6 - ADI Method, a Fast Implicit Method for 3D USS HC Problems

All of the fast methods presented in the previous chapter allowed solution to one dimensional unsteady state problem. In this chapter a fast method will be presented to allow for multidimensional fast solutions. The method presented here is called the Alternating Direction Implicit method (ADI) and is based on the Crank-Nicolson Method of solving one-dimensional problems.

The beauty of the Crank-Nicolson Method is that it results in a tridiagonal matrix that is efficiently solved using the tridiagonal matrix algorithm. If the Crank-Nicolson approach for a one-dimensional problem (the averaging of forward and current position derivatives) is extended to a two or three dimensional problem, a real mess results. Rather than an equation with three unknown (i.e. new) temperatures at a grid point, each grid point's corresponding equation has five unknown temperatures for a two-dimensional problem and seven for a three-dimensional problem and neither forms a tridiagonal set of equations. Therefore, to make use of the Crank-Nicolson Method (i.e. tridiagonal matrix algorithm), one must take a modified approach. That approach is to alternately apply the method in one direction at a time. This is called the Alternating Implicit Direction method.

Figure 1 shows a computation flow diagram for an ADI Macro that will run as a VBA in Microsoft Excel®.

6.1 - ADI Method, a Fast Implicit Method for 3D USS HT

The Alternating Direction Implicit (ADI) Method of solving PDQ's is based on the Crank-Nicolson Method of solving one-dimensional problems. The Crank-Nicolson Method creates a coincidence of the position and the time derivatives by averaging the position derivative for the old and the new temperatures while using the forward derivative for the time derivative as shown in Figure. 1.

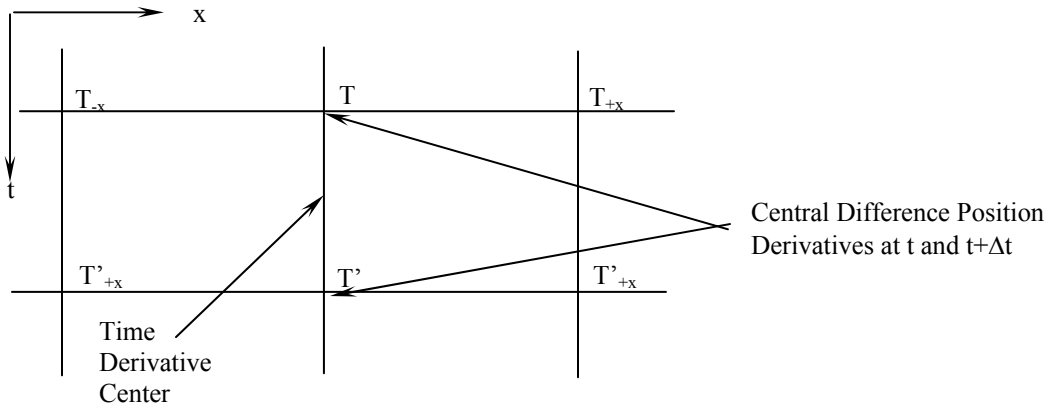

Figure.1 Locations of the Time and Position Derivatives for Crank-Nicolson Method for solving a 1D USS HT problem

The Heat Equation

$$\alpha\left[\frac{\partial^2 T}{\partial x^2}\right]=\frac{\partial T}{\partial t} \tag{6.1}$$

then becomes

$$\alpha\left[\frac{\dfrac{\left[T_{+x}-2T+T_{-x}\right]}{\Delta x^2}+\dfrac{\left[T'_{+x}-2T'+T'_{-x}\right]}{\Delta x^2}}{2}\right]=\frac{T'-T}{\Delta t} \tag{6.2}$$

The complete enumeration of Eq. (6.2) for nine unknown temperatures in the x-direction leads to the familiar Eqs.(6.3)-(6.11). This set of equations forms a tridiagonal coefficient matrix that is solved easily using the Tridiagonal Matrix Algorithm.

1. $$b'T'_1+c'T'_2=fT_0+(1-2f)T_1+fT_2-a'T'_0=d_1 \tag{6.3}$$
2. $$a'T'_1+b'T'_2+c'T'_3=fT_1+(1-2f)T_2+fT_3=d_2 \tag{6.4}$$
3. $$a'T'_2+b'T'_3+c'T'_4=fT_2+(1-2f)T_3+fT_4=d_3 \tag{6.5}$$
4. $$a'T'_3+b'T'_4+c'T'_5=fT_3+(1-2f)T_4+fT_5=d_4 \tag{6.6}$$
5. $$a'T'_4+b'T'_5+c'T'_6=fT_4+(1-2f)T_5+fT_6=d_5 \tag{6.7}$$
6. $$a'T'_5+b'T'_6+c'T'_7=fT_5+(1-2f)T_6+fT_7=d_6 \tag{6.8}$$
7. $$a'T'_6+b'T'_7+c'T'_8=fT_6+(1-2f)T_7+fT_8=d_7 \tag{6.9}$$
8. $$a'T'_7+b'T'_8+c'T=fT_7+(1-2f)T_8+fT_9=d_8 \tag{6.10}$$
9. $$a'T'_8+b'T'_9=fT_8+(1-2f)T_9+fT_{10}-c'T'_{10}=d_9 \tag{6.11}$$

where

$$f=\frac{\alpha\Delta t}{2\Delta x^2} \tag{6.12}$$

6.2 - Extending the Crank-Nicolson Method to Three Dimensions: Not ADI

At an interior point (x, y, z) the Heat Equation for three dimensions is

$$\alpha\left[\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}+\frac{\partial^2 T}{\partial z^2}\right]=\frac{\partial T}{\partial t}\ . \tag{6.13}$$

Were the Crank-Nicolson Method extended to all three dimensions, there would be a total of seven unknown temperatures in each equation describing a grid point in the interior of the solid: three as before for an interior point in the x-direction T'_{-x}, T', T'_{+x} and two new temperatures in both the y direction (T'_{-y}, T'_{+y}) and the z direction (T'_{-z}, T'_{+z}) as shown in Eq. [14].

$$\frac{\alpha}{2\Delta x^2}\left\{\begin{array}{c}\left[\left(T_{-x}-2T+T_{+x}\right)+\left(T'_{-x}-2T'+T'_{+x}\right)\right]\\ +\left[\left(T_{-y}-2T+T_{+y}\right)+\left(T'_{-y}-2T'+T'_{+y}\right)\right]\\ +\left[\left(T_{-z}-2T+T_{+z}\right)+\left(T'_{-z}-2T'+T'_{+z}\right)\right]\end{array}\right\}=\frac{T'-T}{\Delta t} \tag{6.14}$$

Collecting the terms of Eq. (6.14) into unknown temperatures on the LHS and the known temperatures on the RHS gives Eq.(6.15).

$$\begin{bmatrix} fT'_{x-1,y,z}-2fT'_{x,y,z}+fT'_{x+1,y,z}\\ fT'_{x,y-1,z}-2fT'_{x,y,z}+fT'_{x,y+1,z}\\ fT'_{x,y,z-1}-2fT'_{x,y,z}+fT'_{x,y,z+1}\end{bmatrix}-T'_{x,y,z}$$

$$=\begin{bmatrix} -fT_{x-1,y,z}+2fT_{x,y,z}-fT_{x+1,y,z}\\ -fT_{x,y-1,z}+2fT_{x,y,z}-fT_{x,y+1,z}\\ -fT_{x,y,z-1}+2fT_{x,y,z}-fT_{x,y,z+1}\end{bmatrix}-T_{x,y,z}=d_{x,y,z} \tag{6.15}$$

or more succinctly,

$$\begin{bmatrix} fT'_{x-1,y,z}+fT'_{x+1,y,z}\\ fT'_{x,y-1,z}+fT'_{x,y+1,z}\\ fT'_{x,y,z-1}+fT'_{x,y,z+1}\end{bmatrix}-(6f-1)T'_{x,y,z}$$

$$=\begin{bmatrix} -fT_{x-1,y,z}-fT_{x+1,y,z}\\ -fT_{x,y-1,z}-fT_{x,y+1,z}\\ -fT_{x,y,z-1}-fT_{x,y,z+1}\end{bmatrix}+(6f-1)T_{x,y,z}=d \tag{6.16}$$

For a 3D USS HT problem involving a cubic solid divided into 10 increments in each direction the 0^{th} and 10^{th} locations would be boundaries leaving 9x9x9 = 729 unknown temperatures and 729 such equations. Of course, as the point of interest moves next to a boundary, some of the unknown temperatures on the LHS are known and move to the RHS as in Eqs. (6.3) and (6.11) for the 1D USS HT case. The matrix for simultaneous solution is a 729x729 with many rows containing 7 coefficients for interior grid points, some rows with 6 coefficients for points along the boundary faces, fewer rows with 5 coefficients along the boundary edges, and eight rows with 4 coefficients at the boundary corners. Clearly, this is a huge numerical solution problem. If one calculates the sparse nature of the coefficient matrix as shown in Table 1, less than one percent of the matrix contains a value. This is called a sparse matrix and there are special methods for solving them but there are also pitfalls to solving such a matrix. This entire complexity is completely eliminated by the ADI Method described next.

Table 1. Sparseness of a Crank-Nicolson-Type Solution to a 10^3 Increment 3D USS HC Problem

Description	Number of Forms of this Type	Unknown Temperatures	Number of Points in Each Form	Total Coefficients in Each Form
Interior	1	7	7*7*7	2401
Faces	6	6	7*7	1764
Edges	12	5	7	420
Corners	8	4	1	32
Total				4617
Total Matrix Elements			729*729	531441
Matrix Sparseness				0.87%

6.3 - ADI: Extending the Crank-Nicolson Idea to Three Dimensions

The ADI Method simply applies the Crank-Nicolson Method in one direction at a time. That is, for any of the many line of points in the x-direction the Heat Equation is

$$\frac{\alpha}{\Delta x^2}\left[\frac{\left(T_{-x}-2T+T_{+x}\right)+\left(T'_{-x}-2T'+T'_{+x}\right)}{2}+\left(T_{-y}-2T+T_{+y}\right)+\left(T_{-z}-2T+T_{+z}\right)\right]=\frac{T'-T}{\Delta t} \qquad (6.17)$$

For any one line in the x direction, the set of equations are the same as Eqs. (6.3)– (6.11) except that the old temperatures in the y and z directions become part of the RHS. Therefore, new temperatures along the line of consideration may be determined by Crank-Nicolson as for any 1D USS HT problem. Once all the temperatures along all the lines in the x direction are updated in a new temperature array, the process is repeated in the y direction followed by the z direction according to Eqs. (6.18) and (6.19). This completes one computational cycle by the ADI method. Figure 1 shows a schematic diagram of the computational organization. Appendix 6A is a copy of the ADI VBA code corresponding to Figure 2 in 2 directions: x and y. The code runs as a Visual Basic Application (VBA) in Microsoft Excel®.

$$\frac{\alpha}{\Delta x^2}\left[\left(T_{-x}-2T+T_{+x}\right)+\frac{\left(T_{-y}-2T+T_{+y}\right)+\left(T'_{-y}-2T'+T'_{+y}\right)}{2}+\left(T_{-z}-2T+T_{+z}\right)\right]=\frac{T'-T}{\Delta t} \tag{6.18}$$

$$\frac{\alpha}{\Delta x^2}\left[\left(T_{-x}-2T+T_{+x}\right)+\left(T_{-y}-2T+T_{+y}\right)+\frac{\left(T_{-z}-2T+T_{+z}\right)+\left(T'_{-z}-2T'+T'_{+z}\right)}{2}\right]=\frac{T'-T}{\Delta t} \tag{6.19}$$

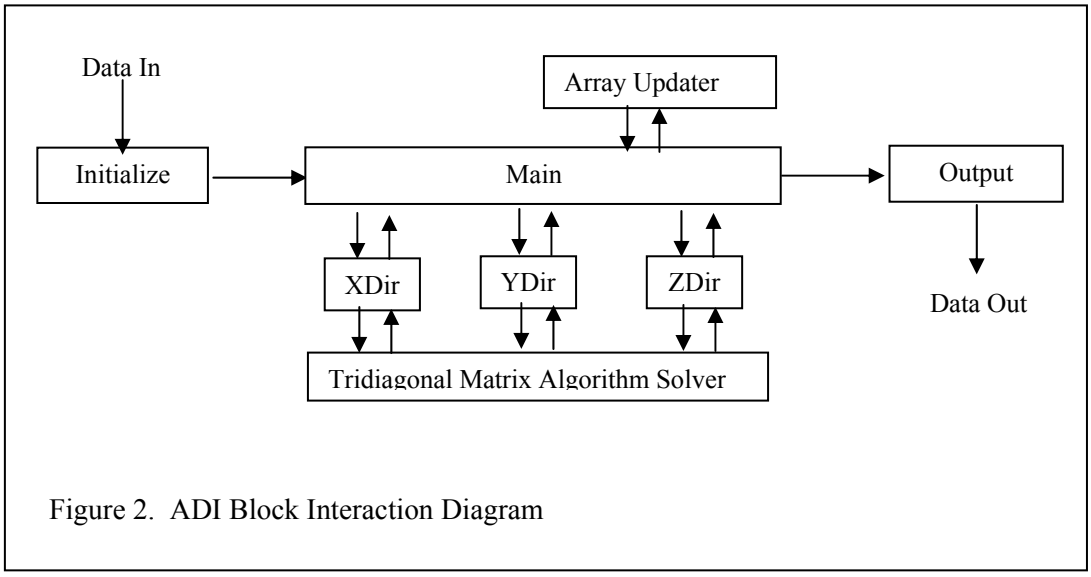

Figure 2. ADI Block Interaction Diagram

6.4 – Other PDQ Methods

In the world of PDS solving, all of the methods described in this volume are considered elementary, but provide a valuable foundation for learning more advanced methods or, perhaps more likely, a means of quickly and inexpensively approximating a solution to a partial differential equation. The chief concerns leading to more advanced methods include: minimizing error while increasing step size and better handling of curved boundaries so that jagged increments are not needed. In addition to the *Finite Difference Methods* that have been the exclusive consideration in this volume, a second approach called the *Finite Element Method* is commonly employed. Commercial code that solves PDQs can cost many tens of thousands of dollars. Having an introductory knowledge of how such problems are solved, is a tremendous help in setting up and running such programs. Most of the questions that must be answered involved the following:

- geometry of the system (Auto grid builders will place a mesh of grid points),
- nature of the boundary conditions for the system's bounding surfaces,
- physical properties of the material including viscous or thermal generation terms, and
- initial conditions
- time span.

Example 6a - Model for the Welding of Two Flat Plates.

Consider two flat plates to be welded. The following assumptions are made:

- The solution is considered symmetrical about the weld line.
- Heat transfer across the two plates along the weld line is considered to occur as though the plates were continuous whether they are yet welded, or not.
- The outside edges parallel to the weld are considered to be at such distance that they may be considered to be at constant temperature.
- There are no variations in the thickness direction.
- No allowance is made for the heat of fusion

Solution

Figure 3 shows a sketch for deriving the governing differential equations. A heat balance for the non-weld-line element shown in Figure 1 gives the following:

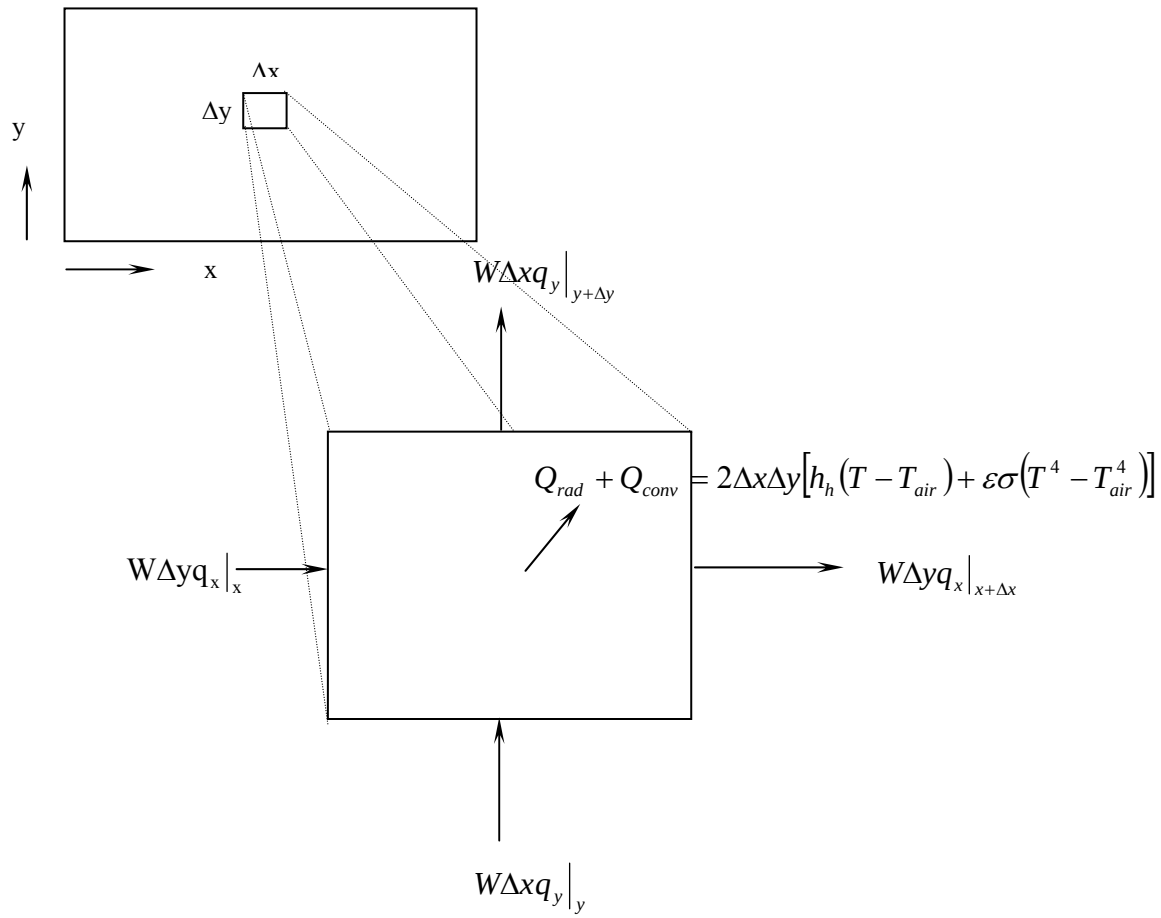

Figure 3. Sketch of the Heat Balance for an Interior Element not to be Welded

Rate In - Rate Out + Gen Rate = Accumulation Rate (6.20)

expressed as

$$W\Delta y\left(q_x\big|_x - q_x\big|_{x+\Delta x}\right) + W\Delta x\left(q_y\big|_y - q_y\big|_{y+\Delta y}\right) + 2\Delta x\Delta y\left[h_h\left(T - T_{air}\right) + \varepsilon\sigma\left(T^4 - T_{air}^4\right)\right] = W\Delta x\Delta y\rho Cp\frac{\partial T}{\partial t} \tag{6.21}$$

Dividing by WΔxΔy and taking the limit as the Δx and Δy approach zero and substituting Fourier's Law of Heat Conduction gives

$$\alpha\left[\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right] - \frac{2}{W\rho Cp}\left[h_h\left(T - T_{air}\right) + \varepsilon\sigma\left(T^4 - T_{air}^4\right)\right] = \frac{\partial T}{\partial t} \tag{6.22}$$

where $\alpha = k/\rho Cp$.

Approximating the second partials with central differences and the time partial with a forward difference and solving for the new temperature T' gives

$$T' = T + \frac{\alpha\Delta t}{\Delta x^2}\left(T_{-x} + T_{+x} + T_{-y} + T_{+y} - 4T\right) - \frac{2\Delta t}{W\rho Cp}\left[h_h\left(T - T_{air}\right) + \varepsilon\sigma\left(T^4 - T_{air}^4\right)\right] \tag{6.23}$$

The same equation may be derived for an interior element being welded. It will be the same as Eq. (6.23) (3) except for an additional generation term that accounts for the welding power, Wp, applied to the element.

$$T' = T + \frac{\alpha\Delta t}{\Delta x^2}\left(T_{-x} + T_{+x} + T_{-y} + T_{+y} - 4T\right) - \frac{2\Delta t}{W\rho Cp}\left[h\left(T - T_{air}\right) + \varepsilon\sigma\left(T^4 - T_{air}^4\right)\right] + \frac{Wp\Delta t}{\rho Cp} \tag{6.24}$$

Of course, Wp, must be turned on only for that portion of the plates being welded at a particular time.

Equations similar to Eqs. (6.23) and (6.24) may be derived for the right and left edges of the plates.

These equations are the same as Eqs. (6.23) and (6.24) except for the following:

- There are both convective and radiative losses from the exposed ends.
- The heat balance is performed around a half-element thickness in the weld direction (x-dir). Full welding power is applied to this half thickness at both the start and the end of the weld.

The resulting equation for the non-weld-line elements along the left edge is

$$T' = T + \frac{\alpha\Delta t}{\Delta x^2}\left(T_{-y} - 2T + T_{+y}\right) - \frac{\Delta t}{\rho Cp}\left(\frac{2h_v}{W} + \frac{h_h}{\Delta x}\right)\left(T - T_{air}\right) - \frac{2\alpha\Delta t}{\Delta x^2}\left(T - T_{+x}\right) - \frac{\varepsilon\sigma\Delta t}{\rho Cp}\left(\frac{2}{\Delta x} + \frac{1}{W}\right)\left(T^4 - T_{air}^4\right) \tag{6.25}$$

and the resulting equation for the weld-line element along the left edge is

$$T' = T + \frac{\alpha\Delta t}{\Delta x^2}\left(T_{-y} - 2T + T_{+y}\right) - \frac{\Delta t}{\rho Cp}\left(\frac{2h_v}{W} + \frac{h_h}{\Delta x}\right)\left(T - T_{air}\right) - \frac{2\alpha\Delta t}{\Delta x^2}\left(T - T_{+x}\right) - \frac{\varepsilon\sigma\Delta t}{\rho Cp}\left(\frac{2}{\Delta x} + \frac{1}{W}\right)\left(T^4 - T_{air}^4\right) + \frac{2Wp\Delta t}{\rho Cp} \tag{6.26}$$

The weld power is turned off and on using a control variable of 0 or 1. The control variable is determined by logical statements along the x-direction of the spreadsheet. The first of these logical statements resides in cell B38 and describes if the weld is complete. Cell A38 is to the left of the plate and is set at a value of 1 to initiate welding at column B. The contents of cell B38 is

```
=IF(s=0,0,IF(B38=1,1,IF(OR(AB24>mp,B24>mp),1,0)))
```
(6.27)

This expression is filled right to the right end of the plate. The variable “mp” is the melting point for welding.

The second logical statement also spread across the x-direction of the plate model starting in cell B40 is the control value for power in that column (location of x).

```
=IF(s=0,0,IF(B38=1,0,IF(AND(A38=1,C38=0,B24<mp),1,0)))
```
(6.28)

Appendix 5A

3D USS HT by ADI Method Code with Comments

```
Dim a, b, c, f, IC, counter, T(100, 100, 100), Tnew(100, 100, 100)
Dim B1C, B2C, B3C, B4C, B5C, B6C, nlops, ncols, nrows, npans
Sub initilize()
'Clear old
Range("c4:bb500").Select
    Selection.ClearContents
    Range("G3").Select
    Cells(2, 5).Value = 0
'Input all data from sheet
IC = Cells(4, 2).Value
ncols = Cells(5, 2).Value
nrows = Cells(6, 2).Value
npans = Cells(7, 2).Value
nlops = Cells(8, 2).Value
f = Cells(9, 2).Value
B1C = Cells(10, 2).Value
B2C = Cells(11, 2).Value
B3C = Cells(12, 2).Value
B4C = Cells(13, 2).Value
B5C = Cells(14, 2).Value
B6C = Cells(15, 2).Value
counter = 0
'Set up initial T's and BC's
For k = 1 To npans
For j = 1 To nrows
For i = 1 To ncols
    T(k, j, i) = IC
Next i
Next j
Next k
'Setup BC#1 and #2
  For k = 0 To npans + 1
    For j = 0 To nrows + 1
      T(k, j, 0) = B1C
      T(k, j, ncols + 1) = B2C
    Next j
  Next k
'Setup BC#3 and #4
  For k = 0 To npans + 1
    For i = 0 To ncols + 1
      T(k, 0, i) = B3C
      T(k, nrows + 1, i) = B4C
    Next i
  Next k
'Setup BC#5 and #6
  For j = 0 To nrows + 1
    For i = 0 To ncols + 1
      T(0, j, i) = B5C
      T(npans + 1, j, i) = B6C
    Next i
  Next j
'Write initial values
  Call output
'Set LHS matrix coeff's
  a = -f
  b = 2 * (1 + f)
  c = -f
End Sub
```

```
Sub main()
'Start time loops
For n = 1 To nlops
'Execute x-direction passes
  For j = 1 To nrows
    For k = 1 To npans
     Call Xdir(j, k)
    Next k
  Next j
  Call update'Replace all T's with Tnew's
'Execute y-direction passes
  For i = 1 To ncols
    For k = 1 To npans
      Call Ydir(i, k)
    Next k
  Next i
  Call update'Replace all T's with Tnew's
'Execute z-direction passes
  For i = 1 To ncols
    For j = 1 To nrows
      Call Zdir(i, j)
    Next j
  Next i
  Call update'Replace all T's with Tnew's
'Write out results LABELS
  Cells(4, 4 + ncols / 2).Value = "z-dir y vs x"
  Cells(4, 4 + ncols + 3 + nrows / 2).Value = "x-dir z vs y"
  Cells(4, 4 + ncols + 3 + nrows + 3 + npans / 2).Value = "y-dir z vs x"
  Call output‘Writeout results
Next n
'Update the counter for time calculation on the sheet
  counter = counter + nlops
  Cells(2, 5).Value = counter
End Sub
```

```
Sub update() 'Move all new T's into existing T array
For k = 1 To npans
  For j = 1 To nrows
    For i = 1 To ncols
      T(k, j, i) = Tnew(k, j, i)
    Next i
  Next j
Next k
End Sub
```

```
Sub Xdir(j, k) ' x-dir ADI
Dim d(100), v(100)
For i = 1 To ncols
    d(i) = f * T(k, j, i - 1) + 2 * (1 - f) * T(k, j, i) + f * T(k, j, i + 1)
Next i
d(1) = d(1) - a * B1C
d(ncols) = d(ncols) - c * B2C
Call Tridiag(ncols, d, v)
' fill v's into Tnew array
For i = 1 To ncols
  Tnew(k, j, i) = v(i)
Next i
End Sub
```

```
Sub Ydir(i, k) 'y-dir ADI
Dim d(100), v(100)
For j = 1 To nrows
    d(j) = f * T(k, j - 1, i) + 2 * (1 - f) * T(k, j, i) + f * T(k, j + 1, i)
Next j
d(1) = d(1) - a * B3C
d(nrows) = d(nrows) - c * B4C
```



```
Call Tridiag(nrows, d, v)
' fill v's into Tnew array
For j = 1 To nrows
  Tnew(k, j, i) = v(j)
Next j
End Sub
```

```
Sub Zdir(i, j)  'z-dir ADI
Dim d(100), v(100)
For k = 1 To npans
    d(k) = f * T(k - 1, j, i) + 2 * (1 - f) * T(k, j, i) + f * T(k + 1, j, i)
Next k
d(1) = d(1) - a * B3C
d(npans) = d(npans) - c * B4C
Call Tridiag(npans, d, v)
' fill v's into Tnew array
For k = 1 To npans
  Tnew(k, j, i) = v(k)
Next k
End Sub
```

```
Sub Tridiag(n, d, v)  'Tridaigonal Algorithm
Dim beta(100), gamma(100)
beta(1) = b
gamma(1) = d(1) / beta(1)
For i = 2 To n
  beta(i) = b - a * c / beta(i - 1)
  gamma(i) = (d(i) - a * gamma(i - 1)) / beta(i)
Next i
v(n) = gamma(n)
For i = n - 1 To 1 Step -1
  v(i) = gamma(i) - c * v(i + 1) / beta(i)
Next i
End Sub
```

```
Sub output() 'Output results in z-direction  (first column  x vs y)
For k = 0 To npans + 1
  For j = 0 To nrows + 1
    For i = 0 To ncols + 1
      Cells(k * (nrows + 3) + 5 + j, 4 + i).Value = T(k, j, i)
    Next i
  Next j
Next k
'Output results in x-direction  (second column  y vs z)
For i = 0 To ncols + 1
  For k = 0 To npans + 1
    For j = 0 To nrows + 1
      Cells(i * (npans + 3) + 5 + k, (ncols + 3) + 4 + j).Value = T(k, j, i)
    Next j
  Next k
Next i
'Output results in y-direction  (third column  x vs z)
For j = 0 To nrows + 1
  For k = 0 To npans + 1
    For i = 0 To ncols + 1
      Cells(j * (ncols + 3) + 5 + i, (ncols + 3) + (nrows + 3) + 4 + k).Value = T(k,
j, i)
    Next i
  Next k
Next j
End Sub
```
