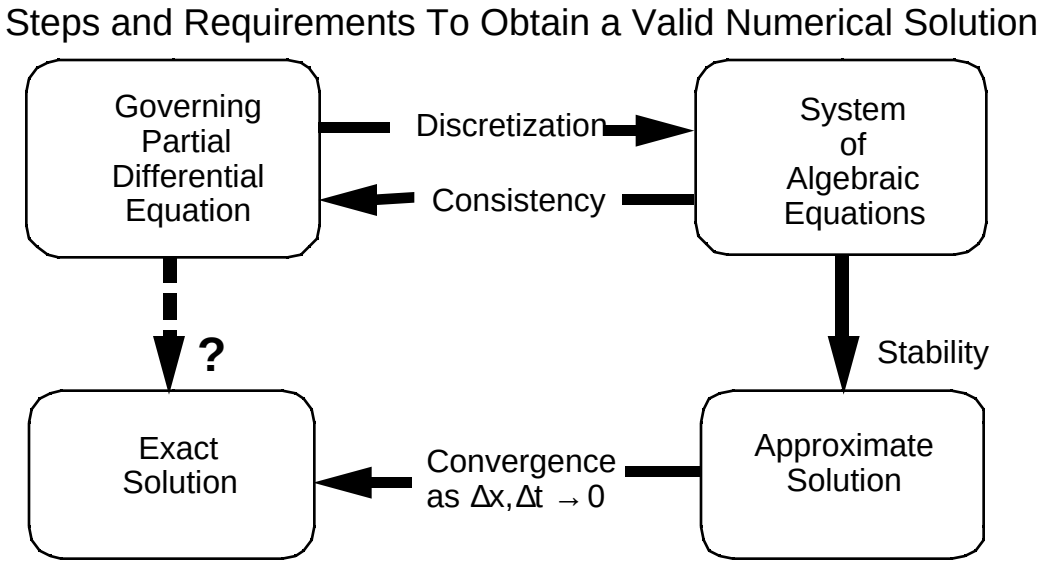


Figure 8-3. Overall procedure used to develop a CFD solution procedure.

Successful numerical methods for partial differential equations demand that the physical features of the PDE be reflected in the numerical approach. The selection of a particular finite difference approximation depends on the physics of the problem being studied. In large part the *type* of the PDE is crucial, and thus a determination of the type, *i.e*. elliptic, hyperbolic, or parabolic is extremely important. The mathematical type of the PDE must be used to construct the numerical scheme for approximating partial derivatives. Some advanced methods obscure the relationship, but it must exist. Consider the example given in Fig. 8-4 illustrating how information in a grid must be used.

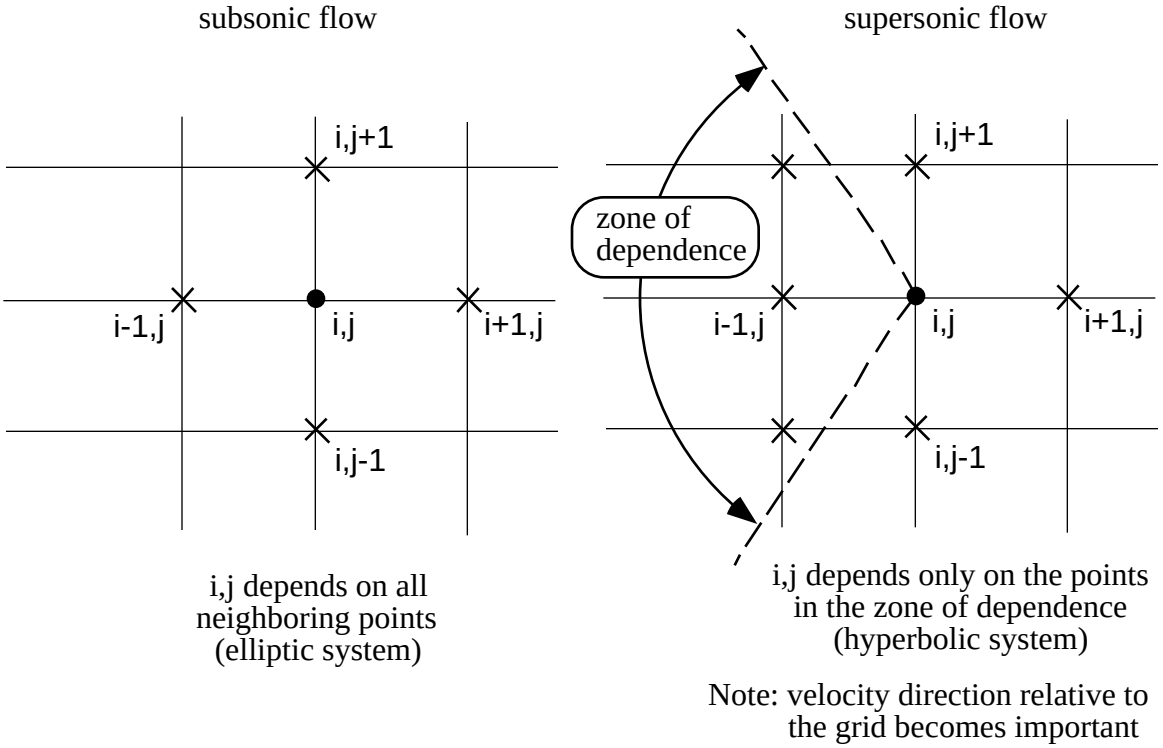


Figure 8-4. Connection between grid points used in numerical method and equation type.

Any scheme that fails to represents the physics correctly will fail when you attempt to obtain a solution. Furthermore, remember, in this case we were looking at a uniformly spaced cartesian grid. In actual “real life” applications we have to consider much more complicated non-uniform grids in non-Cartesian coordinate systems. In this section we use simple uniform Cartesian grid systems to illustrate the ideas. The necessary extensions of the methods illustrated in this chapter are outlined in the next chapter.

In Fig. 8-3 above, we introduced several important terms requiring definition and discussion:

- discretization
- consistency
- stability
- convergence

Before defining the terms, we provide an example using the heat equation:

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}. \tag{8-21}$$

We discretize the equation using a forward difference in time, and a central difference in space following the notation shown in the following sketch:

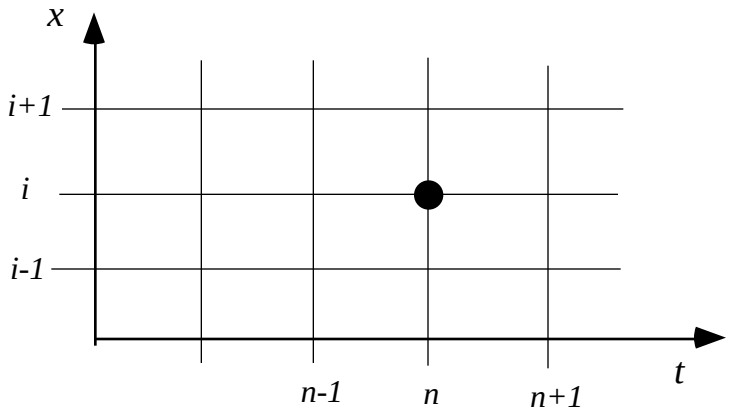


Figure 8-5. Grid nomenclature for discretization of heat equation.

The heat equation can now be written as:

$$\underbrace{\frac{\partial u}{\partial t} - \alpha \frac{\partial^2 u}{\partial x^2}}_{\text{PDE}} = \underbrace{\frac{u_i^{n+1} - u_i^n}{\Delta t} - \frac{\alpha}{(\Delta x)^2}\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right)}_{\text{FDE}} +$$

$$\underbrace{\left[-\frac{\partial^2 u}{\partial t^2}\bigg|_i^n \frac{\Delta t}{2} + \alpha \frac{\partial^4 u}{\partial x^4}\bigg|_i^n \frac{(\Delta x)^2}{12} + \ldots\right] = 0}_{\text{Truncation Error}} \tag{8-22}$$

where we use the superscript to denote time and the subscript to denote spatial location. In Eq. (8-18) the partial differential equation (PDE) is converted to the related finite difference equation (FDE). The truncation error is $O(\Delta t) + O(\Delta x)^2$ or $O\left[\Delta t, (\Delta x)^2\right]$. An understanding of the truncation error for a particular scheme is important.

Using the model equation give here, we define the terms in the schematic given above:

discretization

This is the process of replacing derivatives by finite difference approximations. Replace continuous derivatives with an approximation at a discrete set of points (the mesh). This introduces an error due to the truncation error arising from the finite difference approximation

and any errors due to treatment of BC's. A reexamination of the Taylor series representation is worthwhile in thinking about the possible error arising from the discretization process:

$$\frac{\partial f}{\partial x} = \frac{f(x_0 + \Delta x) - f(x_0 - \Delta x)}{2\Delta x} + \underbrace{\frac{\Delta x^2}{6}\frac{\partial^3 f}{\partial x^3}}_{\substack{\text{formally valid for } \Delta x \to 0, \\ \text{but when } \Delta x = \text{finite, } \partial^3 f/\partial x^3 \\ \text{can be big for rapidly changing} \\ \text{solutions (shock wave cases)}}} . \qquad (8\text{-}5a)$$

Thus we see that the size of the truncation error will depend locally on the solution. In most cases we expect the discretization error to be larger than round-off error.

consistency

A finite-difference representation of a PDE is *consistent* if the difference between the PDE and its difference representation vanishes as the mesh is refined, i.e.,

$$\lim_{mesh \to 0} (\text{PDE} - \text{FDE}) = \lim_{mesh \to 0} (\text{T.E.}) = 0 \qquad (8\text{-}23)$$

When might this be a problem? Consider a case where the truncation error is O($\Delta t/\Delta x$). In this case we must let the mesh go to zero just such that:

$$\lim_{\Delta t, \Delta x \to 0} \left(\frac{\Delta t}{\Delta x} \right) = 0 \qquad (8\text{-}24)$$

Some finite difference representations have been tried that weren't consistent. An example cited by Tannehill, *et al*,[1] is the DuFort-Frankel differencing of the wave equation.

stability

A stable numerical scheme is one for which errors from any source (round-off, truncation) are not permitted to grow in the sequence of numerical procedures as the calculation proceeds from one marching step, or iteration, to the next, thus:

errors grow → unstable

errors decay → stable

and

- Stability is normally thought of as being associated with marching problems.
- Stability requirements often dictate allowable step sizes.
- In many cases a stability analysis can be made to define the stability requirements.

convergence

The solution of the FDE's should approach the solution of the PDE as the mesh is refined. In the case of a linear equation there is a theorem which proves that the numerical solution to the FDE is in fact the solution of the original partial differential equation.

> **Lax Equivalence Theorem**[1] (linear, initial value problem): For a properly posed problem, with a consistent finite difference representation, stability is the necessary and sufficient condition for convergence.

In practice, numerical experiments must be conducted to determine if the solution appears to be converged with respect to mesh size.[*] Machine capability and computing budget (time as well as money) dictate limits to the mesh size. Many, many results presented in the literature are not completely converged with respect to the mesh.

So far we have represented the PDE by an FDE at the point *i,n*. The PDE is now a set of algebraic equations written at each mesh point. If the grid is (in three dimensions) defined by a grid with *IMAX*, *JMAX* and *KMAX* mesh points in each direction, then we have a grid with $IMAX \times JMAX \times KMAX$ grid points. This can be a very large number. A typical recent case computed by one of my students was for the flow over a simple aircraft forebody. The calculation required 198,000 grid points. Thus the ability to carry out aerodynamic analysis using finite difference methods depends on the ability to solve large systems of algebraic equations efficiently.

We need to obtain the solution for the values at each grid point. We now consider how this is actually accomplished. Since the computer requirements and approach are influenced by the mathematical type of the equation being solved, we illustrate the basic types of approaches to the solution with two examples.

1st example - typical parabolic/hyperbolic PDEs

Explicit Scheme: Consider the finite difference representation of the heat equation given above in Eq. (8-18). Using the notation shown in the Fig. 8-6 below, we write the finite difference representation as:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{\alpha}{(\Delta x)^2}\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right) \tag{8-25}$$

and the solution at time step *n* is known. At time *n*+1 there is only one unknown.

[*] This is convergence with respect to grid. Another convergence requirement is associated with the satisfaction of the solution of a system of equations by iterative methods on a fixed grid.

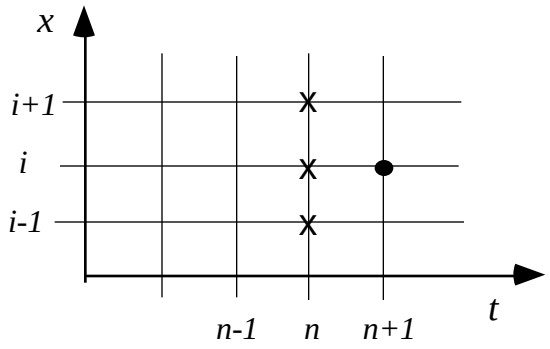


Figure 8-6. Grid points used in typical explicit calculation.

We solve for the value of u at i and the n+1 time step:

$$u_i^{n+1} = u_i^n + \alpha\left(\frac{\Delta t}{(\Delta x)^2}\right)\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right) \tag{8-26}$$

and thus at each i on n+1 we can solve for u_i^{n+1} algebraically, without solving a system of equations. This means that we can solve for each new value explicitly in terms of known values from the previous time step. This type of algorithm is known as an explicit scheme. It is a very straight forward procedure. To summarize:

- The algebra is simple.
- The bad news for non-vector computers: stability requirements require very small steps sizes.
- The good news: this scheme is easily vectorized* and a natural for massively parallel computation.

Implicit Scheme: Now consider an alternate finite difference representation of the heat equation given above in Eq. (8-18). Use the notation shown in the Fig. 8-7 below to define the location of grid points used to define the finite difference representation.

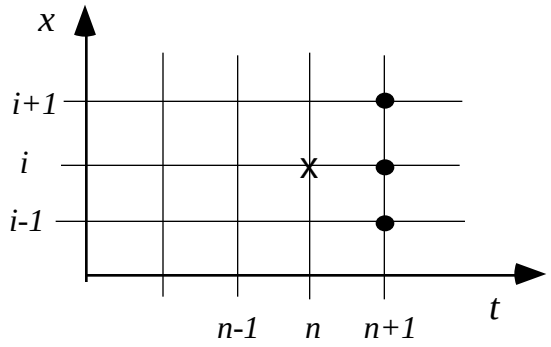


Figure 8-7. Grid points used in typical implicit calculation.

* see Chapter 3 for a brief discussion of vectorization.

Now we write the finite difference representation as:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{\alpha}{(\Delta x)^2}\left(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}\right) \tag{8-27}$$

where we use the spatial derivative at time n+1. By doing this we obtain a system where at each i on n+1, u_i^{n+1} depends on all the values at n+1. Thus we need to find the values along n+1 simultaneously. This leads to a system of algebraic equations that must be solved. For our model problem this system is linear. We can see this more clearly by rearranging Eq. (8-27). Defining

$$\lambda = \alpha \frac{\Delta t}{(\Delta x)^2} \tag{8-28}$$

we can re-write Eq.(8-27) after some minor algebra as:

$$-\lambda u_{i-1}^{n+1} + (1 + 2\lambda)u_i^{n+1} - \lambda u_{i+1}^{n+1} = u_i^n \qquad \text{for } i = 1,\ldots, N\,. \tag{8-29}$$

This can be put into a matrix form to show that it has a particularly simple form:

$$\begin{bmatrix} (1+2\lambda) & -\lambda & 0 & 0 & 0 & 0 & 0 \\ -\lambda & (1+2\lambda) & -\lambda & 0 & \ddots & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ \ddots & \ddots & -\lambda & (1+2\lambda) & -\lambda & \ddots & \ddots \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & \ddots & 0 & -\lambda & (1+2\lambda) & -\lambda \\ 0 & 0 & 0 & 0 & 0 & -\lambda & (1+2\lambda) \end{bmatrix} \begin{bmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_i^{n+1} \\ \vdots \\ u_{N-1}^{n+1} \\ u_N^{n+1} \end{bmatrix} = \begin{bmatrix} u_1^n \\ u_2^n \\ \vdots \\ u_i^n \\ \vdots \\ u_{N-1}^n \\ u_N^n \end{bmatrix} \tag{8-30}$$

Equation (8-30) is a special type of matrix form known as a tridiagonal form. A particularly easy solution technique is available to solve this form. Known as the Thomas algorithm, the details are described in Section 8.5 and a routine called **tridag** is described in Appendix H-1. Many numerical methods are tailored to be able to produce this form.

The approach that leads to the formulation of a problem requiring the simultaneous solution of a system of equations is known as an *implicit* scheme. To summarize:

- The solution of a system of equations is required at each step.
- The good news: stability requirements allow a large step size.
- The not so good news: this scheme is harder to vectorize/parallelize.

A common feature for both explicit and implicit methods for parabolic and hyperbolic equations:

- A large number of mesh points can be treated, you only need the values at a small number of marching stations at any particular stage in the solution. This means you can obtain the solution with a large number of grid points using a relatively small amount of memory. Curiously, some recent codes don't take advantage of this last fact.

2nd Example - elliptic PDE

We use Laplace's equation as the model problem for elliptic PDE's:

$$\varphi_{xx} + \varphi_{yy} = 0 \tag{8-31}$$

and consider the grid shown below in Figure 8-8.

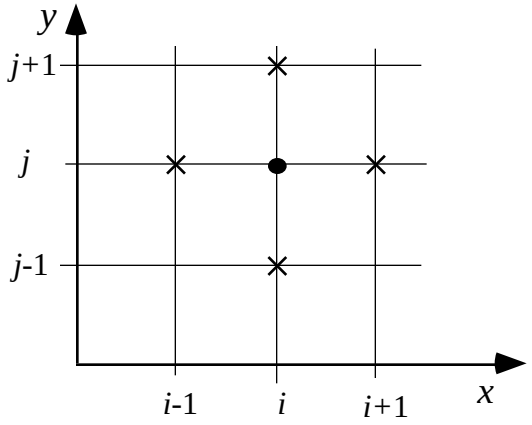


Figure 8-8. Grid points used in a typical representation of an elliptic equation.

Use the second order accurate central difference formulas at i,j:

$$\varphi_{xx} = \frac{\varphi_{i+1,j} - 2\varphi_{i,j} + \varphi_{i-1,j}}{(\Delta x)^2} + O(\Delta x)^2 \tag{8-32}$$

and:

$$\varphi_{yy} = \frac{\varphi_{i,j+1} - 2\varphi_{i,j} + \varphi_{i,j-1}}{(\Delta y)^2} + O(\Delta y)^2, \tag{8-33}$$

and substitute these expressions into the governing equation:

$$\frac{\varphi_{i+1,j} - 2\varphi_{i,j} + \varphi_{i-1,j}}{(\Delta x)^2} + \frac{\varphi_{i,j+1} - 2\varphi_{i,j} + \varphi_{i,j-1}}{(\Delta y)^2} = 0 \tag{8-34}$$

Solve this equation for φ_{ij}:

$$\varphi_{i,j} = \frac{(\Delta y)^2}{2\left[(\Delta x)^2 + (\Delta y)^2\right]}\left(\varphi_{i+1,j} + \varphi_{i-1,j}\right) + \frac{(\Delta x)^2}{2\left[(\Delta x)^2 + (\Delta y)^2\right]}\left(\varphi_{i,j+1} + \varphi_{i,j-1}\right) \tag{8-35}$$

where if $\Delta x = \Delta y$:

$$\varphi_{i,j} = \frac{1}{4}\left(\varphi_{i+1,j} + \varphi_{i-1,j} + \varphi_{i,j+1} + \varphi_{i,j-1}\right) \tag{8-36}$$

This expression illustrates the essential physics of flows governed by elliptic PDE's:

- φ_{ij} depends on all the values around it
- *all* values of φ must be found simultaneously
- computer storage requirements are much greater than those required for parabolic/hyperbolic PDE's

Because of the large number of mesh points required to resolve the flowfield details, it is generally not practical to solve the system of equations arising from applying the above equation at each mesh point directly. Instead, an iterative procedure is usually employed. In this procedure an initial guess for the solution is made and then each mesh point in the flowfield is updated repeatedly until the values satisfy the governing equation. This iterative procedure can be thought of as having a time-like quality, which has been exploited in many solution schemes to find the steady flowfield.

A Note on Conservation Form

Care must be taken if the flowfield has discontinuities (shocks). In that case the correct solution of the partial differential equation will only be obtained if the conservative forms of the governing equations are used.

8.3 Other approaches, including the finite volume technique

Finite difference methods are the most well known methods in CFD. However other methods have also proven successful, and one method in particular, the finite volume technique, actually forms the basis for most current successful codes. The other methods in use are categorized as finite element and spectral. Each method eventually leads to a large set of algebraic equations, just as with the finite difference methods. See References 1 and 3 for more details of the latter two methods. In US aircraft aerodynamics work they don't currently have an impact.

The finite volume method *is* important. Instead of discretizing the PDE, select the integral form of the equations. Recall that each conservation law had both differential and integral statements. The integral form is more fundamental, not depending on continuous partial derivatives.

Example of Finite Volume Approach (Fletcher,[3] vol. I, pg.105-116, Tannehill, *et al*,[1] pg 71-76)

Consider the general conservation equation (in two dimensions for our example analysis):

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} = 0 . \qquad (8\text{-}37)$$

Pick the particular form to be conservation of mass:

$$\begin{aligned} \mathbf{q} &= \rho \\ \mathbf{F} &= \rho u, \\ \mathbf{G} &= \rho v \end{aligned} \tag{8-38}$$

and recall that this conservation law could also come from the integral statement:

$$\frac{\partial}{\partial t}\iiint \rho \, dV = -\oiint \rho \mathbf{V} \cdot \mathbf{n} \, dS . \tag{8-39}$$

Introducing the notation defined above and assuming two dimensional flow, the conservation law can be rewritten as:

$$\frac{\partial}{\partial t}\iint \mathbf{q} \, dV + \oint \mathbf{H} \cdot \mathbf{n} \, dS = 0 \tag{8-40}$$

where

$$\mathbf{H} = (\mathbf{F}, \mathbf{G}) = \rho \mathbf{V} \tag{8-41}$$

and

$$\begin{aligned} H_x &= F = \rho u \\ H_y &= G = \rho v \end{aligned} . \tag{8-42}$$

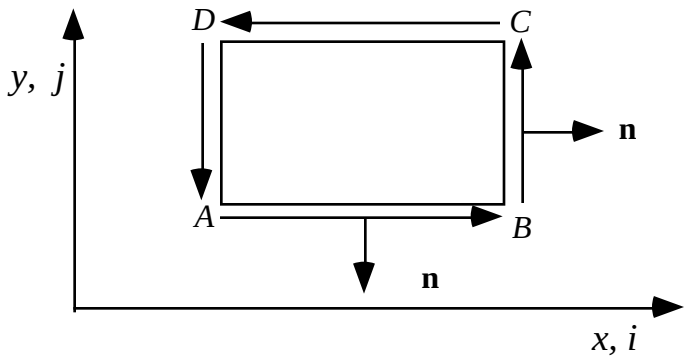


Figure 8-9. Basic nomenclature for finite volume analysis.

Using the definition of **n** in Cartesian coordinates, and considering for illustration the Cartesian system given in Fig. 8-9, we can write:

$$\begin{aligned} \mathbf{H} \cdot \mathbf{n} \, dS &= \left(H_x \mathbf{i} + H_y \mathbf{j}\right) \cdot \mathbf{n} \, dS \\ &= \left(F \mathbf{i} + G \mathbf{j}\right) \cdot \mathbf{n} \, dS \end{aligned} \tag{8-43}$$

along *AB*, $n = -j$, $dS = dx$, and:

$$\mathbf{H} \cdot \mathbf{n} \, dS = -G \, dx \tag{8-44}$$

along *BC*, $n = i$, $dS = dy$, and:

$$\mathbf{H} \cdot \mathbf{n} \, dS = F \, dy \tag{8-45}$$

or in general:

$$\mathbf{H}\ \mathbf{n}\,dS = F\,dy - G\,dx\,. \tag{8-46}$$

Using the general grid shown in the Fig. 8-10, our integral statement, Eq. (8-40) can be written as:

$$\frac{\partial}{\partial t}\left(A q_{j,k}\right) + \sum_{AB}^{DA}\left(F\Delta y - G\Delta x\right) = 0. \tag{8-47}$$

Here A is the area of the quadrilateral $ABCD$, and $q_{i,j}$ is the average value of q over $ABCD$.

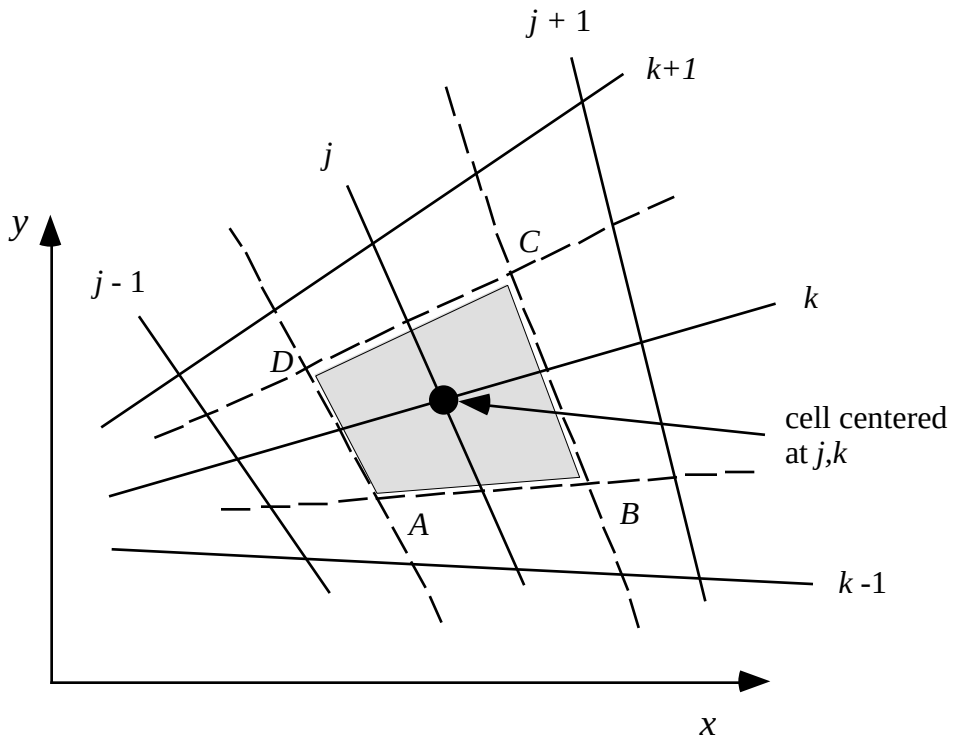


Figure 8-10. Circuit in a general grid system.

Now define the quantities over each face. For illustration consider AB:

$$\begin{aligned}
\Delta y_{AB} &= y_B - y_A \\
\Delta x_{AB} &= x_B - x_A \\
F_{AB} &= \frac{1}{2}\left(F_{j,k-1} + F_{j,k}\right)\ , \\
G_{AB} &= \frac{1}{2}\left(G_{j,k+1} + G_{j,k}\right)
\end{aligned} \tag{8-48}$$

and so on over the other cell faces.

Assuming A is not a function of time, and combining:

$$\begin{aligned} A\frac{\partial q_{j,k}}{\partial t} &+ \frac{1}{2}\left(F_{j,k-1} + F_{j,k}\right)\Delta y_{AB} - \frac{1}{2}\left(G_{j,k-1} + G_{j,k}\right)\Delta x_{AB} \\ &+ \frac{1}{2}\left(F_{j,k} + F_{j+1,k}\right)\Delta y_{BC} - \frac{1}{2}\left(G_{j,k} + G_{j+1,k}\right)\Delta x_{BC} \\ &+ \frac{1}{2}\left(F_{j,k} + F_{j,k+1}\right)\Delta y_{CD} - \frac{1}{2}\left(G_{j,k} + G_{j,k+1}\right)\Delta x_{CD}. \\ &+ \frac{1}{2}\left(F_{j-1,k} + F_{j,k}\right)\Delta y_{DA} - \frac{1}{2}\left(G_{j-1,k} + G_{j,k}\right)\Delta x_{DA} \\ &= 0 \end{aligned} \tag{8-49}$$

Supposing the grid is regular cartesian as shown in Fig. 8-11. Then $A = \Delta x \Delta y$, and along:

$$\begin{aligned} &AB: && \Delta y = 0, \quad \Delta x_{AB} = \Delta x \\ &BC: && \Delta x = 0, \quad \Delta y_{BC} = \Delta y \\ &CD: && \Delta y = 0, \quad \Delta x_{CD} = -\Delta x \\ &DA: && \Delta x = 0, \quad \Delta y_{DA} = -\Delta y \end{aligned} \tag{8-50}$$

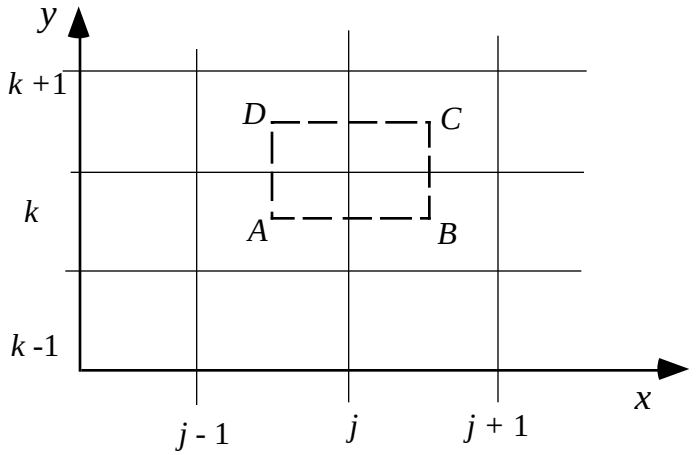


Figure 8-11. General finite volume grid applied in Cartesian coordinates.

Thus, in Eq. (8-49) we are left with:

$$\begin{aligned} \Delta x \Delta y \frac{\partial q_{j,k}}{\partial t} &- \frac{1}{2}\left(G_{j,k-1} + G_{j,k}\right)\Delta x + \frac{1}{2}\left(F_{j,k} + F_{j+1,k}\right)\Delta y \\ &+ \frac{1}{2}\left(G_{j,k} + G_{j,k+1}\right)\Delta x - \frac{1}{2}\left(F_{j-1,k} + F_{j,k}\right)\Delta y = 0 \end{aligned} \tag{8-51}$$

Collecting terms:

$$\frac{\partial q_{j,k}}{\partial t} + \underbrace{\frac{F_{j+1,k} - F_{j-1,k}}{2\Delta x} + \frac{G_{j,k+1} - G_{j,k-1}}{2\Delta y}}_{\text{for this reversion to Cartesian coordinates the equation just reduces to simple central differences of the original partial differential equation}} = 0 \tag{8-52}$$

or:

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} = 0 \,. \tag{8-53}$$

Thus, and at first glance remarkably, the results of the finite volume approach can lead to the exact same equations to solve as the finite difference method on a simple Cartesian mesh. However, the interpretation is different:*

- Finite difference: approximates the governing equation at a point
- Finite volume: " " " " over a volume
- Finite volume is the most physical in fluid mechanics codes, and is actually used in most codes.
- Finite difference methods were developed earlier, the analysis of methods is easier and further developed.

Both the finite difference and finite volume methods are very similar. However, there are differences. They are subtle but important. We cite three points in favor of the finite volume method compared to the finite difference method:

- Good conservation of mass, momentum, and energy using integrals when mesh is finite size
- Easier to treat complicated domains (integral discretization [averaging] easier to figure out, implement, and interpret)
- Average integral concept much better approach when the solution has shock waves (i.e. the partial differential equations assume continuous partial derivatives).

Finally, special considerations are needed to implement some of the boundary conditions in this method. The references, in particular Fletcher,[3] should be consulted for more details.

8.4 Boundary conditions

So far we have obtained expressions for interior points on the mesh. However, the actual geometry of the flowfield we wish to analyze is introduced through the boundary conditions. We use an elliptic PDE problem to illustrate the options available for handling boundary conditions.

Consider the flow over a symmetric airfoil at zero angle of attack, as shown in Fig. 8-12.

* Summarized from Professor B. Grossman's unpublished CFD notes.

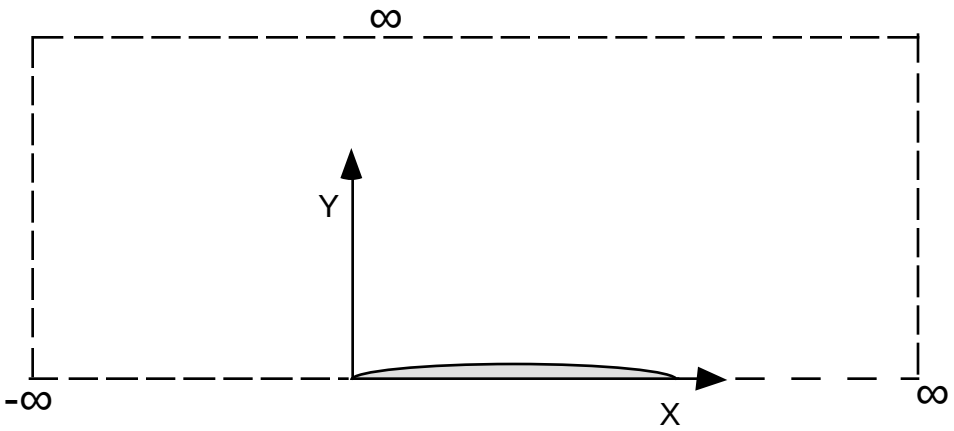


Figure 8-12. Example of boundary condition surface requiring consideration.

Here, because there is no lift, symmetry allows us to solve only the top half of the region. If φ is a perturbation potential [see Chap. 2, Eq. (2-123)],

$$\begin{aligned} U &= U_\infty + \varphi_x \\ V &= \varphi_y \end{aligned}, \tag{8-54}$$

then far away from the surface,

$$u = v = 0 \tag{8-55}$$

or

$$\varphi \to 0 \quad \text{as} \quad x^2 + y^2 \to \infty. \tag{8-56}$$

For a lifting airfoil, the farfield potential must take the form of a potential vortex singularity with a circulation equal to the circulation around the airfoil.

The boundary condition on the surface of primary interest is the flow tangency condition, where the velocity normal to the surface is specified. In most cases the velocity normal to the surface is zero.

Consider ways to handle the farfield BC

There are several possibilities:

A. "go out" far enough (?) and set $\varphi = 0$ for $\varphi \to 0$, as the distance from the body goes to infinity (or $v = 0$, $u = 0$ where these are the perturbation velocities, or $u = U_\infty$ if it is the total velocity).

   How good is this? This method is frequently used, although clearly it requires numerical experimentation to ensure that the boundary is "far enough" from the body. In lifting cases this can be on the order of 50 chord lengths in two-dimensions. In

addition, this approach leads to excessive use of grid points in regions where we normally aren't interested in the details of the solution.

B. Transform the equation to another coordinate system, and satisfy the boundary condition explicitly at infinity (details of this approach are given in Chap. 9).

Figure 8-13 demonstrates what we mean. In the ξ system the physical distance from 0 to infinity is transformed to the range from 0 to 1. Although this approach may lead to efficient use of grid points, the use of the resulting highly stretched grid in the physical plane may result in numerical methods that lose accuracy, and even worse, do not converge during an iterative solution.

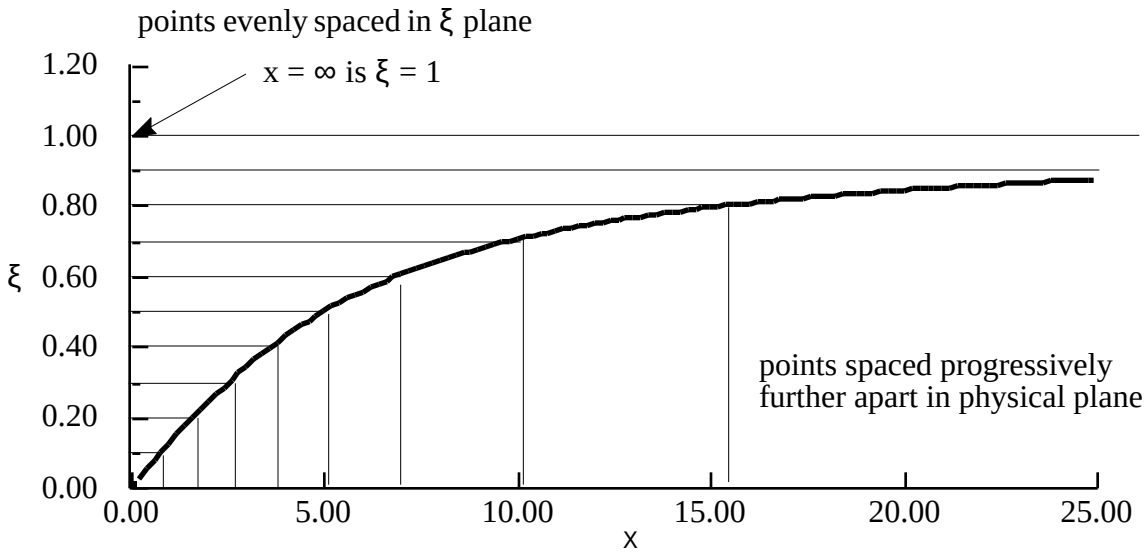


Figure 8-13. One example of a way to handle the farfield boundary condition.

C. Blocks of Grids are sometimes used, a dense "inner" grid and a "coarse" outer grid. In this approach the grid points are used efficiently in the region of interest. It is a simple version of the adaptive grid concept, where the the grid will adjust automatically to concentrate points in regions of large flow gradients.

D. Match the numerical solution to an analytic approximation for the farfield boundary condition.

This is emerging as the standard way to handle the farfield boundary conditions. It allows the outer boundary to be placed at a reasonable distance from the body, and properly done, it ensures that the boundary numerical solution reflects the correct physics at the boundary. This has been found to be particularly important in the solution

of the Euler equations. Effort is still underway to determine the best way to implement this approach.

To summarize this discussion on farfield boundary conditions:

- BC's on the FF boundary are important, and can be especially important for Euler codes which march in time to a steady state final solution.
- How to best enforce the FF BC is still under study - research papers are still being written describing new approaches.

Consider ways to handle the nearfield BC

There are are also several ways to approach the satisfaction of boundary conditions on the surface. Here we discuss three.

A. Use a standard grid and allow the surface to intersect grid lines in an irregular manner. Then, solve the equations with BC's enforced between node points. Figure 8-14 illustrates this approach. In the early days of CFD methodology development this approach was not found to work well, and the approach discussed next was developed. However, using the finite volume method, an approach to treat boundary conditions imposed in this manner was successfully developed (primarily by NASA Langley and its contractors and grantees). It has not become a popular approach, and is considered to lead to an inefficient use of grid points. Many grid points end up inside the body.

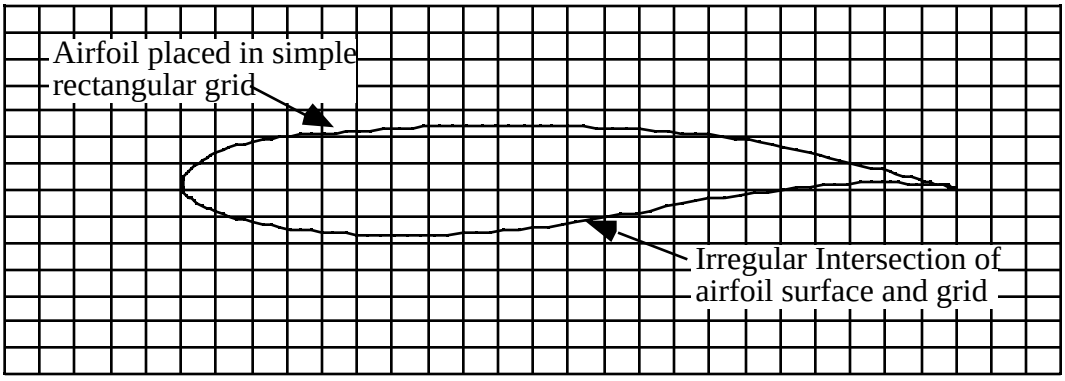
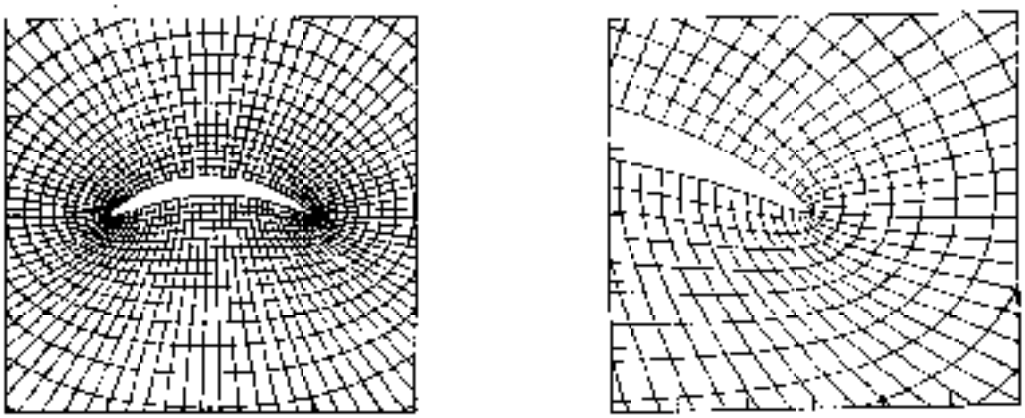


Figure 8-14. Surface passing through a general grid.

B. The most popular approach to enforcing surface boundary conditions is to use a coordinate system constructed such that the surface of the body is a coordinate surface. An example of this approach is shown in Figure 8-15. This is currently the method of choice and by far the most popular approach employed in CFD. It works well. However, it complicates the problem formulation. To use this approach, grid generation became an area of study by itself. Grid generation is discussed in Chapter 9.



a. Entire geometry b. Closeup of the trailing edge

Figure 8-15. Body conforming grid for easy application of BCs on curved surfaces.[10]

C. Another approach is to use thin airfoil theory boundary conditions, as described in detail in Chapter 6. This eliminates many of the problems associated with the first two approaches. It is expedient, but at some loss in accuracy (but very likely not that much, as shown in Chap. 6, Fig. 6-14).

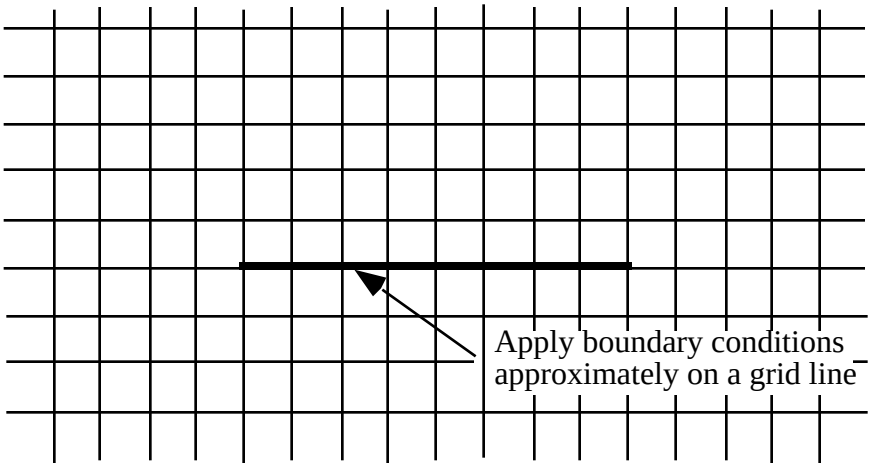


Figure 8-16. Approximate approach to boundary condition specification.

Finite difference representation of the BC's

After defining a coordinate system, the finite difference representation of the boundary condition must be written down. Using Laplace's Equation as an example, consider that there are normally two types of boundary conditions associated with the boundary: 1) the Dirichlet problem, where φ is specified on the boundary, and 2) the Neumann problem, where $\partial\varphi/\partial n$ is specified. If the Dirichlet problem is being solved, the value on the boundary is simply specified and no special difference formulas are required.

When the solution requires that the gradient normal to the surface be specified, a so-called "dummy row" is the easiest way to implement the boundary condition. As an example, following Moran,[4] consider a case where the normal velocity, *v*, is set to zero at the outer boundary. The boundary is at grid line *j* = *NY*. Assume that another row is added at *j* = *NY* + 1, as indicated in Fig. 8-17.

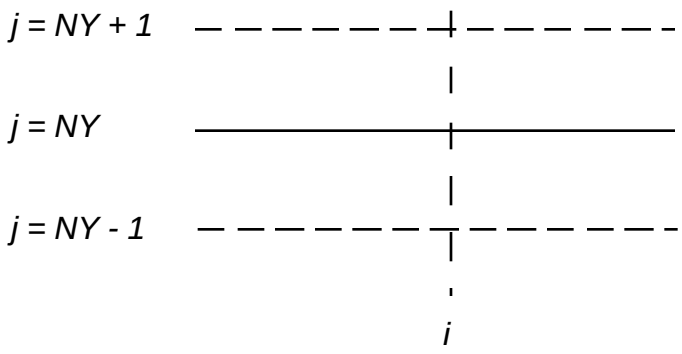


Figure 8-17. Boundary condition at farfield.

The required boundary condition at *j* = NY is:

$$\frac{\partial \varphi}{\partial n} = 0 = \frac{\varphi_{i,NY+1} - \varphi_{i,NY-1}}{Y_{NY+1} - Y_{NY-1}} + O(\Delta Y)^2 \tag{8-57}$$

and to ensure that the boundary condition is satisfied, simply define:

$$\varphi_{i,NY+1} \equiv \varphi_{i,NY-1}. \tag{8-58}$$

The equations are then solved up to Y_{NY}, and whenever you need φ at *NY*+1, simply use the value at *NY*-1.

Now, we present an example demonstrating the application of thin airfoil theory boundary conditions at the surface. Recall that the boundary condition is:

$$v = \frac{\partial \varphi}{\partial y} = U_\infty \frac{df}{dx} \tag{8-59}$$

where $y_{\text{foil}} = f(x)$. Assuming that in the computer code *v* has been nondimensionalized by U_∞, the boundary condition is:

$$\frac{\partial \varphi}{\partial y} = \frac{df}{dx} \tag{8-60}$$

and the grid near the surface is defined following Fig. 8-18.

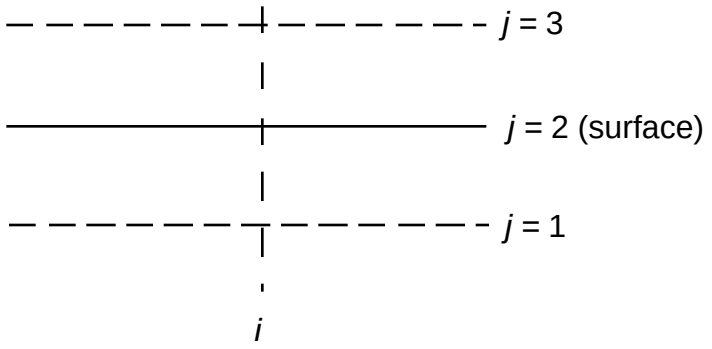


Figure 8-18. Boundary condition at surface.

Writing the derivative in terms of central differences at j =2,

$$\frac{\varphi_{i,3} - \varphi_{i,1}}{Y_3 - Y_1} = \frac{df}{dx} \tag{8-61}$$

we solve for $\varphi_{i,1}$:

$$\varphi_{i,1} = \varphi_{i,3} - \left(Y_3 - Y_1\right)\frac{df}{dx}. \tag{8-62}$$

Note that since j = 1 is a dummy row, you can select the grid spacing such that the spacing is equal on both sides of j = 2, resulting in second order accuracy. Thus, as in the previous example, anytime we need $\varphi_{i,1}$ we use the value given by Eq. (8-62). Using these boundary condition relations, the boundary conditions are identically satisfied. Note also that this approach is the reason that in many codes the body surface corresponds to the second grid line, j = 2.

Imposition of boundary conditions is sometimes more difficult than the analysis given here suggests. Specifically, both the surface and farfield boundary conditions for the pressure in the Navier-Stokes and Euler equations can be tricky.

8.5 Solution of Algebraic Equations

We now know how to write down a representation of the PDE at each grid point. The next step is to solve the resulting system of equations. Recall that we have one algebraic equation for each grid point. The system of algebraic equations may, or may not, be linear. If they are nonlinear, the usual approach is to form an approximate linear system, and then solve the system iteratively to obtain the solution of the original nonlinear system. The accuracy requirement dictates the number of the grid points required to obtain the solution. Previously, we assumed that linear equation solution subroutines were available, as discussed in Chapter 3. However, the development of CFD methods requires a knowledge of the types of algebraic systems of equations.

Recall that linear algebraic equations can be written in the standard form:

$$\mathbf{A}x = b\,. \tag{8-63}$$

For an inviscid two-dimensional solution, a grid of 100 x 30 is typical. This is 3000 grid points, and results in a matrix 3000 x 3000. In three dimensions, 250,000 300,000 grid points are common, 500,000 points are not uncommon, and a million or more grid points are often required. Clearly, you can't expect to use classical direct linear equation solvers for systems of this size.

Standard classification of algebraic equations depends on the characteristics of the elements in the matrix **A.** If **A**:

1. contains few or no zero coefficients, it is called dense,
2. contains many zero coefficients, it is called sparse,
3. contains many zero coefficients, *and* the non-zero coefficients are close to the main diagonal: the **A** matrix is called sparse and banded.

Dense Matrix

For a dense matrix direct methods are appropriate. Gauss elimination is an example of the standard approach to these systems. LU decomposition[11] is used in program **PANEL**, and is an example of a standard method for solution of a dense matrix. These methods are not good for large matrices (> 200-400 equations). The run time becomes huge, and the results may be susceptible to round-off error.

Sparse and Banded

Special forms of Gauss elimination are available in many cases. The most famous banded matrix solution applies to so-called tridiagonal systems:

$$\begin{bmatrix} b_1 & c_1 & & & & & \\ a_2 & b_2 & c_2 & & & 0 & \\ & \ddots & \ddots & \ddots & & & \\ & & a_i & b_i & c_i & & \\ & & & \ddots & \ddots & \ddots & \\ & 0 & & & a_{N-1} & b_{N-1} & c_{N-1} \\ & & & & & a_N & b_N \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_i \\ \vdots \\ d_{N-1} \\ d_N \end{bmatrix} \tag{8-64}$$

The algorithm used to solve Eq. (8-64) is known as the Thomas algorithm. This algorithm is very good and widely used. The Thomas algorithm is given in detail in the sidebar, and a sample subroutine, **tridag**, is described in App H-1.

Solution of tridiagonal systems of equations

The Thomas Algorithm is a special form of Gauss elimination that can be used to solve tridiagonal systems of equations. When the matrix is tridiagonal, the solution can be obtained in O(n) operations, instead of O($n^3/3$). The form of the equation is:

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i \qquad i = 1,\ldots,n$$

where a_1 and c_n are zero. The solution algorithm[12] starts with k = 2,....,n:

$$m = \frac{a_k}{b_{k-1}}$$

$$b_k = b_k - m c_{k-1} .$$

$$d_k = d_k - m d_{k-1}$$

Then:

$$x_n = \frac{d_n}{b_n}$$

and finally, for $k = n$ - 1,...1:

$$x_k = \frac{d_k - c_k x_{k+1}}{b_k} .$$

In CFD methods this algorithm is usually coded directly into the solution procedure, unless machine optimized subroutines are employed on a specific computer.

General Sparse

These matrices are best treated with iterative methods. In this approach an initial estimate of the solution is specified (often simply 0), and the solution is then obtained by repeatedly updating the values of the solution vector until the equations are solved. This is also a natural method for solving nonlinear algebraic equations, where the equations are written in the linear equation form, and the coefficients of the **A** matrix are changed as the solution develops during the iteration. Many methods are available.

There is one basic requirement for iterative solutions to converge. The elements on the diagonal of the matrix should be large relative to the values off the diagonal. The condition can be give mathematically as:

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \qquad (8\text{-}65)$$

and for at least one row:

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \tag{8-66}$$

A matrix that satisfies this condition is diagonally dominant, and, for an iterative method to converge, the matrix must be diagonally dominant. One example from aerodynamics of a matrix that arises which is not diagonally dominant is the matrix obtained in the monoplane equation formulation for the solution of the lifting line theory problem.

One class of iterative solution methods widely used in CFD is "relaxation." As an example, consider Laplace's Equation. Start with the discretized form, Eq.(8-31). The iteration proceeds by solving the equation at each grid point *i,j* at an iteration *n*+1 using values found at iteration *n*. Thus the solution at iteration *n*+1 is found from:

$$\varphi_{i,j}^{n+1} = \frac{1}{4}\left[\varphi_{i+1,j}^{n} + \varphi_{i-1,j}^{n} + \varphi_{i,j+1}^{n} + \varphi_{i,j-1}^{n}\right]. \tag{8-67}$$

The values of φ are computed repeatedly until they are no longer changing. The "relaxation" of the values of φ to final converged values is roughly analogous to determining the solution for an unsteady flow approaching a final steady state value, where the iteration cycle is identified as a time-like step. This is an important analogy. Finally, the idea of "iterating until the values stop changing" as an indication of convergence is not good enough. Instead, we must check to see if the finite difference representation of the partial differential equation using the current values of φ actually satisfies the partial differential equation. In this case, the value of the equation should be zero, and the actual value of the finite difference representation is know as the *residual*. When the residual is zero, the solution has converged. This is the value that should be monitored during the iterative process. Generally, as done in **THINFOIL**, the maximum residual and its location in the grid, and the average residual are computed and saved during the iterative process to examine the convergence history.

Note that this method uses all old values of φ to get the new value of φ. This approach is known as *point Jacoby* iteration. You need to save all the old values of the array as well as the new values. This procedure converges only very slowly to the final converged solution.

A more natural approach to obtaining the solution is to use new estimates of the solution as soon as they are available. Figure 8-19 shows how this is done using a simply programmed systematic sweep of the grid. With a conventional sweep of the grid this becomes:

$$\varphi_{i,j}^{n+1} = \frac{1}{4}\left[\varphi_{i+1,j}^{n} + \varphi_{i-1,j}^{n+1} + \varphi_{i,j+1}^{n} + \varphi_{i,j-1}^{n+1}\right]. \tag{8-68}$$

This scheme is called the *point Gauss-Seidel* iteration. It also elliminates the need to store all the old iteration values as well as all the new iteration results, which was required with the point Jacoby method.

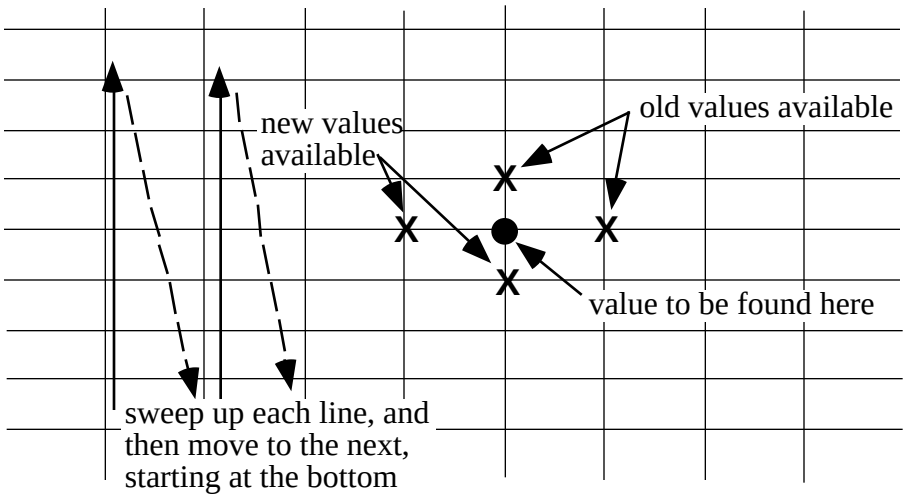


Figure 8-19. Grid sweep approach to implement the Gauss-Seidel solution iteration scheme.

The point Gauss-Seidel iteration procedure also converges slowly. One method of speeding up the convergence is to make the change to the value larger than the change indicated by the normal Gauss-Seidel iteration. Since the methods that have been described are known as relaxation methods, the idea of increasing the change is known as successive over-relaxation, or SOR. This is implemented by defining an intermediate value:

$$\hat{\varphi}_{i,j}^{n+1} = \frac{1}{4}\left[\varphi_{i+1,j}^{n} + \varphi_{i-1,j}^{n+1} + \varphi_{i,j+1}^{n} + \varphi_{i,j-1}^{n+1}\right] \tag{8-69}$$

and then obtaining the new value as:

$$\varphi_{i,j}^{n+1} = \varphi_{i,j}^{n} + \omega\left(\hat{\varphi}_{i,j}^{n+1} - \varphi_{i,j}^{n}\right). \tag{8-70}$$

The parameter ω is a relaxation parameter. If it is unity, the basic Gauss-Seidel method is recovered. How large can we make it? For most model problems, a stability analysis (presented in the next section) indicates that $\omega < 2$ is required to obtain a converging iteration. The best value of ω depends on the grid and the actual equation. For most cases of practical interest the best values of ω must be determined through numerical experimentation. Figure 8-20 presents an example of the manner in which the solution evolves with iterations. The value of φ after 2000 iterations is approached very gradually. The figure also illustrates the time-like nature of the iteration.

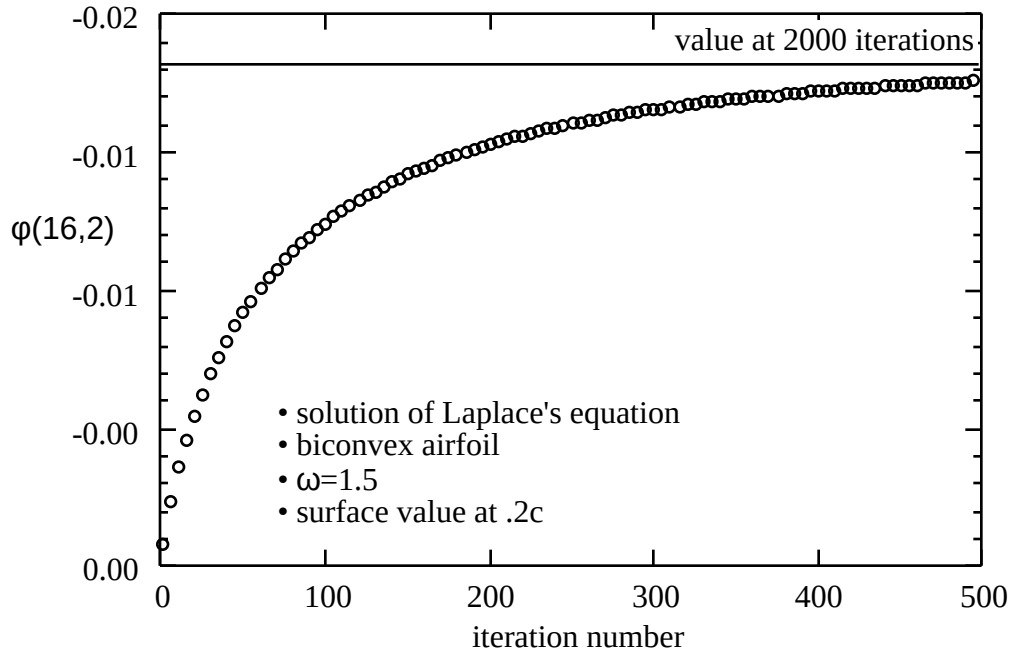


Figure 8-20. Typical variation of φ during solution iteration.

Another way to speed up the iteration is to sweep the flowfield a “line” at a time rather than a point at a time. Applying over-relaxation to this process, the so-called successive line over-relaxtion, or SLOR, process is obtained. In this method a system of equations must be solved at each line. Figure 8-21 illustrates this approach. The method is formulated so that the system of equations is tridiagonal, and the solution is obtained very efficiently. This approach provides a means of spreading the information from new values more quickly than the point by point sweep of the flowfield. However, all of these approaches result in a very slow approach to the final value during the iterations.

The effect of the value of the over relaxation parameter is shown in Figure 8-22. Here, the convergence level is compared for various values of ω. Notice that as convergence requirements are increased, the choice of ω becomes much more important. Unfortunately, the choice of ω may not only be dependent on the particular numerical method, but also on the particular problem being solved.

Mathematically, the convergence rate of an iterative process depends on the value of the so-called *spectral radius* of the matrix relating the value of the unknowns at one iteration to the values of the unknowns at the previous iteration. The spectral radius is the absolute value of the largest eigenvalue of the matrix. The spectral radius must be less than one for the iterative process to converge. The smaller the value of the spectral radius, the faster the convergence.

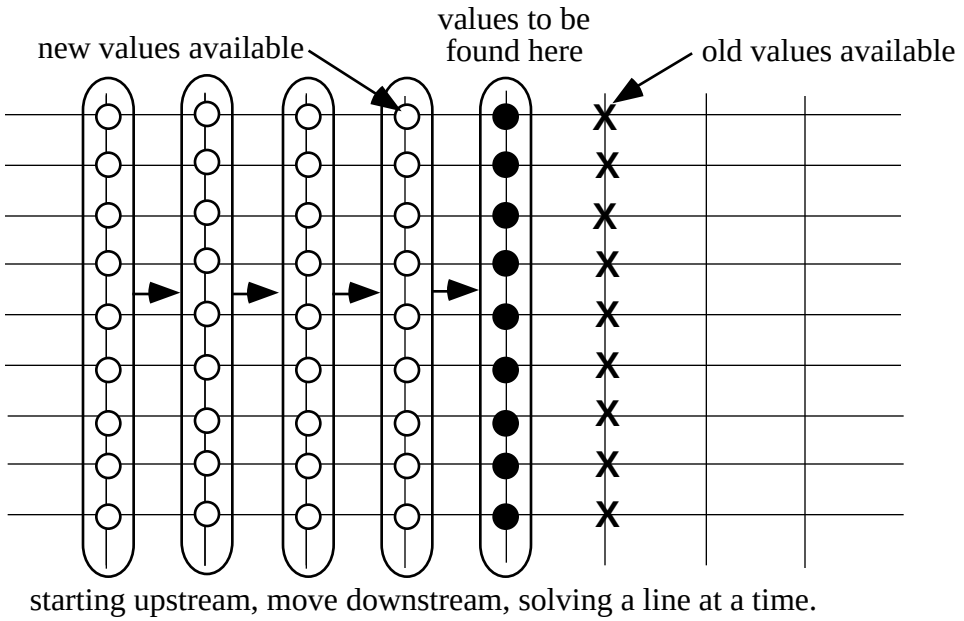


Figure 8-21. Solution approach for SLOR.

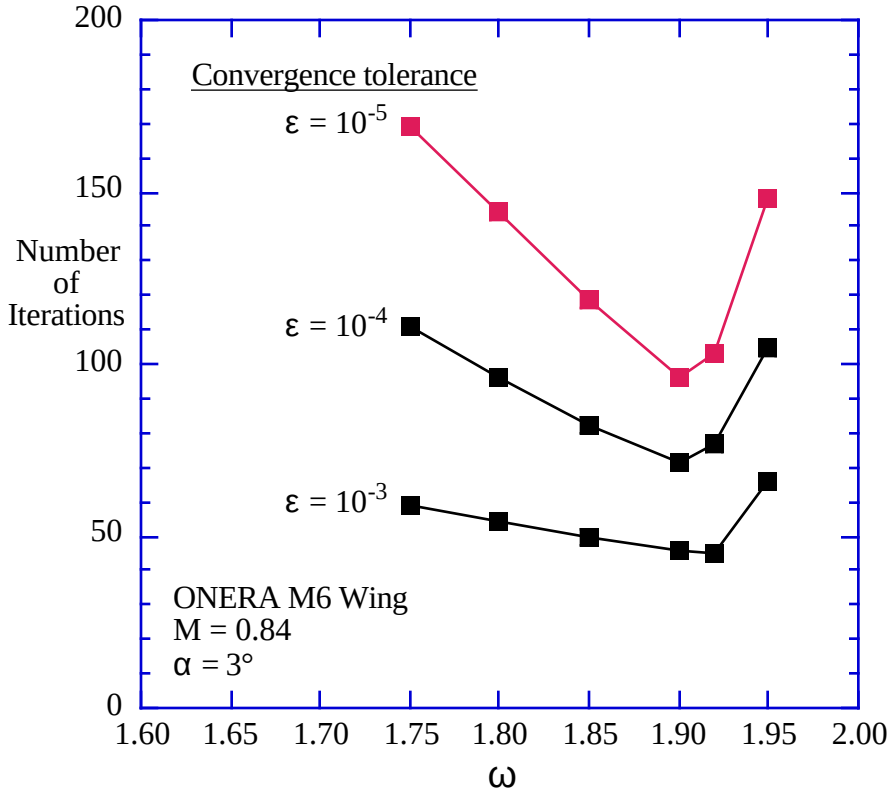


Figure 8-22. Effect of the value of ω on the number of iterations required to achieve various levels of convergence.[13]

Another way to spread the information rapidly is to alternately sweep in both the *x* and *y* direction. This provides a means of obtaining the final answers even more quickly, and is known

as an alternating direction implicit or ADI method. Figure 8-23 illustrates the modification to the SLOR method that is used to implement an ADI scheme. Several different methods of carrying out the details of this iteration are available. The traditional approach for linear equations is known as the Peaceman-Rachford method, and is described in standard textbooks, *e.g.*, Ames[8] and Isaacson and Keller.[14] This approach is also known as an approximate factorization or "AF" scheme. It is known as AF1 because of the particular approach to the factorization of the operator. A discussion of ADI including a computer program is given in the first edition of the *Numerical Recipes* book.[15]

Another approach has been found to be more robust for nonlinear partial differential equations, including the case of mixed sub- and supersonic flow. In this case the time-like nature of the approach to a final value is used explicitly to develop a robust and rapidly converging iteration scheme. This scheme is known as AF2. This method was first proposed for steady flows by Ballhaus, et al,[16] and Catherall[17] provided a theoretical foundation and results from numerical experiments. A key aspect of ADI or any AF scheme is the use of a sequence of relaxation paramters rathers a single value, as employed in the SOR and SLOR methods. Typically, the sequence repeats each eight to eleven iterations.

Holst[10] has given an excellent review and comparison of these methods. Figure 8-24, from Holst,[10] shows how the different methods use progressively "better" information at a point to find the solution with the fewest possible iterations. The advantage is shown graphically in Figure 8-25, and is tabulated in Table 8-1 (also from Holst[10]). Program **THINFOIL**, described in Section 8.7, uses these methods and App. G-1 contains a description of the theoretical implementation of these methods. Further details are given in Chapter 11, Transonic Aerodynamics.

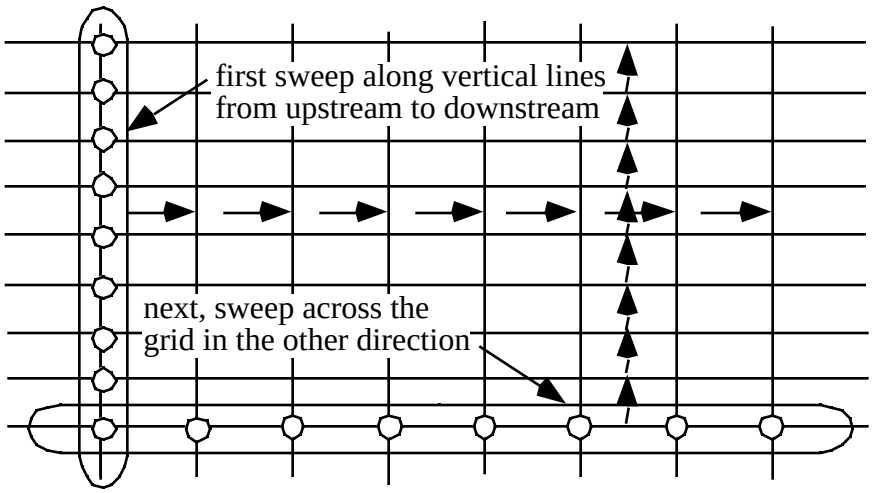


Figure 8-23. ADI Scheme solution approach.

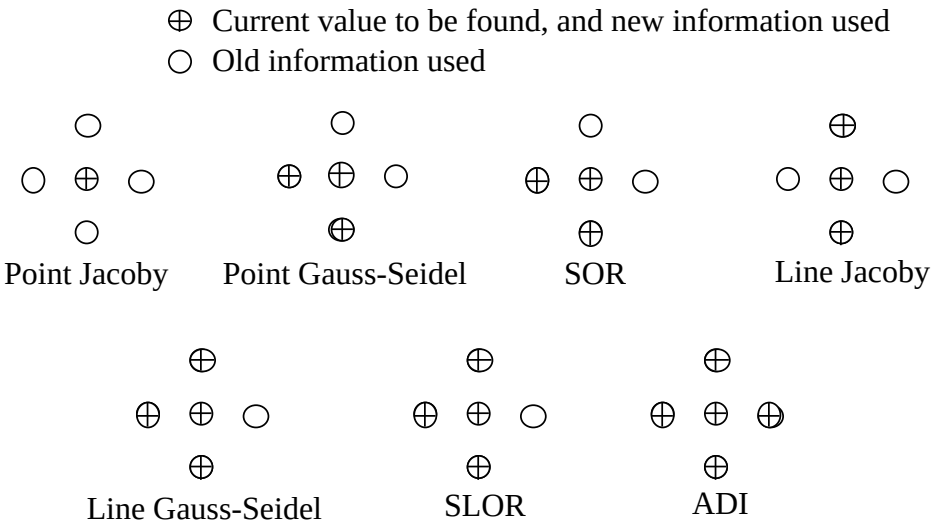


Figure 8-24. Stencil of information (Holst[10])

In addition to these methods, solutions can be obtained more rapidly by using so-called *multigrid* methods. These methods accelerate the convergence iterative procedures by using a sequence of grids of different densities and have become one of the most important techniques used to solve field problems of all types. The overall levels of the solution are established by the solution on a crude grid, while the details of the solution are established on a series of finer grids. Typically, one iteration is made on each successively finer grid, until the finest grid is reached. Then, one iteration is made on each successively courser grid. This process is repeated until the solution converges. This procedure can reduce the number of fine grid iterations from possibly thousands, as shown above, to from 10 to 30 iterations.

This approach to the solution of partial differential equations was highly developed by Jameson[18] for the solution of computational aerodynamics problems. He used the multigrid approach together with an alternating direction method in an extremely efficient algorithm for the two-dimensional transonic flow over an airfoil.

The details of the multigrid method are, as they say, beyond the scope of this chapter, and the reader should consult the standard literature for more details. This includes the original treatise on the subject by Brandt[19] (which includes an example FORTRAN program), another tutorial which includes a FORTRAN code,[20] and more recent presentations by Briggs[21] and Wesseling.[22] The most recent *Numerical Recipes*[11] book also includes a brief description and sample program.

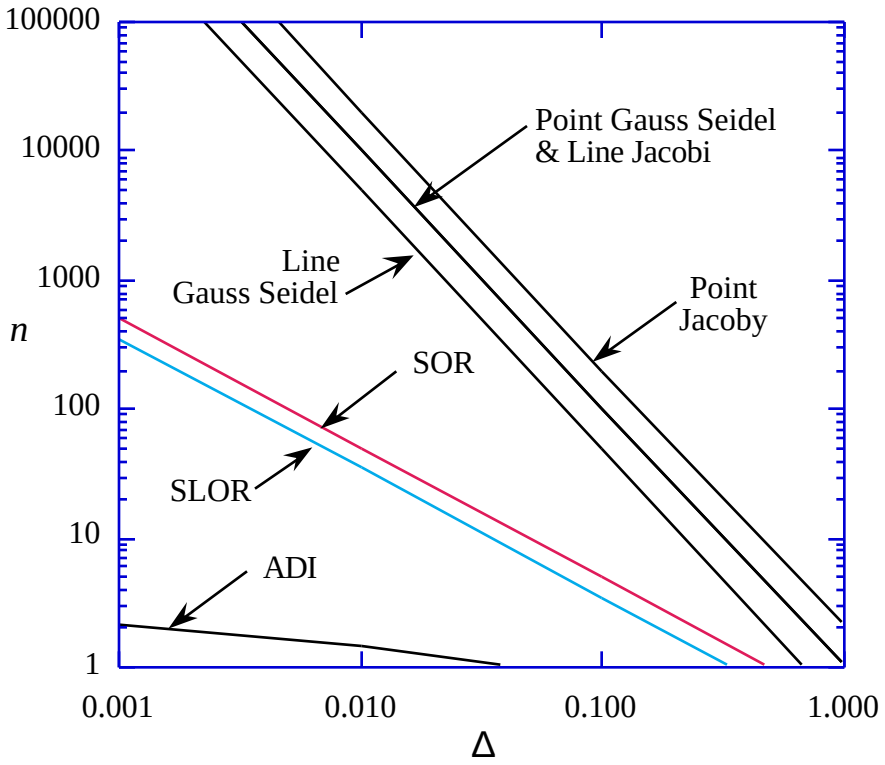


Figure 8-25. Comparison of convergence rates of various relaxation schemes (Holst[10]). This is the number of iterations estimated to be required to reduce the residual by one order of magnitude

Table 8-1
Convergence rate estimates for various relaxation schems (Holst[10])

Algorithm	Number of iterations required for a one order-of-magnitude reduction in error
Point - Jacobi	$2/\Delta^2$
Point - Gauss - Seidel	$1/\Delta^2$
SOR	$1/(2\Delta)$
Line - Jacobi	$1/\Delta^2$
Line - Gauss - Seidel	$1/(2\Delta^2)$
SLOR	$1/(2\sqrt{2}\Delta)$
ADI	$-\log(\Delta/2)/1.55$

To carry out the solution to large systems of equations, the standard numerical procedures require that the approach be generalized slightly from the one given above. Specificaly, we define an operator, such that the partial differential equation is written (continuing to use Laplace's equation as an example):

$$L\varphi = 0 \tag{8-71}$$

where

$$L = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}. \tag{8-72}$$

To solve this equation, we re-write the iteration scheme expressions given above in Equation. (8-70) as:

$$N \underbrace{C_{i,j}^n}_{\substack{n^{th}\text{ iteration}\\ \text{correction}}} + \omega \underbrace{L\varphi_{i,j}^n}_{\substack{n^{th}\text{ iteration}\\ \text{residual, = 0}\\ \text{when converged}\\ \text{solution is achieved}}} = 0. \tag{8-63}$$

This form is known as the standard or delta form. *C* is given by

$$C_{i,j}^n = \varphi_{1,j}^{n+1} - \varphi_{1,j}^n. \tag{8-64}$$

The actual form of the *N* operator depends on the specific scheme chosen to solve the problem.

8.6 Stability Analysis

The analysis presented above makes this approach to solving the governing equations for flowfields appear deceptively simple. In many cases it proved impossible to obtain solutions. Frequently the reason was the choice of an inherently *unstable* numerical algorithm. In this section we present one of the classical approaches to the determination of stability criteria for use in CFD. These types of analysis provide insight into grid and stepsize requirements (the term stepsize tends to denote time steps, whereas a grid size is thought of as a spatial size). In addition, this analysis is directly applicable to a linear equation. Applications in nonlinear problems are not as fully developed.

Fourier or Von Neumann Stability Analysis

Consider the heat equation used previously,

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} \tag{8-17}$$

and examine the stability of the explicit representation of this equation given by Eq. (8-21). Assume at $t = 0$, that an error, possibly due to finite length arithmetic, is introduced in the form:

$$\underbrace{u(x,t)}_{\text{"error" is introduced}} = \psi(t) \underbrace{e^{j\beta x}}_{\text{actually could be a series, take one term here}} \tag{8-75}$$

where:

$$\beta - \text{a real constant}$$
$$j = \sqrt{-1}$$

Here we restate the explicit finite difference representation,

$$\frac{u(x,t+\Delta t) - u(x,t)}{\Delta t} = \alpha \frac{u(x+\Delta x,t) - 2u(x,t) + u(x-\Delta x,t)}{(\Delta x)^2}. \tag{8-21}$$

Substitute Eq. (8-75) into this equation, and solve for $\psi(t + \Delta t)$. Start with

$$\frac{\psi(t+\Delta t)e^{j\beta x} - \psi(t)e^{j\beta x}}{\Delta t} = \alpha \frac{\psi(t)}{(\Delta x)^2}\left\{e^{j\beta(x+\Delta x)} - 2e^{j\beta x} + e^{j\beta(x-\Delta x)}\right\} \tag{8-76}$$

and collecting terms:

$$\psi(t+\Delta t)e^{j\beta x} = \psi(t)e^{j\beta x} + \alpha \frac{\Delta t}{(\Delta x)^2}\psi(t)e^{j\beta x}\underbrace{\left\{e^{j\beta\Delta x} - 2 + e^{-j\beta\Delta x}\right\}}_{-2+\underbrace{e^{j\beta\Delta x}+e^{-j\beta\Delta x}}_{2\cos\beta\Delta x}} \tag{8-77}$$

Note that the $e^{j\beta x}$ term cancels, and Eq. (8-77) can be rewritten:

$$\begin{aligned}\psi(t+\Delta t) &= \psi(t)\left[1 + \alpha \frac{\Delta t}{(\Delta x)^2}\left(-2 + 2\cos\beta\Delta x\right)\right] \\ &= \psi(t)\left[1 - 2\alpha \frac{\Delta t}{(\Delta x)^2}\left(1 - \underbrace{\cos\beta\Delta x}_{\substack{\text{double angle formula} \\ = 1 - 2\sin^2\beta\frac{\Delta x}{2}}}\right)\right]\end{aligned} \tag{8-78}$$

which reduces to:

$$\begin{aligned}\psi(t+\Delta t) &= \psi(t)\left[1 + 2\alpha \frac{\Delta t}{(\Delta x)^2}\left(-1 + 2\sin^2\beta\frac{\Delta x}{2}\right)\right] \\ &= \psi(t)\left[1 - 4\alpha \frac{\Delta t}{(\Delta x)^2}\sin^2\beta\frac{\Delta x}{2}\right]\end{aligned}. \tag{8-79}$$

Now look at the ratio of $\psi(t + \Delta t)$ to $\psi(t)$, which is defined as an amplification factor G,

$$G = \frac{\psi(t+\Delta t)}{\psi(t)} = \left[1 - 4\alpha \frac{\Delta t}{(\Delta x)^2} \sin^2 \beta \frac{\Delta x}{2}\right]. \tag{8-80}$$

For stability the requirement is clearly:

$$|G| < 1, \tag{8-81}$$

which means that the error introduced decays. For arbitrary β, what does this condition mean? Observe that the maximum value of the sine term is one. Thus, the condition for stability will be:

$$\left| 1 - 4\alpha \underbrace{\frac{\Delta t}{(\Delta x)^2}}_{\lambda} \right| < 1 \tag{8-82}$$

and the limit will be:

$$|1 - 4\lambda| = 1. \tag{8-83}$$

The largest λ that can satisfy this requirement is:

$$1 - 4\lambda = -1$$

or

$$-4\lambda = -2. \tag{8-84}$$

and

$$\lambda = \frac{1}{2}$$

Thus, the largest λ for $|G| < 1$ means

$$\lambda = \alpha \frac{\Delta t}{(\Delta x)^2} < \frac{1}{2} \tag{8-85}$$

or:

$$\alpha \frac{\Delta t}{(\Delta x)^2} < \frac{1}{2}. \tag{8-86}$$

This sets the condition on Δt and Δx for stability of the model equation. This is a real restriction. It can be applied locally for nonlinear equations by assuming constant coefficients. An analysis of the implicit formulation, Eq. (8-23), demonstrates that the implicit formulation is unconditionally stable.

Is this restriction on Δt and Δx real? Rightmyer and Morton[23] provided a dramatic example demonstrating this criteria. Numerical experiments can quickly demonstrate how important this condition is. Figure 8-26 repeats the analysis of Rightmyer and Morton,[23] demonstrating the validity of the analysis. The initial and boundary conditions used are:

$$u(x,0 = \phi(x)\,(given) \quad \text{for} 0 \le x \le \pi$$
$$u(0,t) = 0, \quad u(\pi,t) = 0 \quad \text{for } t > 0$$

Figure 8-26a presents the development of the solution and shows the particular choice of initial value shape, ψ, using a value of λ < 1/2: 5/11. Figure 8-26b-d provide the results for a value of λ > 1/2: 5/9. Theoretically, this stepsize will lead to an unstable numerical method, and the figure demonstrates that this is, in fact, the case.

Our model problem was parabolic. Another famous example considers a hyperbolic equation. This is the wave equation, where *c* is the wave speed:

$$\frac{\partial^2 u}{\partial t^2} - c\frac{\partial^2 u}{\partial x^2} = 0\,. \tag{8-87}$$

This equation represents one-dimensional acoustic disturbances. The two-dimensional small disturbance equation for the potential flow can also be written in this form for supersonic flow. Recall,

$$\left(1 - M_\infty^2\right)\varphi_{xx} + \varphi_{yy} = 0 \tag{8-88}$$

or when the flow is supersonic:

$$\varphi_{xx} - \frac{1}{\left(M_\infty^2 - 1\right)}\varphi_{yy} = 0 \tag{8-89}$$

and we see here that *x* is the timelike variable for supersonic flow.

Performing an analysis similar to the one above, the stability requirement for Eq. (8-87) is found to result in a specific parameter for stability:

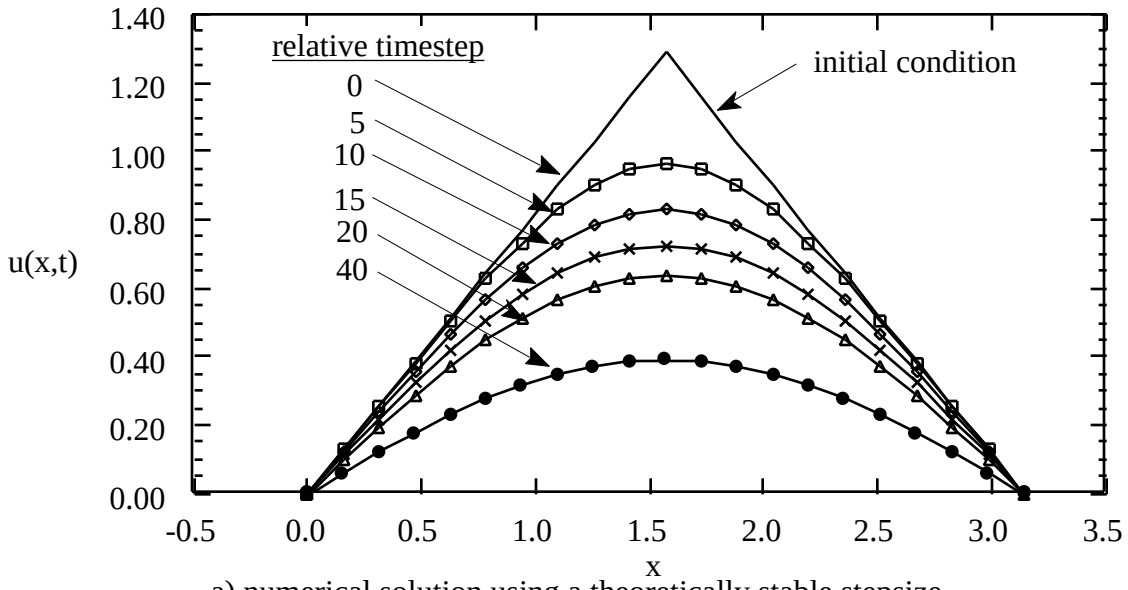
$$\nu = c\frac{\Delta t}{\Delta x} \tag{8-90}$$

which is known as the Courant number. For many explicit schemes for hyperbolic equations, the stability requirement is found to be

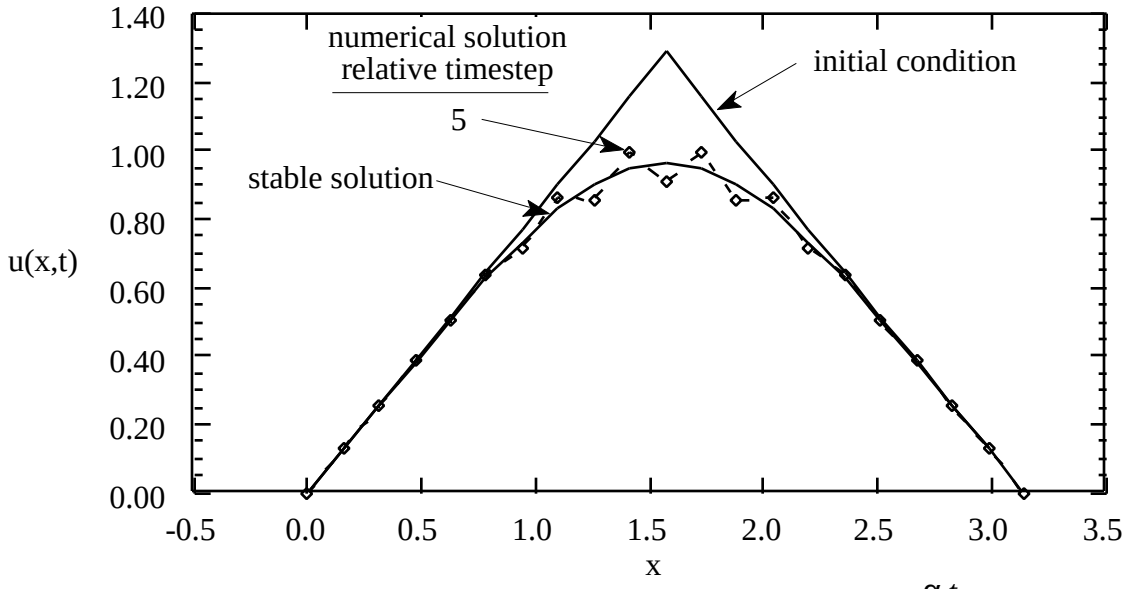
$$|\nu| \le 1. \tag{8-91}$$

This requirement is known as the CFL condition, after its discoverers: Courant, Friedrichs, and Levy. It has a physical interpretation. The analytic domain of influence must lie within the numerical domain of influence.

Recalling that the evolution of the solution for an elliptic system had a definite time-like quality, a stability analysis for elliptic problems can also be carried out. For the SOR method, that analysis leads to the requirement that the over-relaxation factor, ω, be less than two.

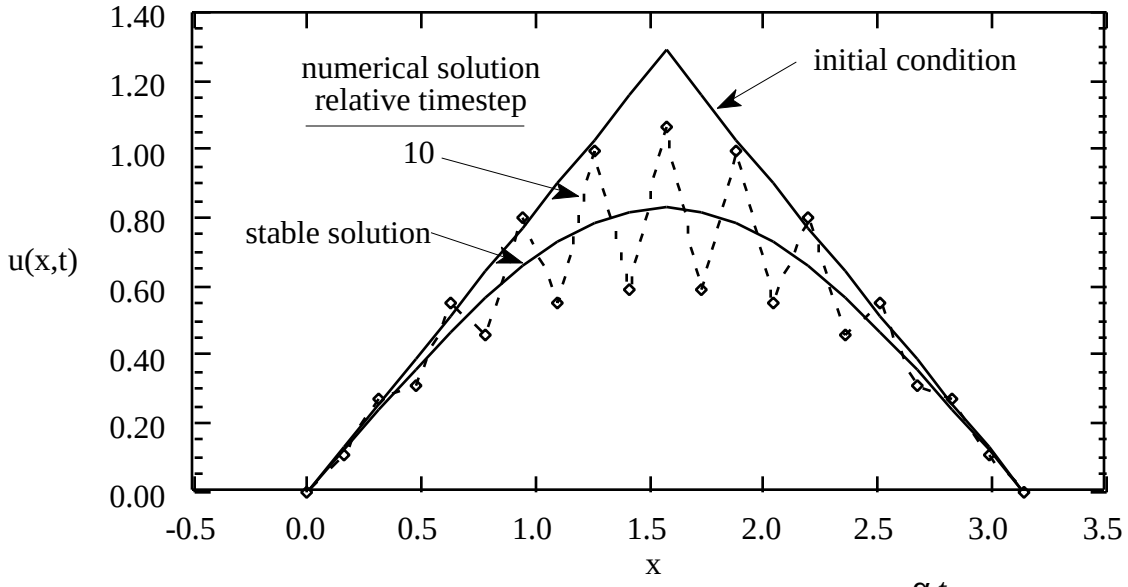


a) numerical solution using a theoretically stable stepsize.

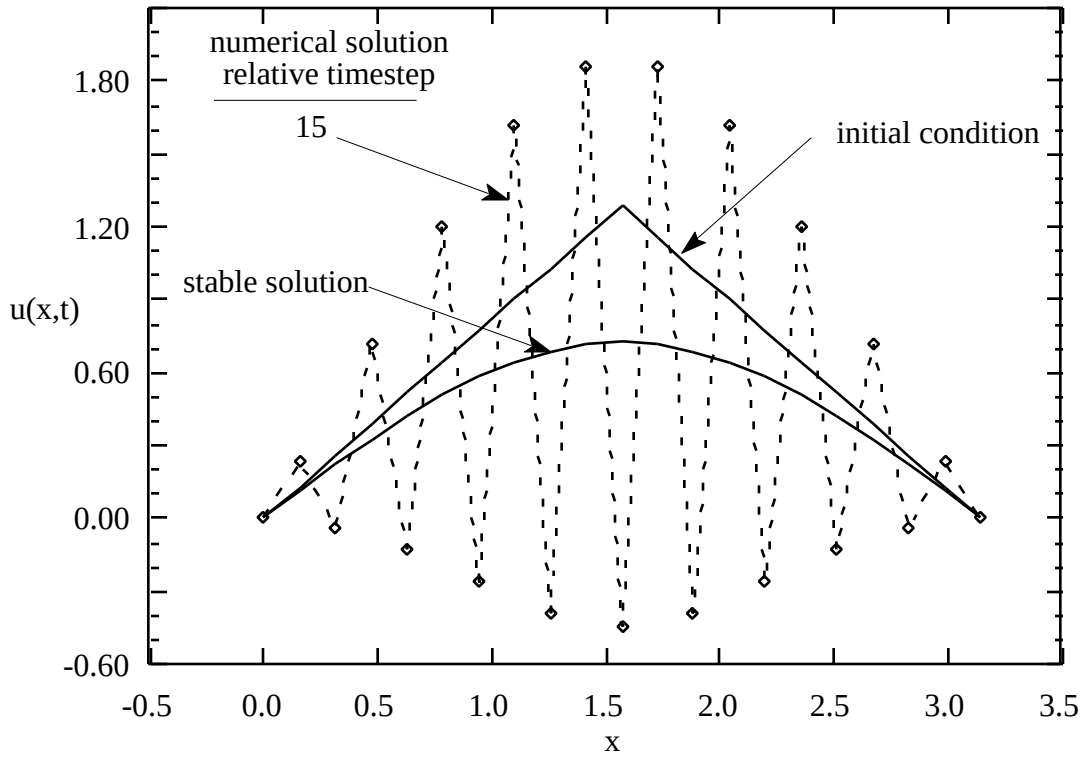


b) numerical solution using a theoreticaly unstable stepsize, $\frac{\alpha t}{(\Delta x)^2} = 5$.

Figure 8-26. Demonstration of the step size stability criteria on numerical solutions.



c) numerical solution using a theoreticaly unstable stepsize, $\frac{\alpha t}{(\Delta x)^2} = 10$.



d) numerical solution using a theoreticaly unstable stepsize, $\frac{\alpha t}{(\Delta x)^2} = 15$.

Figure 8-26. Demonstration of the step size stability criteria on numerical solutions (concluded).

8.7 Program THINFOIL

An example of the solution of Laplace's Equation by finite differences is demonstrated in the program **THINFOIL**. This program offers the users options of SOR, SLOR, AF1 and AF2 to solve the system of algebraic equations for the flow over a biconvex airfoil at zero angle of attack. An unevenly spaced grid is used to concentrate grid points near the airfoil. The program and the theory are described in Appendix G-1. It can be used to study the effects of grid boundary location, number of grid points, and relaxation factor, ω.

Figure 8-27 provides the convergence history for the case for which the comparison with the exact solution is given below. Using SOR, this shows that hundreds of iterations are required to reduce the maximum change between iteration approximately three orders of magnitude. This is about the minimum level of convergence required for useful results. A check against results converged further should be made. The reader should compare this with the other iteration options.

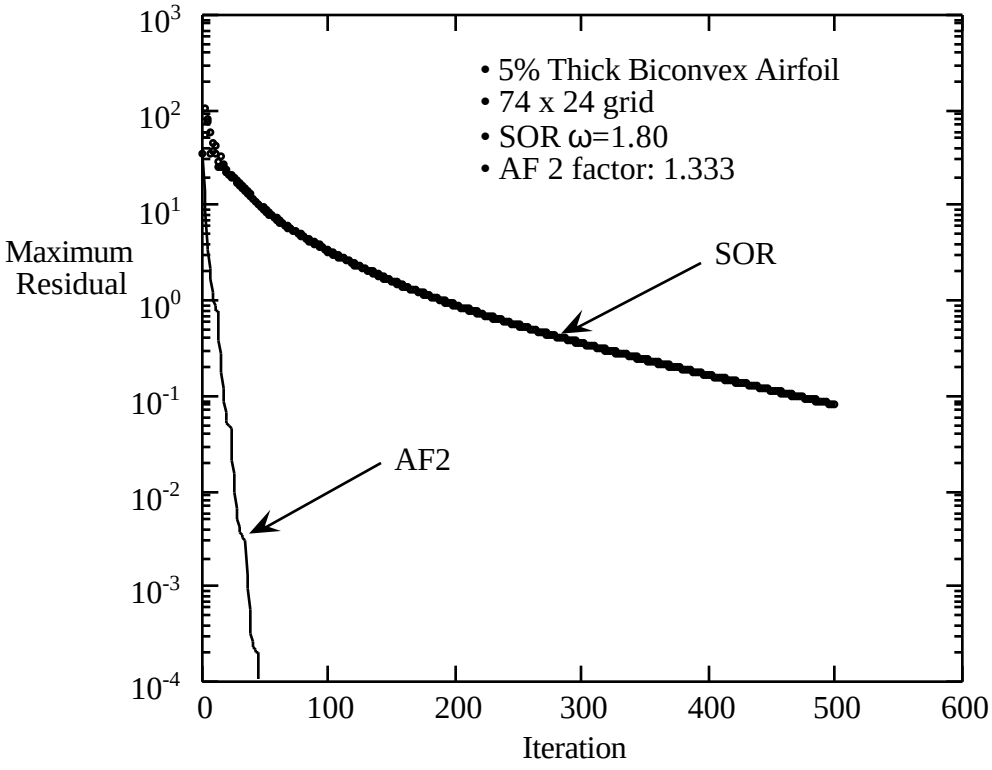


Figure 8-27. Convergence history during relaxation solution.

The convergence history presented above is actually the maximum residual of φ at each iteration. The solution is assumed to have converged when the residual goes to zero. Typical

engineering practice is to consider the solution converged when the residual is reduced by 3 or 4 orders of magnitude. However, a check of the solution obtained at a conventional convergence level with a solution obtained at much smaller residual (and higher cost) level should be made before conducting an extensive analysis for a particular study.

The solution for a 5% thick biconvex airfoil obtained with **THINFOIL** is presented in Figure 8-28, together with the exact solution. For this case the agreement with the exact solution is excellent. The exact solution for a biconvex airfoil is given by Milton Van Dyke,[24] who cites Milne-Thompson[25] for the derivation.

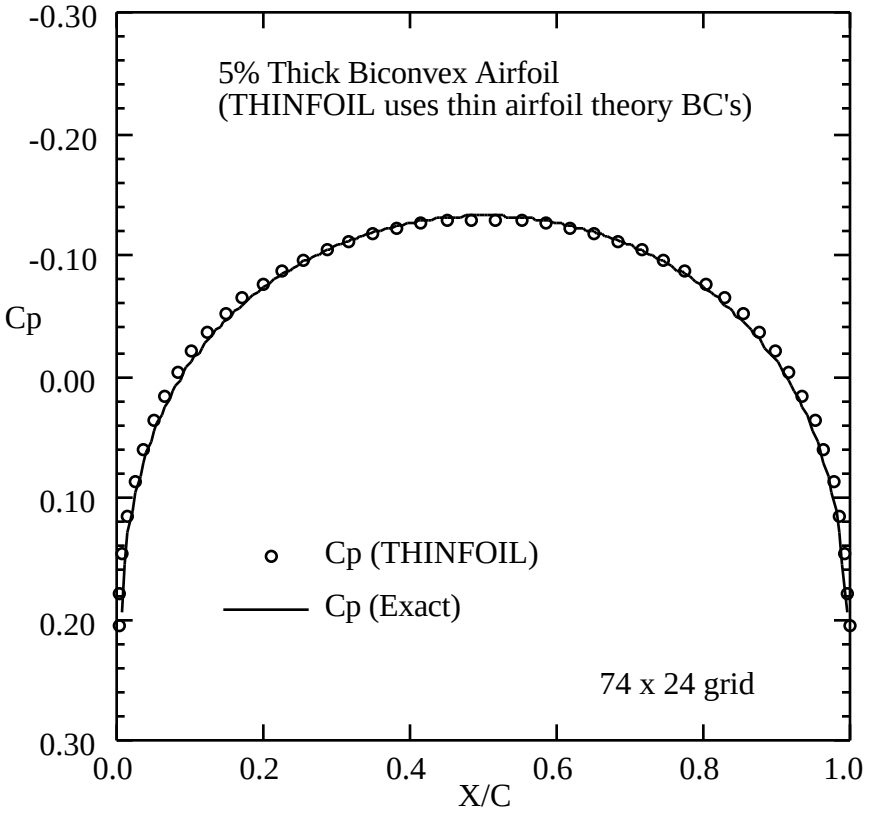


Figure 8-28. Comparison of numerical solution with analytic solution for a biconvex airfoil.

The material covered in this chapter provides a very brief introduction to an area which has been the subject of an incredible amount of research in the last thirty years. Extensions to include ways to treat flows governed by nonlinear partial differential equations are described after some basic problems in establishing geometry and grids are covered in the next chapter.

8.8 Exercises

1. How accurate are finite difference approximations? Over one cycle of a sine wave, compare first and second order accurate finite difference approximations of the 1st derivative and the second order accurate 2nd derivative of the shape with the exact values. How small does the step size have to be for the numerical results to accurate to 2 significant figure? 4 figures? 6? What conclusions about step size can you make?
2. Get some experience with the solution of Laplace's Equation using finite differences.
   i) Download a copy of **THINFOIL** from the web page
   ii) Make it run on your PC.
   iii) Study the program to understand the procedure.

   Pick as a baseline case: Xmin=-2.2, Xmax=3.2, Ymax=2.4, and NUP=14, NDOWN=14, NON=30, NABOVE=18

   iv) Run SOR with $\omega = 1.6$ and see how many iterations to "convergence"

   v) Run with $\omega = 1.0, 1.50, 1.75, 1.90, 1.99$ (400 iterations max)

   vi) Plot the convergence history as a function of iteration for each ω. Note that it is standard procedure to plot the log of the residual. See examples in the text.

   vii) For one ω, increase the number of grid points and compare (watch dimensions)

   - convergence rate with the same ω case above
   - the surface pressure distribution results for the two grids

   viii) Draw conclusions about SOR as a numerical method for solving PDE's.

   ix) Repeat the studies using SLOR, AF1 and AF2. What do you conclude about the relative convergence times and solution accuracy?
3. Examine the effect of the number of grid points on the solution obtained using program **THINFOIL**. How many grid points are required for a grid converged solution?
4. Examine the effect of the location of the farfield boundary condition on the solution obtained using program **THINFOIL**. What do you conclude?
5. Change the farfield boundary condition to set $\varphi = 0$, instead of $\partial\varphi/\partial n = 0$. How does this affect the solution? the convergence rate?
6. Modify program **THINFOIL** to obtain the solution to the flow over an NACA 4-Digit airfoil thickness shape. Address the following issues:
   i) store the boundary condition values before the calculation begins instead of recomputing each time the BC needs the value
   ii) recognizing that the slope at the leading edge is infinite, assess two methods of avoiding numerical problems
   - place the leading edge between grid points
   - use Riegels' factor to modify the slope boundary condition, replacing *df/dx* by

$$\frac{df/dx}{\sqrt{1+\left(df/dx\right)^2}}$$

8.9 References

[1] Tannehill, J.C., Anderson, D.A., and Pletcher, R.H., *Computational Fluid Mechanics and Heat Transfer*, 2nd Ed., Taylor & Francis, New York, 1997. (on the first edition the order of the authors was Anderson, D.A., Tannehill, J.C., and Pletcher, R.H)

[2] Hoffman, K.A., and Chiang, S.T., *Computational Fluid Dynamics for Engineers*, in two volumes, Engineering Education System, PO Box 20078, Wichita, KS, 67208-1078. 1993.

[3] Fletcher, C.A.J., *Computational Techniques for Fluid Dynamics*, Vol. 1: "Fundamental and General Techniques," Vol. II, "Specific Techniques for Different Flow Categories," Springer-Verlag, Berlin, 1988.

[4] Moran, J., *An Introduction to Theoretical and Computational Aerodynamics*, John Wiley & Sons, New York, 1984.

[5] Hirsch, C., *Numerical Computation of Internal and External Flows*, Vol. 1, "Fundamentals of Numerical Discretization," 1988, and Vol. 2, "Computational Methods for Inviscid and Viscous Flows," 1990, John Wiley & Sons, New York.

[6] Anderson, J.D., Jr., *Computational Fluid Dynamics: The Basics with Applications*, McGraw-Hill, Inc., New York, 1995.

[7] Smith, G.D., *Numerical Solution of Partial Differential Equations: Finite Difference Methods*, 3rd Ed., Clarendon Press, Oxford, 1985.

[8] Ames, W.F., *Numerical Methods for Partial Differential Equations*, 2nd Ed., Academic Press, New York, 1977.

[9] Versteeg, H.K., and Malalasekera, W., *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*, Addison Wesley Longman, Ltd., Harlow, England, 1995.

[10] Holst, T.L., "Numerical Computation of Transonic Flow Governed by the Full-Potential Equation," VKI Lecture Series on Computational Fluid Dynamics, Rhode-St.-Genese, March, 1983.

[11] Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vettering, W.T., *Numerical Recipes in FORTRAN: The Art of Scientific Computing*, 2nd ed., Cambridge University Press, Cambridge, 1992.

[12] Conte, S.D., and deBoor, C., *Elementary Numerical Analysis*, McGraw-Hill, New York, 1972.

[13] Mason, W.H., Mackenzie, D., Stern, M., Ballhaus, W.F., and Frick, J., "An Automated Procedure for Computing the Three-Dimensional Transonic Flow Over Wing-Body Combinations, Including Viscous Effects," Vol. II Program User's Manual and Code Description, AFFDL-TR-77-122, Feb. 1978.

[14] Isaacson, E., and Keller, H.B., *Analysis of Numerical Methods*, John Wiley, New York, 1966.

[15] Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vettering, W.T., *Numerical Recipes: The Art of Scientific Computing (FORTRAN Version)*, Cambridge University Press, Cambridge, 1989.

[16] Ballhaus, W.F., Jameson, A., and Albert, J., "Implicit Approximate Factorisation Schemes for the Efficient Solution of Steady Transonic Flow Problems," AIAA J., Vol. 16, No. 6, June 1978, pp. 573-579. (also AIAA Paper 77-634, 1977).

[17] Catherall, D., "Optimum Approximate-Factorisation Schemes for 2D Steady Potential Flows," AIAA Paper 81-1018, 1981.

[18] Jameson, A., "Acceleration of Transonic Potential Flow Calculations on Arbitrary Meshes by the Multiple Grid Method," AIAA Paper 79-1458, *Proceedings of the AIAA 4th Computational*

Fluid Dynamics Conference, AIAA, New York, 1979, pp. 122-146.

[19] Brandt, A., "Multi-Level Adaptive Solutions to Boundary-Value Problems," *Mathematics of Computation*, Vol. 31, No. 138, April 1977, pp. 333-390.

[20] Stuben, K., and Trottenberg, U., "Multigrid Methods: Fundamental Algorithms, Model Problem Analysis and Applications," in *Multgrid Methods*, ed. by W. Hackbusch and U. Trottenberg, Lecture Notes in Mathematics Vol. 960, Springer-Verlag, Berlin, 1982. pp. 1-176.

[21] Briggs, W.L., *A Multigrid Tutorial*, SIAM, Philadelphia, 1987

[22] Wesseling, P., *An Introduction to Multigrid Methods*, John Wiley, Chichester, 1992.

[23] Richtmyer, R.D., and Morton, K.W., *Difference Methods for Initial-Value Problems*, 2nd Ed., Interscience, New York, 1967 pp. 4-9.

[24] Van Dyke, M., *Perturbation Methods in Fluid Mechanics*, *Annotated Edition*, The Parabolic Press, Stanford, 1975, problem 4.5, page 74.

[25] Milne-Thompson, *Theoretical Hydrodynamics*, 5th ed., (1968), reprinted by Dover Publications, Inc., New York, 1996, Section 6.51, pp. 177-179.