Hoffmann & Chiang

Subroutine tridiag2 in file tmix3d.f

APPENDIX B:

Tridiagonal System of Equations

The formulation of many implicit methods for a scalar PDE results in the following equation:

$$a_i^n u_{i-1}^{n+1} + b_i^n u_i^{n+1} + c_i^n u_{i+1}^{n+1} = D_i^n \tag{B-1}$$

Once this equation is applied to all the nodes at the advanced level, a system of linear algebraic equations is obtained. When these equations are represented in a matrix form, the coefficient matrix is tridiagonal. We will take advantage of the tridiagonal nature of the coefficient matrix and review a very efficient solution procedure.

To see the matrix formulation of the equations, consider the Laasonen implicit formulation of our diffusion model equation, i.e., Equation (3-12), presented here as

$$(d)u_{i-1}^{n+1} - (2d+1)u_i^{n+1} + (d)u_{i+1}^{n+1} = -u_i^n \tag{B-2}$$

where $d = \frac{\alpha \Delta t}{(\Delta x)^2}$ is the diffusion number. Define the following coefficients of (B-2) according to the formulation of (B-1):

$$\begin{aligned} a &= d \\ b &= -(2d+1) \\ c &= d \\ D &= \text{RHS} \end{aligned}$$

Applying Equation (B-2) to all the grid points will result in the following set of linear algebraic equations:

$$i = 2 \qquad a_2 u_1 + b_2 u_2 + c_2 u_3 = D_2 \tag{B-3}$$

$$i = 3 \qquad a_3u_2 + b_3u_3 + c_3u_4 = D_3$$

$$i = 4 \qquad a_4u_3 + b_4u_4 + c_4u_5 = D_4$$

$$\vdots$$

$$i = IM2 \qquad a_{IM2}u_{IM3} + b_{IM2}u_{IM2} + c_{IM2}u_{IM1} = D_{IM2}$$

$$i = IM1 \qquad a_{IM1}u_{IM2} + b_{IM1}u_{IM1} + c_{IM1}u_{IM} = D_{IM1} \tag{B-4}$$

where $IM1 = IM - 1$, $IM2 = IM - 2$, and so on. In addition, note that the superscript $n + 1$ has been dropped from the equations. Assume that Dirichlet boundary conditions are imposed and, therefore, the values of the dependent variable u at $i = 1$ (the lower boundary) and at $i = IM$ (the upper boundary) are given. Then the first equation, (B-3), and the last equation, (B-4), can be written as:

$$b_2u_2 + c_2u_3 = D_2 - a_2u_1$$

and

$$a_{IM1}u_{IM2} + b_{IM1}u_{IM1} = D_{IM1} - c_{IM1}u_{IM}$$

Now, the set of equations in matrix formulation is

$$\begin{bmatrix} b_2 & c_2 & & & & & \\ a_3 & b_3 & c_3 & & & & \\ & a_4 & b_4 & c_4 & & & \\ & & & & & & \\ & & & & a_{IM2} & b_{IM2} & c_{IM2} \\ & & & & & a_{IM1} & b_{IM1} \end{bmatrix} \begin{bmatrix} u_2 \\ u_3 \\ u_4 \\ | \\ u_{IM2} \\ u_{IM1} \end{bmatrix} = \begin{bmatrix} D_2 - a_2u_1 \\ D_3 \\ D_4 \\ | \\ D_{IM2} \\ D_{IM1} - c_{IM1}u_{IM} \end{bmatrix}$$

Applying other types of boundary conditions does not change the tridiagonal form of the coefficient matrix, and can be easily implemented. An example is given below, where the Neumann boundary condition at $i = IM$ is applied. Impose the boundary condition $\frac{\partial u}{\partial x} = 0$ (at $i = IM$). Then

$$\frac{\partial u}{\partial x} = \frac{u_{IMP1} - u_{IM}}{\Delta x}$$

or

$$u_{IM} = u_{IMP1}$$

Note that we have introduced a fictitious boundary at $IMP1 = IM + 1$. The grid points are shown in Figure B-1. Now, at $i = IM$ (the upper boundary), we have

$$a_{IM}u_{IM1} + b_{IM}u_{IM} + c_{IM}u_{IMP1} = D_{IM}$$

or

$$a_{IM}u_{IM1} + (b_{IM} + c_{IM})u_{IM} = D_{IM}$$

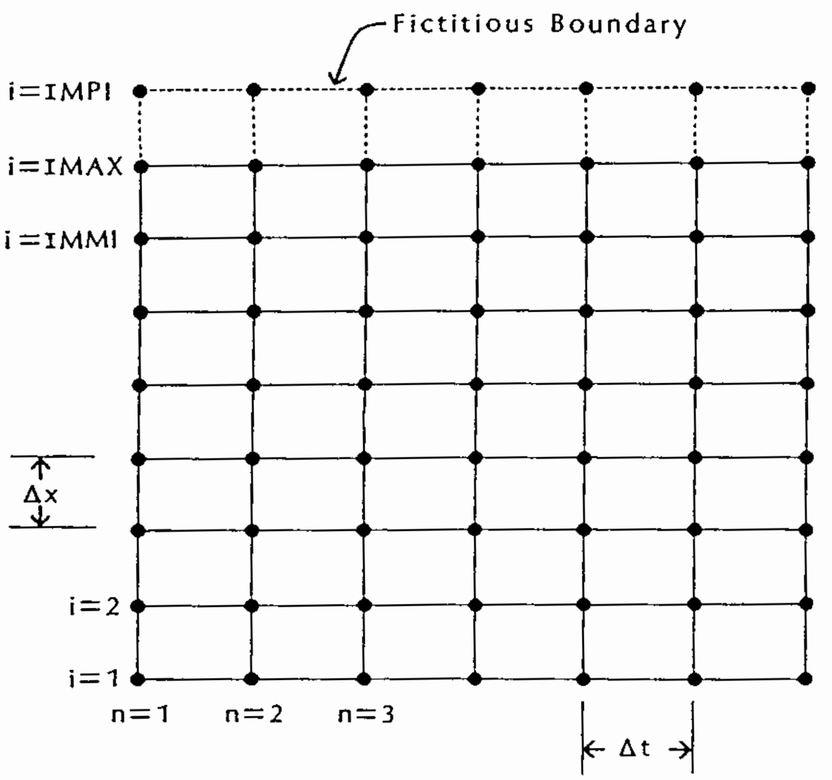

Figure B-1. Illustration of grid points with fictitious boundary.

and the matrix representation takes the following form:

$$\begin{bmatrix} b_2 & c_2 & & & & \\ a_3 & b_3 & c_3 & & & \\ & a_4 & b_4 & c_4 & & \\ & & & & & \\ & & & a_{IM1} & b_{IM1} & c_{IM1} \\ & & & & a_{IM} & (b+c)_{IM} \end{bmatrix} \begin{bmatrix} u_2 \\ u_3 \\ u_4 \\ | \\ u_{IM1} \\ u_{IM} \end{bmatrix} = \begin{bmatrix} D_2 - a_2 u_1 \\ D_3 \\ D_4 \\ | \\ D_{IM1} \\ D_{IM} \end{bmatrix}$$

In the example above, a first-order approximation for the gradient was used. If a second-order approximation is utilized, then

$$\frac{\partial u}{\partial x} = \frac{u_{IMP1} - u_{IM1}}{2\Delta x}$$

or

$$u_{IMP1} = u_{IM1}$$

Thus, the equation at $i = IM1$ is modified according to

$$a_{IM} u_{IM1} + b_{IM} u_{IM} + c_{IM} u_{IM1} = D_{IM}$$

or

$$(a_{IM} + c_{IM}) u_{IM1} + b_{IM} u_{IM} = D_{IM}$$

To proceed with the solution technique, assume a solution of the form

$$u_i = -H_i u_{i+1} + G_i \tag{B-5}$$

where u_i is unknown, u_{i+1} is known from the imposed boundary condition, and H_i and G_i are yet to be determined. Apply Equation (B-5) at node $i-1$, then

$$u_{i-1} = -H_{i-1} u_i + G_{i-1} \tag{B-6}$$

Upon substitution of (B-6) into Equation (B-1), one has

$$a_i(-H_{i-1} u_i + G_{i-1}) + b_i u_i + c_i u_{i+1} = D_i$$

or

$$(b_i - a_i H_{i-1}) u_i + c_i u_{i+1} = D_i - a_i G_{i-1}$$

from which

$$u_i = -\frac{c_i}{b_i - a_i H_{i-1}} u_{i+1} + \frac{D_i - a_i G_{i-1}}{b_i - a_i H_{i-1}} \tag{B-7}$$

Comparing Equations (B-7) with (B-5), one concludes that

$$H_i = \frac{c_i}{b_i - a_i H_{i-1}} \tag{B-8}$$

and

$$G_i = \frac{D_i - a_i G_{i-1}}{b_i - a_i H_{i-1}} \tag{B-9}$$

Now that H_i and G_i have been determined, the recursion equation, (B-5), can be used to solve for all the unknowns.

To see how this procedure is applied, consider the parabolic equation investigated in Chapter 3, i.e., the suddenly accelerated plane. At the lower boundary $i = 1$, $u_1 = UWALL$ is specified for all times; therefore, (B-5) at $i = 1$ becomes

$$u_1 = -H_1 u_2 + G_1$$

Since this equation must hold for all u_2, $H_1 = 0$ and $G_1 = u_1 = UWALL$. With the values of H_1 and G_1 provided from the boundary condition, Equations (B-8) and (B-9) can be solved for the values of H_i and G_i at the second node. Subsequently, (B-8) and (B-9) are sequentially applied to all grid points to obtain the values of H_i and G_i. Note that the computation of H_i and G_i starts from the lower boundary and proceeds upward. Now, Equation (B-5) is used for the computation of u_i. This calculation is performed inward from the upper boundary. At $i = IM$, Equation (B-5) provides

$$u_{IM1} = -H_{IM1} u_{IM} + G_{IM1}$$

In this equation, u_{IM} is specified from the upper boundary condition, with H and G at $IM1$ previously determined. Once u_{IM1} is computed, Equation (B-5) is applied to compute u_{IM2} and so on. The solution procedure may be coded in the program or as a subroutine.

(1977, Chap. 5), and, at a more advanced level, by Duff (1981). Packages for implementing sparse Gauss elimination are available; the Harwell programs MA28, etc., discussed by Duff (pp. 1–29), are recommended.

6.2.2 Tridiagonal Systems: Thomas Algorithm

The use of three-point finite difference formulae or finite elements with linear interpolation leads, after splitting (Sect. 8.2), to a tridiagonal structure for $\underline{A}$ in (6.23). The use of higher-order finite difference schemes or higher-order finite elements produces a larger bandwidth in $\underline{A}$. The Thomas algorithm is suitable for solving (6.23) when $\underline{A}$ is tridiagonal. The extension of the Thomas algorithm to the case when $\underline{A}$ is pentadiagonal is described in Sect. 6.2.4. For systems of equations, $\underline{A}$ has a block (tridiagonal) structure typically. The treatment of $\underline{A}$ for this case is considered in Sect. 6.2.5.

When the nonzero elements lie close to the main diagonal it is useful to consider variants of Gauss elimination that take advantage of the banded nature of $\underline{A}$. An example is provided by the convection-diffusion problem of Sect. 9.3. Using centred difference formulae the following algorithm (in the present notation) is obtained

$$-(1+0.5R_{\text{cell}})v_{i-1}+2v_i-(1-0.5R_{\text{cell}})v_{i+1}=0 \ , \tag{6.27}$$

which, when repeated for every node, gives

$$\begin{bmatrix} b_1 & c_1 & & & & & \\ a_2 & b_2 & c_2 & & & & \\ & & \cdot & \cdot & \cdot & & \\ & & a_i & b_i & c_i & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & a_{N-1} & b_{N-1} & c_{N-1} \\ & & & & & a_N & b_N \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \cdot \\ v_i \\ \cdot \\ v_{N-1} \\ v_N \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \cdot \\ d_i \\ \cdot \\ d_{N-1} \\ d_N \end{bmatrix} , \tag{6.28}$$

where $a_i=-(1+0.5R_{\text{cell}})$, $b_i=2$, $c_i=-(1-0.5R_{\text{cell}})$. Nonzero values of d_i are associated with source terms, or, for d_1 and d_N, with boundary conditions. All terms in $\underline{A}$, other than those shown, are zero. The change in notation, particularly in relation to b_i, from that in Sect. 6.2.1, may be noted.

The Thomas algorithm for solving (6.28) consists of two parts (Fig. 6.17). First (6.28) is manipulated into the form

$$\begin{bmatrix} 1 & c'_1 & & & & \\ & 1 & c'_2 & & & \\ & & \cdot & \cdot & \cdot & \\ & & & 1 & c'_i & \\ & & & \cdot & \cdot & \cdot \\ & & & & 1 & c'_{N-1} \\ & & & & & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ \cdot \\ \cdot \\ v_i \\ \cdot \\ \cdot \\ v_N \end{bmatrix} = \begin{bmatrix} d'_1 \\ \cdot \\ \cdot \\ d'_1 \\ \cdot \\ \cdot \\ d'_N \end{bmatrix} ,$$

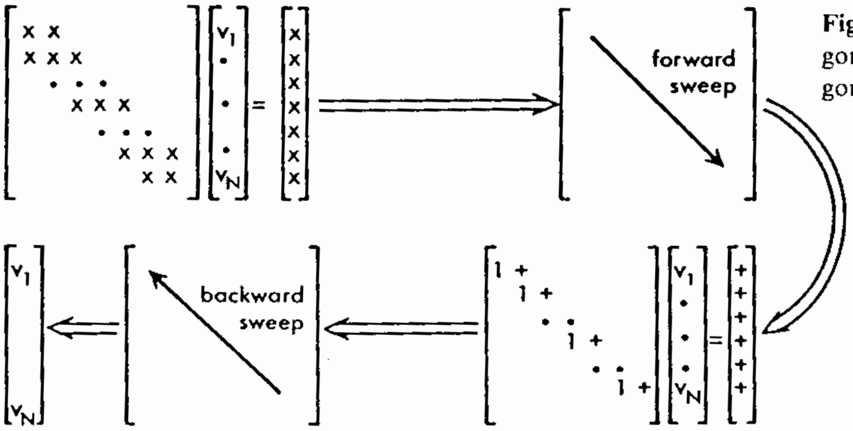

Fig. 6.17. The Thomas algorithm for solving a tridiagonal system of equations

i.e. the a_i coefficients have been eliminated and the b_i coefficients normalised to unity. For the first equation

$$c_1' = \frac{c_1}{b_1}, \quad d_1' = \frac{d_1}{b_1}, \tag{6.29}$$

and for the general equation

$$\begin{aligned} c_i' &= \frac{c_i}{b_i - a_i c_{i-1}'}, \\ d_i' &= \frac{d_i - a_i d_{i-1}'}{b_i - a_i c_{i-1}'}. \end{aligned} \tag{6.30}$$

The equations are modified, as in (6.30), in a forward sweep (Fig. 6.17). The second stage consists of a back-substitution (backward sweep in Fig. 6.17),

$$\begin{aligned} &v_N = d_N' \quad \text{and} \\ &v_i = d_i' - v_{i+1} c_i' . \end{aligned} \tag{6.31}$$

The Thomas algorithm is particularly economical; it requires only $5N-4$ operations (multiplications and divisions). But to prevent ill-conditioning (and hence round-off contamination) it is necessary that

$$|b_i| > |a_i| + |c_i| .$$

The use of splitting for multidimensional problems (Sect. 8.2) typically generates tridiagonal systems of equations that can be solved efficiently using the Thomas algorithm.

6.2.3 BANFAC/BANSOL: Narrowly Banded Gauss Elimination

When $\underset{\sim}{A}$ is narrowly banded subroutines BANFAC and BANSOL are suitable for performing Gauss elimination. Subroutine BANFAC (Fig. 6.18) carries out the

Time integration of potential temperature equation in the ARPS:

$$\rho^* \frac{\theta'^{\tau+\Delta\tau} - \theta'^{\tau}}{\Delta\tau} = -\left[\overline{\overline{\rho}^{\zeta} x_\xi y_\eta \delta_\zeta \bar{\theta} w}^{\zeta}\right]^{\tau} + f'_\theta . \tag{3.4e}$$

$$f'_\theta = -ADVT^t + \sqrt{G}\, D_\theta^{t-\Delta t} + \sqrt{G}\, S_\theta^t , \tag{3.5e}$$

where *ADVT* is the advection terms, and D_θ the diffusion / mixing term.

Note that D_θ is evaluated at time level t-Δt.

3.2.4. Special treatment of vertical mixing

Given the vertical mixing coefficients K_{mv} and K_{Hv} that are based, in the PBL, on the length scale l in Eq.(2.43), vertical turbulent mixing often results in a linear stability constraint more severe than that associated with advection, especially when the vertical resolution is high. This problem is dealt with by using an implicit scheme for the vertical mixing terms, wherein these mixing terms are represented as the weighted average of their values at $t + \Delta t$ and $t - \Delta t$, and the resultant tridiagonal equations are solved as in Eq.(A3). Paegle et al. (1976) showed in a 1-D boundary layer model that the implicit scheme is as accurate as the explicit scheme though much more efficient. In the ARPS, due to the use of mode-splitting scheme for acoustic waves, the implicit treatment of vertical mixing is done in two steps - the mixing terms are first integrated without the acoustic terms, their contributions to the time tendencies are added to the forcing terms which are then used in the small time steps to arrive at the final solution.

Control parameters related to vertically implicit treatment of turbulent mixing in the ARPS (from `arps.input`):

```
c#######################################################################

c   trbvimp    Option for implicit treatment of vertical mixing
c            = 0, vertical explicit (default);
c            = 1, vertical implicit
c
c   alfcoef    Time average weighting coefficient used
c              in the vertically implicit mixing (default is 1.0)
c
c#######################################################################

 &turbulence
   tmixopt   = 4,
   trbisotp  = 1,
   tkeopt    = 1,
   trbvimp   = 1,
   alfcoef   = 0.25,
   prantl    = 1.0,
   tmixcst   = 0.0,
   kmlimit   = 1.0,
 &END
```

$$\delta_t u_i^n = \sigma\left[\alpha\, \delta_{xx} u_i^{n+1} + (1-\alpha)\, \delta_{xx} u_i^n\right]$$

alfcoef $= 1-\alpha$

alfcoef $=1$ – explicit

$=0$ – fully implicit

Also read:

Durran: Sec. 3.4.1.

Hoffmann: Sec 4.3 on Diffusion equation