

The field of computational transonics is relatively new. Until 1970, the attempts by scientists to numerically solve the TSD equation were successful only in the subsonic regime. Even a mildly transonic flow with a small supersonic region could not be solved at that time. In deed, the more costly numerical solution of the Euler equations was the only way transonic flows could be analyzed prior to 1970.

In 1971, Murman and Cole developed solution techniques for solving the TSD equation. They recognized the fact that the TSD equation changes in character from an elliptic equation to hyperbolic in supersonic regions, and devised suitable numerical approximations for these two regions. Their technique also allowed the shock waves to be captured as part of the numerical procedure. This pioneering work lead to a rapid development of the field of computational transonics. By 1972, transonic flow over lifting airfoils could be studied.

In 1973, Murman demonstrated that their pioneering scheme did not numerically conserve mass across shock waves, and developed techniques to improve the 1971 scheme. By 1975, Bailey, Steger and Ballhaus, three researchers at the NASA Ames Research Center developed techniques for lifting transonic flow over three-dimensional isolated wings and wing-body configurations, using the 3-D TSD equation. With the increased computing power, more ambitious analyses such as 3-D transonic flow over a complete aircraft became possible, through efforts of Boppe at Grumman and others.

The pioneering work of Murman and Cole also lead to successful numerical schemes for the 2-D and 3-D full potential equations which will be covered in the next chapter. It also paved the way for studying 2-D unsteady transonic flows using the TSD theory. In 1977, Ballhaus and Goorjian developed a procedure for solving 2-D transonic flow over thin airfoils undergoing oscillatory motion. By 1983, 3-D unsteady analyses using the small disturbance theory (Borland at Boeing Military Aircraft Co., Goorjian and Guruswamy at NASA Ames, Batina at NASA Langley), as well as the 2-D and 3-D full potential equation (Malone and Sankar at Lockheed Georgia Co.) became feasible.

4.2 Overview of the TSD Solution Procedure

The TSD equation may be written as

$$\frac{\partial P}{\partial x} + \frac{\partial Q}{\partial y} = 0$$

where,

$$P = (1 - M_\infty^2)\varphi_x - (\gamma + 1)\frac{M_\infty^2}{2}(\varphi_x)^2$$

$$Q = \varphi_y$$

(4.1)

Numerical solution of this equation (or the FPE equation considered in the next chapter) involves the following steps:

(a) Selection of a set of points called "nodes" in the x-y plane, where the governing equation will be solved. The region of interest is a large region surrounding the airfoil (represented as a slit). Because the disturbances associated with the airfoil are felt for large distances away from the airfoil, this region must be 5 to 6 chord lengths wide in all directions surrounding the slit.

This process of selecting the points where the TSD equation will be solved is known as grid generation.

(b) Devising a numerical approximation to equation (4.1) which converts the PDE into a set of nonlinear algebraic equations at the nodes selected in step 1. This step is known as discretization.

(c) Solving the nonlinear algebraic equations from step (b) for the numerical value of φ at the nodes. This is usually iteratively done, starting with some arbitrary guess for the solution at the node. The process by which φ converges from an arbitrary initial guess to the solution which satisfies the discretized form developed in step (b) is known as the relaxation procedure.

(d) Once the values of φ are known at all the nodes, we can find other quantities such as disturbance velocities, surface pressure C_p distribution, the lift, drag and pitching moment experienced by the airfoil, the total number of supersonic points in the flow field, etc. This process is known as the post-processing step.

Of course, there is no need to wait until the iterative process in step (c) has fully converged. We can always postprocess the intermediate solution to compute lift, drag etc., and monitor if these values are converging to fixed values.

We now consider the various steps in the solution process in detail.

4.3 Grid generation

The grid generation for solving the TSD equation for 2-D isolated airfoils is very simple, because of the fact that the airfoil is modeled as a slit, placed over the x- axis.

In order to obtain reasonably accurate solutions, the nodes must be placed fairly close to each other. From experience, it is known that there must be at least 50 points over the slit. We will place these points uniformly, spaced apart by a distance $c/50$, where c is the chord length. For convenience, the chord length is always taken to be unity. In the direction normal to the slit, close to the slit, the spacing must also be comparable to the x- spacing, say $1/50$ (assuming a chord length equal to unity).

If we use uniformly spaced nodes in both the x- and y- directions, a quick calculation reveals that we will need 400×400 nodes (160,000) for a grid that is 8 chord lengths wide, and 8 chord lengths tall. We can avoid using such excessive number of nodes by gradually, geometrically increasing the grid spacing both in the x- and y- directions away from the slit. In other words, away from the slit, use

$$\begin{aligned}\Delta x_{next} &= K_1 \Delta x_{previous} \\ \Delta y_{next} &= K_2 \Delta y_{previous}\end{aligned}\tag{4.2}$$

Here K_1 and K_2 are stretching factors, between 1.05 and 1.15. A large stretching factor will mean that we need to use fewer grid points. On the other hand, on such rapidly stretched grids, the solution accuracy will be compromised, as will be demonstrated later. Figure 4.1 shows a sample grid.

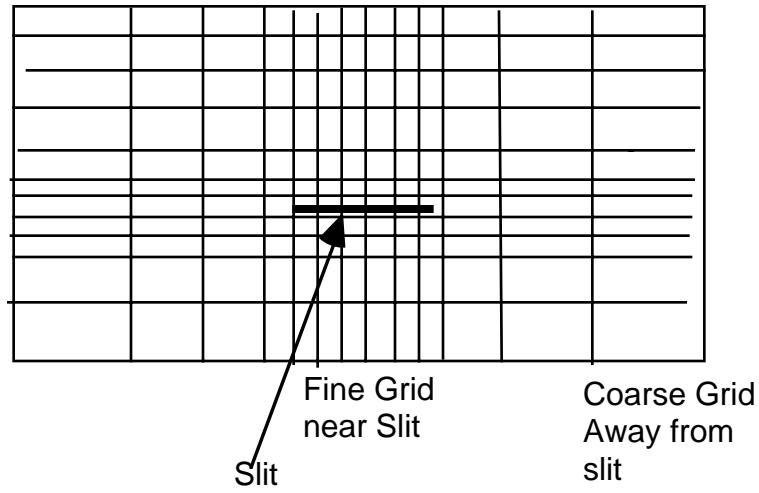


Figure 4.1 Algebraically stretched grid for the 2-D TSD solution

A convenient way of referencing these nodes is needed. These nodes may, of course, be numbered consecutively as 1, 2, 3, ... and so forth. A more common numbering system is to identify each node with two indices (i,j) . The first index, i refers to the vertical line on which the node lies, while the second index j refers to the horizontal grid line on which the node lies. Figure 4.2 shows a typical node, and its four neighbors, along with the distance separating these nodes.

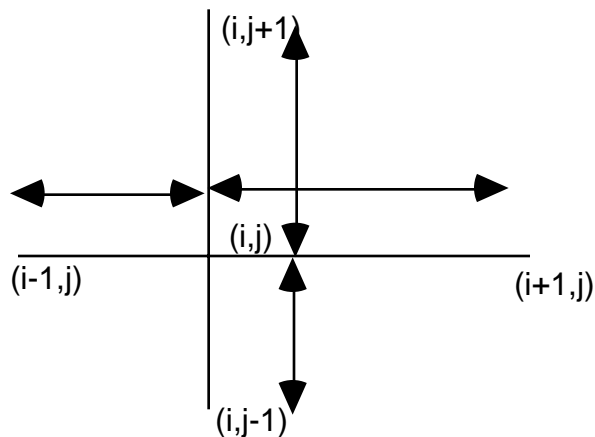


Figure 4.2 The node (i,j) and its neighbors

In the above figure the distance separating the nodes (i,j) and $(i-1,j)$ is Δx_i . The horizontal distance separating the node (i,j) and $(i+1,j)$ is Δx_{i+1} . Likewise, the vertical distances that separate (i,j) from $(i,j-1)$ and $(i,j+1)$ are Δy_j and Δy_{j+1} , respectively. These

spacings are computed during the grid generation phase, and are stored as one-dimensional arrays, for later use during the discretization.

For the sake of applying the solid wall boundary conditions, it is recommended that the slit itself be placed halfway between two horizontal grid lines, as shown below.

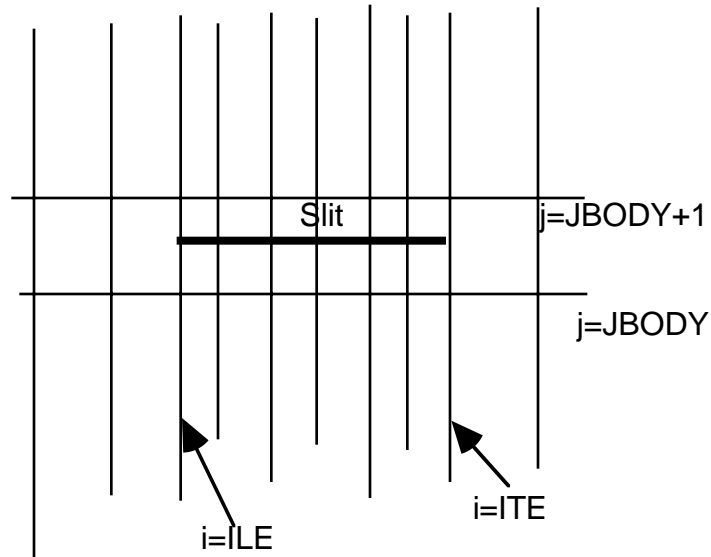


Figure 4.3 The slit is placed halfway between lines $j=JBODY$ and $j=JBODY+1$

Note that the slit is placed between the two vertical lines $i=ILE$ and $i=ITE$. The numerical values of ILE , ITE , $JBODY$ are all prescribed or computed during the grid generation phase, and stored for later use during the discretization phase, and the post-processing phase when the air loads are computed.

Appendix A shows a sample FORTRAN subprogram for generating a stretched Cartesian grid. This subroutine may be improved further. For example, it assumes a uniform grid spacing between the leading edge and trailing edge (ILE and ITE). For better accuracy near the leading and trailing edges, a cosine distribution may be more suitable.

4.4 Discretization of the TSD equation on a stretched Cartesian grid

We next turn our attention to the discretization of equation (4.1), i.e. converting the TSD into an algebraic equation linking a node (i,j) to its neighbors. This process may be done in a number of ways. Three common approaches to discretization are (a) finite

volume method, (b) finite element method, and (c) finite difference method. The reader is referred to several excellent text books on numerical analysis (e.g. books by Anderson, Tannehill and Pletcher, P. J. Roache, Hirsch) for further information on these techniques. In the present work, the finite difference method is used for its simplicity, and also because Murman and Cole's original work used this type of discretization.

The finite difference method handles the governing PDE one term at a time. Each term is expanded using a Taylor series (or other convergent series such as Fourier series expansion) in terms on the values of the dependent variable stored at the nodal points, surrounding and including a typical node (i,j) . The order of accuracy depends on how many terms in the series expansion are kept. *Fundamental to this approach is the assumption that the individual terms in the PDE are smooth, differentiable functions of x and y , in the vicinity of the node (i,j) . This assumption, of course is not always valid in the vicinity of the shocks.*

Consider the term $\partial P/\partial x$ appearing in equation (4.1). This derivative is computed by expanding the function P in the vicinity of the node (i,j) . In particular, we expand P at the half-nodes $(i+1/2,j)$ and $(i-1/2,j)$ about the node (i,j) . The reason for choosing the half-nodes rather than $(i \pm 1,j)$ will be explained later.

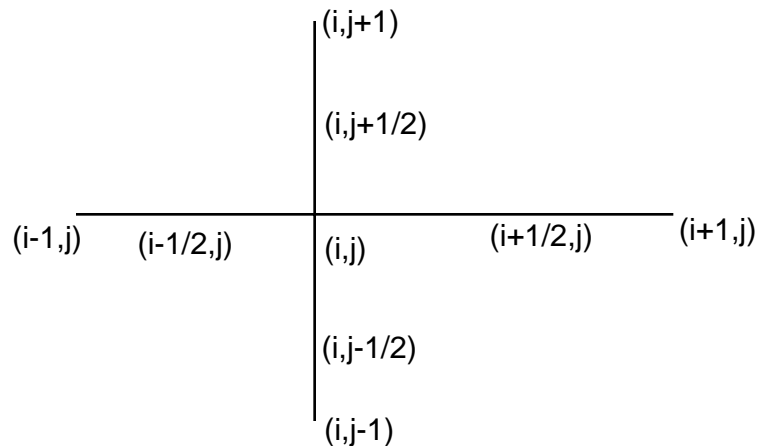


Figure 4.4 Half Nodes are used to discretize the TSD equation

Recognizing the fact that the half-node $(i+1/2,j)$ is separated from the node (i,j) by a distance $(\Delta x_{i+1})/2$, we write,

$$P_{i+1/2,j} = P_{i,j} + \frac{\Delta x_{i+1}}{2} (P_x)_{i,j} + \frac{(\Delta x_{i+1})^2}{8} (P_{xx})_{i,j} + O(\Delta x_{i+1}^3) \quad (4.3)$$

Note that we have truncated the Taylor series after the first three terms. Omitting the remaining terms in this infinite series creates a numerical error, known as the truncation error.

The term P at $(i-1/2, j)$ may likewise be expanded about (i, j) , noting the fact that these two points are separated by a distance $\Delta x_i / 2$.

$$P_{i-1/2, j} = P_{i, j} - \frac{\Delta x_i}{2} (P_x)_{i, j} + \frac{(\Delta x_i)^2}{8} (P_{xx})_{i, j} + O(\Delta x_i^3) \quad (4.4)$$

Subtracting (4.4) from (4.3) and rearranging the resulting form, we obtain the following expression for $\partial P / \partial x$, the first term in our PDE.

$$(P_x)_{i, j} = \frac{P_{i+1/2, j} - P_{i-1/2, j}}{\left(\frac{\Delta x_i + \Delta x_{i+1}}{2}\right)} - \left(\frac{\Delta x_{i+1} - \Delta x_i}{4}\right) (P_{xx})_{i, j} + O(\Delta x^2) \quad (4.5)$$

If we keep only the first term on the right hand side (4.5) and drop all the other terms, it appears that our truncation error will be of the order Δx_i . In other words, our approximation will appear to be a first order approximation. This is indeed so, when the grid is highly stretched, and the spacings Δx_{i+1} and Δx_i are vastly different. If the spacing is uniform the approximation is clearly second order accurate. In our analysis, the grid is only mildly stretched so that the stretching factor appearing in equation (4.2) is given by

$$K_1 = 1 + O(\Delta x)$$

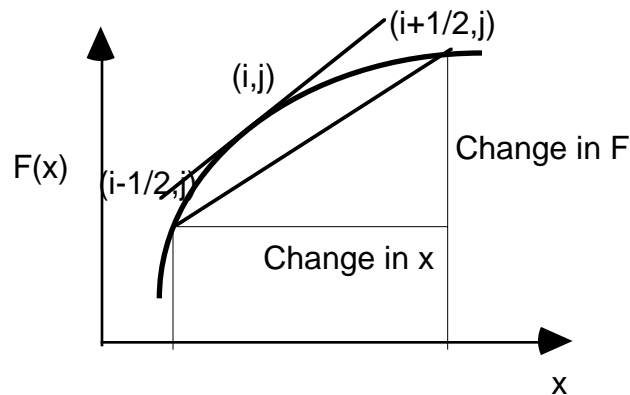
It can be easily shown that the truncation error involves only the second powers of the grid spacing Δx_{i+1} or Δx_i . In other words, to a second order accuracy, on mildly stretched grids,

$$\boxed{(P_x)_{i, j} = \frac{P_{i+1/2, j} - P_{i-1/2, j}}{\left(\frac{\Delta x_{i+1} + \Delta x_i}{2}\right)}} \quad (4.6)$$

It is cumbersome to resort to Taylor series expansions every time we need to evaluate the derivative of a function. The following thumb rule is easy to use and remember:

$$\begin{aligned} (x - \text{Derivative of } F)_{i,j} &= \frac{F_{\text{Right}} - F_{\text{Left}}}{\text{Distance between Nodes to the Right and Left}} \\ (y - \text{Derivative of } G)_{i,j} &= \frac{G_{\text{Above}} - G_{\text{Below}}}{\text{Distance between Nodes Above and Below}} \end{aligned} \quad (4.7)$$

This thumb rule is graphically illustrated below:



**Figure 4.5 Graphical Representation of the Derivative of $F(x)$,
 $dF/dx \approx \Delta F/\Delta x$**

Using this thumb rule, we can evaluate $\partial Q/\partial y$ at the node (i,j) , as well as the quantities $P_{i \pm 1/2,j}$.

Exercise 4.1 Verify for yourselves the expressions given in equation (4.8). Use the thumb rule given above.

$$\begin{aligned}
(Q_y)_{i,j} &= \frac{Q_{i,j+1/2} - Q_{i,j-1/2}}{\left(\frac{\Delta y_{j+1} + \Delta y_j}{2}\right)} \\
P_{i+1/2,j} &= \left((1 - M_\infty^2) \phi_x - \frac{\gamma+1}{2} M_\infty^2 \phi_x^2 \right)_{i+1/2,j} \\
&= (1 - M_\infty^2) \frac{\phi_{i+1,j} - \phi_{i,j}}{\Delta x_{i+1}} - \frac{\gamma+1}{2} M_\infty^2 \left(\frac{\phi_{i+1,j} - \phi_{i,j}}{\Delta x_{i+1}} \right)^2 \\
P_{i-1/2,j} &= (1 - M_\infty^2) \frac{\phi_{i,j} - \phi_{i-1,j}}{\Delta x_i} - \frac{\gamma+1}{2} M_\infty^2 \left(\frac{\phi_{i,j} - \phi_{i-1,j}}{\Delta x_i} \right)^2 \\
Q_{i,j+1/2} &= (\phi_y)_{i,j+1/2} = \frac{\phi_{i,j+1} - \phi_{i,j}}{\Delta y_{j+1}} \\
Q_{i,j-1/2} &= \frac{\phi_{i,j} - \phi_{i,j-1}}{\Delta y_j}
\end{aligned} \tag{4.8}$$

In summary, a second order accurate numerical approximation to the 2-D transonic small disturbance equation is

$$\frac{P_{i+1/2,j} - P_{i-1/2,j}}{\Delta x_i + \Delta x_{i+1}} + \frac{Q_{i,j+1/2} - Q_{i,j-1/2}}{\Delta y_j + \Delta y_{j+1}} = 0 \tag{4.9}$$

The reason for using half points to compute P and Q may now become clear. Choice of these half points leads to very compact expressions for the terms P_x and Q_y at our node (i,j) involving only the node and its four neighbors. If we had computed P and Q at the full nodes ($i \pm 1, j$) and ($i, j \pm 1$), then, our final form would have involved nodes far away such as ($i \pm 2, j$) and ($i, j \pm 2$). In general, use of a compact form (compact "stencil") involving just a handful of nodes closely spaced together is likely to yield better results than over a large stencil involving nodes far apart.

4.5 Shock Fitting vs. Shock Capturing

Use of the compact form given by equation (4.9) also ensures that discontinuities such as shock waves (which are modeled as large, but differentiable changes in the flow velocity) will be "captured" across a small number of points, rather than smeared over a large number of points.

The strategy of modeling shock waves as smooth phenomena by the application of the finite difference form of the governing equations in the vicinity of, and across a shock wave is known as "shock capturing". The alternative is "shock fitting" which involves treating the shock surface as a discontinuity and applying special jump conditions (in lieu of the governing equations). Murman and Cole's original work, and almost all subsequent transonic flow analyses have used the "shock capturing" approach because of its inherent simplicity. Shock fitting is more accurate, but is somewhat cumbersome to program.

4.9 Murman and Cole Scheme

When Murman and Cole (and several others) used equation (4.9) to solve compressible flows, they observed the following. This form gave correct solutions when the flow was subsonic. In transonic regions, equation (4.9) either did not work, or gave rise to nonphysical expansion shocks. Recall that our TSD equation can model both compression shocks and expansion shocks. This apparent inability of equation (4.9) to model transonic phenomena lead many researchers to abandon TSD theory altogether, and use computationally intensive Euler analyses (e.g. Magnus and Yoshihara in 1970). As stated in the introductory remarks, the breakthrough in modern computational transonics occurred when the Murman-Cole scheme was discovered.

Murman and Cole observed that the equation (4.9) requires that the velocity potential at the node (i,j) surrounds on its entire neighborhood, upstream, downstream, above and below. This dependency is physically correct only in subsonic regions of the flow, where the PDE is elliptic as shown in Chapter III.

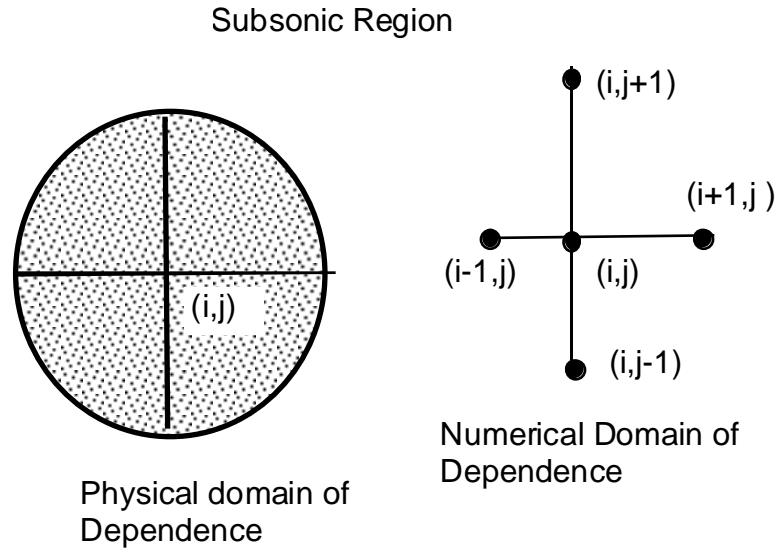


Figure 4.6 In elliptic regions, the node (i,j) depends on the surrounding region

In supersonic regions, however, the point P is not influenced by points outside the "Mach" cone, a wedge formed by the two characteristic lines, discussed in Chapter III.

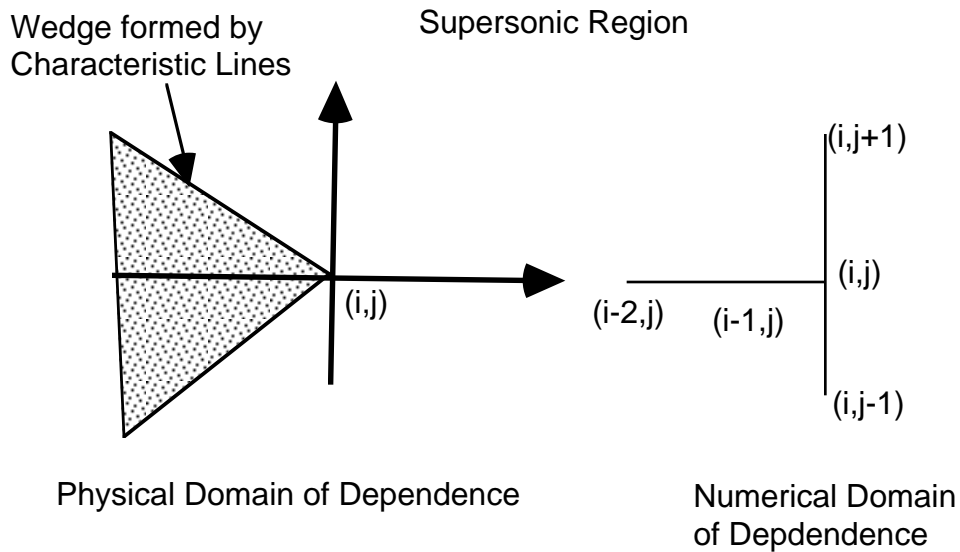


Figure 4.7 In supersonic regions, the node (i,j) should depend primarily on the information within the characteristic cone

Murrman and Cole argued that in supersonic regions, the finite difference approximation should reflect the upstream bias, required by the physics of the flow.

They proposed that in supersonic regions, the i - indices in equation (4.9) be shifted to the left, resulting in the following approximation:

$$\frac{P_{i-1/2,j} - P_{i-3/2,j}}{\Delta x_i + \Delta x_{i+1}} + \frac{Q_{i,j+1/2} - Q_{i,j-1/2}}{\Delta y_j + \Delta y_{j+1}} = 0 \quad (4.10)$$

Exercise 4.2 Show that the first term in (4.10) is only a first order approximation to $\partial P/\partial x$ at the node (i,j) , even on uniformly spaced grids.

In chapter III, it was mentioned that the sign of the quantity A , given by

$$A = 1 - M_\infty^2 - (\gamma + 1)M_\infty^2 \phi_x \quad (4.11)$$

determines if the TSD equation is elliptic or hyperbolic. In the Murman-Cole scheme, A is first numerically evaluated as

$$A \approx 1 - M_\infty^2 - (\gamma + 1)M_\infty^2 \frac{\phi_{i+1,j} - \phi_{i-1,j}}{\Delta x_i + \Delta x_{i+1}} \quad (4.12)$$

If this quantity A at the node (i,j) is positive, then equation (4.9) is to be used. If A is zero or negative, then equation (4.10) is to be used.

Switching between the two forms is made easier by defining a switching function $\mu_{i,j}$ as 1 if A is less than 0, and 0 if A is greater than zero. The total number of nodes within the entire grid where A is less than zero is a measure of the number of supersonic points in the flow. Then, the TSD equation is modeled as

$$\begin{aligned} & \frac{P_{i+1/2,j} - P_{i-1/2,j}}{\Delta x_i + \Delta x_{i+1}} + \frac{Q_{i,j+1/2} - Q_{i,j-1/2}}{\Delta y_{j+1} + \Delta y_j} \\ & + \mu_{i,j} \left[\frac{P_{i-1/2,j} - P_{i-3/2,j}}{\Delta x_i + \Delta x_{i+1}} - \frac{P_{i+1/2,j} - P_{i-1/2,j}}{\Delta x_i + \Delta x_{i+1}} \right] = 0 \end{aligned} \quad (4.13)$$

Notice that equation (4.13) becomes equivalent to equation (4.9) in subsonic regions, and equivalent to equation (4.10) in supersonic regions.

Murman and Cole's simple fix to equation (4.9) made computation of transonic flows feasible, at a fraction of the cost associated with Euler analyses. It captured shock waves across three grid points in the x-direction, because the stencils (4.9) and (4.10) both span only three grid points in the x-direction. It always leads to compression shocks. Both lifting and nonlifting airfoils could be analyzed using this technique.

4.10 Drawbacks of the Murman-Cole Scheme

Although easy to use, the 1971 Murman-Cole scheme had two major drawbacks, which limited its use.

(a) As seen in exercise (4.2), this scheme was only first order accurate in supersonic regions. A large number of points are needed in the x-direction to achieve an acceptable level of accuracy, because the error is proportional to Δx , and not to Δx^2 ! This is the reason a number of nodes (50 or more) are needed over the airfoil surface in transonic analyses. Subsonic analyses can be done with as few as 30 points on the airfoil.

(b) In a subsequent work in 1973, Murman pointed out that his scheme predicted shock waves that were weaker than expected from an Euler analysis. That is, the Mach number ahead of the shock was lower, and the pressure jump across the shock was weaker than expected from an Euler analysis. The shock location was predicted to be upstream of the expected location, from an inviscid analysis.

If our analysis (Euler or TSD) had included viscous effects, then the boundary layer growth over the airfoil and the attendant loss in lift, and the rapid thickening of the boundary layer at the foot of the shock will produce effects similar to that of Murman and Cole scheme. That is, in viscous flows, the shock is weaker, the lift is smaller and the shock position is more upstream than in an inviscid analysis.

Thus, Murman and Cole scheme spuriously modeled the effects of viscosity. In fact, the inviscid predictions of Murman-Cole scheme more closely matched experiments than the more accurate Euler analysis. This agreement is entirely spurious. Nevertheless, many analysts use the Murman and scheme and its full potential equation

based cousins (e.g. the FLO22 code developed by Jameson, 1975) even today because of this feature.

Murman showed that these discrepancies with the Euler analysis are to a large part due to the fact that his scheme did not conserve mass, in a numerical sense. Consider the integral of the governing equation over the entire computational domain. We expect our numerical solution to satisfy divergence theorem, which states

$$\iint_{\text{Entire Domain}} (P_x + Q_y) dx dy = \oint_{\text{Outer Boundaries}} [P\vec{i} + Q\vec{j}] \cdot \vec{n} dS \quad (4.14)$$

The area integral on the left hand side may be converted into a sum over a number of subdomains, each domain enclosing a typical node (i,j) as shown:

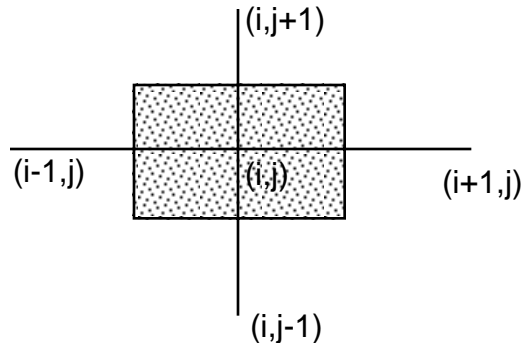


Figure 4.8 Subdomain or Control Volume surrounding a node (i,j)

The boundaries of the above subdomain, or control volume passes through the half nodes. Then, the left side of equation (4.14) may be written as a sum over all the cells

$$\sum_i \sum_j (\text{Numerical Approximation to 4.1}) (\Delta x_i + \Delta x_{i+1}) (\Delta y_j + \Delta y_{j+1}) \quad (4.15)$$

The product of grid spacing appearing in equation (4.15) model (four times) the area of the control volume.

If we had used only equation (4.9) at all nodes, it is easy to show that the above summation over all the cells reduces to boundary integral over the outer edges of the computational domain. For example, the first term in equation (4.9) results in the following:

$$\begin{aligned}
& \sum_{i=2} \sum_{j=2} \frac{P_{i+1/2,j} - P_{i-1/2,j}}{\Delta x_i + \Delta x_{i+1}} (\Delta x_i + \Delta x_{i+1}) (\Delta y_j + \Delta y_{j+1}) \\
&= \sum_{i=2} \sum_{j=2} (P_{i+1/2,j} - P_{i-1/2,j}) (\Delta y_j + \Delta y_{j+1}) \\
&= \sum_{j=2} \left((\Delta y_j + \Delta y_{j+1}) \sum_{i=2} (P_{i+1/2,j} - P_{i-1/2,j}) \right) \\
&= \sum_j \left((\Delta y_j + \Delta y_{j+1}) \left(P_{2\frac{1}{2},j} - P_{1\frac{1}{2},j} + P_{3\frac{1}{2},j} - P_{3\frac{1}{2},j} + \dots \right) \right) \\
&= \sum_j \left((\Delta y_j + \Delta y_{j+1}) \left(P_{\text{IMAX}-\frac{1}{2},j} - P_{1\frac{1}{2},j} \right) \right) \\
&= \sum_j P_{\text{IMAX}-\frac{1}{2},j} (\Delta y_j + \Delta y_{j+1}) - \sum_j P_{1\frac{1}{2},j} (\Delta y_j + \Delta y_{j+1})
\end{aligned} \tag{4.16}$$

The right side is simply a numerical approximation to the boundary integral $\oint P \vec{i} \cdot \vec{n} dS$ over the left and right boundaries of the computational domain, respectively.

Only these two boundaries contribute, because on the top and bottom boundaries, the unit normal vector \vec{n} is normal to \vec{i} .

Similarly, the second term in the discrete approximation (4.9) reduces to a line integral over the top and bottom boundaries. We thus conclude that equation (4.9) satisfies the divergence theorem in a numerical sense.

What happened to the P and Q values at the interior points? Because of the nature of equation (4.9), P and Q contributions from adjacent cells simply cancel out. Such a sum where all intermediate components vanish leaving only the first and last term is known as the telescopic sum.

Exercise 4.3 Show that the use of equation (4.10) everywhere will also satisfy the divergence theorem. Assume the summation begins at $i = 3$.

Now observe what happens if we abruptly switch from from (4.9) to (4.10) in just a few cells within the computational domain, as recommended by Murman and Cole. The telescopic sum features independently satisfied by (4.9) and (4.10) are destroyed in the vicinity of the boundaries of these cells where switch occurs. Then the divergence theorem will never be satisfied, and our area integral will never reduce to purely boundary integrals over the outer edges of the computational domain.

Numerical schemes which satisfy the Divergence theorem in a numerical sense are called conservative. They guarantee that mass will not be spuriously generated within the computational domain. We are discussing "conservation of mass" because our TSD is a descendent of the continuity equation. In these schemes, mass flowing in through the outer boundaries will equal mass flowing out through the outer boundaries. The 1971 Murman-Cole scheme is non-conservative. It yields non-zero interior point contributions at the cells where flow changes from subsonic to supersonic (sonic line) and where flow changes from supersonic to subsonic (shocks). It is this nonphysical production of mass near shock waves that causes the shocks to be weaker, than would be expected from conservation of mass arguments.

4.11 Murman's Conservative Form

In 1973, Murman proposed a simple way for making the original scheme conservative. He examined the first and last term in (4.13). The first term is identical to that in equation (4.9) and satisfies the telescopic sum property. It is the last term in equation 4.13, which abruptly appears only in supersonic regions which destroys the telescopic sum property. Murman proposed that this last term be slightly modified leading to the following discrete approximation to the 2-D TSD equation:

$$\frac{P_{i+1/2,j} - P_{i-1/2,j}}{\Delta x_i + \Delta x_{i-1}} + \frac{Q_{i,j+1/2} - Q_{i,j-1/2}}{\Delta y_j + \Delta y_{j+1}} - \left[\mu_{i,j} \frac{P_{i+1/2,j} - P_{i-1/2,j}}{\Delta x_i + \Delta x_{i-1}} - \mu_{i-1,j} \frac{P_{i-1/2,j} - P_{i-3/2,j}}{\Delta x_i + \Delta x_{i-1}} \right] = 0 \quad (4.17)$$

It is easy to see that equation (4.17) reduces to equation (4.9) in subsonic regions where $\mu_{i,j}$ and $\mu_{i-1,j}$ are both zero. Similarly, this equation reduces to equation (4.10) when both $\mu_{i,j}$ and $\mu_{i-1,j}$ are unity. Thus, the above equation preserves the essential features of the Murman and Cole scheme. Unlike (4.10), equation (4.17) is conservative. We already know that the first and second term in (4.17) leads to telescopic sums, and are numerically conservative. To see why the third term is also conservative, let

$$G_{i,j} = -\mu_{i,j} (P_{i+1/2,j} - P_{i-1/2,j})$$

Then it follows that

$$G_{i-1,j} = -\mu_{i-1,j} (P_{i-1/2,j} - P_{i-3/2,j})$$

Equation (4.17) may then be viewed as

$$\frac{P_{i+1/2,j} - P_{i-1/2,j}}{\Delta x_i + \Delta x_{i+1}} + \frac{Q_{i,j+1/2} - Q_{i,j-1/2}}{\Delta y_j + \Delta y_{j+1}} + \frac{G_{i+1/2,j} - G_{i-1/2,j}}{\Delta x_i + \Delta x_{i+1}} = 0 \quad (4.18)$$

The third term in equation (4.18) is similar in form to the first term. It is easily demonstrated that this third term will yield a telescopic sum when summed over all the cells. The summation will result only in contributions from a boundary integral, no matter how many times μ changes its numerical magnitude between 0 and 1 from cell to cell. In other words, equation (4.18) and its original form (4.17) are numerically conservative.

Some additional observations on Murman's conservative form are now in order.

- (a) The conservative scheme is still first order accurate in supersonic regions. A large number of cells, and a small grid spacing in the x- direction should be used to achieve acceptable accuracy.
- (b) Consider the situation where $\mu_{i,j}$ is zero (subsonic point) and $\mu_{i-1,j}$ is 1 (supersonic point). Such a situation occurs across a shock wave, and the point (i,j) is known as the shock point.

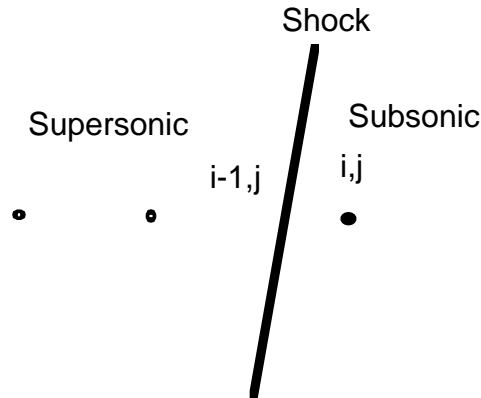


Figure 4.9 Shock Point

At the shock point equation (4.17) reduces to

$$\frac{P_{i+1/2,j} - P_{i-1/2,j}}{\Delta x_i + \Delta x_{i+1}} + \frac{Q_{i,j+1/2} - Q_{i,j-1/2}}{\Delta y_{j+1} + \Delta y_j} + \frac{P_{i-1/2,j} - P_{i-3/2,j}}{\Delta x_i + \Delta x_{i+1}} = 0 \quad (4.19)$$

Because P involves derivatives of ϕ , the above approximation will involve the four nodes (i-2,j), (i-1,j), (i,j) and (i+1,j) in the x-direction, rather than the three nodes used in the Murman-Cole scheme. Thus, the shocks captured by the 1973 Murman conservative form will be more smeared than the shocks from the 1971 original form, although stronger.

4.12 Application of Solid Wall Boundary Conditions

We now turn our attention to the implementation of boundary conditions at the boundaries, starting with the slit. From on, we will consider only the discretized form given by equation (4.17), because of its ability to conserve mass. Consider first a typical node above the slit, on the grid line $j = \text{JBODY}+1$. One must avoid taking derivatives of

the velocity potential ϕ across the slit, because the function and its y-derivative are both discontinuous at the slit.

The first term and last set of terms involving P need no special care since they involve only a derivative in the x- direction. The terms involving Q contain y-derivatives of ϕ and require care. We make use of the solid wall boundary condition derived in Chapter III:

$$\frac{\partial \phi}{\partial y} = V_{\infty} \frac{dY}{dx} \quad (4.20)$$

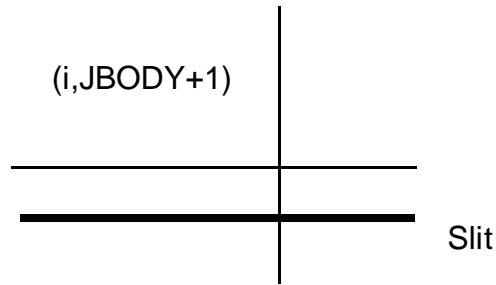


Figure 4.10 Treatment of Nodes above the Slit

Making use of (4.20), the terms involving Q in equation (4.17) may be written as

$$\frac{Q_{i,JBODY+3/2} - V_{\infty} \left. \frac{dY}{dx} \right|_{i,UPPER}}{\Delta y_j + \Delta y_{j+1}} \quad (4.21)$$

The nodes underneath the slit ($j = JBODY$) are similarly handled. Note that the solid surface boundary conditions thus enter our analysis in a natural, straightforward manner, and influence our calculations.

4.13 Application of Murman Conservative Form across the Cut

As discussed in Chapter III, in lifting airfoil calculations, the velocity potential has a discontinuity equal to the circulation Γ along the cut. This cut is chosen to be the line starting from the trailing edge and ending at the downstream boundary. Across this cut ϕ is discontinuous.

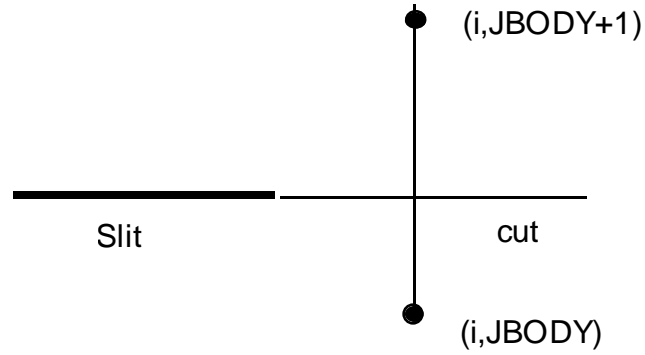


Figure 4.12 Treatment of Nodes above the Slit

Consider a typical node $(i, JBODY+1)$ just above the cut, downstream of the slit trailing edge. Before applying equation (4.17) at this node, the circulation is first computed as

$$\Gamma = \phi_{ITE,JBODY+1} - \phi_{ITE,JBODY} \quad (4.22)$$

Next, the terms involving Q in equation (4.17) are written as

$$\frac{Q_{i,JBODY+1/2} - Q_{i,JBODY-1/2}}{\Delta y_{j+1} + \Delta y_j} = \frac{\left(\frac{\phi_{i,JBODY+1} - \phi_{i,JBODY}}{\Delta y_{JBODY+1}} - \frac{\phi_{i,JBODY} - \phi_{i,JBODY} - \Gamma}{\Delta y_{JBODY}} \right)}{\Delta y_{JBODY+1} + \Delta y_{JBODY}} \quad (4.23)$$

In other words, we subtract the jump in ϕ before computing the y -derivative. The points underneath the cut ($j = JBODY$) are handled in a similar manner.

4.14 Application of Far Field Boundary Conditions

In Chapter III, the appropriate boundary conditions were derived for the outer boundary, both for subsonic and supersonic freestream Mach numbers. These boundary conditions are briefly summarized in their discretized form below.

Subsonic Freestream:

At all the points on the outer (upstream, downstream, top and bottom) boundaries, the following equation is applied:

$$\Phi_{\text{Far Field}} = \frac{\Gamma}{2\pi} \arctan\left(\sqrt{1 - M_\infty^2} \frac{y}{x}\right) \quad (4.23)$$

Here Γ is the circulation found from equation (4.22). The coordinates of the far field may be measured from any convenient reference point on the slit, such as the quarter-chord point.

Equation (4.23) is usually applied following every iterative step in the relaxation procedure, after the values of ϕ at the interior points are evaluated.

Supersonic freestream, upstream boundary ($i = 1$):

If the freestream Mach number is greater than 1, then the effects of the airfoil can not be felt at the upstream boundary. In such a case, we set ϕ to be zero at the upstream boundary.

Supersonic freestream, downstream boundary ($i = \text{IMAX}$):

In this case, we have two choices. We can either apply the compatibility equations derived from the method of characteristics (see Chapter III) or apply the governing equations. The easiest thing to do is to apply the governing equations, as given by equation (4.17). In other words, we treat the downstream boundary like any other interior point. We need not prescribe any information downstream, because equation (4.17) does not require any information downstream of our downstream boundary.

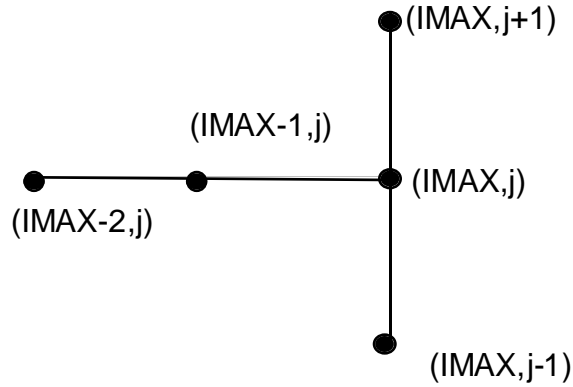


Figure 4.13 For supersonic outflow, equation (4.17) depends only nodes shown

Supersonic freestream, top boundary ($j = JMAX$):

On this boundary, we can use the method of characteristics to arrive at the following compatibility relation, as discussed in Chapter III.

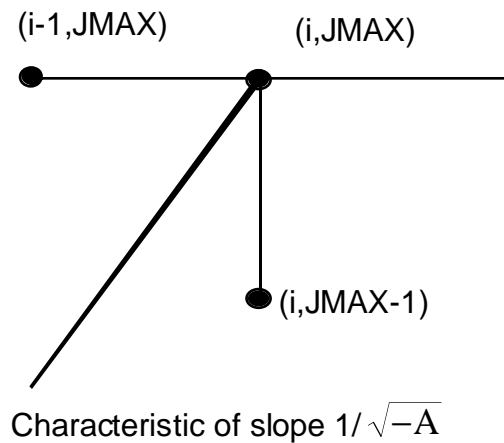


Figure 4.14 At the top boundary the method of characteristic may be used

$$\varphi = \text{constant} \quad (4.24)$$

Thus φ may on the boundaries may be set to the value of φ in the interior, along the characteristic. Since this characteristic may not exactly pass through an interior

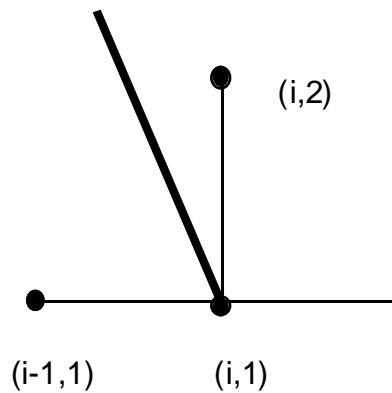
node, linear interpolation of ϕ at the two interior nodes between which the characteristic passes may be necessary.

Supersonic freestream, bottom boundary:

This boundary is treated in a manner similar to the top boundary, using the following compatibility relation, obtained from the method of characteristics (see Chapter III):

$$\phi = \text{constant} \tag{4.25}$$

Characteristic of slope : $-1/\sqrt{-A}$



4.15 Relaxation Procedure for solving the Discretized form of the TSD Equation:

We finally turn our attention to solving equation (4.17) at every interior point. Because P is a nonlinear function of ϕ , equation (4.17) is a nonlinear algebraic equation linking ϕ at a typical node (i,j) to its neighbors. The task on hand is to solve a very large number of nonlinear simultaneous equations to determine ϕ at all the interior nodes. Clearly, this large set of nonlinear equations applied at 5000 or more interior nodes can not be directly solved. The usual practice is to use an iterative approach, *similar to the Newton-Raphson iterative method* used to find the roots of nonlinear algebraic equations.

Let 'n' be the index that we use to keep track of the iterations. Then the symbol $\phi_{i,j}^n$ represents the value of ϕ at the node (i,j) at the end of iteration 'n'. The symbol $\phi_{i,j}^{n+1}$ represents the value of ϕ at the node (i,j) at the end of the next iteration 'n+1'. We use the symbol $\Delta\phi_{i,j}$ to represent the change in the value of ϕ at the node (i,j) between any two successive iteration. That is,

$$\Delta\phi_{i,j} = \phi_{i,j}^{n+1} - \phi_{i,j}^n \quad (4.28)$$

If our iterative process converges, then $\Delta\phi_{i,j}$ will go to zero at all the interior nodes after a sufficiently large number of iterations. At the same time, equation (4.17) will be satisfied at all the interior points in the computational domain.

Starting guess for ϕ :

We need to start our iterative process with some estimated values of ϕ in the interior. A robust iterative process will (hopefully) converge to the correct solution regardless of our initial guess. For lack of anything else, we set ϕ at all the interior points to zero to start our calculations. That is,

$$\phi_{i,j} = 0 \quad (4.29)$$

Linearization of the Discretized form of the TSD Equation:

The next step in the relaxation procedure is to arrive at a system of linear equations for $\Delta\phi$, one equation per interior node. This is accomplished as follows. Consider a term such as $Q_{i,j+1/2}$ that appears in our discrete form. This term may be expressed as follows:

$$\begin{aligned} Q_{i,j+1/2}^{n+1} &= \phi_y \Big|_{i,j+1/2}^{n+1} = \frac{\phi_{i,j+1}^{n+1} - \phi_{i,j}^{n+1}}{\Delta y_{j+1}} \\ &= \frac{\phi_{i,j+1}^n - \phi_{i,j}^n}{\Delta y_{j+1}} + \frac{\Delta\phi_{i,j+1} - \Delta\phi_{i,j}}{\Delta y_{j+1}} \\ &= Q_{i,j+1/2}^n + \frac{\Delta\phi_{i,j+1} - \Delta\phi_{i,j}}{\Delta y_{j+1}} \end{aligned} \quad (4.30)$$

Notice that we have just expressed an unknown quantity Q in terms of its known value from a previous iteration 'n', and unknown values of $\Delta\phi$ at the node points. We can similarly express Q at the half node $(i,j-1/2)$ at the iteration level 'n+1' as the sum of two parts, one involving Q at the 'n' level, plus an expression involving $\Delta\phi$.

Next, consider a term such as P at the half node $(i+1/2,j)$ at the iteration level 'n+1'. This term also may be expressed in terms of P at a known iteration level, and expressions involving $\Delta\phi$.

$$\begin{aligned}
P_{i+1/2,j}^{n+1} &= \left(1 - M_\infty^2\right)\phi_x - \frac{(\gamma + 1)}{2}M_\infty^2(\phi_x)^2 \Big|_{i+1/2,j}^{n+1} \\
&= \left(1 - M_\infty^2\right)(\phi^n + \Delta\phi)_x - \frac{(\gamma + 1)}{2}M_\infty^2\left((\phi^n + \Delta\phi)_x\right)^2 \Big|_{i+1/2,j} \\
&\cong \left(1 - M_\infty^2\right)\phi_x - \frac{(\gamma + 1)}{2}M_\infty^2(\phi_x)^2 \Big|_{i+1/2,j}^n + \left[\left(1 - M_\infty^2\right) - (\gamma + 1)M_\infty^2\phi_x^n\right](\Delta\phi)_x \Big|_{i+1/2,j} \\
&= P_{i+1/2,j}^n + A_{i+1/2,j}^n(\Delta\phi)_x \Big|_{i+1/2,j} \\
&= P_{i+1/2,j}^n + A_{i+1/2,j}^n \frac{\Delta\phi_{i+1,j} - \Delta\phi_{i,j}}{\Delta x_{i+1}}
\end{aligned} \tag{4.31}$$

When each and every term in equation (4.17) is similarly expressed as a known term at the iteration level 'n', and an unknown quantity involving $\Delta\phi$, after grouping the known terms on the right hand side, and the unknown quantities on the left hand side, the following linear system of equations for $\Delta\phi$ results:

$$\begin{aligned}
&B_{i,j}\Delta\phi_{i-2,j} + C_{i,j}\Delta\phi_{i-1,j} + D_{i,j}\Delta\phi_{i,j} + \\
&E_{i,j}\Delta\phi_{i+1,j} + F_{i,j}\Delta\phi_{i,j-1} + G_{i,j}\Delta\phi_{i,j+1} = R_{i,j}^n
\end{aligned} \tag{4.32}$$

where $R_{i,j}$ is called the residual, and is simply the negative of the left side of equation (4.17), computed at the previous iteration 'n'. The coefficients B, C etc. are all easily found as follows:

$$\begin{aligned}
B_{i,j} &= A_{i-\frac{3}{2},j} \frac{\mu_{i-1,j}}{\Delta x_{i-1}(\Delta x_i + \Delta x_{i+1})} \\
C_{i,j} &= (1 - \mu_{i,j}) \frac{A_{i-1/2,j}}{\Delta x_i(\Delta x_i + \Delta x_{i+1})} \\
&\quad - \mu_{i-1,j} \frac{1}{(\Delta x_i + \Delta x_{i+1})} \left(\frac{A_{i-1/2,j}}{\Delta x_i} + \frac{A_{i-3/2,j}}{\Delta x_{i-1}} \right) \\
E_{i,j} &= (1 - \mu_{i,j}) \frac{A_{i+1/2,j}}{\Delta x_{i+1}(\Delta x_i + \Delta x_{i+1})} \\
F_{i,j} &= \frac{1}{\Delta y_j(\Delta y_j + \Delta y_{j+1})} \\
G_{i,j} &= \frac{1}{\Delta y_{j+1}(\Delta y_j + \Delta y_{j+1})} \\
D_{i,j} &= -(1 - \mu_{i,j}) \frac{\left(\frac{A_{i-1/2,j}}{\Delta x_i} + \frac{A_{i+1/2,j}}{\Delta x_{i+1}} \right)}{(\Delta x_i + \Delta x_{i+1})} \\
&\quad + \mu_{i-1,j} \frac{1}{(\Delta x_i + \Delta x_{i+1})} \left(\frac{A_{i-1/2,j}}{\Delta x_i} \right) - (F_{i,j} + G_{i,j})
\end{aligned} \tag{4.33}$$

Exercise 4.4: Verify the expression given in equation (4.33)

Equation (4.32) may be viewed as the following matrix system of equations:

$$[M]\{\Delta\phi\} = \{R\} \tag{4.34}$$

where M is a sparse, banded matrix with six diagonals, one corresponding to each of the coefficients defined in equation (4.34). This matrix may be stored in the memory of a computer system simply by storing the six diagonals defined above.

Numerical Solution of Equation (4.34)

Although M is a sparse matrix with lots of zeros, it is still a formidable matrix to invert using conventional techniques such as the Gaussian elimination scheme. Such schemes require arithmetic operations of the order $O(N^3)$ where N is the number of equations. Since we are solving $\Delta\phi$ at several thousand nodes, N is very large and a direct inversion is not recommended.

Since we have already made some approximations while deriving equation (4.33) such as throwing away $\Delta\phi^2$ terms, we can make additional approximations on the left hand side of (4.34). Our goal is to drive the right hand side of (4.34) to zero. We can simplify our solution process by replacing the matrix M on the left hand side with another matrix N that is easy to invert and which is closely related to M . In other words, we solve instead,

$$[N]\{\Delta\phi\} = \{R\} \quad (4.35)$$

What criteria should be used to select the matrix N ? From numerical analysis, it is known that any matrix N will do, provided the absolute values of all the eigenvalues of the matrix $[I] - [N]^{-1}[M]$ is less than unity. Some commonly used $[N]$ matrices are discussed next. We will not attempt to prove that these matrices satisfy the criterion $[I] - [N]^{-1}[M]$ is less than unity.

Point Jacobi Scheme:

In this approach, the matrix $\{N\}$ contains only the diagonal elements of $[M]$. In other words, we solve

$$D_{i,j}\Delta\phi_{i,j} = R_{i,j}$$

Or,

$$\Delta\phi_{i,j} = \frac{R_{i,j}}{D_{i,j}} \quad (4.36)$$

The Point Jacobi scheme converges in all situations, although very slowly. For a typical lifting transonic flow, several thousand iterations are usually needed for R and $\Delta\phi$ to be driven to zero.

Gauss Seidel Scheme:

In this scheme, the N is either the lower triangular part of M , or the upper triangular part of M . If the lower triangular part is chosen, we solve

$$B_{i,j}\Delta\phi_{i-2,j} + C_{i,j}\Delta\phi_{i-1,j} + D_{i,j}\Delta\phi_{i,j} + F_{i,j}\Delta\phi_{i,j-1} = R_{i,j}$$

Or,

$$\Delta\phi_{i,j} = \frac{R_{i,j} - B_{i,j}\Delta\phi_{i-2,j} - C_{i,j}\Delta\phi_{i-1,j} - F_{i,j}\Delta\phi_{i,j-1}}{D_{i,j}} \quad (4.37)$$

Note that equation (4.37) may be easily implemented in a double Do-loop over i and j , in any high level programming language.

If the upper triangular part is chosen, we solve

$$D_{i,j}\Delta\phi_{i,j} + E_{i,j}\Delta\phi_{i+1,j} + G_{i,j}\Delta\phi_{i,j+1} = R_{i,j}$$

Or,

$$\Delta\phi_{i,j} = \frac{R_{i,j} - E_{i,j}\Delta\phi_{i+1,j} - G_{i,j}\Delta\phi_{i,j+1}}{D_{i,j}} \quad (4.38)$$

Equation (4.38) is also easily implemented on computers. For subsonic flows, both equation (4.37) and (4.38) work well. In transonic flows, where the information propagates from the upstream points $(i-1,j)$ $(i-2,j)$ to downstream points, equation (4.37) works well, while (4.38) may diverge.

Point Successive Over-Relaxation (SOR) Scheme:

This method is identical to the Gauss-Seidel scheme discussed in equations (4.37) and (4.38) above, except the residual R on the right hand side is multiplied by an

over-relaxation factor ω , between 1 and 2. This factor is chosen by trial and error for best convergence.

Successive Line Over-Relaxation Scheme:

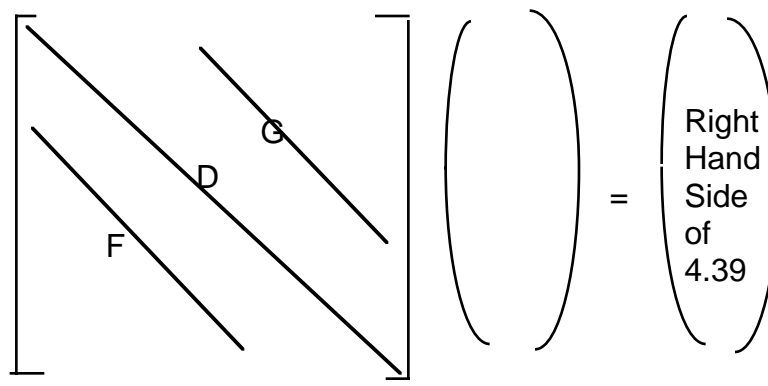
In this scheme, five of the six diagonals that constitute the M matrix are kept, and only the term involving the coefficient $F_{i,j}$ is neglected. The right hand side is multiplied by a relaxation factor ω , between 1 and 2. We therefore solve

$$B_{i,j}\Delta\phi_{i,-2j} + C_{i,j}\Delta\phi_{i-1,j} + D_{i,j}\Delta\phi_{i,j} + F_{i,j}\Delta\phi_{i,j-1} + G_{i,j}\Delta\phi_{i,j+1} = \omega R_{i,j}$$

Or,

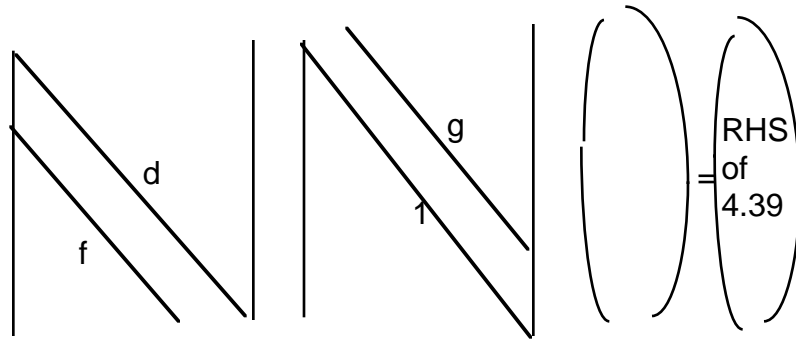
$$D_{i,j}\Delta\phi_{i,j} + F_{i,j}\Delta\phi_{i,j-1} + G_{i,j}\Delta\phi_{i,j+1} = \omega R_{i,j} - B_{i,j}\Delta\phi_{i,-2j} - C_{i,j}\Delta\phi_{i-1,j} \quad (4.39)$$

In the SLOR scheme, we solve for the $\Delta\phi$ values on all the nodes on a vertical grid line, $i = \text{constant}$, simultaneously. When we are solving for $\Delta\phi$ at an i - station, the $\Delta\phi$ values at the previous stations $(i-1,j)$ and $(i-2,j)$ are already known and may be brought to the right side as shown in equation (4.39). The resulting form is a tri-diagonal system of equations linking $\Delta\phi$ at the nodes (i,j) , $(i,j-1)$ and $(i,j+1)$:



$$\begin{bmatrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix} \begin{pmatrix} \\ \\ \\ \\ \end{pmatrix} = \begin{pmatrix} \text{Right} \\ \text{Hand} \\ \text{Side} \\ \text{of} \\ 4.39 \end{pmatrix} \quad (4.40)$$

The Thomas algorithm is simply the Gaussian elimination scheme, intelligently implemented, taking advantage of the fact that the matrix N has many zero elements. In this scheme, the matrix N is factored into a product of a lower-triangular L- matrix, and an upper-triangular matrix U as shown.



(4.41)

The elements f , d and g of the LU form are recursively related to the original elements of matrix on the left side of 4.40. These may be stored as one-dimensional vectors (or arrays). Specifically,

$$f_j = F_{i,j}$$

$$d_j = D_{i,j} - f_j g_{j-1}$$

$$g_j = \frac{G_{i,j}}{d_j}$$

for $j = 3, \dots, JMAX - 1$

with

$$f_2 = 0$$

$$d_2 = D_{i,2}$$

$$g_2 = \frac{G_{i,2}}{d_2}$$

(4.42)

Once the elements of L- and U- matrices are known from 4.42 , we can solve (4.41) easily.

Approximate Factorization Methods:

In this approach, the $\{N\}$ matrix we choose is one that is easily factored into smaller (tridiagonal, or LU) matrices. Approximate factorization methods are much

faster than the SLOR scheme, but are somewhat more difficult to program. For an excellent approximate factorization scheme for solving the 2-D TSD equation, the reader is referred to an article by Ballhaus, Jameson and Albert in the *AIAA Journal*, 1978.

Concluding Remarks

In this chapter, we discussed how to solve numerically the TSD equation and predict lifting transonic potential flows. We discussed techniques for grid generation, techniques for discretizing the TSD equation so that it reduces to an algebraic form and techniques for iteratively solving the resulting algebraic equations.