APPENDIX

A

SUBROUTINE FOR SOLVING A TRIDIAGONAL SYSTEM OF EQUATIONS

Subroutine SY solves a tridiagonal system of equations following the Thomas algorithm described in Chapter 4. To use the subroutine, the equations must be of the form

$$\begin{bmatrix} D_{IL} & A_{IL} & & \\ B_I & D_I & A_I & \\ & \ddots & \ddots & \ddots \\ & & B_{IU} & D_{IU} \end{bmatrix} \begin{bmatrix} U_{IL} \\ U_I \\ | \\ U_{IU} \end{bmatrix} = \begin{bmatrix} C_{IL} \\ C_I \\ | \\ C_{IU} \end{bmatrix} \tag{A.1}$$

The call statement for subroutine SY is of the form

$$\text{CALL SY}(IL, IU, B, D, A, C)$$

where B, D, A, and C are the array names for the singly subscripted real variables $B(I)$, $D(I)$, $A(I)$, $C(I)$. The variables IL and IU are unsubscripted integer variables. The arrays must be defined for subscripts ranging from IL to IU according to

B, Coefficient behind (to the left of) the main diagonal
D, Coefficient on the main diagonal
A, Coefficient ahead (to the right of) the main diagonal
C, Element in the constant vector

The equations in the system are ordered according to the value of the subscript. The variable *IL* corresponds to the subscript of the first equation in the system and *IU* corresponds to the subscript of the last equation in the system. The number of equations in the system is $IU - IL + 1$. *The solution vector,* **U**, *is returned to the calling program in the* C *array.* That is, the constant vector **C** is overwritten in the subroutine with the solution. The *D* array is also altered by the subroutine. *A* and *B* remain unchanged.

LISTING OF SUBROUTINE SY

```
C...
      SUBROUTINE SY(IL,IU,BB,DD,AA,CC)
      DIMENSION AA(1),BB(1),CC(1),DD(1)
C...
C...SUBROUTINE SY SOLVES TRIDIAGONAL SYSTEM BY ELIMINATION
C...IL = SUBSCRIPT OF FIRST EQUATION
C...IU = SUBSCRIPT OF LAST EQUATION
C...BB = COEFFICIENT BEHIND DIAGONAL
C...DD = COEFFICIENT ON DIAGONAL
C...AA = COEFFICIENT AHEAD OF DIAGONAL
C...CC = ELEMENT OF CONSTANT VECTOR
C...
C...ESTABLISH UPPER TRIANGULAR MATRIX
C...
      LP = IL+1
      DO 10 I = LP,IU
      R = BB(I)/DD(I-1)
      DD(I) = DD(I)-R*AA(I-1)
   10 CC(I) = CC(I)-R*CC(I-1)
C...
C...BACK SUBSTITUTION
C...
      CC(IU) = CC(IU)/DD(IU)
      DO 20 I = LP,IU
      J = IU-I+IL
   20 CC(J) = (CC(J)-AA(J)*CC(J+1))/DD(J)
C...
C...SOLUTION STORED IN CC
C...
      RETURN
      END
```

APPENDIX

B

SUBROUTINES FOR SOLVING BLOCK TRIDIAGONAL SYSTEMS OF EQUATIONS

The subroutines described here for solving block tridiagonal systems of equations were provided by Sukumar R. Chakravarthy of Rockwell International Science Center. Subroutine NBTRIP solves a block tridiagonal system of equations of the form

$$\begin{bmatrix} B_{IL} & C_{IL} & & \\ A_I & B_I & C_I & \\ & \ddots & \ddots & \ddots \\ & & A_{IU} & B_{IU} \end{bmatrix} \begin{bmatrix} X_{IL} \\ X_I \\ \vdots \\ X_{IU} \end{bmatrix} = \begin{bmatrix} D_{IL} \\ D_I \\ \vdots \\ D_{IU} \end{bmatrix} \tag{B.1}$$

Subroutine PBTRIP solves a periodic block tridiagonal system of equations in the form

$$\begin{bmatrix} B_{IL} & C_{IL} & & A_{IL} \\ A_I & B_I & C_I & \\ & \ddots & \ddots & \ddots \\ C_{IU} & & A_{IU} & B_{IU} \end{bmatrix} \begin{bmatrix} X_{IL} \\ X_I \\ \vdots \\ X_{IU} \end{bmatrix} = \begin{bmatrix} D_{IL} \\ D_I \\ \vdots \\ D_{IU} \end{bmatrix} \tag{B.2}$$

The block matrices A, B, and C are $N \times N$ matrices at every point I with N being an integer greater than 1. Note that for $N = 1$, the Thomas algorithm of Appendix A can be employed. The right-hand side vector D has length N at each point I. The total number of I points at which the matrices are defined (denoted by NI) is given by

$$NI = (IU - IL + 1) \tag{B.3}$$

The matrices A, B, and C are dimensioned as

$$A(N, N, NI)$$
$$B(N, N, NI)$$
$$C(N, N, NI)$$

while the vector D is dimensioned as

$$D(N, NI)$$

The call statement for subroutine NBTRIP is

CALL NBTRIP($A, B, C, D, IL, IU, ORDER$)

with arguments defined by

A, Subdiagonal block matrix
B, Diagonal block matrix
C, Superdiagonal block matrix
D, Right-hand side vector
IL, Lower value of I for which matrices are defined
IU, Upper value of I for which matrices are defined
$ORDER$, N (order can be any integer greater than 1)

The solution (X) is returned to the calling program by overwriting the D vector with the X vector. The calling statement for subroutine PBTRIP is

CALL PBTRIP($A, B, C, D, IL, IU, ORDER$)

with the same arguments as subroutine NBTRIP. However, if $ORDER$ is greater than 5 a dimension statement must be changed in this subroutine (see listing of subroutine).

Subroutines NBTRIP and PBTRIP employ no pivoting strategy in their elimination schemes. It should be noted that a specialized subroutine for solving a block tridiagonal system of equations can be written for each value of N which will be faster than the general subroutines given here.

LISTING OF SUBROUTINE NBTRIP

```
C...
C...SUBROUTINE TO SOLVE NON-PERIODIC BLOCK TRIDIAGONAL
C...SYSTEM OF EQUATIONS WITHOUT PIVOTING STRATEGY
C...WITH THE DIMENSIONS OF THE BLOCK MATRICES BEING
C...N x N (N IS ANY NUMBER GREATER THAN 1).
```

```
C...
      SUBROUTINE NBTRIP(A,B,C,D,IL,IU,ORDER)
      INTEGER ORDER,ORDSQ
      DIMENSION A(1),B(1),C(1),D(1)

C...
C...A = SUB DIAGONAL MATRIX
C...B =     DIAGONAL MATRIX
C...C = SUP DIAGONAL MATRIX
C...D = RIGHT HAND SIDE VECTOR
C...IL = LOWER VALUE OF INDEX FOR WHICH MATRICES ARE DEFINED
C...IU = UPPER VALUE OF INDEX FOR WHICH MATRICES ARE DEFINED
C...     (SOLUTION IS SOUGHT FOR BTRI(A,B,C)*X = D
C...     FOR INDICES OF X BETWEEN IL AND IU (INCLUSIVE).
C...     SOLUTION WRITTEN IN D VECTOR (ORIGINAL CONTENTS
C...     ARE OVERWRITTEN)).
C...ORDER = ORDER OF A,B,C MATRICES AND LENGTH OF D VECTOR
C...     AT EACH POINT DENOTED BY INDEX I
C...     (ORDER CAN BE ANY INTEGER GREATER THAN 1).
C...
C...THE MATRICES AND VECTORS ARE STORED IN SINGLE SUBSCRIPT FORM
C...
      ORDSQ = ORDER**2
C...
C...FORWARD ELIMINATION
C...
      I = IL
      IOMAT = 1+(I-1)*ORDSQ
      IOVEC = 1+(I-1)*ORDER
      CALL LUDECO(B(IOMAT),ORDER)
      CALL LUSOLV(B(IOMAT),D(IOVEC),D(IOVEC),ORDER)
      DO 100 J=1,ORDER
      IOMATJ = IOMAT+(J-1)*ORDER
      CALL LUSOLV(B(IOMAT),C(IOMATJ),C(IOMATJ),ORDER)
  100 CONTINUE
  200 CONTINUE
      I = I+1
      IOMAT = 1+(I-1)*ORDSQ
      IOVEC = 1+(I-1)*ORDER
      I1MAT = IOMAT-ORDSQ
      I1VEC = IOVEC-ORDER
      CALL MULPUT(A(IOMAT),D(I1VEC),D(IOVEC),ORDER)
      DO 300 J=1,ORDER
      IOMATJ = IOMAT+(J-1)*ORDER
      I1MATJ = I1MAT+(J-1)*ORDER
      CALL MULPUT(A(IOMAT),C(I1MATJ),B(IOMATJ),ORDER)
  300 CONTINUE
      CALL LUDECO(B(IOMAT),ORDER)
      CALL LUSOLV(B(IOMAT),D(IOVEC),D(IOVEC),ORDER)
      IF(I.EQ.IU) GO TO 500
      DO 400 J=1,ORDER
      IOMATJ = IOMAT+(J-1)*ORDER
      CALL LUSOLV(B(IOMAT),C(IOMATJ),C(IOMATJ),ORDER)
  400 CONTINUE
      GO TO 200
  500 CONTINUE
C...
C...BACK SUBSTITUTION
C...
      I = IU
  600 CONTINUE
      I = I-1
      IOMAT = 1+(I-1)*ORDSQ
      IOVEC = 1+(I-1)*ORDER
      I1VEC = IOVEC+ORDER
      CALL MULPUT(C(IOMAT),D(I1VEC),D(IOVEC),ORDER)
      IF (I.GT.IL) GO TO 600
C...
      RETURN
      END
```

LISTING OF SUBROUTINE PBTRIP

```
C...
C...SUBROUTINE TO SOLVE PERIODIC BLOCK TRIDIAGONAL
C...SYSTEM OF EQUATIONS WITHOUT PIVOTING STRATEGY.
C...EACH BLOCK MATRIX MAY BE OF DIMENSION N WITH
C...N ANY NUMBER GREATER THAN 1.
C...
      SUBROUTINE PBTRIP(A,B,C,D,IL,IU,ORDER)
      INTEGER ORDER,ORDSQ
      DIMENSION A(1),B(1),C(1),D(1)
      DIMENSION AD(25),CD(25)
C...
C...A = SUB DIAGONAL MATRIX
C...B =     DIAGONAL MATRIX
C...C = SUP DIAGONAL MATRIX
C...D = RIGHT HAND SIDE VECTOR
C...IL = LOWER VALUE OF INDEX FOR WHICH MATRICES ARE DEFINED
C...IU = UPPER VALUE OF INDEX FOR WHICH MATRICES ARE DEFINED
C...     (SOLUTION IS SOUGHT FOR BTRI(A,B,C)*X = D
C...     FOR INDICES OF X BETWEEEN IL AND IU (INCLUSIVE).
C...     SOLUTION WRITTEN IN D VECTOR (ORIGINAL CONTENTS
C...     ARE OVERWRITTEN)).
C...ORDER = ORDER OF A,B,C MATRICES AND LENGTH OF D VECTOR
C...     AT EACH POINT DENOTED BY INDEX I
C...     (ORDER CAN BE ANY INTEGER GREATER THAN 1)
```

```
C...      (ARRAYS AD AND CD MUST BE AT LEAST OF LENGTH ORDER**2)
C...      (CURRENT LENGTH OF 25 ANTICIPATES MAXIMUM ORDER OF 5).
C...
      IS = IL+1
      IE = IU-1
      ORDSQ = ORDER**2
      IUMAT = 1+(IU-1)*ORDSQ
      IUVEC = 1+(IU-1)*ORDER
      IEMAT = 1+(IE-1)*ORDSQ
      IEVEC = 1+(IE-1)*ORDER
C...
C...FORWARD ELIMINATION
C...
      I = IL
      IOMAT = 1+(I-1)*ORDSQ
      IOVEC = 1+(I-1)*ORDER
      CALL LUDECO(B(IOMAT),ORDER)
      CALL LUSOLV(B(IOMAT),D(IOVEC),D(IOVEC),ORDER)
      DO 10 J=1,ORDER
      IOMATJ = IOMAT+(J-1)*ORDER
      CALL LUSOLV(B(IOMAT),C(IOMATJ),C(IOMATJ),ORDER)
      CALL LUSOLV(B(IOMAT),A(IOMATJ),A(IOMATJ),ORDER)
   10 CONTINUE
C...
      DO 200 I = IS,IE
      IOMAT = 1+(I-1)*ORDSQ
      IOVEC = 1+(I-1)*ORDER
      I1MAT = IOMAT-ORDSQ
      I1VEC = IOVEC-ORDER
      DO 20 J=1,ORDSQ

      IOMATJ = J-1+IOMAT
      IUMATJ = J-1+IUMAT
      AD(J) = A(IOMATJ)
      CD(J) = C(IUMATJ)
      A(IOMATJ) = 0.0
      C(IUMATJ) = 0.0
   20 CONTINUE
      CALL MULPUT(AD,D(I1VEC),D(IOVEC),ORDER)
      DO 22 J=1,ORDER
      IOMATJ = IOMAT+(J-1)*ORDER
      I1MATJ = I1MAT+(J-1)*ORDER
      CALL MULPUT(AD,C(I1MATJ),B(IOMATJ),ORDER)
      CALL MULPUT(AD,A(I1MATJ),A(IOMATJ),ORDER)
   22 CONTINUE
      CALL LUDECO(B(IOMAT),ORDER)
      CALL LUSOLV(B(IOMAT),D(IOVEC),D(IOVEC),ORDER)
      DO 24 J=1,ORDER
      IOMATJ = IOMAT+(J-1)*ORDER
      CALL LUSOLV(B(IOMAT),C(IOMATJ),C(IOMATJ),ORDER)
      CALL LUSOLV(B(IOMAT),A(IOMATJ),A(IOMATJ),ORDER)
   24 CONTINUE
      CALL MULPUT(CD,D(I1VEC),D(IUVEC),ORDER)
      DO 26 J=1,ORDER
      IUMATJ = IUMAT+(J-1)*ORDER
      I1MATJ = I1MAT+(J-1)*ORDER
      CALL MULPUT(CD,A(I1MATJ),B(IUMATJ),ORDER)
      CALL MULPUT(CD,C(I1MATJ),C(IUMATJ),ORDER)
   26 CONTINUE
  200 CONTINUE
C...
      DO 30 J=1,ORDSQ
      IUMATJ = J-1+IUMAT
      AD(J) = A(IUMATJ)+C(IUMATJ)
   30 CONTINUE
      CALL MULPUT(AD,D(IEVEC),D(IUVEC),ORDER)
      DO 32 J=1,ORDER
      IUMATJ = IUMAT+(J-1)*ORDER
      IEMATJ = IEMAT+(J-1)*ORDER
      CALL MULPUT(AD,C(IEMATJ),B(IUMATJ),ORDER)
      CALL MULPUT(AD,A(IEMATJ),B(IUMATJ),ORDER)
   32 CONTINUE
      CALL LUDECO(B(IUMAT),ORDER)
      CALL LUSOLV(B(IUMAT),D(IUVEC),D(IUVEC),ORDER)
C...
C...BACK SUBSTITUTION
C...
      DO 40 IBAC = IL,IE
      I = IE-IBAC+IL
      IOMAT = 1+(I-1)*ORDSQ
      IOVEC = 1+(I-1)*ORDER
      I1VEC = IOVEC+ORDER
      CALL MULPUT(A(IOMAT),D(IUVEC),D(IOVEC),ORDER)
      CALL MULPUT(C(IOMAT),D(I1VEC),D(IOVEC),ORDER)
   40 CONTINUE
C...
      RETURN
      END

C...
C...SUBROUTINE TO CALCULATE L-U DECOMPOSITION
C...OF A GIVEN MATRIX A AND STORE RESULT IN A
C...(NO PIVOTING STRATEGY IS EMPLOYED)
C...
      SUBROUTINE LUDECO(A,ORDER)
      INTEGER ORDER
      DIMENSION A(ORDER,1)

C...
      DO 8 JC=2,ORDER
```

```
    8 A(1,JC) = A(1,JC)/A(1,1)
      JRJC = 1
   10 CONTINUE
      JRJC = JRJC+1
      JRJCM1 = JRJC-1
      JRJCP1 = JRJC+1
      DO 14 JR=JRJC,ORDER
      SUM = A(JR,JRJC)
      DO 12 JM=1,JRJCM1
   12 SUM = SUM-A(JR,JM)*A(JM,JRJC)
   14 A(JR,JRJC) = SUM
      IF (JRJC.EQ.ORDER) RETURN
      DO 18 JC = JRJCP1,ORDER
      SUM = A(JRJC,JC)
      DO 16 JM=1,JRJCM1
   16 SUM = SUM-A(JRJC,JM)*A(JM,JC)
   18 A(JRJC,JC) = SUM/A(JRJC,JRJC)
      GO TO 10
      END


C...
C...SUBROUTINE TO MULTIPLY A VECTOR B BY A MATRIX A
C...SUBTRACT RESULT FROM ANOTHER VECTOR C AND STORE
C...RESULT IN C.  THUS VECTOR C IS OVERWRITTEN.
C...
      SUBROUTINE MULPUT(A,B,C,ORDER)
      INTEGER ORDER
      DIMENSION A(1),B(1),C(1)

C...
      DO 200 JR=1,ORDER
      SUM = 0.0
      DO 100 JC=1,ORDER
      IA = JR+(JC-1)*ORDER
  100 SUM = SUM+A(IA)*B(JC)
  200 C(JR) = C(JR)-SUM
C...
      RETURN
      END


C...
C...SUBROUTINE TO SOLVE LINEAR ALGEBRAIC SYSTEM OF
C...EQUATIONS A*C=B AND STORE RESULTS IN VECTOR C.
C...MATRIX A IS INPUT IN L-U DECOMPOSITION FORM.
C...(NO PIVOTING STRATEGY HAS BEEN EMPLOYED TO
C...COMPUTE THE L-U DECOMPOSITION OF THE MATRIX A).
C...
      SUBROUTINE LUSOLV(A,B,C,ORDER)
      INTEGER ORDER
      DIMENSION A(ORDER,1),B(1),C(1)
```

```
C...
C...FIRST L(INV)*B
C...
      C(1) = C(1)/A(1,1)
      DO 14 JR=2,ORDER
      JRM1 = JR-1
      SUM = B(JR)
      DO 12 JM=1,JRM1
   12 SUM = SUM-A(JR,JM)*C(JM)
   14 C(JR) = SUM/A(JR,JR)
C...
C...NEXT U(INV) OF L(INV)*B
C...
      DO 18 JRJR=2,ORDER
      JR = ORDER-JRJR+1
      JRP1 = JR+1
      SUM = C(JR)
      DO 16 JMJM = JRP1,ORDER
      JM = ORDER-JMJM+JRP1
   16 SUM = SUM-A(JR,JM)*C(JM)
   18 C(JR) = SUM
C...
      RETURN
      END
```

APPENDIX

C

THE MODIFIED STRONGLY IMPLICIT PROCEDURE

This appendix describes the Modified Strongly Implicit (MSI) procedure (Schneider and Zedan, 1981) for solving a class of elliptic PDE's. The overall strategy of this procedure was described in Chapter 4. This appendix supplies further details. Schneider and Zedan (1981) presented the procedure as a means for solving the algebraic equations arising from the finite-difference representation of the elliptic equation

$$\frac{\partial}{\partial x}\left(k_x\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(k_y\frac{\partial u}{\partial y}\right)=q(x,y) \tag{C.1}$$

which governs two-dimensional steady-state heat conduction when u is the temperature. In the above, k_x and k_y are thermal conductivities for heat flow in the x and y directions, respectively, and $q(x,y)$ is a source term accounting for possible heat generation. It should be clear that a wide variety of problems are governed by equations of the form given by Eq. (C.1). With $k_x = k_y =$ constant and $q(x,y) \neq 0$, Eq. (C.1) becomes the Poisson equation. With $k_x = k_y =$ constant and $q(x,y) = 0$, Eq. (C.1) reduces to the Laplace equation. Only numerical examples for the solution to the Laplace equation were presented in Schneider and Zedan (1981). Examples presented employed Dirichlet, Neumann, and Robins (convective) boundary conditions.

The algorithm is developed to handle a nine-point finite-difference representation of Eq. (C.1) and treats the five-point representation as a special case. A nine-point [see Eq. (4.114)] representation of Eq. (C.1) can be written in the general form

$$A^1_{i,j}u_{i,j+1} + A^2_{i,j}u_{i+1,j+1} + A^3_{i,j}u_{i+1,j} + A^4_{i,j}u_{i+1,j-1} + A^5_{i,j}u_{i,j-1} + A^6_{i,j}u_{i-1,j-1}$$
$$+ A^7_{i,j}u_{i-1,j} + A^8_{i,j}u_{i-1,j+1} + A^9_{i,j}u_{i,j} = q_{i,j} \qquad \text{(C.2)}$$

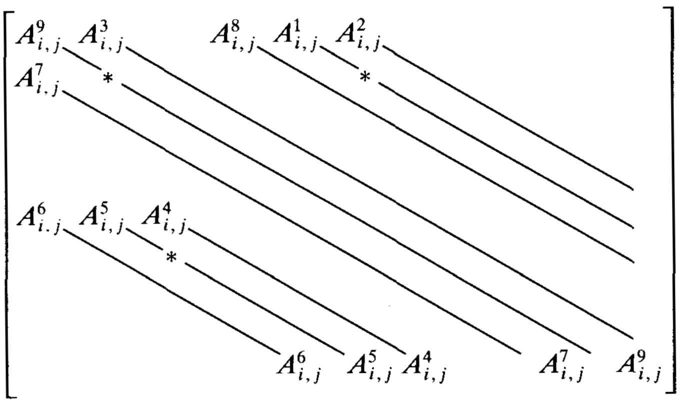
The i, j subscript refers to location within the grid network rather than the matrix row-column designation. Note that superscripts are used to identify the coefficients in the difference equation written for the general point (i, j). The five-point representation becomes a special case in which

$$A^2_{i,j} = A^4_{i,j} = A^6_{i,j} = A^8_{i,j} = 0$$

The equations can be written in the form

$$[A]\mathbf{u} = \mathbf{C} \qquad \text{(C.3)}$$

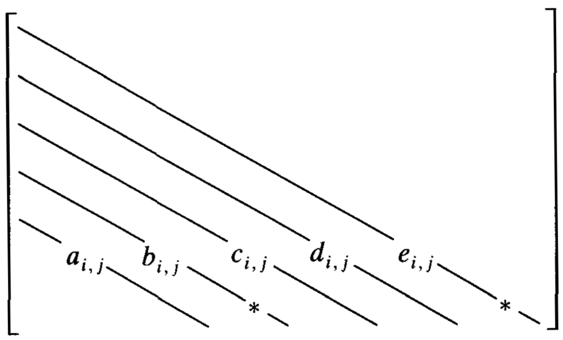
where the coefficient matrix has the form

$[A] =$ 

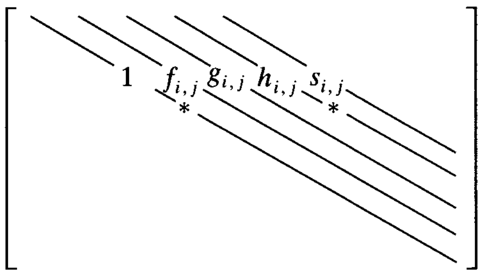
For reference, the diagonals corresponding to grid points having the same value for the i index (same grid column) and are identified by an asterisk. We now construct a matrix

$$[B] = [A + P]$$

such that $[B]$ can be decomposed into upper and lower triangular matrices, $[L]$ and $[U]$. We require that the original nine coefficients ($A^1_{i,j}$ through $A^9_{i,j}$) remain unchanged as $[A + P]$ is constructed. The $[L]$ and $[U]$ matrices have the form

$[L] =$ 

$[U] =$



Again the asterisk is used to identify diagonals corresponding to grid points having the same value for the i index.

The equations to be used to determine the coefficients of $[L]$ and $[U]$ such that the original nine coefficients in $[A]$ remain unchanged in $[B]$ are

$$a_{i,j} = A^6_{i,j} \tag{C.3a}$$

$$a_{i,j}f_{i-1,j-1} + b_{i,j} = A^5_{i,j} \tag{C.3b}$$

$$b_{i,j}f_{i,j-1} + C_{i,j} = A^4_{i,j} \tag{C.3c}$$

$$a_{i,j}h_{i-1,j-1} + b_{i,j}g_{i,j-1} + d_{i,j} = A^7_{i,j} \tag{C.3d}$$

$$a_{i,j}s_{i-1,j-1} + b_{i,j}h_{i,j-1} + c_{i,j}g_{i+1,j-1} + d_{i,j}f_{i-1,j} + e_{i,j} = A^9_{i,j} \tag{C.3e}$$

$$b_{i,j}s_{i,j-1} + c_{i,j}h_{i+1,j-1} + e_{i,j}f_{i,j} = A^3_{i,j} \tag{C.3f}$$

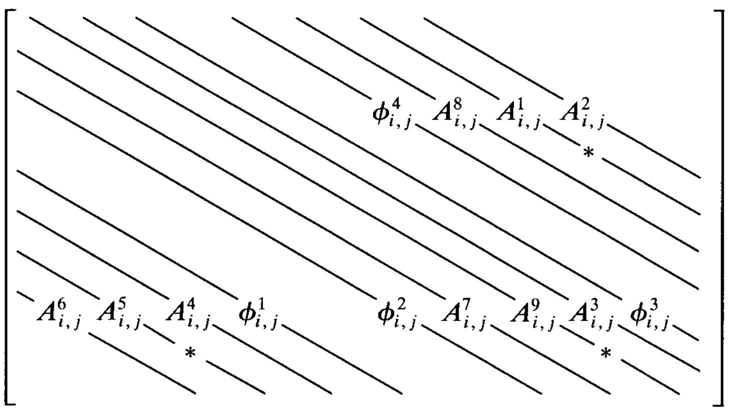
$$d_{i,j}h_{i-1,j} + e_{i,j}g_{i,j} = A^8_{i,j} \tag{C.3g}$$

$$d_{i,j}s_{i-1,j} + e_{i,j}h_{i,j} = A^1_{i,j} \tag{C.3h}$$

$$e_{i,j}s_{i,j} = A^2_{i,j} \tag{C.3i}$$

The modified coefficient matrix $[B] = [A + P]$ has the form

$[B] =$



where the asterisk has the same meaning as before.

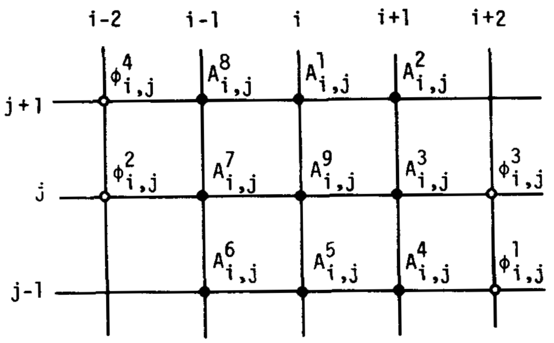


Figure C.1 The numerical molecule for the MSI procedure for a nine-point formulation, points labeled $A^2_{i,j}, A^4_{i,j}, A^6_{i,j}, A^8_{i,j}, \phi^2_{i,j}, \phi^3_{i,j}$ are eliminated when a five-point formulation is used.

The elements in $[B]$ denoted by $\phi^1_{i,j}$, $\phi^2_{i,j}$, $\phi^3_{i,j}$, and $\phi^4_{i,j}$ are determined from

$$\phi^1_{i,j} = c_{i,j} f_{i+1,j-1} \tag{C.4a}$$

$$\phi^2_{i,j} = a_{i,j} g_{i-1,j-1} \tag{C.4b}$$

$$\phi^3_{i,j} = c_{i,j} s_{i+1,j-1} \tag{C.4c}$$

$$\phi^4_{i,j} = d_{i,j} g_{i-1,j} \tag{C.4d}$$

The numerical molecule associated with the modified matrix $[B]$ is shown schematically in Fig. C.1.

Schneider and Zedan (1981) employed Taylor-series expansions to obtain values of $u_{i-2,j}$, $u_{i+2,j}$, $u_{i+2,j-1}$, and $u_{i-2,j+1}$ in terms of u's in the original nine-point molecule to partially cancel the influence of the additional $(\phi_{i,j})$ terms in the $[B]$ matrix. These are

$$u_{i-2,j} = -u_{i,j} + 2u_{i-1,j} \tag{C.5a}$$

$$u_{i+2,j} = -u_{i,j} + 2u_{i+1,j} \tag{C.5b}$$

$$u_{i+2,j-1} = -2u_{i,j} + 2u_{i+1,j} + u_{i,j-1} \tag{C.5c}$$

$$u_{i-2,j+1} = -2u_{i,j} + 2u_{i-1,j} + u_{i,j+1} \tag{C.5d}$$

Other "extrapolation" schemes for obtaining values outside the original molecule may work equally well. The use of such approximations affects only the approach to convergence of the iterative sequence and not the final converged solution.

An iterative parameter α is employed to implement partial cancellation of the influence of the $\phi_{i,j}$ terms appearing in $[B]$. This is done by using a modified representation for the nine-point scheme in the form

$$A^5_{i,j}u_{i,j-1} + A^7_{i,j}u_{i-1,j} + A^9_{i,j}u_{i,j} + A^1_{i,j}u_{i,j+1} + A^3_{i,j}u_{i+1,j}$$
$$+ A^6_{i,j}u_{i-1,j-1} + A^8_{i,j}u_{i-1,j+1} + A^2_{i,j}u_{i+1,j+1}$$
$$+ A^4_{i,j}u_{i+1,j-1} + \phi^1_{i,j}\left[u_{i+2,j-1} - \alpha(-2u_{i,j} + 2u_{i+1,j} + u_{i,j-1})\right]$$

$$+ \phi^2_{i,j}\left[u_{i-2,j} - \alpha(-u_{i,j} + 2u_{i-1,j})\right] + \phi^3_{i,j}\left[u_{i+2,j} - \alpha(-u_{i,j} + 2u_{i+1,j})\right]$$
$$+ \phi^4_{i,j}\left[u_{i-2,j+1} - \alpha(-2u_{i,j} + 2u_{i-1,j} + u_{i,j+1})\right] = q_{i,j} \qquad \text{(C.6)}$$

Equations (C.3) and (C.4) are modified to include the partial cancellation indicated in Eq. (C.6) and rearranged to permit the explicit evaluation of the elements of $[L]$ and $[U]$:

$$a_{i,j} = A^6_{i,j} \qquad \text{(C.7}a\text{)}$$

$$b_{i,j} = \frac{A^5_{i,j} - a_{i,j}f_{i-1,j-1} - \alpha A^4_{i,j}f_{i+1,j-1}}{1 - \alpha f_{i,j-1}f_{i+1,j-1}} \qquad \text{(C.7}b\text{)}$$

$$c_{i,j} = A^4_{i,j} - b_{i,j}f_{i,j-1} \qquad \text{(C.7}c\text{)}$$

$$d_{i,j} = \frac{A^7_{i,j} - a_{i,j}h_{i-1,j-1} - b_{i,j}g_{i,j-1} - 2\alpha a_{i,j}g_{i-1,j-1}}{1 + 2\alpha g_{i-1,j}} \qquad \text{(C.7}d\text{)}$$

$$e_{i,j} = A^9_{i,j} - a_{i,j}s_{i-1,j-1} - b_{i,j}h_{i,j-1} - c_{i,j}g_{i+1,j-1} - d_{i,j}f_{i-1,j}$$
$$+ \alpha\left(2\phi^1_{i,j} + \phi^2_{i,j} + \phi^3_{i,j} + 2\phi^4_{i,j}\right) \qquad \text{(C.7}e\text{)}$$

$$f_{i,j} = \frac{A^3_{i,j} - b_{i,j}s_{i,j-1} - c_{i,j}h_{i+1,j-1} - 2\alpha\left(\phi^1_{i,j} + \phi^3_{i,j}\right)}{e_{i,j}} \qquad \text{(C.7}f\text{)}$$

$$g_{i,j} = \frac{A^8_{i,j} - d_{i,j}h_{i-1,j}}{e_{i,j}} \qquad \text{(C.7}g\text{)}$$

$$h_{i,j} = \frac{A^1_{i,j} - d_{i,j}s_{i-1,j} - \alpha\phi^4_{i,j}}{e_{i,j}} \qquad \text{(C.7}h\text{)}$$

$$s_{i,j} = \frac{A^2_{i,j}}{e_{i,j}} \qquad \text{(C.7}i\text{)}$$

The $\phi_{i,j}$'s appearing in the above are evaluated as indicated in Eqs. (C.4) using the values of a, b, c, d, f, g, and s obtained from Eqs. (C.7). Note that the $\phi_{i,j}$'s are needed in Eqs. (C.7) and should be evaluated as soon as the evaluation of $d_{i,j}$ is complete. The results obtained by Schneider and Zedan (1981) indicate that the MSI procedure is not extremely sensitive to the choice of α. Values of α between 0.3 and 0.6 worked well in their calculations.

It is important to observe that when the MSI procedure is used for the five-point difference representation,

$$A^2_{i,j} = A^4_{i,j} = A^6_{i,j} = A^8_{i,j} = 0 \qquad \text{(C.8)}$$

and, as a result,

$$a_{i,j} = s_{i,j} = \phi^2_{i,j} = \phi^3_{i,j} = 0 \qquad \text{(C.9)}$$

The iterative sequence is developed as follows. Adding $[P]\mathbf{u}$ to both sides of Eq. (C.3) gives

$$[A + P]\mathbf{u} = \mathbf{C} + [P]\mathbf{u} \qquad \text{(C.10)}$$

We evaluate the unknowns on the right-hand side at the n iteration level to write

$$[A + P]\mathbf{u}^{n+1} = \mathbf{C} + [P]\mathbf{u}^n \tag{C.11}$$

Decomposing $[A + P]$ into the $[L]$ and $[U]$ matrices gives

$$[L][U]\mathbf{u}^{n+1} = \mathbf{C} + [P]\mathbf{u}^n \tag{C.12}$$

Defining an intermediate vector $\mathbf{V}^{n+1}$ by

$$\mathbf{V}^{n+1} = [U]\mathbf{u}^{n+1} \tag{C.13}$$

we can employ the two-step process

Step 1: $$[L]\mathbf{V}^{n+1} = \mathbf{C} + [P]\mathbf{u}^n \tag{C.14a}$$

Step 2: $$[U]\mathbf{u}^{n+1} = \mathbf{V}^{n+1} \tag{C.14b}$$

The elements of $[P]$ are simply the $\phi^1, \phi^2, \phi^3, \phi^4$ (only ϕ^1 and ϕ^4 when the five-point scheme is used) values determined from Eqs. (C.4).

Alternatively, we can define a difference vector

$$\boldsymbol{\delta}^{n+1} = \mathbf{u}^{n+1} - \mathbf{u}^n \tag{C.15}$$

and a residual vector

$$\mathbf{R}^n = [A]\mathbf{u}^n - \mathbf{C} \tag{C.16}$$

so that Eq. (C.11) becomes

$$[A + P]\boldsymbol{\delta}^{n+1} = -\mathbf{R}^n \tag{C.17}$$

Replacing $[A + P]$ by the $[L][U]$ product gives

$$[L][U]\boldsymbol{\delta}^{n+1} = -\mathbf{R}^n$$

Defining an intermediate vector $\mathbf{W}^{n+1}$ by

$$\mathbf{W}^{n+1} = [U]\boldsymbol{\delta}^{n+1} \tag{C.18}$$

the solution procedure can again be written as a two-step process:

Step 1: $$[L]\mathbf{W}^{n+1} = -\mathbf{R}^n \tag{C.19a}$$

Step 2: $$[U]\boldsymbol{\delta}^{n+1} = \mathbf{W}^{n+1} \tag{C.19b}$$

The processes represented by Eqs. (C.14) and (C.19) consist of a forward substitution to determine $\mathbf{V}^{n+1}$ or $\mathbf{W}^{n+1}$ followed by a backward substitution to obtain $\mathbf{u}^{n+1}$ or $\boldsymbol{\delta}^{n+1}$. The coefficients remain unchanged for the iterative process. The right-hand side of the Step 1 equation is then updated and the procedure is repeated.

APPENDIX

D
FINITE-VOLUME DISCRETIZATION FOR GENERAL CONTROL VOLUMES

The finite-volume method enforces conservation principles in integral form to fixed regions in space known as control volumes. The purpose of this appendix is to illustrate how this can be carried out for general control volumes for which the boundaries may not necessarily intersect in an orthogonal manner. The main points can be readily illustrated by considering two examples in two dimensions, that of mass conservation in steady, incompressible flow and thermal energy conservation.

The approach is largely the same regardless of the shape of the control volume. The grid may be structured or unstructured. We will utilize the generally nonorthogonal but structured grid illustrated in Fig. D.1. in which control volumes are quadrilaterals. The boundaries of control volumes are placed approximately halfway between grid points. More precisely, the coordinates of the corners of the control volumes (points a, b, c, d) are taken as the average of the coordinates of the four surrounding grid points. That is,

$$x_a = (x_{i,j-1} + x_{i+1,j-1} + x_{i+1,j} + x_{i,j})/4$$

$$y_a = (y_{i,j-1} + y_{i+1,j-1} + y_{i+1,j} + y_{i,j})/4$$

The coordinates of points b, c, d are located in a similar manner. The corners are connected by straight lines to form the control volume. This illustrates a cell vertex or nodal point scheme, the procedure employed in Chapter 3 (Section 3.4.4). In passing we note that other choices for the establishment of control volumes could have been made. For example, we could have considered the

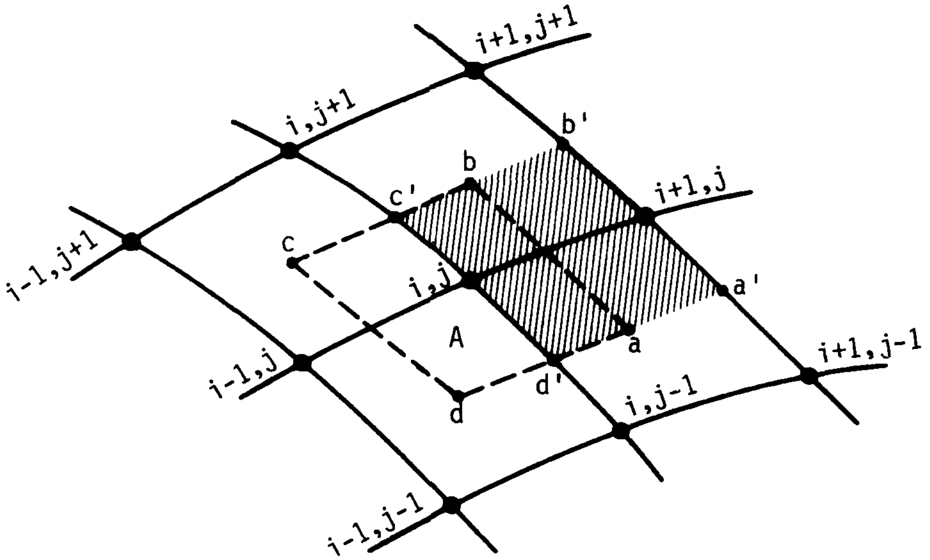


Figure D.1 Control volume for finite-volume discretization; shaded area is secondary volume used in representation of boundary derivatives.

intersecting solid lines in Fig. D.1 to be the control volume boundaries and placed "grid" points (where variables will be evaluated) in the center of the volumes. This would have been a cell-centered scheme, the scheme utilized in Chapter 5 (Section 5.7).

For a control volume, conservation of mass in steady, incompressible, two-dimensional flow requires

$$\oiint_S \mathbf{V} \cdot \mathbf{n}\, dS = 0 \tag{D.1}$$

where $\mathbf{n}$ is a unit vector normal to the control volume (positive when pointing outward) and $\mathbf{V}$ is the velocity vector, $\mathbf{V} = u\mathbf{i} + v\mathbf{j}$. The integral represents the net volume rate of flow out of the volume S. For two-dimensional flow, the "volume" is formed by including a unit depth normal to the x-y plane. This unit depth will be omitted in the equations that follow. Equation (D.1) serves as a model for conservation statements that require evaluation of surface fluxes that can be represented in terms of simple functions of the dependent variables themselves.

In two dimensions, we can represent $\mathbf{n}dS$ as $\mathbf{i}dy - \mathbf{j}dx$ (omitting the unit depth) for an integration path around the boundary in a *counter-clockwise* direction. Thus, Eq. (D.1) can be written as

$$\oint\!\!\!\oint_S (u\,dy - v\,dx) = 0 \tag{D.2}$$

To evaluate this integral for the control volume A in Fig. D.1, we need to represent the velocity components on the boundaries of the control volume. One choice is to use the average of the velocity components at nodes on either side of the boundary. Thus, for boundary a-b we could use

$$u_{ab} = u_{i+1/2,j} \cong (u_{i+1,j} + u_{i,j})/2 \qquad v_{ab} = v_{i+1/2,j} \cong (v_{i+1,j} + v_{i,j})/2$$

Thus,

$$\begin{aligned}\oint\!\!\!\oint_S &(u\,dy - v\,dx)\\ &\cong u_{i+1/2,j}\Delta y_{ab} - v_{i+1/2,j}\Delta x_{ab} + u_{i,j+1/2}\Delta y_{bc} - v_{i,j+1/2}\Delta x_{bc}\\ &\quad + u_{i-1/2,j}\Delta y_{cd} - v_{i-1/2,j}\Delta x_{cd} + u_{i,j-1/2}\Delta y_{da} - v_{i,j-1/2}\Delta x_{da}\end{aligned} \tag{D.3}$$

where the increments in the coordinates indicated above must be evaluated very carefully as $\Delta x_{ab} = x_b - x_a$, $\Delta y_{ab} = y_b - y_a$, for example. Some of the increments will be positive and some will be negative as the integration proceeds around the control volume. Note that if the control volume is a rectangle with sides aligned with the Cartesian coordinate system, one of the coordinate increments will be zero along each boundary and some of the velocity components will cancel resulting in the central representation

$$\oint\!\!\!\oint_S (u\,dy - v\,dx) \cong (u_{i+1,j} - u_{i-1,j})\Delta y_{ab} + (v_{i,j+1} - v_{i,j-1})\Delta x_{da} \tag{D.4}$$

Many conservation statements required in applications govern unsteady phenomena and others contain fluxes that depend on derivatives. The integral form of the 2-D heat equation will be used as a model to illustrate how such terms can be handled for a control volume of arbitrary shape. The differential form of the 2-D heat equation is given by Eq. (3.92a)

$$\rho c \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right) = \nabla \cdot (k\nabla T)$$

The corresponding integral form is

$$\int\!\!\int\!\!\int_R \rho c \frac{\partial T}{\partial t} dR + \oint\!\!\!\oint_S \mathbf{q}\cdot\mathbf{n}\,dS = 0 \tag{D.5}$$

where $\mathbf{q} = -k\dfrac{\partial T}{\partial x}\mathbf{i} - k\dfrac{\partial T}{\partial y}\mathbf{j}$. Following the strategy employed with the surface integral that appeared in the previous example, we can write Eq. (D.5) as

$$\int\!\!\int\!\!\int_R \rho c \frac{\partial T}{\partial t} dR + \oint\!\!\!\oint_S \left(-k\frac{\partial T}{\partial x}\,dy + k\frac{\partial T}{\partial y}\,dx\right) = 0 \tag{D.6}$$

The term on the left containing the time derivative can be evaluated by assuming that the temperature at point (i, j) is the mean value for the volume and then using a forward-time difference to obtain

$$\rho c \frac{\left(T_{i,j}^{n+1} - T_{i,j}^{n}\right)}{\Delta t} R_{abcd} \tag{D.7}$$

where R_{abcd} is the volume of the control volume. In this 2-D example, we will take the volume to be the area of the 2-D control volume in the x-y plane, A_{abcd}, times a unit depth normal to that plane. As before, the unit depth will be omitted in the equations to follow. We will determine the area of the quadrilateral region A_{abcd} as one-half the magnitude of the cross products of its diagonals,

$$A_{abcd} = 0.5|(\Delta x_{db} \Delta y_{ac} - \Delta y_{db} \Delta x_{ac})|$$

Note that for a rectangular region with boundaries aligned with the Cartesian coordinate system, this results in $A_{abcd} = \Delta x \Delta y$.

To evaluate the surface integral in Eq. (D.6) we will first discretize by approximating the integrand on the boundaries. When doing this, a decision must be made as to the time level at which the boundary fluxes are to be evaluated. This determines whether the scheme will be explicit or implicit. Reasonable choices include time levels n, $n + 1$ or an average of the two. Selecting time level n,

$$\begin{aligned}
\oint\!\!\!\oint_S \left(-k \frac{\partial T}{\partial x} dy + k \frac{\partial T}{\partial y} dx \right) &\cong -k \frac{\partial T}{\partial x}\bigg)_{i+1/2,j}^{n} \Delta y_{ab} \\
&+ k \frac{\partial T}{\partial y}\bigg)_{i+1/2,j}^{n} \Delta x_{ab} - k \frac{\partial T}{\partial x}\bigg)_{i,j+1/2}^{n} \Delta y_{bc} \\
&+ k \frac{\partial T}{\partial y}\bigg)_{i,j+1/2}^{n} \Delta x_{bc} - k \frac{\partial T}{\partial x}\bigg)_{i-1/2,j}^{n} \Delta y_{cd} \\
&+ k \frac{\partial T}{\partial y}\bigg)_{i-1/2,j}^{n} \Delta x_{cd} - k \frac{\partial T}{\partial x}\bigg)_{i,j-1/2}^{n} \Delta y_{da} + k \frac{\partial T}{\partial y}\bigg)_{i,j-1/2}^{n} \Delta x_{da}
\end{aligned} \tag{D.8}$$

For rectangular volumes whose boundaries align with the Cartesian coordinate system, the derivatives at boundaries are readily represented by central differences utilizing neighboring nodal values of temperature. This is demonstrated in Chapter 3 (Section 3.4.4). However, it is possible to develop appropriate representations for derivatives at control volume boundaries by integral methods in a manner that is not restricted to Cartesian or even orthogonal grids. The method makes use of results that can be obtained from application of the divergence theorem,

$$\iiint_R \nabla \cdot \mathbf{V}\, dS = \oiint_S \mathbf{V} \cdot \mathbf{n}\, dS$$

Letting $\mathbf{V} = T\mathbf{i}$ and applying the divergence theorem gives

$$\iiint_R \frac{\partial T}{\partial x}\, dR = \oiint_S T dy \tag{D.9}$$

where again we are making use of the fact that $\mathbf{n}\, dS$ can be represented by $\mathbf{i}dy - \mathbf{j}dx$. In a like manner, letting $\mathbf{V} = T\mathbf{j}$, we observe

$$\iiint_R \frac{\partial T}{\partial y}\, dR = -\oiint_S T dx \tag{D.10}$$

Further, by multiplying the first expression by $\mathbf{i}$ and the second by $\mathbf{j}$ and adding, we can see that

$$\iiint_R \nabla T dR = \oiint_S T\mathbf{n}\, dS \tag{D.11}$$

These results provide a way in which derivatives of temperature on the control volume boundary can be represented by integrating the temperature itself around the boundaries of a suitable volume. Suppose, for example, that we wish to represent $\left.\frac{\partial T}{\partial x}\right)^n_{i+1/2,j}$ and $\left.\frac{\partial T}{\partial y}\right)^n_{i+1/2,j}$ for an interior control volume such as volume A in Fig. D.1. We first establish a secondary volume (area in this 2-D example) in such a manner that the point at which the representation is desired, $i + \frac{1}{2}, j$ in this case, is approximately in the center. The shaded area in Fig. D.1 will serve that purpose. This procedure assumes that $\left.\frac{\partial T}{\partial x}\right)^n_{i+1/2,j}$ and $\left.\frac{\partial T}{\partial y}\right)^n_{i+1/2,j}$ are mean values for the secondary area $A_{a'b'c'd'}$. Thus, it follows from the results above that

$$\left.\frac{\partial T}{\partial x}\right)^n_{i+1/2,j} \cong \frac{1}{A_{a'b'c'd'}} \oiint_S T dy \quad \text{and} \quad \left.\frac{\partial T}{\partial y}\right)^n_{i+1/2,j} \cong -\frac{1}{A_{a'b'c'd'}} \oiint_S T dx$$

The coordinates of the secondary volume are established by employing suitable averages of the coordinates of neighboring points. The determination of the coordinates of points a, b, c, d has already been discussed. The coordinates are found to be averages of the coordinates of the four neighboring nodal points. The location of a' can be determined by averaging the coordinates of points $(i + 1, j)$ and $(i + 1, j - 1)$. The coordinates of point b' can be determined by averaging the coordinates of points $(i + 1, j)$ and $(i + 1, j + 1)$. The coordinates of points c' and d' are determined as averages in a similar manner. We next approximate the line integrals as, for example,

$$\left.\frac{\partial T}{\partial x}\right)_{i+1/2,j}^{n} \cong \frac{1}{A_{a'b'c'd'}} \oint_S T\,dy \cong T_{i+1,j}^{n}\Delta y_{a'b'} + T_b^n \Delta y_{b'c'} + T_{i,j}^{n}\Delta y_{c'd'} + T_a^n \Delta y_{d'a'} \tag{D.12}$$

where T_a^n and T_b^n are determined as averages of the four neighboring nodal temperatures. Notice that for a rectangular volume with boundaries that align with the Cartesian coordinate system, Eq. (D.12) reduces to

$$\frac{T_{i+1,j}^{n} - T_{i,j}^{n}}{\Delta x}$$

To complete the discretization of Eq. (D.8), seven more equations similar to Eq. (D.12) must be developed. This requires that a different secondary area be established for approximating derivatives on each of the four sides of the original control volume labeled A in Fig. D.1. This is computationally intensive, but the result is general and not restricted to orthogonal grids.