SEP - 9 1987

ORNL
MASTER COPY

ornl

ORNL-6393

OAK RIDGE
NATIONAL
LABORATORY

MARTIN MARIETTA

IMPSOR, A Fully Vectorized FORTRAN Code for Three Dimensional Moving Boundary Problems with Dirichle or Neumann Boundary Conditions

M. A. Williams
D. G. Wilson

OPERATED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

Printed in the United States of America. Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road, Springfield, Virginia 22161
NTIS price codes—Printed Copy: A03; Microfiche A01

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL-6393

Engineering Physics and Mathematics Division

Mathematical Sciences Section

IMPSOR, A FULLY VECTORIZED FORTRAN CODE FOR THREE DIMENSIONAL MOVING BOUNDARY PROBLEMS WITH DIRICHLET OR NEUMANN BOUNDARY CONDITIONS

M. A. Williams*
D. G. Wilson

*Laboratory Graduate Program Participant from
Lehigh University, Bethlehem, Pennsylvania

Date Published: August 1987

Research sponsored by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy under contract number DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc. and the Laboratory Graduate Participation Program under contract number DE-AC05-7OR00033 between the U.S. Department of Energy and Oak Ridge Associated Universities.

Prepared by the
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831
operated by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under Contract No. DE-AC05-84OR21400

Table of Contents

Abstract v

1. Introduction 1

2. Input and Output Specifications 1

3. Implementation Details 10

References 21

Appendix A. Enthalphy Formulation 23

Appendix B. Numerical Formulation 27

Appendix C. Successive Overrelaxation Algorithm 33

IMPSOR, A FULLY VECTORIZED FORTRAN CODE FOR THREE DIMENSIONAL MOVING BOUNDARY PROBLEMS WITH DIRICHLET OR NEUMANN BOUNDARY CONDITIONS

M. A. Williams
D. G. Wilson

This report documents IMPSOR, a FORTRAN code that implements finite difference methods for multidimensional moving boundary problems with Dirichlet or Neumann boundary conditions on a Cray X-MP. The geometry of the spatial domain is a rectangular parallelepiped. The dimensions of the parallelepiped are specified by the user. Dirichlet or Neumann boundary conditions may be specified on each face of the box independently. The initial conditions are arbitrary. The user defines the initial and boundary conditions as well as the thermal and physical properties of the problem. Additionally, several parameters of the numerical method, e.g. degree of implicitness, time step size, are also specified by the user.

1. Introduction

This report documents IMPSOR, a FORTRAN code that implements finite difference methods for multidimensional moving boundary problems with Dirichlet or Neumann boundary conditions on a Cray X-MP. The name IMPSOR indicates that IMPlicit finite difference schemes may be used with an SOR solver. Explicit difference schemes may also be chosen depending on an input parameter specified by the user. Implicit finite difference schemes yield a mildly nonlinear system of equations to be solved at each time step. If an implicit scheme is chosen, the nonlinear system is solved using an SOR algorithm described in Appendix C. The performance of this algorithm is discussed in [2]. The geometry of the spatial domain is restricted to rectangular parallelepipeds. The dimensions of the parallelepiped are specified by the user. Dirichlet or Neumann boundary conditions may be specified on each face of the box independently. The initial conditions are arbitrary. The user defines the initial and boundary conditions as well as the thermal and physical properties of the problem. Additionally, several parameters of the numerical method, e.g. degree of implicitness, time step size, are also specified by the user.

In section 2, we describe the format of the user-supplied input file and the program-generated output files. In section 3, we discuss implementation details of the numerical method. A standard implementation of an SOR algorithm inhibits vectorization on a Cray X-MP. We describe how these problems are circumvented in IMPSOR. The equations of the underlying mathematical model are presented in Appendix A. The discretization of the spatial domain and the spatial differencing is summarized in Appendix B. The SOR nonlinear system solver is explained in Appendix C.

2. Input and Output Specifications

In this section we describe the input data file that must be supplied by the user and the output files produced by IMPSOR.

2.1 Input Data File

The thermal and physical properties of the problem, the parameters of the numerical method, and the initial and boundary conditions are all defined by the user in an input data file named *gin*. The input data file contains one or more data sets defining these variables. Each data set contains 18 records. The data sets within an input file are processed sequentially. In this subsection, we describe the FORTRAN format used to read each record and the variables contained in that record. The FORTRAN format statement is presented opposite the record number. We express the units of the variables in the MKS system. However, any consistent set of units may be used. FORTRAN unit 5 is assigned to file *gin*, the input file.

Record 1 format (i5)

Record 1 contains the value to be assigned to the checkpoint indicator. The program has the capability of writing partially processed data to a scratch file. The scratch file can be read later and the computation restarted. The value of the checkpoint indicator is set as follows.

0 - new computation; neither read nor write checkpoint

1 - new computation; write checkpoint but do not read one

2 - old computation; read checkpoint but do not write one

3 - old computation; read checkpoint and also write one

Record 2-4 format (1p2e12.5)

Records 2 through 4 define the location of the parallelepiped. Record 2 contains minimum value of the x coordinate in columns 1-12 and the maximum value of the x coordinate in columns 13-24. Records 3 and 4 contain the minimum and maximum values of the y and z coordinates, respectively. The units are in meters.

Record 5 format (6i5)

Record 5 contains the number of nodes and refinement levels in each direction. The number of nodes in the x, y, and z directions must be in columns 1-5, columns 6-10, and columns 11-15, respectively. The refinement levels in the x, y, and z directions must be in columns 16-20, columns 21-25, and columns 26-30, respectively. The code provides for automatic refinement of the mesh. If the number of nodes in the x direction is nx and the refinement level in the x direction is nxr, then the total number of nodes in the x direction used in the computation is $nx \cdot (nxr \cdot 2 + 1)$. However, the output file will only contain data at points corresponding to the nx unrefined mesh points. Similar considerations hold for other directions.

Record 6 format (1p3e12.5)

The constant density, and the solid and liquid heat capacities are specified on record 6. The density appears in columns 1-12. The solid heat capacity appears in columns 13-24 and the liquid heat capacity appears in columns 25-36. The units for density are kg/m^3. The units for heat capacity are kJ/kg-°C.

Record 7 format (1p2e12.5)

The thermal conductivity of the solid and liquid are specified on record 7. The conductivity of the solid is in columns 1-12. The conductivity of the liquid is in columns 13-24. The units for the conductivities are kg/ms - °C.

Record 8 format (1p3e12.5)

Record 8 contains the latent heat, the critical or melt temperature, and the initial temperature. The latent heat is specified in columns 1-12, the critical temperature in columns 13-24, and the initial temperature in columns 25-36. The units are °C.

Record 9 format (1p3e12.5)

Record 9 contains the output time step size, the amount of time to simulate, and the maximum number of time steps allowed. The output time step size is the time interval at which data are written to the output file. If zero is specified and all refinement levels are zero, then the output time step size, *dtout*, is set equal to the time step size used in the numerical method (see below). If zero is specified and the refinement levels are not zero, then *dtout* is set equal to the time step size used in the numerical method with the unrefined mesh. The output time step size is specified in columns 1-12. The amount of time to simulate the phase change process, *tquit*, is specified in columns 13-24. The maximum number of time steps allowed, *steps*, is specified in columns 25-36. If the input value of *tquit* is zero, then *tquit* will be set to *dtout* · *steps*. If the input value of *tquit* is nonzero, then the input value of *steps* will be ignored.

Record 10 format (i5,1p2e12.5)

Record 10 contains parameters required by the numerical method. The SOR algorithm for solving the nonlinear system requires an initial approximation. The code allows two choices: either the solution from the previous time step or the result of an explicit update. The first field (columns 1-5) of this record specifies which will be used as the initial

approximation. If the value in this field is 0, the solution from the previous time step is used. Otherwise, the result of an explicit update is used. The second field (columns 6-17) is the degree of implicitness of the algorithm to be used. If this parameter is 0, the method is fully explicit. If it is 1, the method is fully implicit. Any value between 0 and 1 may be chosen. The last field (columns 18-29) is the tolerance. The SOR algorithm is halted when the supremum of the difference between two successive iterates is less than the specified tolerance.

Record 11 format (i5,1p2e12.5)

The first field (columns 1-5) contains an indicator for whether the material is initially solid or liquid if the initial temperature is equal to the critical temperature (see record 8). A value of 1 indicates liquid. A value of 0 indicates solid. The second field (columns 6-17) is the relaxation parameter to be used in the SOR algorithm. This value should be between 1 and 2. The third data field (columns 18-29), *tmult*, controls the time step size. The maximum allowable time step size for an explicit method is calculated based on the parameters of the problem. The time step size used in the problem is set equal to the explicit time step multiplied by *tmult*.

Record 12-17 format (i5,1pe12.5)

Records 12 through 17 set the boundary conditions for the east, west, north, south, lower, and upper faces of the parallelepiped, respectively. The first field (columns 1-5) indicates the type of boundary condition imposed. A value of 1 indicates Dirichlet boundary conditions. A value of

0 indicates Neumann boundary conditions. The second field (columns 6-17) is the boundary value to be imposed. If Neumann boundary conditions are chosen, a positive number indicates a flux into the box. A negative number indicates a flux out of the box.

Record 18 format (i5)

Record 18 contains an end of file indicator. If the value in this field (columns 1-5) is 1, there are more data sets to be processed. Otherwise, execution is terminated after the current data set is processed.

A sample input data file is given in Table 2.1.

Table 2.1

0	(check point indicator)
0.0 e+00 1.2 e+00	(min & max x values)
0.0 e+00 1.0 e+00	(min & max y values)
0.0 e+00 0.8 e+00	(min & max z values)
21 15 45 0 0 0	(nx, ny, nz and refinement levels)
1.0 e 00 1.0 e 00 1.0 e 00	(rho, sol & liq ht capacities)
1.0 e -2 1.0 e -3	(sol & liq conductivities)
1.5 e 01 0.0 e 00 0.0 e 00	(latent ht, tcr, initial temp)
0.0 e 00 1.5 e 00 0.0 e 00	(dtout, tquit, last time step)
1 0.0 e-00 1.0 e-04	(ipcsw,theta,eps)
1 1.0 e 00 1.0 e 00	(imush,relax,tmult)
0 0.0 e 00	(east wall temp flag, east bc)
0 0.0 e 00	(west wall temp flag, west bc)
0 0.0 e 00	(north wall temp flag, north bc)
0 0.0 e 00	(south wall temp flag, south bc)
1 -1.0 e 01	(lower wall temp flag, lower bc)
0 0.0 e 00	(upper wall temp flag, upper bc)
0	(indicator of more data sets)

2.2 Output Files

IMPSOR generates two output files: *outsor* and *grautsor*. The first is designed to be read by the user. The second is suitable for a graphics post-processor.

All data read in from the input data file and some calculated values are written to *outsor* in the form *variable* = *value*. The initial temperature distribution on the lower face is also written to *outsor*. For each time step, the temperature and enthalpy distributions and the number of SOR iterations required for convergence are written to *outsor*. The beginning of the *outsor* file generated by IMPSOR with the sample input file in section 2.1 follows.

```
time at beginning of first data set  2.18100e-03
1 INPUT DATA and initial data.
    xmin,       xmax,       ymin,       ymax,       zmin,       zmax
   0.        1.20000e+00  0.        1.00000e+00  0.        8.00000e-01

nx =  21, ny =  15, nz =  45, nxny =  391, ntot =17986, nsw =   1

frstx =  392, lastx =17985, lasty =17965, lastz =17595

rho =  1.000e+00, cps =  1.000e+00,  cpl =  1.000e+00
rks =  1.000e-02, rkl =  1.000e-03, rkmax =  1.000e-02, rkavg =  5.500e-03

h =  1.500e+01, tcr =  0.      , t0 =  0.      , e0 =  1.500e+01

dtout =  3.160e-02, tquit =  1.500e+00, steps =  0.

ipcsw =   1, theta =  5.000e-01, eps =  1.000e-04

imush =   1, relax =  1.000e+00, tmult =  5.000e+00

dx,dy,dz,dt:   5.71429e-02  6.66667e-02  1.77778e-02  1.58025e-02

iet =   0, bce =  0.
```

```
iet =  0, bce = 0.

ist =  0, bcs = 0.

inrt =  0, bcn = 0.

iut =  0, bcu = 0.

ilt =  1, bcl = -1.000e+01
x-coords of first values output:  8.57143e-02 2.57143e-01 4.28571e-01
time = 1.58025e-02 dt = 1.58025e-02 # of iter =  1

begin k level =   3, z = 2.66667e-02
 row j =   3, y = 1.00000e-01
t(*,j,k): 0.        0.        0.        0.        0.        0.        0.
e(*,j,k): 1.50000e+01 1.50000e+01 1.50000e+01 1.50000e+01 1.50000e+01 1.50000e+01 1.50000e+01
 row j =   6, y = 3.00000e-01
t(*,j,k): 0.        0.        0.        0.        0.        0.        0.
e(*,j,k): 1.50000e+01 1.50000e+01 1.50000e+01 1.50000e+01 1.50000e+01 1.50000e+01 1.50000e+01
 row j =   9, y = 5.00000e-01
t(*,j,k): 0.        0.        0.        0.        0.        0.        0.
e(*,j,k): 1.50000e+01 1.50000e+01 1.50000e+01 1.50000e+01 1.50000e+01 1.50000e+01 1.50000e+01
 row j =  12, y = 7.00000e-01
t(*,j,k): 0.        0.        0.        0.        0.        0.        0.
e(*,j,k): 1.50000e+01 1.50000e+01 1.50000e+01 1.50000e+01 1.50000e+01 1.50000e+01 1.50000e+01
```

All parameters used for the current execution of IMPSOR are also written to *groutsor*. Five data records containing this information are written at the top of each output data set. Then one data record is written for each time step and grid point containing the data set number, time, x-coordinate index, y-coordinate index, z-coordinate, temperature, and

enthalpy. If the x-coordinate index is ix and the mesh width in the x direction is dx, then the x-coordinate is given by $x_0 + ix \cdot dx$. Here x_0 is the minimum x value specified on input record 2. The beginning of the *groutsor* file generated by IMPSOR with the sample input file in section 2.1 follows.

```
1-0.50000e+01 0        0        0.       0.       0.
1-0.40000e+01 0.       0.12000e+01 0.       0.10000e+01 0.50000e+01
1-0.30000e+01 0.       0.80000e+00 0.       0.       0.70000e+01
1-0.20000e+01 0.10000e+01 0.10000e+01 0.10000e+01 0.10000e+01
1-0.10000e+01 0.10000e-01 0.10000e-02 0.       0.15000e+02
1 0.15802e-01 1        1        0.26667e-01 0.       0.15000e+02
1 0.15802e-01 2        1        0.26667e-01 0.       0.15000e+02
1 0.15802e-01 3        1        0.26667e-01 0.       0.15000e+02
1 0.15802e-01 4        1        0.26667e-01 0.       0.15000e+02
1 0.15802e-01 5        1        0.26667e-01 0.       0.15000e+02
1 0.15802e-01 6        1        0.26667e-01 0.       0.15000e+02
1 0.15802e-01 7        1        0.26667e-01 0.       0.15000e+02
1 0.15802e-01 1        2        0.26667e-01 0.       0.15000e+02
1 0.15802e-01 2        2        0.26667e-01 0.       0.15000e+02
1 0.15802e-01 3        2        0.26667e-01 0.       0.15000e+02
1 0.15802e-01 4        2        0.26667e-01 0.       0.15000e+02
1 0.15802e-01 5        2        0.26667e-01 0.       0.15000e+02
1 0.15802e-01 6        2        0.26667e-01 0.       0.15000e+02
1 0.15802e-01 7        2        0.26667e-01 0.       0.15000e+02
1 0.15802e-01 1        3        0.26667e-01 0.       0.15000e+02
```

3. Implementation Details

Most of the computing time used by the numerical method is spent in the iterative solution of the nonlinear system. Hence, we concentrate on reducing the computation time used by the SOR algorithm described in Appendix C. We briefly summarize the numerical method here.

3.1 Numerical Method

Partitioning the spatial domain and discretizing the governing equations yields the following nonlinear system at each time step.

$$e_{ijk}^{n+1} + \xi_{ijk}^{n+1} T_{ijk}^{n+1} = \zeta_{ijk}^{n+1}$$

where

$$\begin{aligned}\xi_{ijk}^{n+1} \equiv & \frac{\theta \Delta t}{\rho \Delta x^2} (k_{i+\frac{1}{2},jk}^{n+1} + k_{i-\frac{1}{2},jk}^{n+1}) \\ & + \frac{\theta \Delta t}{\rho \Delta y^2} (k_{i,j+\frac{1}{2},k}^{n+1} + k_{i,j-\frac{1}{2},k}^{n+1}) \\ & + \frac{\theta \Delta t}{\rho \Delta z^2} (k_{ij,k+\frac{1}{2}}^{n+1} + k_{ij,k-\frac{1}{2}}^{n+1}) ,\end{aligned}$$

and

$$\begin{aligned}\zeta_{ijk}^{n+1} \equiv & b_{ijk}^{n} + \frac{\theta \Delta t}{\rho \Delta x^2} (k_{i+\frac{1}{2},jk}^{n+1} T_{i+1,jk}^{n+1} + k_{i-\frac{1}{2},jk}^{n+1} T_{i-1,jk}^{n+1}) \\ & + \frac{\theta \Delta t}{\rho \Delta y^2} (k_{i,j+\frac{1}{2},k}^{n+1} T_{i,j+1,k}^{n+1} + k_{i,j-\frac{1}{2},k}^{n+1} T_{i,j-1,k}^{n+1}) \\ & + \frac{\theta \Delta t}{\rho \Delta z^2} (k_{ij,k+\frac{1}{2}}^{n+1} T_{ij,k+1}^{n+1} + k_{ij,k-\frac{1}{2}}^{n+1} T_{ij,k-1}^{n+1}) .\end{aligned}$$

Here subscripts denote the location in the spatial mesh and the superscript denotes the time level. See Appendix B for details. The parameter θ denotes the degree of implicitness of the numerical scheme. This nonlinear system is solved using a successive overrelaxation (SOR) scheme. Let τ_{ijk}^{p} denote the p th iterate of T_{ijk}^{n+1} and for compactness, define

$$z_{ijk}^{p} = b_{ijk}^{n} + \frac{\theta \Delta t}{\rho \Delta x^2} (k_{i+1/2,jk}^{p} \tau_{i+1,jk}^{p} + k_{i-1/2,jk}^{p+1} \tau_{i-1,jk}^{p+1})$$
$$+ \frac{\theta \Delta t}{\rho \Delta y^2} (k_{i,j+1/2,k}^{p} \tau_{i,j+1,k}^{p} + k_{i,j-1/2,k}^{p+1} \tau_{i,j-1,k}^{p+1})$$
$$+ \frac{\theta \Delta t}{\rho \Delta z^2} (k_{ij,k+1/2}^{p} \tau_{ij,k+1}^{p} + k_{ij,k-1/2}^{p+1} \tau_{ij,k-1}^{p+1}) , \tag{3.1}$$

and

$$c_{ijk}^{p} = \frac{\theta \Delta t}{\rho \Delta x^2} (k_{i+1/2,jk}^{p} + k_{i-1/2,jk}^{p+1})$$
$$+ \frac{\theta \Delta t}{\rho \Delta y^2} (k_{i,j+1/2,k}^{p} + k_{i,j-1/2,k}^{p+1})$$
$$+ \frac{\theta \Delta t}{\rho \Delta z^2} (k_{ij,k+1/2}^{p} + k_{ij,k-1/2}^{p+1}) . \tag{3.2}$$

For a given node, temperature and thermal conductivity values are lagged one iteration at advanced points, but not at previous points. This constitutes a Gauss-Seidel iteration. Then the iterative form of the equation to be solved is

$$e_{ijk}^{p+1} + c_{ijk}^{p} \tau_{ijk}^{p+1} = z_{ijk}^{p} . \tag{3.3}$$

The quantity z_{ijk}^{p} is computed. Then a new approximation to the temperature, τ_{ijk}^{p+1} is calculated using z_{ijk}^{p} . Details may be found in Appendix C.

3.2 Cray X-MP Implementation

We first consider an implementation with enthalpies and temperatures are stored in three-dimensional arrays $e(nx, ny, nz)$ and $t(nx, ny, nz)$, respectively. Psuedo-code for this SOR algorithm may be written as follows.

```
while ( time < tquit ) do
  for k = 1 to nz
    for j = 1 to ny
      for i = 1 to nx
        compute b(i, j, k)
```

```
        end
      end
    end
    while (snorm > tolerance) do
      for k = 1 to nz
        for j = 1 to ny
          for i = 1 to nx
            compute thermal conductivities
            compute z(i,j,k) and c(i,j,k)
            compute SOR temperature, tnew
            if i = j = k = 1
            then snorm = | tnew - t(i,j,k) |
            else snorm = max{ snorm, | tnew - t(i,j,k) | }
            set t(i,j,k) = tnew
          end
        end
      end
    endwhile
    output data
    time = time + Δt
  endwhile
```

The details of computing z and c are omitted here. The computation of z and c is different at boundary nodes than it is at interior nodes. This does not present any difficulty. Boundary nodes are easily recognized, i.e. $i = 1$ corresponds to a west boundary node, $i = nx$ corresponds to an east boundary node, etc. and calculations are modified accordingly.

Code based on the above pseudo-code could be implemented on either a serial or vector computer. However, the efficiency on a serial machine may be very different from the efficiency on a vector processor. We are interested in an efficient vector implementation for the Cray X-MP. The Cray X-MP performs identical calculations on long vectors very quickly. Calculations in the innermost loops are candidates for vectorization. The number of passes through the loop is the vector length. The speed of the Cray X-MP increases as the vector length increases. Vector lengths of at least 20 are required to acheive any significant increase in speed. In the above psuedo-code, the length of the vectors in the innermost loop is only nx. Interchanging the order of the loops might increase the vector

length. However the longest vector length possible using the three-dimensional array data structure is $\max\{nx, ny, nz\}$. This is the motivation for storing the temperature and enthalpy values in vectors.

An ordering of the nodes unfolds the three dimensional enthalpy and temperature arrays into vectors. It converts an $nx \times ny \times nz$ array into a vector of length $nx \cdot ny \cdot nz$. In the natural ordering, nodes are consecutively numbered from left to right, front to back, and bottom to top. Suppose i, j, and k are the row, column and level indices of the three dimensional array. Then the corresponding index, l say, of the vector is $l \equiv i + (j-1)nx + (k-1)nx \cdot ny$. Also, given the index l of the vector, i, j, and k are given by

$$
\begin{aligned}
k &= \text{Integer}\,[(l-1)/(nx\ ny)] + 1 \\
j &= \text{Integer}\,[(l-(k-1)nx\ ny)/nx] + 1 \\
i &= l - (k-1)nx\ ny - (j-1)nx\ .
\end{aligned}
$$

Using this notation, the discrete equations for interior nodes may be written as

$$
\begin{aligned}
e_l^{n+1} = {} & \frac{\theta \Delta t}{\rho \Delta x^2}\left[k_{E,l}^{n+1}(T_{l+1}^{n+1} - T_l^{n+1}) - k_{W,l}^{n+1}(T_l^{n+1} - T_{l-1}^{n+1})\right] \\
& + \frac{\theta \Delta t}{\rho \Delta y^2}\left[k_{N,l}^{n+1}(T_{l+nx}^{n+1} - T_l^{n+1}) - k_{S,l}^{n+1}(T_l^{n+1} - T_{l-nx}^{n+1})\right] \\
& + \frac{\theta \Delta t}{\rho \Delta z^2}\left[k_{U,l}^{n+1}(T_{l+nxny}^{n+1} - T_l^{n+1}) - k_{L,l}^{n+1}(T_l^{n+1} - T_{l-nxny}^{n+1})\right] \\
& + b_l^n\,,
\end{aligned}
\tag{3.4}
$$

where

$$
\begin{aligned}
b_l^n = {} & e_l^n + \frac{(1-\theta)\Delta t}{\rho \Delta x^2}\left[k_{E,l}^{n}(T_{l+1}^{n} - T_l^{n}) - k_{W,l}^{n}(T_l^{n} - T_{l-1}^{n})\right] \\
& + \frac{(1-\theta)\Delta t}{\rho \Delta y^2}\left[k_{N,l}^{n}(T_{l+nx}^{n} - T_l^{n}) - k_{S,l}^{n}(T_l^{n} - T_{l-nx}^{n})\right] \\
& + \frac{(1-\theta)\Delta t}{\rho \Delta z^2}\left[k_{U,l}^{n}(T_{l+nxny}^{n} - T_l^{n}) - k_{L,l}^{n}(T_l^{n} - T_{l-nxny}^{n})\right].
\end{aligned}
\tag{3.5}
$$

Here

$$k_{E,l} \equiv (k(T_l) + k(T_{l+1}))/2, \qquad k_{W,l} \equiv (k(T_l) + k(T_{l-1}))/2,$$

$$k_{N,l} \equiv (k(T_l) + k(T_{l+nx}))/2, \qquad k_{S,l} \equiv (k(T_l) + k(T_{l-nx}))/2,$$

$$k_{U,l} \equiv (k(T_l) + k(T_{l+nxny}))/2, \quad \text{and} \quad k_{L,l} \equiv (k(T_l) + k(T_{l-nxny}))/2.$$

Pseudo-code for the SOR algorithm using this ordering is as follows.

```
while ( time <tquit ) do
  for l = 1 to nx · ny · nz
    compute b(l)
  end
  while (snorm >tolerance ) do
    for l = 1 to nx · ny · nz
      compute k_E,l , k_W,l , k_N,l , k_S,l , k_U,l , and k_L,l
      compute z(l) and c(l)
      compute SOR temperature, tnew
      if l = 1
      then snorm = | tnew − t(l) |
      else snorm = max { snorm , | tnew − t(l) | }
      set t(l) = tnew
    end
  endwhile
  output data
  time = time + Δt
endwhile
```

The vector length in this pseudo-code is $nx \cdot ny \cdot nz$. Thus, if all calculations could be vectorized, one expects a Cray X-MP implementation based on this pseudo-code to run faster than a Cray X-MP implementation based on the previous psuedo-code.

We now focus on vectorizing the calculations in the innermost loop. To insure that vectorization is not inhibited, interdependencies among vector elements must be avoided and calculations at interior and boundary nodes must be identical. Computing the Gauss-Seidel iterate of $T(i)$ requires the current iterate of $T(i-nxny)$, $T(i-nx)$, and $T(i-1)$. This creates an interdependency among vector elements. The interdependency can be eliminated by renumbering the nodes using a red/black ordering. In a red/black ordering, odd numbered nodes are thought of as "red" and even numbered nodes as thought of as

"black." We require an odd number of nodes per row and column. Thus all neighbors of a given node are of the opposite color. Nodes of one color are updated and then nodes of the opposite color are updated. This decreases the vector length by a factor of 2. In our implementation, red nodes are updated first using black nodal values from the previous time step. Black nodes are then updated using the new red nodal values. In effect, we make two Jacobi-type updates to get one Gauss-Seidel iteration. For a more complete description of red/black ordering and a discussion of its effects on convergence of Gauss-Seidel iterations, see [3].

When the nodes are numbered with a natural or red/black ordering, boundary nodes are scattered throughout the temperature and enthalpy vectors. Thus, arranging the computation so that boundary and interior node calculations are identical is more difficult. In our implementation, boundary values are stored in the vectors along with the nodal values. Consider a three dimensional parallelepiped with the mesh described in Appendix B. Boundary data is required along the faces of this box. "Boundary data nodes" are placed along these faces to store the boundary data. Then the total number of nodes in x, y, and z directions is $(nx+2)$, $(ny+2)$, and $(nz+2)$, respectively. When unfolding the three dimensional array, a boundary data node is numbered as any other node. Thus the first and last $(nx+2)\cdot(ny+2)$ elements of the vector contain boundary data from the top and bottom of the box. If Dirichlet boundary conditions are imposed, then the temperature at a boundary data node is stored at the appropriate place in the temperature vector. If Neumann boundary conditions are imposed, then $\frac{dt}{d^*} flux$ is stored in the appropriate location in the temperature vector. Here d^* is either dx, dy, or dz depending on at which face the flux is imposed. The user specifies the type of boundary condition imposed on each face and the constant boundary values (cf section 2.1, records 12-17).

Rewriting the nonlinear system gives some insight into the modifications required at boundary nodes. Substituting the expressions for z and c into equation (3.3) and rearranging terms yields the following expression.

$$
\begin{aligned}
e_l^{p+1} = \; & \frac{\theta \Delta t}{\rho \Delta x^2} [k_{E,l}^p (\tau_{l+1}^p - \tau_l^{p+1}) - k_{W,l}^{p+1} (\tau_l^{p+1} - \tau_{l-1}^{p+1})] \\
& + \frac{\theta \Delta t}{\rho \Delta y^2} [k_{N,l}^p (\tau_{l+nx}^p - \tau_l^{p+1}) - k_{S,l}^{p+1} (\tau_l^{p+1} - \tau_{l-nx}^{p+1})] \\
& + \frac{\theta \Delta t}{\rho \Delta z^2} [k_{U,l}^p (\tau_{l+nxny}^p - \tau_l^{p+1}) - k_{L,l}^{p+1} (\tau_l^{p+1} - \tau_{l-nxny}^{p+1})] \\
& + b_l^n ,
\end{aligned}
\tag{3.6}
$$

where

$$
\begin{aligned}
b_l^n = \; & e_l^n + \frac{(1-\theta)\Delta t}{\rho \Delta x^2} [k_{E,l}^n (T_{l+1}^n - T_l^n) - k_{W,l}^n (T_l^n - T_{l-1}^n)] \\
& + \frac{(1-\theta)\Delta t}{\rho \Delta y^2} [k_{N,l}^n (T_{l+nx}^n - T_l^n) - k_{S,l}^n (T_l^n - T_{l-nx}^n)] \\
& + \frac{(1-\theta)\Delta t}{\rho \Delta z^2} [k_{U,l}^n (T_{l+nxny}^n - T_l^n) - k_{L,l}^n (T_l^n - T_{l-nxny}^n)] .
\end{aligned}
\tag{3.7}
$$

Here

$$
\begin{aligned}
k_{E,l} &\equiv (k(T_l) + k(T_{l+1}))/2, & k_{W,l} &\equiv (k(T_l) + k(T_{l-1}))/2, \\
k_{N,l} &\equiv (k(T_l) + k(T_{l+nx}))/2, & k_{S,l} &\equiv (k(T_l) + k(T_{l-nx}))/2, \\
k_{U,l} &\equiv (k(T_l) + k(T_{l+nxny}))/2, \quad \text{and} & k_{L,l} &\equiv (k(T_l) + k(T_{l-nxny}))/2.
\end{aligned}
$$

Suppose Dirichlet boundary conditions are imposed along the west wall. If node l is a west boundary node, $T(l-1)$ is the boundary temperature. Then the only modifications required are that $k_{W,l}^{p+1}(\tau_l^{p+1} - \tau_{l-1}^{p+1})$ be replaced by $2 \cdot k_{W,l}^{p+1}(\tau_l^{p+1} - \tau_{l-1}^{p+1})$ in (3.6) and

$k^n_{W,l}(T^n_l - T^n_{l-1})$ be replaced by $2 \cdot k^n_{W,l}(T^n_l - T^n_{l-1})$ in (3.7) for all west boundary nodes. See Appendix B for details. Similar modifications occur along other walls where Dirichlet boundary conditions are imposed.

Now suppose Neumann boundary conditions are imposed along the west wall. If node l is a west boundary node, $T(l-1)$ contains the imposed flux. This flux may be added to b^n_l before the iteration begins. Also, the $k_{W,l}$ term must be eliminated from both (3.6) and (3.7). Equivalently, we may replace $k^{p+1}_{W,l}(\tau^{p+1}_l - \tau^{p+1}_{l-1})$ by $0 \cdot k^{p+1}_{W,l}(\tau^{p+1}_l - \tau^{p+1}_{l-1})$ in (3.6) and $k^n_{W,l}(T^n_l - T^n_{l-1})$ by $0 \cdot k^n_{W,l}(T^n_l - T^n_{l-1})$ in (3.7) for all west boundary nodes. Similar modifications occur along other walls where Neumann boundary conditions are imposed. This motivates the creation of boundary mask vectors.

Seven boundary mask vectors are created to distinguish between interior nodes, boundary nodes, and boundary data nodes. The mask vector corresponding to the boundary data nodes, *nbdy* is defined as follows.

$$nbdy(i) = \begin{cases} 1 & \text{if node } i \text{ is a boundary data node} \\ 0 & \text{otherwise} \end{cases}$$

The other six mask vectors correspond to the six faces of the parallelepiped. If Dirichlet boundary conditions are applied on the east face, then the mask vector corresponding to the east face, *nxrt*, is defined as follows.

$$nxrt(i) = \begin{cases} 2 & \text{if node } i \text{ is an east boundary node} \\ 0 & \text{if node } i \text{ is a boundary data node} \\ 1 & \text{otherwise} \end{cases}$$

If Neumann boundary conditions are applied on the east face, then *nxrt* is modified as follows.

$$nxrt(i) = \begin{cases} 0 & \text{if node } i \text{ is an east boundary node} \\ 0 & \text{if node } i \text{ is a boundary data node} \\ 1 & \text{otherwise} \end{cases}$$

The other mask vectors, $nxlft$, $nyrt$, $nylft$, $nzrt$, and $nzlft$, are similarly defined.

With the mask vectors so defined, computation of z, c, and b can be identical for interior, boundary, and boundary data nodes. After each of the thermal conductivities, $k_{W,l}$, $k_{E,l}$, $k_{S,l}$, $k_{N,l}$, $k_{L,l}$, and $k_{U,l}$, are computed as described in [3], they are replaced by $nxlft(l) \cdot k_{W,l}$, $nxrt(l) \cdot k_{E,l}$, $nylft(l) \cdot k_{S,l}$, $nyrt(l) \cdot k_{N,l}$, $nzlft(l) \cdot k_{L,l}$, and $nzrt(l) \cdot k_{U,l}$, respectively. Prior to the iteration, some additional processing must be done at boundary nodes where Neumann boundary conditions are imposed. The imposed flux, which is stored in $T(l-1)$ is simply added to b_l^n. Psuedo-code for this algorithm follows.

```
initialize variables
start(1) = nx · ny + 1
start(2) = start(1) + 1
while ( time < tquit ) do
  for l = 1 to nx · ny · nz
    compute b(l)
  end
  Add contributions from Neumann boundary conditions to b
  while (snorm > tolerance) do
    for i = 1 to 2
      for l = start(i) to (nx+1)·(ny+1)·(nz+1) by 2
        compute k_E,l, k_W,l, k_N,l, k_S,l, k_U,l, and k_L,l
        set k_E,l = nxrt(l)·k_E,l
        set k_W,l = nxlft(l)·k_W,l
        set k_N,l = nyrt(l)·k_N,l
        set k_S,l = nylft(l)·k_S,l
        set k_U,l = nzrt(l)·k_U,l
        set k_L,l = nzlft(l)·k_L,l
        compute z(l) and c(l)
        compute SOR temperature, tnew
        if l = 1
        then snorm = | tnew − t(l) |
        else snorm = max { snorm, | tnew − t(l) | }
        set t(l) = nbdy(l)·tnew
      end
    end
  endwhile
  output data
  time = time + Δt
endwhile
```

Computation of the thermal conductivities and *tnew* is described in detail in [3]. The i loop controls the red/black ordering. The l loop is executed first for the red nodes and

then for the black. As was previously mentioned, this eliminates vector dependencies. However, the CFT compiler will still see a vector dependency in the l loop. In our FORTRAN implementation, the compiler directive CDIR IVDEP is issued preceding the l loop. This instructs the compiler to ignore any apparent vector dependencies and allow vector execution of the loop.

This version of the SOR algorithm requires more storage than the version using three dimensional arrays. The additional storage enables us to create longer vectors and vectorize all calculations. Thus, we feel the small increase in storage is justified. Performance results of this algorithm are presented and discussed in [2].

References

[1] Elliott, C. M. and J. R. Ockendon, *Weak and Variational Methods for Moving Boundary Problems*, Pitman, Boston 1982.

[2] Williams, M. A., "Iterative Solution of a Nonlinear System Arising in Phase Change Problems," Oak Ridge National Laboratory, Report ORNL-6398, 1987.

[3] Williams, M. A. and D. G. Wilson, "Vectorized Difference Schemes for a Three Dimensional Enthalpy Formulation for Phase Change Problems," Oak Ridge National Laboratory, Report ORNL/TM-10034, 1986.

Appendix A. Enthalpy Formulation

The enthalpy formulation of a Stefan problem consists of a parabolic partial differential equation and an auxiliary relationship. The partial differential equation relates changes in enthalpy to temperature gradients. The auxiliary relationship expresses enthalpy as a function of temperature. The enthalpy formulation we consider is expressed as follows. Determine a temperature distribution $T(x,t)$ and an enthalpy distribution $e(x,t)$ that satisfy

$$\rho e_t = div(k(T)\, grad\, T), \tag{A.1}$$

where

$$e = \begin{cases} \int_{T_{cr}}^{T} c(\tau)\, d\tau & T < T_{cr}, \\ [0, H] & T = T_{cr}, \\ \int_{T_{cr}}^{T} c(\tau)\, d\tau + H & T > T_{cr}. \end{cases} \tag{A.2}$$

Here e is the enthalpy, T is the temperature, $k(T)$ is the thermal conductivity, $c(T)$ is the heat capacity, ρ is the constant density, and H is the latent heat. x is a vector denoting position in the spatial domain. Note that enthalpy is multi-valued when $T = T_{cr}$.

If the heat capacities are constant in the solid and liquid, the enthalpy/temperature relation (A.2) reduces to

$$e = \begin{cases} c_{Sol}(T - T_{cr}) & T < T_{cr}, \\ [0, H] & T = T_{cr}, \\ c_{Liq}(T - T_{cr}) + H & T > T_{cr}. \end{cases} \tag{A.3}$$

Here c_{Sol} and c_{Liq} are the specific heats of the solid and liquid respectively. See Figure 1. In this case, the relation can also be expressed as

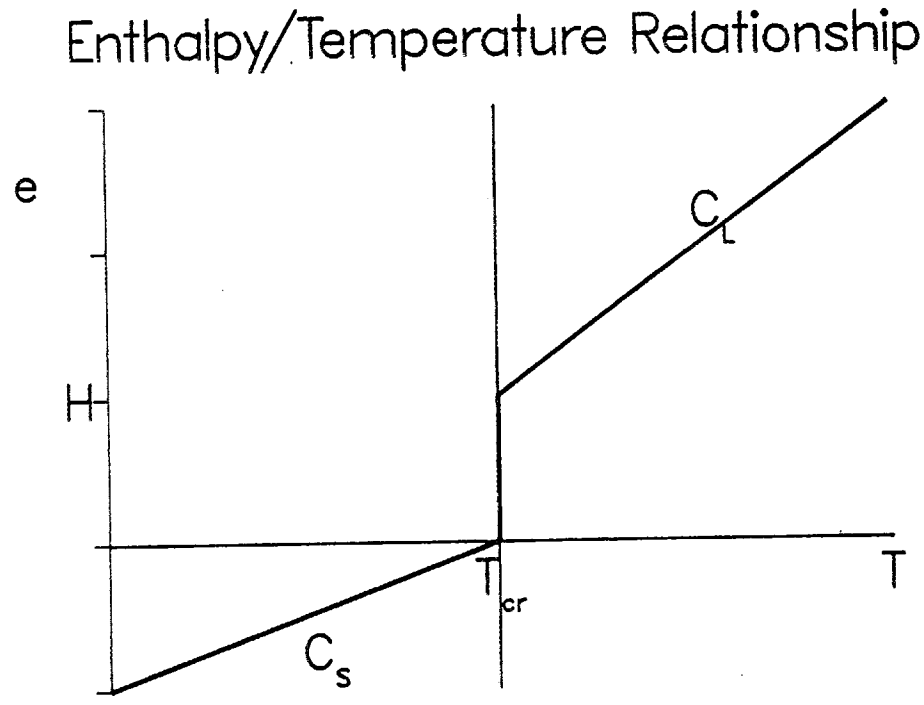

Figure 1. Enthalpy/Temperature Relationship

$$T = \begin{cases} T_{cr} + e / c_{Sol} & e \leqslant 0, \\ T_{cr} & 0 < e < H, \\ T_{cr} + (e-H) / c_{Liq} & e \geqslant H, \end{cases} \tag{A.4}$$

where temperature is a single-valued function of enthalpy.

The partial differential equation (A.1) is derived from the First Law of Thermodynamics (conservation of energy) and Fourier's Law. Let q denote the heat flux. Then these relationships can be expressed as follows:

$$\rho e_t = -div(q), \qquad \text{(Conservation of Energy)}$$

$$q = -k(T)\, grad\, T. \qquad \text{(Fourier's Law)}$$

It is useful to think of the partial differential equation in this way when formulating the numerical problem.

The spatial domain we consider is a rectangular parallelepiped. Thus we assume a Cartesian coordinate system. Let q_1, q_2, and q_3 denote the x, y, and z components of q, respectively. Then the conservation of energy law, Fourier's Law, and the partial differential equation (A.1) can be expressed in Cartesian coordinates as follows:

$$\rho e_t = -\left[(q_1)_x + (q_2)_y + (q_3)_z\right], \tag{A.5}$$

$$q = \left[q_1, q_2, q_3\right]^T = -k(T)\left[T_x, T_y, T_z\right]^T, \tag{A.6}$$

$$\rho e_t = (k(T)T_x)_x + (k(T)T_y)_y + (k(T)T_z)_z. \tag{A.7}$$

In Appendix B we discuss an implicit finite difference approximation to these equations.

Appendix B. Numerical Formulation

In this section, we discuss a numerical formulation of the mathematical problem. We first describe how the spatial domain is partitioned. Then we describe an implicit finite difference approximation of the governing partial differential equation. Finally, we present the mildly nonlinear system of equations resulting from the implicit approximation.

B.1 Discretization of the Spatial Domain

We partition the rectangular parallelepiped $[0, lx] \times [0, ly] \times [0, lz]$ as follows. Given positive integers nx, ny, and nz, we set $\Delta x = lx/nx$, $\Delta y = ly/ny$, and $\Delta z = lz/nz$. (We require that nx and ny be odd for reasons to be explained later in this section.) Let $x_j = j\,\Delta x$ for $j = 0, 1, 2, \ldots, nx$, $y_j = j\,\Delta y$ for $j = 0, 1, 2, \ldots, ny$, and $z_j = j\,\Delta z$ for $j = 0, 1, 2, \ldots, nz$. We partition the intervals $[0, lx]$, $[0, ly]$, and $[0, lz]$ as follows. Denote by I_j the interval (x_{j-1}, x_j) for $j = 1, 2, \ldots, nx$, by J_j the interval (y_{j-1}, y_j) for $j = 1, 2, \ldots, ny$, and by K_j the interval (z_{j-1}, z_j) for $j = 1, 2, \ldots, nz$. The I, J, and K intervals are the intervals for which the model provides approximations for average enthalpy and temperature values. Denote by L_l the interval $(x_l - \Delta x/2,\ x_l + \Delta x/2)$ for $l = 1, 2, \ldots, nx - 1$, by M_l the interval $(y_l - \Delta y/2,\ y_l + \Delta y/2)$ for $l = 1, 2, \ldots, ny - 1$, and by P_l the interval $(z_l - \Delta z/2,\ z_l + \Delta z/2)$ for $l = 1, 2, \ldots, nz - 1$. Denote by L_0 the interval $(0, \Delta x/2)$ and by L_{nx} the interval $(lx - \Delta x/2,\ lx)$. Similarly, denote by M_0 the interval $(0, \Delta y/2)$, by M_{ny} the interval $(ly - \Delta y/2,\ ly)$, by P_0 the interval $(0, \Delta z/2)$, and by P_{nz} the interval $(lz - \Delta z/2,\ lz)$. The L, M, and P intervals are intervals for which the model requires approximate values of the thermal conductivities. Figure 2 shows how the I and L intervals interleave.

$$\begin{array}{ccccccccccccccccc}
\cdot & I_1 & \cdot & I_2 & \cdot & I_3 & \cdot & I_4 & \cdot & I_5 & \cdot & I_6 & \cdot & I_7 & \cdot & I_8 & \cdot \quad \ldots \\
\hline
L_0 & \cdot & L_1 & \cdot & L_2 & \cdot & L_B. & \cdot & L_4 & \cdot & L_5 & \cdot & L_6 & \cdot & L_7 & \cdot & L_8 \quad \ldots
\end{array}$$

Figure 2. Interleaved Intervals I_j and L_j for $j = 0, 1, \cdots$.

The domain $[0, lx] \times [0, ly] \times [0, lz]$ is partitioned by $I_i \times J_j \times K_k$ for $i = 1, 2, \ldots, nx$, $j = 1, 2, \ldots, ny$ and $k = 1, 2, \ldots, nz$. Nodes are located in the centers of the three dimensional cells defined by $I_i \times J_j \times K_k$ for $i = 1, 2, \ldots, nx$, $j = 1, 2, \ldots, ny$, and $k = 1, 2, \ldots, nz$. For example consider a $5 \times 5 \times 5$ grid. All cross sections parallel to the xy-, xz-, or yz-planes through $(i + ½)\,\Delta z$ or $(i + ½)\,\Delta y$ or $(i + ½)\,\Delta x$ for $i = 0, 1, \ldots, 4$ are as depicted in Figure 3.

Node ijk is an interior node if $ieps\{2, 3, \ldots, nx-1\}$, $jeps\{2, 3, \ldots, ny-1\}$, and $keps\{2, 3, \ldots, nz-1\}$. Otherwise, node ijk is a boundary node.

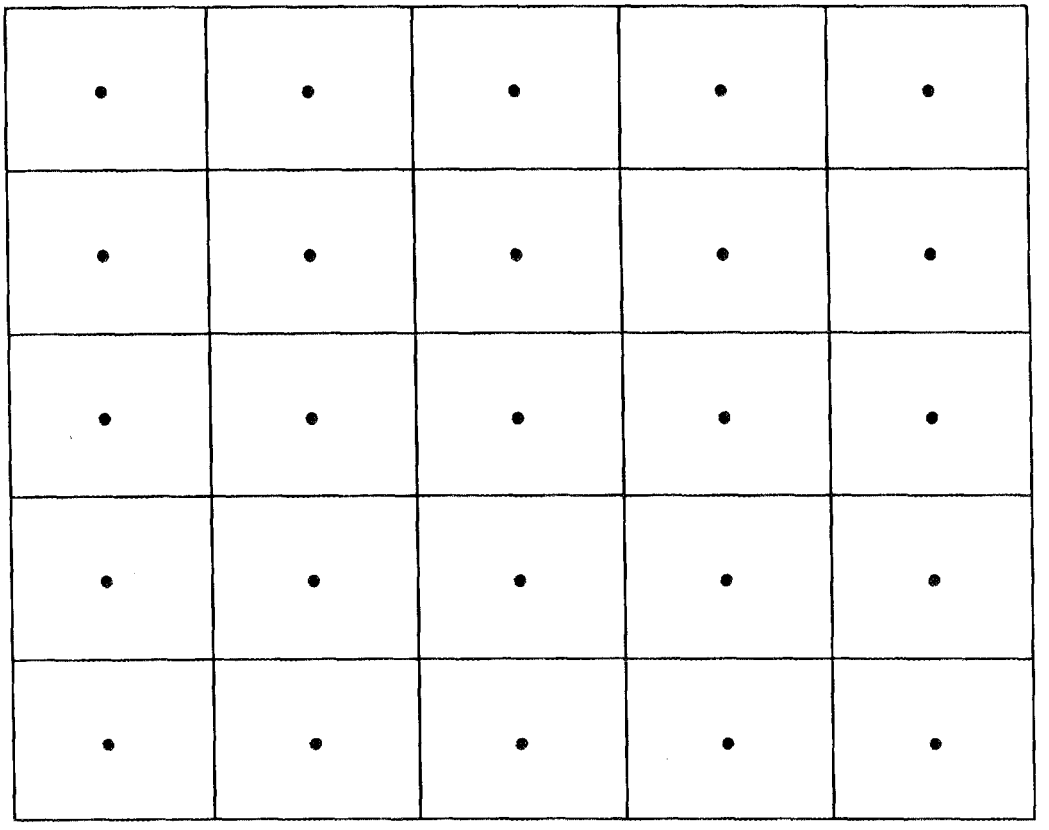

Figure 3. Cross Section of a $5 \times 5 \times 5$ grid.

B.2 Finite Difference Approximation to the Partial Differential Equation

We discretize the partial differential equation (A.7) by first discretizing the conservation of energy relation (A.5). The time derivative and the spatial derivatives are replaced by standard forward difference quotients. The following is a discrete analog of (A.5).

$$\rho \frac{e_{ijk}^{n+1} - e_{ijk}^{n}}{\Delta t} = \frac{q_{i-\frac{1}{2},jk}^{n+\theta} - q_{i+\frac{1}{2},jk}^{n+\theta}}{\Delta x} + \frac{q_{i,j-\frac{1}{2},k}^{n+\theta} - q_{i,j+\frac{1}{2},k}^{n+\theta}}{\Delta y} + \frac{q_{ij,k-\frac{1}{2}}^{n+\theta} - q_{ij,k+\frac{1}{2}}^{n+\theta}}{\Delta z} \tag{B.1}$$

Here $\theta \in [0, 1]$ and

$$F^{n+\theta} \equiv (1-\theta) F^{n} + \theta F^{n+1} .$$

$q_{i-\frac{1}{2},jk}$ denotes the heat flux into the cell $I_i \times J_j \times K_k$ in the x-direction and $q_{i+\frac{1}{2},jk}$ denotes the heat flux out of the cell $I_i \times J_j \times K_k$ in the x-direction. Similarly, $q_{i,j-\frac{1}{2},k}$ and $q_{i,j+\frac{1}{2},k}$ denote the flux in and out of the (ijk)th cell in the y-direction and $q_{ij,k-\frac{1}{2}}$ and $q_{ij,k+\frac{1}{2}}$ denote the flux in and out of the (ijk)th cell in the z-direction. We may think of the (ijk)th equation (B.1) as representing an energy balance associated with the (ijk)th node. Thus we may speak of an equation "at a node."

All nodes, including boundary nodes, are centered in a cell with dimensions $\Delta x \times \Delta y \times \Delta z$. Thus equation (B.1) holds at both interior and boundary nodes.

We next discretize Fourier's Law using centered difference quotients as follows:

$$q_{i-\frac{1}{2},jk} \approx k_{i-\frac{1}{2},jk} \left(\frac{T_{i-1,jk} - T_{ijk}}{\Delta x} \right)$$

$$q_{i+\frac{1}{2},jk} \approx k_{i+\frac{1}{2},jk} \left(\frac{T_{ijk} - T_{i+1,jk}}{\Delta x} \right) .$$

We take $k_{i\pm\frac{1}{2},jk}$ to be

$$k_{i\pm\frac{1}{2},jk} = \frac{k(T_{ijk}) + k(T_{i\pm1,jk})}{2} .$$

Similar expressions hold for $q_{i,j-\frac{1}{2},k}$, $q_{i,j+\frac{1}{2},k}$, $q_{ij,k-\frac{1}{2}}$, and $q_{ij,k+\frac{1}{2}}$.

Some of these expressions for flux are not defined at boundary nodes. At west boundary nodes, nodes with $i = 1$, $q_{i-\frac{1}{2},jk}$ is not defined in this way since T_{0jk} is not defined. Similar statements apply at other boundary nodes. Expressions for flux at boundary nodes will be discussed below. Substituting these expressions for flux into (B.1), expanding and rearranging terms gives the following discrete equation at each interior node.

$$
\begin{aligned}
e_{ijk}^{n+1} = {} & \frac{\theta \Delta t}{\rho \Delta x^2} [k_{i+\frac{1}{2},jk}^{n+1}(T_{i+1,jk}^{n+1} - T_{ijk}^{n+1}) - k_{i-\frac{1}{2},jk}^{n+1}(T_{ijk}^{n+1} - T_{i-1,jk}^{n+1})] \\
& + \frac{\theta \Delta t}{\rho \Delta y^2} [k_{i,j+\frac{1}{2},k}^{n+1}(T_{i,j+1,k}^{n+1} - T_{ijk}^{n+1}) - k_{i,j-\frac{1}{2},k}^{n+1}(T_{ijk}^{n+1} - T_{i,j-1,k}^{n+1})] \\
& + \frac{\theta \Delta t}{\rho \Delta z^2} [k_{ij,k+\frac{1}{2}}^{n+1}(T_{ij,k+1}^{n+1} - T_{ijk}^{n+1}) - k_{ij,k-\frac{1}{2}}^{n+1}(T_{ijk}^{n+1} - T_{ij,k-1}^{n+1})] \\
& + b_{ijk}^{n} ,
\end{aligned}
\tag{B.2}
$$

where

$$
\begin{aligned}
b_{ijk}^{n} = {} & e_{ijk}^{n} + \frac{(1-\theta)\Delta t}{\rho \Delta x^2} [k_{i+\frac{1}{2},jk}^{n}(T_{i+1,jk}^{n} - T_{ijk}^{n}) - k_{i-\frac{1}{2},jk}^{n}(T_{ijk}^{n} - T_{i-1,jk}^{n})] \\
& + \frac{(1-\theta)\Delta t}{\rho \Delta y^2} [k_{i,j+\frac{1}{2},k}^{n}(T_{i,j+1,k}^{n} - T_{ijk}^{n}) - k_{i,j-\frac{1}{2},k}^{n}(T_{ijk}^{n} - T_{i,j-1,k}^{n})] \\
& + \frac{(1-\theta)\Delta t}{\rho \Delta z^2} [k_{ij,k+\frac{1}{2}}^{n}(T_{ij,k+1}^{n} - T_{ijk}^{n}) - k_{ij,k-\frac{1}{2}}^{n}(T_{ijk}^{n} - T_{ij,k-1}^{n})] .
\end{aligned}
\tag{B.3}
$$

At boundary nodes, equivalent fluxes must be calculated from the boundary conditions. If the incoming flux is specified on the west wall, $q = q_{bdy}(0, y, z, t)$, then $q_{i-\frac{1}{2},jk}^{n} \equiv q_{bdy}(0, (j-\frac{1}{2})\Delta y, (k-\frac{1}{2})\Delta z, n\Delta t)$. Flux boundary conditions on other walls are treated similarly.

If temperature is specified on the west wall, $T = T_{bdy}(0, y, z, t)$, then we approximate $q_{i-\frac{1}{2},jk}^{n}$ as follows:

$$q^n_{i-\frac{1}{2},jk} \approx \left[\frac{k(T^n_{ijk}) + k(T^n_{0jk})}{2}\right]\left[\frac{T^n_{0jk} - T^n_{ijk}}{\Delta x / 2}\right].$$

Here T^n_{0jk} denotes the boundary temperature, $T^n_{0jk} \equiv T_{bdy}(0, (j-\frac{1}{2})\Delta y, (k-\frac{1}{2})\Delta z, n\Delta t)$. $(\Delta x / 2)$ appears in the denominator instead of Δx because the distance between the boundary and the boundary node is $(\Delta x / 2)$. Temperature boundary conditions on other walls are treated similarly.

Equation (B.2) may be rewritten in a different form. For compactness, define

$$\begin{aligned}\xi^{n+1}_{ijk} \equiv\ & \frac{\theta\Delta t}{\rho\Delta x^2}(k^{n+1}_{i+\frac{1}{2},jk} + k^{n+1}_{i-\frac{1}{2},jk}) \\ &+ \frac{\theta\Delta t}{\rho\Delta y^2}(k^{n+1}_{i,j+\frac{1}{2},k} + k^{n+1}_{i,j-\frac{1}{2},k}) \\ &+ \frac{\theta\Delta t}{\rho\Delta z^2}(k^{n+1}_{ij,k+\frac{1}{2}} + k^{n+1}_{ij,k-\frac{1}{2}}),\end{aligned}$$

and

$$\begin{aligned}\zeta^{n+1}_{ijk} \equiv\ & b^n_{ijk} + \frac{\theta\Delta t}{\rho\Delta x^2}(k^{n+1}_{i+\frac{1}{2},jk}\, T^{n+1}_{i+1,jk} + k^{n+1}_{i-\frac{1}{2},jk}\, T^{n+1}_{i-1,jk}) \\ &+ \frac{\theta\Delta t}{\rho\Delta y^2}(k^{n+1}_{i,j+\frac{1}{2},k}\, T^{n+1}_{i,j+1,k} + k^{n+1}_{i,j-\frac{1}{2},k}\, T^{n+1}_{i,j-1,k}) \\ &+ \frac{\theta\Delta t}{\rho\Delta z^2}(k^{n+1}_{ij,k+\frac{1}{2}}\, T^{n+1}_{ij,k+1} + k^{n+1}_{ij,k-\frac{1}{2}}\, T^{n+1}_{ij,k-1}).\end{aligned}$$

Then, equation (B.2) can be written as

$$e^{n+1}_{ijk} + \xi^{n+1}_{ijk}\, T^{n+1}_{ijk} = \zeta^{n+1}_{ijk}.$$

Again, appropriate modifications must be made to ξ and ζ at boundary nodes.

Appendix C. Successive Overrelaxation Algorithm

C. M. Elliott and J. R. Ockendon [1] suggest using successive overrelaxation (SOR) to solve simultaneously for T_{ijk}^{n+1} and e_{ijk}^{n+1}. Let τ_{ijk}^{p} denote the pth iterate of T_{ijk}^{n+1} and for compactness, define

$$z_{ijk}^{p} = b_{ijk}^{n} + \frac{\theta \Delta t}{\rho \Delta x^2} \left(k_{i+\frac{1}{2},jk}^{p} \, \tau_{i+1,jk}^{p} + k_{i-\frac{1}{2},jk}^{p+1} \, \tau_{i-1,jk}^{p+1} \right)$$
$$+ \frac{\theta \Delta t}{\rho \Delta y^2} \left(k_{i,j+\frac{1}{2},k}^{p} \, \tau_{i,j+1,k}^{p} + k_{i,j-\frac{1}{2},k}^{p+1} \, \tau_{i,j-1,k}^{p+1} \right)$$
$$+ \frac{\theta \Delta t}{\rho \Delta z^2} \left(k_{ij,k+\frac{1}{2}}^{p} \, \tau_{ij,k+1}^{p} + k_{ij,k-\frac{1}{2}}^{p+1} \, \tau_{ij,k-1}^{p+1} \right) ,$$

and

$$c_{ijk}^{p} = \frac{\theta \Delta t}{\rho \Delta x^2} \left(k_{i+\frac{1}{2},jk}^{p} + k_{i-\frac{1}{2},jk}^{p+1} \right) + \frac{\theta \Delta t}{\rho \Delta y^2} \left(k_{i,j+\frac{1}{2},k}^{p} + k_{i,j-\frac{1}{2},k}^{p+1} \right)$$
$$+ \frac{\theta \Delta t}{\rho \Delta z^2} \left(k_{ij,k+\frac{1}{2}}^{p} + k_{ij,k-\frac{1}{2}}^{p+1} \right) .$$

Note that for a given node, temperature and thermal conductivity values are lagged one iteration at advanced points, but not at previous points. This constitutes a Gauss-Seidel iteration. Our iterative form of the equation to be solved is

$$e_{ijk}^{p+1} + c_{ijk}^{p} \, \tau_{ijk}^{p+1} = z_{ijk}^{p} . \tag{C.1}$$

Starting with an initial approximation to T_{ijk}^{n+1}, τ_{ijk}^{0}, a candidate temperature, $\tilde{\tau}_{ijk}^{p}$ is computed for successive p's from

$$\tilde{\tau}_{ijk}^{p+1} = \begin{cases} \dfrac{c_{sol} \, T_{cr} + z_{ijk}^{p}}{c_{ijk}^{p} + c_{sol}} & \text{if } z_{ijk}^{p} - c_{ijk}^{p} \, T_{cr} \leqslant 0 , \\ T_{cr} & \text{if } 0 < z_{ijk}^{p} - c_{ijk}^{p} \, T_{cr} < H , \\ \dfrac{c_{liq} \, T_{cr} + z_{ijk}^{p}}{c_{ijk}^{p} + c_{liq}} & \text{if } z_{ijk}^{p} - c_{ijk}^{p} \, T_{cr} \geqslant H . \end{cases} \tag{C.2}$$

This assignment will be explained below.

Another candidate temperature, $\hat{\tau}$, is calculated using SOR from

$$\hat{\tau}_{ijk}^{p+1} = \tau_{ijk}^{p} + \omega(\tilde{\tau}_{ijk}^{p+1} - \tau_{ijk}^{p}), \tag{C.3}$$

where $\omega \in [1, 2)$ is the relaxation parameter. A discussion of the optimal choice for ω can be found in [1].

The SOR temperature, $\hat{\tau}_{ijk}^{p+1}$, is chosen if $\hat{\tau}_{ijk}^{p+1}$ and τ_{ijk}^{p} both lie on the same side of T_{cr}; the Gauss-Seidel temperature, $\tilde{\tau}_{ijk}^{p+1}$, is chosen otherwise. Precisely stated

$$\tau_{ijk}^{p+1} = \begin{cases} \tilde{\tau}_{ijk}^{p+1}, & \text{if } (\tau_{ijk}^{p} - T_{cr})(\hat{\tau}_{ijk}^{p+1} - T_{cr}) \leqslant 0, \\ \hat{\tau}_{ijk}^{p+1}, & \text{if } (\tau_{ijk}^{p} - T_{cr})(\hat{\tau}_{ijk}^{p+1} - T_{cr}) > 0. \end{cases}$$

This choice avoids oscillation about T_{cr}. Once the iteration has converged to a sufficient tolerance, after $p = R$ iterations say, $\|\tau^{R} - \tau^{R-1}\|_{\infty}$ is less than ϵ and we set $T_{ijk}^{n+1} = \tau_{ijk}^{R}$ and calculate e_{ijk}^{n+1} using

$$e_{ijk}^{n+1} = z_{ijk}^{R-1} - c_{ijk}^{R-1} T_{ijk}^{n+1}. \tag{C.4}$$

To explain the assignment of τ_{ijk}^{p+1} in (C.2), we first define $f(s) \equiv z_{ijk}^{p} - c_{ijk}^{p} s$ and note from (C.1) that $f(\tau_{ijk}^{p+1})$ will be the $(p+1)$th approximation to the enthalpy. It is easy to show that if $f(T_{cr}) \leqslant 0$, then $f(\tau_{ijk}^{p+1}) \leqslant 0$; if $0 < f(T_{cr}) < H$, then $0 < f(\tau_{ijk}^{p+1}) < H$ and if $f(T_{cr}) \geqslant H$, then $f(\tau_{ijk}^{p+1}) \geqslant H$. Consider the first branch of (C.2). Since $f(T_{cr}) \leqslant 0$, we have $f(\tau_{ijk}^{p+1}) \leqslant 0$ and the enthalpy/temperature relationship is

$$e_{ijk}^{p+1} = c_{sol}(\tilde{\tau}_{ijk}^{p+1} - T_{cr}).$$

Solving this and (C.1) for $\tilde{\tau}_{ijk}^{p+1}$ yields

$$\tilde{\tau}_{ijk}^{p+1} = \frac{c_{sol} T_{cr} + z_{ijk}^{p}}{c_{ijk}^{p} + c_{sol}}.$$

The assignments of the second and third branches in (C.2) can be justified similarly.

DISTRIBUTION OF ORNL-6393

1. V. Alexiades
2. J. B. Drake
3. G. A. Geist
4. J. A. George
5. L. J. Gray
6.-7. R. F. Harbison/ Math. Sciences Lib.
8. M. T. Heath
9. L. Horton
10.-14. J. K. Ingersoll
15. W. F. Lawkins
16.-20. F. C. Maienschein
21. T. J. Mitchell
22. M. D. Morris
23. E. Ng
24. G. Ostoruchov
25. C. H. Romine
26. R. L. Schmoyer
27. A. D. Solomon
28. S. Thompson
29.-33. R. C. Ward
34.-39. M. A. Williams
40.-44. D. G. Wilson
45. P. Worley
46. T. Wright
47. A. Zucker
48. Central Research Library
49. K-25 Plant Library
50. ORNL Patent Office
51. Y-12 Technical Library Document Reference Section
52. Laboratory Records - RC
53.-54. Laboratory Records Department
55. P. W. Dickson, Jr. (Consultant)
56. G. H. Golub (Consultant)
57. R. M. Haralick (Consultant)
58. D. Steiner (Consultant)

EXTERNAL DISTRIBUTION

59. Dr. Donald M. Austin, ER-7, Applied Mathematical Sciences, Scientific Computing Staff, Office of Energy Research, Office G-437, Germantown Washington, DC 20545

60. Dr. George I. Bell, T-7 Division, Los Alamos National Laboratory, P.O. Box 1663, Los Alamos, New Mexico 87545

61. Professor John Bruch, Department of Mechanical and Environmental Engineering, University of California at Santa Barbara, Santa Barbara, CA 93106

62. Professor Alfredo Bermúdez de Castro, Universidad De Santiago, Dep. de Ecuaciones Funcionales, Facultad de Matematicas, Santiago de Compostela, SPAIN

63. Dr. Alexandre Chorin, Mathematics Department, Lawrence Berkeley Laboratory Berkeley, California 94720

64. Dr. Sam Coriell, U. S. National Bureau of Standards, Mathematical Analysis Section, Washington, DC 20234

65. Dr. James Corones, Ames Laboratory, Iowa State University, Ames, IA 50011

66. Professor Charles M. Elliot, Mathematics Department, University of Sussex, Brighton, UNITED KINGDOM

67. Professor Antonio Fasano, Istituto Matematico Ulisse Dini, V. Le Morgagni 67/A, 50134 Firenze, ITALY

68. Professor Geoffrey C. Fox, Associate Provost for Computing, California Institute of Technology, Pasdena, California 91125

69. Professor Michael Fremond, Laboratoire Central des Pontes et Chaussees, Direction Scientific, 58 Boulevard Lefebvre, F-75732 Paris Cedex 15, FRANCE

70. Professor Dr. Karl-Heinz Hoffman, Angewandte Mathematic I, Universität Augsburg, Memminger Strasse 6, D-8900 Augsburg, WEST GERMANY

71. Dr. Robert Huddleston, Computation Department, Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, California 94550

72. Professor Malvyn Kalos, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, New York 10012

73. Professor David J. Kuck, Center for Supercomputing R&O, University of Illinois, 1384 W. SPringfield Avenue, Urbana, IL 61801

74. Dr. Andrew Lacey, Department of Mathematics, Heriot Watt University, Edinberg EH144AS, Ecosse, UNITED KINGDOM

75. Professor Peter D. Lax, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012

76. Dr. Alex Lehoczky, ES72 Space Science Laboratory, Marshall Space Flight Center, Huntsville, Alabama 35812

77. Professor Marek Niezgodka, Systems Research Institute, ul Newelska 6, 01-447 Warszawa, POLAND

78. Professor John R. Ockendon, Oxford University Computing Center, 24 St. Giles, Oxford OX1 3LB, UNITED KINGDOM

79. Professor Mario Primiccerio, Inst. Matematico Ulisse Dini, Via Morgagni 67/A, I-50134 Firenze, ITALY

80. Professor José Francisco Rodrigues, C.M.A.F., Avenida Prof Gama Pinto 2, 1699 Lisboa, PORTUGAL

81. Professor Subrata Sengupta, Department of Mechanical Engineering, University of Miami at Coral Gables, Coral Gables, FL 33124

82. Professor Dr. Jürgen Sprekels, Angewandte Mathematic I, Universität Augsburg, Memminger Strasse 6, D-8900 Augsburg, WEST GERMANY

83. Dr. Frank Szofran, ES72 Space Science Laboratory, Marshall Space Flight Center, Huntsville, AL 35812

84. Dr. Karl Hermann Tacke, Concast Service Union AG, Toedistrasse 7, 8027 Zurich, SWITZERLAND

85. Professor Alan B. Tayler, Oxford University Computing Center, 19 Parks Road, Oxford OX13 P1, UNITED KINGDOM

86. Dr. Samuel L. Thompson, Computational Physics and Mechanics, Sandia National Laboratory, Albuquerque, New Mexico 87185

87. Professor Rohit Trivedi, Ames Laboratory, Iowa State University, Ames, Iowa 50011

88. Professor Augustor Visintin, C. N. R. Instituto Analisis Numerica, Corso Carlo Alberto 5, 27100 Pavia, ITALY

89. Dr. Vaughan R. Voller, Mineral Resources Research Center, University of Minnesota, 56 East River Road, Minneapolis, Minnesota 55455

90. Professor Robert E. White, Department of Mathematics, North Carolina Stat University, Raleigh, North Carolina 27650

91. Dr. Gene Woodcock, Data Systems Division, International Business Machines, Neighborhood Road, Kingston, New York 12401

92. Office of Assistant Manager for Energy Research and Development, U. S. Department of Energy, Oak Ridge Operations Office, Oak Ridge, TN 37830

93-123. Technical Information Center.