3-D TEMPERATURE DISTRIBUTIONS IN SPOT HEATING OF A CERAMIC BY A FOCUSED MICROWAVE SOURCE

by

Rollin M. Roos

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Mechanical Engineering

APPROVED:

J.R. Thomas, Chairman

E.P. Scott

C.H. Stern

March, 1996
Blacksburg, Virginia

Key Words: Ceramic, Microwave, Joining
3-D TEMPERATURE DISTRIBUTIONS IN SPOT HEATING OF A CERAMIC BY A FOCUSED MICROWAVE SOURCE

by

Rollin M. Roos

J.R. Thomas, Chairman

Mechanical Engineering

(ABSTRACT)

Recent work has described microwave applicators which produce localized spot heating of a workpiece with a Gaussian spatial distribution of microwave power. Applicators of this type provide a nearly ideal heating pattern for joining of ceramics and other applications requiring localized heating, and offer several advantages over conventional methods utilizing surface heat flux.

A three-dimensional mathematical model of a circular volumetric heat source moving through a rectangular block was constructed to address some of the microwave heating concerns—electric field power requirements, uniform heating, and thermal runaway. The partial differential equation from the mathematical model was solved numerically with an implicit finite-difference method. Using experimentally measured dielectric loss properties for alumina, the required electric field strengths to raise the temperature of a localized spot in a rectangular block to 1500K are presented. With the block moving on
an insulating support, the results also show that uniform temperature profiles through the depth of the alumina can be achieved, and thermal runaway can be prevented by choosing an appropriate block velocity and electric field strength. With the block on a highly conductive support, however, the high electric field power required to raise the temperature of the block makes thermal runaway and non-uniform temperature distributions likely. For the published dielectric loss coefficient data for glass, thermal runaway could not be avoided for a stationary block. Calculations suggest that maintaining a constant absorbed power in a moving workpiece could be a way of maintaining the desired maximum temperature within the blocks. From the preceding conclusions, a microwave spot heater appears to be an effective tool for the joining of ceramics.
Acknowledgements

I would like to express my appreciation to my advisor Dr. J.R. Thomas for all of his advice and guidance during this last year and a half. I would also like to thank my committee members Dr. C.H. Stern and Dr. E.P. Scott for their help in writing this thesis.

I would like to thank Dupont for their support in the last year and the Mechanical Engineering Department for their support the previous year.

Also I would like to give a big thanks to my family and especially my sister Elizabeth.
# Table of Contents

Introduction ............................................................................................................ 1

Absorption of Microwave Energy ................................................................. 1

Industrial Applications for Microwave Energy ........................................ 3

Joining ..................................................................................................................... 4

  Reasons for Joining Ceramics ................................................................. 4
  Problems with Conventional Heating ...................................................... 5
  Advantages from Microwave Joining ....................................................... 5
  Previous Joining Work ............................................................................ 5

Spot Heating Applicators .................................................................................. 6

Objective .............................................................................................................. 7

Mathematical Model .......................................................................................... 9

  Geometry ...................................................................................................... 9

  Energy Equation ........................................................................................ 10

  Boundary Conditions ............................................................................... 11

  Heat Generation ....................................................................................... 12

  Electric Field Attenuation .................................................................... 13

Finite Difference Formulation ...................................................................... 14

  Control Volume Geometry .................................................................. 14

  Discretizing of the Energy Equation .................................................. 15

  Boundary Conditions ............................................................................ 17

  Electric Field Attenuation .................................................................. 17
List of Figures and Tables

Figure 1. Dielectric Loss Coefficient for Alumina as a function of Temperature 2
Figure 2. Geometry of the Model 9
Figure 3. Geometry of the Control Volume 14
Table 1. Values for Fixed Parameters 19
Table 2. Grid Size 20
Figure 4. Curve Fit to Dielectric Loss Coefficient for Alumina as a function of Temperature 21
Figure 5. Curve Fit to Dielectric Loss Coefficient for Fused Quartz Glass a Function of Temperature 21
Figure 6. Solution Convergence while the Timestep Decreases 22
Figure 7. Solution Convergence as the Number of Grid Points Increases 23
Figure 8. Time to Heat an Alumina Block to 1500K 24
Figure 9. Electric Field Strength vs. Time 25
Figure 10. Temperature Distribution in the X-Z Plane for $h_c$ of 100 W/m$^2$ and Field Strength of 100,000 V/m 26
Figure 11. Temperature Distribution in the X-Z Plane for $h_c$ of 1000 W/m$^2$ and Field Strength of 250,000 V/m 27
Figure 12. Temperature Distribution in Alumina in the X-Y Plane at z = h with a Stationary Sample 29
Figure 13. Temperature Distribution in Alumina in the X-Z Plane at y = w/2 with a Stationary Sample 30
Figure 14. Temperature Distribution in Alumina in the X-Y Plane at z = h with a Velocity of 0.1 cm/s 31
Figure 15. Temperature Distribution in Alumina in the X-Z Plane at \( y = w/2 \) with a Velocity of 0.1 cm/s

Figure 16. Temperature Distribution in Alumina in the X-Y Plane at \( z = h \) with a Velocity of 0.05 cm/s

Figure 17. Temperature Distribution in Alumina in the X-Z Plane at \( y = w/2 \) with a Velocity of 0.05 cm/s

Figure 18. Temperature Distribution in Alumina in the X-Y Plane at \( z = h \) with a Velocity of 0.02 cm/s

Figure 19. Temperature Distribution in Alumina in the X-Z Plane at \( y = w/2 \) with a Velocity of 0.02 cm/s

Figure 20. Temperature Distribution along the x-axis in Alumina with a Velocity of 0.5 cm/s for Various Time Steps for \( z = h \) and \( y = w/2 \)

Figure 21. Temperature Distribution in Fused Quartz Glass in the X-Y Plane at \( z = h \) with a Stationary Sample

Figure 22. Power Absorbed by the Alumina Block as the Sample Moves

viii
Introduction

Absorption of Microwave Energy

While a high frequency electric field is interacting with a material, heat generation occurs within both conducting and insulating materials, but the mechanism which causes the heat generation is different for both materials. For conducting materials, the electrons can move freely, so when an electric field is applied the electrons move through the material producing a current. The current flowing through the resistance of the material causes heat generation to occur within the material. Since microwaves are mostly reflected by conducting materials, this mechanism is not important, but microwaves can penetrate most insulating materials. For polar insulating materials, the mechanism for heat generation is the dipoles reorienting themselves to the rapidly changing electric field. As the dipoles are rotating to align with the electric field, frictional losses generate heat within the material. The closer the frequency of the electric field change is to the natural frequency at which the orientation can occur, the more the dipoles will rotate which will result in more frictional losses or heat generation.

For modelling these effects, there is a parameter which characterizes the heat generation within a material when a microwave field is applied. The measure of the ability of a dielectric to absorb and to store electrical potential energy is permittivity $\varepsilon$, which has a real and imaginary part. The real part $\varepsilon'$ characterizes the penetration of the microwaves
into the material and is temperature and frequency dependent. The imaginary part $\varepsilon''$ is a measure of the ability of the material to store the potential energy. Using $\varepsilon''$, the heat generation written as [1]

$$\dot{q}_v(x, y, z, T) = 2\pi f \varepsilon''(T) |E_j(z)|^2,$$  \hspace{1cm} [1]

where $f$ is the frequency of the microwave energy and $E_j$ is the local electric field strength. The imaginary part $\varepsilon''$ is also temperature and frequency dependent. For example, Fig. 1 shows the temperature dependence for alumina which was obtained experimentally[2,13].

![Graph showing dielectric loss coefficient for alumina as a function of temperature](image)

**Figure 1.** Dielectric Loss Coefficient for Alumina as a function of Temperature

The large increase of the dielectric loss coefficient at high temperatures, which is characteristic of metal-oxide ceramics, gives rise to the problem of thermal runaway when microwaves are used to heat the material. Thermal runaway is when the heat generation is increasing rapidly which causes the temperatures in the material to increase
uncontrollably. The uncontrollable temperature increase occurs because from Eq. 1 the heat generation is a function of the imaginary part of the dielectric loss which is an increasing function of temperature as shown in Fig 1. The temperature of the sample increases which in turn increases the heat generation, causing the temperature to increase even more. This continues until the temperatures “run away”.

Volumetric heating produced by a microwave field has advantages and disadvantages for materials processing. Since most insulating materials have low thermal conductivity, conventional surface heating techniques are slow, so the major advantage to volumetric heating is a lower heating time. Another advantage is that energy is conserved because only the workpiece is being heated. The major disadvantage is the non-uniform field strength which leads to non-uniform heating of the sample, and this non-uniform heating produces hot spots which could lead to localized thermal runaway. Without a load in microwave cavities, the non-uniformity in field strength occurs from the field pattern within the cavity, but adding a load to the cavity will distort the field pattern even more.

Industrial Applications for Microwave Energy

Current research in industrial applications of microwave heating includes sintering, combustion synthesis, and joining of ceramics. Sintering of ceramics is being researched because more uniform microstructures are obtained and the heating process takes minutes
instead of hours to complete.[3] For joining, the heating process is again faster, and uniform temperatures can be achieved along the joint while heating.[3] Combustion synthesis of ceramics produces powders, carbides, nitrides, and oxides. For this process, the advantage of microwave heating is that the combustion begins at the center of the sample.[3]

Other work includes the nucleation and crystallization of glass. Crystallization for a simple lithium disilicate glass can occur at a more rapid rate.[3] Significant crystallization occurs at temperatures which are lower than reported in the literature for conventional heating.[3]

Joining

Reasons for Joining Ceramics

Ceramics are hard and brittle and this can cause manufacturing problems, such as the expense of machining and difficulty of fabricating near-net-shape components. For these reasons, large ceramics may be manufactured by joining smaller ceramics.

Applications for joining include attachment of components such as piston liners, combustor cans, and turbine rotors. The fabrication of tube assemblies for high
temperature and high-pressure heat exchangers and high temperature and corrosion-resistant radiant burners can also be done with joining. [4]

Problems with Conventional Heating

Current studies show that the strength of the joint produced by conventional methods at high temperatures is low and the reason for this is unknown. [4] Since the thermal conductivity of most insulating materials is low, non-uniform temperatures along the joint can result. This leads to thermal stresses. External heating methods are also slow and inefficient.

Advantages from Microwave Joining

High joint strength, which is sometimes equal to that of the original material, can be obtained with microwave heating [5]. The reasons for the high joint strength is still unknown, and is being studied. The dielectric loss coefficient can be four times larger at high temperatures--800 K to 1500 K--than low temperatures--300 K to 800 K--., and this large coefficient results in more efficient heating. Due to the volumetric heating, uniform temperatures across the joint can potentially be achieved.

Previous Joining Work

Recent work in microwave joining includes Silberglitt’s [6] study which demonstrated the feasibility of microwave joining for fabrication of SSiC-RBSiC tube assemblies for
advanced heat exchangers. For the joining of alumina, Al-Assaf and Clark [7] joined alumina with itself and alumina gel, and Fukushima [5] produced alumina rod joints which had the same bending strength as the as-received rods. Also Fukushima [5] produced joints with Silicon nitride which had 70% of the original strength of the as-received material.

Spot Heating Applicators

Two types of microwave applicators have recently been developed which produce localized spot heating of a workpiece with a Gaussian spatial distribution of microwave power. This localized heating is ideal for joining ceramics.

The quasi-optical focusing gyrotron, which was developed by E.O. Paton Electric Welding Institute in Kiev, Ukraine, can focus a Gaussian beam of centimeter to millimeter wavelength with microwave power up to 200kW[8]. The centimeter to millimeter wavelength is a result of the frequency range--30 to 300 Ghz--and the beam can be focused in several shapes--circular spot, circle, and line. Centimeter wavelengths are absorbed better by some materials, which results in more rapid heating. For example, with the gyrotron one cubic centimeter of alumina could be heated to its melting point in seconds[8].
Another applicator is the open coaxial microwave spot joining applicator developed by Tinga[9] at the University of Alberta, which can focus a Gaussian distribution of microwave energy in a material. The applicator only produces two frequencies--915 and 2450 MHz--and uses a magnetron source. The beam produced is in two spot shapes--circular and rectangular.[9] The circular spot has a radius range of 4 to 12 mm. Unlike the gyrotron, where the beam is projected into the material, the coaxial applicator has to be placed above the material to obtain a beam of microwave energy. One advantage to this is that the material does not have to be fed into an enclosure for heating, which is the case for the gyrotron.

Objective

The objective of the work reported here is to determine the potential for spot heating applicators in materials processing. The feasibility of using these applicators to join ceramics will be the major focus.

This thesis reports development of a 3-D mathematical model of the heat transfer process for a block of material being heated by a focused spot of microwave energy. The model will address some important concerns--electric field strength requirements, uniform heating, and thermal runaway--in the microwave spot heating of materials. For joining in
particular, the concerns are maintaining the joining temperature while the material is moving and a uniform temperature distribution along the joint always.
Mathematical Model

Geometry

The geometry and coordinate system of the model is shown in Fig. 2.

![Figure 2: Geometry of the Model](image)

Moving at velocity $u$, a rectangular block of material is heated by a stationary localized circular spot of microwave energy. The length along the $x$, $y$, and $z$-direction are $l$, $w$, and $h$, respectively. The $z$-direction will be referred to as the depth or thickness of the material.

This model is constructed to simulate the butt-joining of two materials. When joining two materials, one important issue is maintaining the joining temperature along the joint while the material is moved. The other issue is to maintain a uniform temperature
distribution through the depth of the joint. To address these issues, only a single slab of material needs to be modeled. For this model, the joint could be thought of as being at \( y = w/2 \).

For the model, the following assumptions were made:

1. Temperature dependence of all material properties
2. A contact conductance between the supporting material and the sample
3. Radiation and convection heat losses off the top and sides of the block
4. Radiation from the room to the sample was neglected
5. The field strength is attenuated through the depth of the sample
6. The field has a Gaussian distribution in the x-y plane

Energy Equation

We consider a rectangular block of material moving in the positive x-direction at a velocity \( u \) with a circular heat generation. The governing energy equation for this model is [10]

\[
\nabla \cdot (k\nabla T) - \rho C_p u \frac{\partial T}{\partial x} + \dot{q}(x, y, z, T) = \rho C_p \frac{\partial T}{\partial t},
\]

where \( T \) is the absolute temperature at a location \( x, y, z \) at time \( t \), \( C_p \) is the specific heat capacity at constant pressure, \( \rho \) is the density, \( k \) is the thermal conductivity, \( \dot{u} \) is the velocity of motion of the block, and \( \dot{q}(x, y, z, T) \) is the volumetric heating source from
the microwave energy dissipated in the block. The equation describing the amount of microwave energy dissipated will be discussed in the heat generation section.

Boundary Conditions

The model was constructed assuming the block is on a table which is in a room with temperature \( T_a \). From the top and sides, a convective coefficient was used for heat loss between the block and the air. For the heat loss to the table from the bottom of the block, a contact resistance is used. Also, since the block reaches high temperatures, radiative heat losses are considered from the top and sides.

The boundary conditions used for the above energy equation are as follows,

\[
\begin{align*}
  z = 0: & \quad k \frac{\partial T}{\partial z} = h_c (T - T_\infty) \quad [3] \\
  z = h: & \quad -k \frac{\partial T}{\partial z} = h_a (T - T_\infty) + \sigma \varepsilon (T^4 - T_{ref}^4) \quad [4] \\
  x = 0: & \quad k \frac{\partial T}{\partial x} = h_a (T - T_\infty) + \sigma \varepsilon (T^4 - T_{ref}^4) \quad [5] \\
  x = l: & \quad -k \frac{\partial T}{\partial x} = h_a (T - T_\infty) + \sigma \varepsilon (T^4 - T_{ref}^4) \quad [6]
\end{align*}
\]
\[ y = 0: \quad k \frac{\partial T}{\partial y} = h_u (T - T_a) + \sigma \varepsilon (T^4 - T_{ref}^4) \quad [7] \]

\[ y = w: \quad -k \frac{\partial T}{\partial y} = h_u (T - T_a) + \sigma \varepsilon (T^4 - T_{ref}^4) \quad [8] \]

where \( h_u \) is the convective heat transfer coefficient between the block and the surroundings, \( h_c \) is the thermal contact resistance between the block and the supporting material, \( \varepsilon \) is the emissivity of the block material, \( T_a \) is the absolute temperature of the surrounding air, \( T_{ref} \) is the reference temperature of the surroundings, and \( \sigma \) is the Stefan-Boltzmann constant.

Heat Generation

Since both applicators mentioned in the introduction have Gaussian heat source distributions, the heat source distribution in the x-y plane for the model is

\[ \dot{q}(x, y, z, T) = \begin{cases} q_o(x, y, z, T) \exp \left( -\frac{x^2 + y^2}{r_o^2} \right), & x^2 + y^2 \leq r_o^2; \\ 0, & \text{elsewhere}. \end{cases} \quad [9] \]

where \( r_o \) is the outer radius of the heat source and

\[ q_o(x, y, z, T) = 2\pi \varepsilon \sigma (T) \left| E_f(z) \right|^2, \quad [1] \]
where $f$ is the frequency of the microwave energy, $\varepsilon''$ is the dielectric loss coefficient of the material, and $E_f$ is the local electric field strength.

Electric Field Attenuation

An electric field is attenuated as it penetrates a material, and the attenuation can be expressed as [1]

$$\frac{dE_z}{dz} = -\mu(z)E_z$$ \hspace{1cm} [10]

where $E_z$ is the magnitude of the electric field travelling in the $z$-direction. When the material is lossy and $\left(\frac{\sigma}{\omega\varepsilon}\right)^2 << 1$ [1]

$$\mu(z) = \frac{2\pi f}{c} \sqrt{\frac{1}{2}} \left[ 1 + \frac{\varepsilon''}{\varepsilon'} \right]$$ \hspace{1cm} [11]

where $f$ is the microwave frequency and $c$ is the speed of light. $\mu$ is a function of $z$ because $\varepsilon''$ and $\varepsilon'$ are temperature dependent and temperature varies in the $z$-direction.

Integrating Eq. 10 in the $z$-direction through the depth of the material yields

$$E_z = E_{zo} \exp\left[-\int_{h}^{0} \mu(\xi) d\xi\right]$$ \hspace{1cm} [12]

where $E_{zo}$ is the value of the electric field at the surface.
Finite Difference Formulation

Control Volume Geometry

![Figure 3: Geometry of the Control Volume](image)

From Fig. 3, the dashed line cube represents a control volume which is $\Delta x \Delta y \Delta z$. 
Discretizing of the Energy Equation

The energy equation was solved using a fully implicit control volume finite difference scheme. The implicit method was used for stability and the control volume method is used to include the temperature dependence of the properties. The control volume method begins by integrating the energy equation over the control volume shown in Fig. 3, which yields

\[
\iiint_{cv} \int \left( \frac{\partial}{\partial x} \left( k \frac{\partial T}{\partial x} \right) \right) dx dy dz dt + \iiint_{cv} \int \left( \frac{\partial}{\partial y} \left( k \frac{\partial T}{\partial y} \right) \right) dx dy dz dt \\
+ \iiint_{cv} \int \left( \frac{\partial}{\partial z} \left( k \frac{\partial T}{\partial z} \right) \right) dx dy dz dt - u \iiint_{cv} \int \rho C_p \frac{\partial T}{\partial x} dx dy dz dt \\
+ \iiint_{cv} \int \rho (x, y, z, T) dx dy dz dt = \iiint_{cv} \int \rho C_p \frac{\partial T}{\partial t} dx dy dz dt . 
\]

Performing the integration on each conduction term gives, for example,

\[
\iiint_{cv} \int \left( \frac{\partial}{\partial x} \left( k \frac{\partial T}{\partial x} \right) \right) dx dy dz dt = \left[ \left( k \frac{\partial T}{\partial x} \right)_{m+}^{(P+1)} - \left( k \frac{\partial T}{\partial x} \right)_{m-}^{(P+1)} \right] \Delta y \Delta z \Delta t , 
\]

where the partial derivative terms can be approximated as

\[
k \frac{\partial T}{\partial x}_{m+}^{(P+1)} = k_{m+}^{(P+1)} \left( \frac{T_{m+1,n,l}^{(P+1)} - T_{m,n,l}^{(P+1)}}{\Delta x} \right) , 
\]

where \( T_{m+1,n,l}^{(P+1)} \) is the temperature at the advanced timestep \( t_{P+1} \). Since \( k \) is a function of \( T \), we define \( k_{m+} \) as

\[
k_{m+}^{(P+1)} = k \left( \frac{T_{m,n,l}^{(P+1)} + T_{m+1,n,l}^{(P+1)}}{2} \right) . 
\]
Integrating the remaining terms, we get

\[ u \iint \int_{r \in \mathcal{V}} \rho \frac{\partial T}{\partial x} \, dxdydz \, dt = u \left[ \left( \rho C_p T \right)_{m}^{(p+1)} - \left( \rho C_p T \right)_{m-1}^{(p+1)} \right] \Delta y \Delta z \Delta t \]  \quad \text{[17]}

\[ \iint \int_{r \in \mathcal{V}} q(x, y, z, T) \, dxdydz \, dt \equiv \dot{q}(x, y, z, T) \Delta x \Delta y \Delta z \Delta t \] \quad \text{[18]}

and

\[ \iint \int_{r \in \mathcal{V}} \rho \frac{T_0}{\partial t} \, dxdydz \, dt = \left( \rho C_p \right)^{(p+1)} \left[ T_{m,n,l}^{(p+1)} - T_{m,n,l}^{(p)} \right] \Delta x \Delta y \Delta z \] \quad \text{[19]}

For the convective term, an upwind difference scheme was used for stability in the solution. Substituting back into Eq. 13, we have

\[ \left( \rho C_p \right)^{(p+1)} \left( \frac{T_{m,n,l}^{(p+1)} - T_{m,n,l}^{(p)}}{\Delta t} \right) \Delta V = \left[ \left( k_{n+} \right)^{(p+1)} \left( \frac{T_{m+1,n,l}^{(p+1)} - T_{m,n,l}^{(p+1)}}{\Delta x} \right) - k_{n-} \left( \frac{T_{m,n,l}^{(p+1)} - T_{m,n,l-1}^{(p+1)}}{\Delta x} \right) \right] \Delta y \Delta z \]

\[ + \left[ k_{n+} \left( \frac{T_{m,n+1,l}^{(p+1)} - T_{m,n,l}^{(p+1)}}{\Delta y} \right) - k_{n-} \left( \frac{T_{m,n,l}^{(p+1)} - T_{m,n,l-1}^{(p+1)}}{\Delta y} \right) \right] \Delta x \Delta z \]

\[ + \left[ k_{l+} \left( \frac{T_{m,n,l+1}^{(p+1)} - T_{m,n,l}^{(p+1)}}{\Delta z} \right) - k_{l-} \left( \frac{T_{m,n,l}^{(p+1)} - T_{m,n,l-1}^{(p+1)}}{\Delta z} \right) \right] \Delta x \Delta y \]

\[-u \left[ \left( \rho C_p T \right)_{m,n,l}^{(p+1)} - \left( \rho C_p T \right)_{m-1,n,l}^{(p+1)} \right] \Delta y \Delta z \]

\[ + \dot{q}(x, y, z, T) \Delta x \Delta y \Delta z \] \quad \text{[20]}

where all the \( T^{(p+1)} \)'s are unknown temperatures at any given time \( t_p \).
Boundary Conditions

The first derivative for the boundary conditions is represented with a second-order accurate backward-difference approximation [10],

\[
 k \frac{\partial T}{\partial z} = k_{i-1}^{(p+1)} \left( \frac{-3T_{n,m,l}^{(p+1)} + 4T_{n,m,l-1}^{(p+1)} - T_{n,m,l-2}^{(p+1)}}{2\Delta z} \right), \tag{21}
\]

where \( k \) is now evaluated as

\[
 k_{i-1}^{(p+1)} = k(T_{n,m,l-1}^{(p+1)}). \tag{22}
\]

Substituting these approximations into Eq. 4 we get

\[
 -k_{i-1}^{(p+1)} \left( \frac{-3T_{n,m,l}^{(p+1)} + 4T_{n,m,l-1}^{(p+1)} - T_{n,m,l-2}^{(p+1)}}{2\Delta z} \right) \Delta x \Delta y
 = h_a \Delta x \Delta y \left( T_{m,n,l}^{(p+1)} - T_\infty \right) + \Delta x \Delta y \varepsilon \sigma \left[ \left( T_{m,n,l}^{(p+1)} \right)^4 - T_{ref}^4 \right]. \tag{23}
\]

Electric Field Attenuation

From the mathematical model section, Eq. 10 can be numerically integrated, which yields

\[
 E_z = E_{z_0} \exp \left[ -\sum_{i=0}^{n} \mu(T_i) \Delta z \right], \tag{24}
\]

where \( n \) is the total number of nodes in the \( z \)-direction.
Solving for the Temperature Distribution

The above discretization of the energy equation leads to n equations with n unknowns, which can be written in matrix form as

\[ A(T^{(P+1)}) \underline{T}_{m,n,j}^{(P+1)} = \underline{Q}^{(P+1)}(T^{(P+1)}) + \underline{T}_{m,n,j}^{(P)} \]  \hspace{1cm} [25]

where \( A \) is an nxn matrix that is a function of \( T^{(P+1)} \), \( \underline{T}_{m,n,j}^{(P+1)} \) is the vector of the unknown temperatures, \( \underline{Q}^{(P+1)} \) is the heat source vector which is also a function of \( T_{m,n,j}^{(P+1)} \), and \( \underline{T}_{m,n,j}^{(P)} \) is the vector of temperatures from the previous time step. To solve this system of equations, first the A matrix can be inverted and multiplied through Eq. 25

\[ \underline{T}_{m,n,j}^{(P+1)} = A^{-1}(T^{(P+1)})^{-1} \underline{Q}^{(P+1)}(T^{(P+1)}) + A^{-1}(T^{(P+1)})^{-1} \underline{T}_{m,n,j}^{(P)} \]  \hspace{1cm} [26]

These new temperatures are then used to construct a new A matrix, and new temperatures are solved for again. This process continues until the new temperatures converge. The process is then repeated for the next time step. The iterations within each time step are needed to account for the non-linearity of the system which comes from the temperature-dependent properties.

The C code (See Appendix) written to solve the system of linear equations for the unknown temperatures uses a sparse matrix routine obtained from the World Wide Web. \textit{Sparse 1.3} is copyrighted by the authors--Kenneth S. Kundert and Alberto Saniovanni-Vincentelli--and the University of California, Berkeley.
Results

Fixed Parameters

For all results given, some variables were held fixed. Their values are given in Table 1.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Alumina</th>
<th>Glass</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length (x-dir)</td>
<td>4 cm</td>
<td>4 cm</td>
</tr>
<tr>
<td>Width (y-dir)</td>
<td>2 cm</td>
<td>2 cm</td>
</tr>
<tr>
<td>Beam Diameter (2 r₀)</td>
<td>1 cm</td>
<td>1 cm</td>
</tr>
<tr>
<td>Frequency</td>
<td>2.45 GHz</td>
<td>2.45 GHz</td>
</tr>
<tr>
<td>Emissivity</td>
<td>0.6</td>
<td>0.9</td>
</tr>
<tr>
<td>hₜ</td>
<td>10 W/m²K</td>
<td>10 W/m²K</td>
</tr>
<tr>
<td>Tₙ</td>
<td>300K</td>
<td>300K</td>
</tr>
<tr>
<td>Tₙref</td>
<td>0K</td>
<td>0K</td>
</tr>
</tbody>
</table>

The short length and width were chosen to obtain a fine mesh (small delta’s) with a small number of grid points. A sufficiently small number of grid points to keep computation time on the order of hours. The frequency and beam diameter values correspond to those given [9] for the coaxial microwave spot joining applicator. The convective coefficient \( hₜ \) was calculated from the free convection correlations to be the value given in Table 1 even at high temperatures. The emissivities were obtained from a standard reference [11]. The reference temperature \( T_{ref} \) being 0 K is an overestimate of the radiation loss, but when calculations were performed using \( T_{ref} \) equal to 300 K, the
amount of time to heat the sample and the temperature distribution within the sample
differed by less than 5%.

The localized circular spot of microwave energy was located at \( x = 1.5 \text{ cm} \) and \( y = w/2 \)
for both the stationary and moving material results.

The discretizing was the same for all solutions and is shown in Table 2.

<table>
<thead>
<tr>
<th>Direction</th>
<th>Number of Grid Points</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>41</td>
<td>0.1 cm</td>
</tr>
<tr>
<td>y</td>
<td>21</td>
<td>0.1 cm</td>
</tr>
<tr>
<td>z (thin)</td>
<td>4</td>
<td>0.1 cm</td>
</tr>
<tr>
<td>z (thick)</td>
<td>11</td>
<td>0.1 cm</td>
</tr>
</tbody>
</table>

Properties

The thermal properties \( C_p, k, \) and \( \rho \) for alumina, which were obtained from the
literature, and glass \([11]\) were represented by curve fits to experimental data. The
dielectric loss coefficient \( \varepsilon'' \) for alumina\([2]\) and glass\([12]\)--Type 101 clear, fused quartz,
99.97 to 99.98% \( SiO_2 \)--were also curve-fitted from experimental data and are plotted in
Figures 4 and 5, respectively.
Figure 4: Curve Fit to Dielectric Loss Coefficient for Alumina as a function of Temperature

Figure 5: Curve Fit to Dielectric Loss Coefficient for Fused Quartz Glass[12] as a Function of Temperature
For both alumina and glass, the large increase of the dielectric loss coefficient for high temperatures gives rise to the problem of thermal runaway, as described in the Introduction.

Convergence

To verify the model, the number of grid points was increased and the timestep was decreased to see if the solution converged. Figures 6 and 7 show how the solution converges as these increments are decreased.

![Graph showing temperature vs timestep for Node 1 and Node 2](image)

Figure 6: Solution Convergence while the Timestep Decreases

Node 1 and Node 2 are nodal grid points which are representative of the mesh. The convergence for the decreasing timestep is linear, which is expected because a first-order approximation was used for the time derivative.
Stationary Sample

To best model the actual joining process, first a stationary sample was heated to the joining temperature of the material. The important questions for heating the sample are the amount of time to heat to the joining temperature, the electric field strength necessary to get to the joining temperature, and the temperature distribution through the depth of the material to stay uniform while heating. All of the following temperature distributions--stationary or moving--are for discrete times and are not steady state.
Electric Field Strength Requirements

For a stationary sample, computational experiments with a contact resistance of 100 W/m^2 were performed to determine the amount of time required for a given electric field strength to give a maximum temperature of 1500K in the block of alumina. First, two runs were made to see if thickness is important, since the electric field is attenuated with depth. As discussed earlier, the electric field strength exponentially decays as it penetrates the block in the z-direction. The first and second runs were done with thicknesses of 0.3 and 1.0 cm, respectively. The electric field strength was 130,000 V/m and $h_i$ was 10 W/m^2. The results are displayed in Fig. 8.

![Graph showing time vs. thickness](image)

Figure 8: Time to Heat an Alumina Block to 1500K

The results show that the thickness does not significantly affect the amount of time it takes to heat the material to 1500K within the range considered. With a thickness of 0.3
cm (for reasons of computation time), different constant electric field strengths were used, and the amount of time was determined for the material to achieve a maximum temperature of approximately 1500K. These results are shown in Fig. 9.

![Electric Field Strength vs. Time](image)

**Figure 9: Electric Field Strength vs. Time**

To heat the sample to 1500 K in a minute, the electric field strength would have to be 165,000 V/m for an $h_c$ of 100 W/m² and 155,000 V/m for an $h_c$ of 10 W/m². For an $h_c$ of 100 W/m², the curve is approaching an infinite slope as the electric field strength decreases which means that for an electric field strength of 120,000 V/m the amount of time for the sample to reach 1500 K would approach infinity. This gives an approximation to the minimum electric field strength needed to raise the temperature of the sample to 1500 K.
Contact Resistance Effects

One obvious effect is that the electric field strength required to heat the block must increase as the contact conductance $h_c$ increases. If $h_c$ increases, then the heat loss from the block to the support increases, so more heat must be generated within the material to raise the temperature. With a stationary beam, maximum temperature of approximately 1500K, and a thickness of 1 cm, the temperature distribution in the x-z plane was determined for two $h_c$'s, 100 W/m$^2$ and 1000 W/m$^2$, and electric field strengths of 100,000 V/m and 250,000 V/m, respectively. The results are shown in Figures 10 and 11.

![Temperature Distribution](image)

Figure 10: Temperature Distribution in the X-Z Plane for $h_c$ of 100 W/m$^2$ and Field Strength of 100,000 V/m
Figure 11: Temperature Distribution in the X-Z Plane for $h_c$ of 1000 W/m$^2$ and Field Strength of 250,000 V/m

For the smaller value of $h_c$ -- 100 W/m$^2$K -- the temperature distribution through the depth of the material is uniform to within 8%, but for the larger $h_c$ -- 1000 W/m$^2$K -- the distribution is nonuniform to within 70% with a maximum temperature of 2100K and minimum of 600K. These results suggest that the block should be insulated at the bottom to obtain a uniform temperature distribution through the depth of the material. A contact conductance of 100 W/m$^2$K represents an "insulated" case relative to metal against metal [11].
Temperature Distributions for a Moving Sample

Once the joining temperature was obtained, the sample was given a step velocity to further model the joining process. The feasibility of maintaining a uniform temperature distribution through the depth of the material and a maximum temperature of 1500K while the material is moving were issues that needed to be investigated to find the potential for these applicators in joining.

Alumina

Study of these issues began by finding the temperature distribution in the stationary block after heating it until a maximum temperature of 1500K was reached. This was done to simulate the actual joining process where the sample would have to be heated to 1500K before being moved. With an electric field strength of 110,000 V/m and a contact conductance of 100 W/m^2K, the initial temperature distribution is shown in Fig. 12 for the x-y plane and Fig. 13 for the x-z plane.
Figure 12: Temperature Distribution in Alumina in the X-Y Plane at $z = h$ with a Stationary Sample
Figure 13: Temperature Distribution in Alumina in the X-Z Plane at y = w/2 with a Stationary Sample

From the x-y plane, the maximum temperature--approximately 1500K--is at 15 mm and the distribution is symmetric in the y-direction. From the x-z plane, the maximum temperature--approximately 1500K--is relatively constant to within 8% through the depth of the material showing that a uniform temperature distribution can be achieved for a stationary sample. This uniform temperature distribution is important for joining because temperature gradients along the joint can cause thermal stresses.
Using the stationary sample results as the initial temperature distribution, the block was then given a steady velocity and moved 1 cm in the x-direction. With an electric field strength of 110,000 V/m and a contact resistance of 100 W/m^2K, Figures 14 and 15 show the temperature distributions in both the x-y plane and x-z plane for the largest velocity—0.1 cm/s.

Figure 14: Temperature Distribution in Alumina in the X-Y Plane at z = h with a Velocity of 0.1 cm/s
Figure 15: Temperature Distribution in Alumina in the X-Z Plane at $y = w/2$ with a Velocity of 0.1 cm/s

In Fig. 14, the spot of microwave energy is stationary at 15 mm along the length and $w/2$ along the width (the center of the spot is shown by the cross-hairs) while the sample moves through the spot. This is also true for the rest of the x-y plane figures.

It is seen in Fig. 14 that the peak temperature moves 10 mm to a location 25 mm from the left and is now only 1200K. This shows that this velocity is too large for the microwave beam to heat the incoming colder material to 1500K. The peak temperature at 25 mm is only the peak temperature--1500K--from the starting distribution translated 1 cm and
cooled to 1200K. It is seen in Fig. 15 that the temperature distribution in the z-direction is still uniform to within 7%.

A slower velocity--0.05 cm/s--was used to see if the peak temperature of 1500K could be maintained. The results are shown in Figures 16 and 17.

Figure 16: Temperature Distribution in Alumina in the X-Y Plane at z = h with a Velocity of 0.05 cm/s
Figure 17: Temperature Distribution in Alumina in the X-Z Plane at $y = w/2$ with a Velocity of 0.05 cm/s

We see in Fig. 16 that the peak temperature has moved only 5 mm to 20 mm and only dropped to 1450K. This velocity seems to be not quite slow enough for the microwave beam to heat the colder incoming material close to the original 1500K peak, meaning that if the material were moved slower the peak would remain constant at 1500K. Also note that the peak temperature is at the outer radius of the microwave beam. From Fig. 17, the temperature distribution in the z-direction is again uniform to within 6%.

Still a slower velocity—0.02 cm/s—was used to see if the peak temperature could be better maintained. The results are shown in Figures 18 and 19.
Figure 18: Temperature Distribution in Alumina in the X-Y Plane at \( z = h \) with a Velocity of 0.02 cm/s
Figure 19: Temperature Distribution in Alumina in the X-Z Plane at y = w/2 with a Velocity of 0.02 cm/s

Using the slowest velocity, the peak temperature in the x-y plane has only moved 2.5 mm to 17.5 mm and has increased to 2100K. For this velocity, the material stays too long in the microwave beam area, and thermal runaway has started to occur. Once again in the x-z plane, the temperature distribution is uniform to within 6% in the z-direction.

For a velocity the velocity of 0.05 cm/s, snapshots of the temperature distribution at 9 times from the initial point to after the material has been moved 1 cm can be seen in Fig. 20.
Figure 20: Temperature Distribution along the x-axis in Alumina with a Velocity of
0.05 cm/s for Various Time Steps for $z = h$ and $y = w/2$

The top line represents the higher temperatures at $y = w/2$, and the bottom line represents the lower temperatures at the edges--$y = 0$ and $y = w$--of the sample. Initially the
maximum temperature--1500 K--is located at 15 mm from the left, and after the block has moved 5mm or 20 s have passed, the peak temperature increases at the start, decreases after 20 s, and moves to 20 mm. After 20 s, the peak remains at 20 mm but decreases to 1350 K. To maintain a peak temperature of 1500K, the electric field strength could be decreased initially so the temperature does not increase, but after the peak has moved the electric field could be increased to maintain a peak temperature of 1500 K.

Glass
The reason for investigating glass was to see how the temperature distributions would change with different thermal properties. The major reason for considering glass in particular was because there was published data for the dielectric loss properties and thermal properties--k, $C_p$, and $\rho$. Although, the dielectric loss coefficient data that is published (refer to Fig. 5) shows a steep slope of the dielectric loss coefficient with temperature for temperatures above 1200K. This steep slope could lead to thermal runaway.

Moving sample temperature distributions were obtained the same way as in the previous study of alumina. First, a temperature distribution with a maximum temperature of 1500K for a stationary block was found. The temperature distribution for a stationary beam is shown in Fig. 21 for time $t = 250$ s, a contact resistance of 100 W/m^2K, and a field strength of 950,000 V/m.
Figure 21: Temperature Distribution in Fused Quartz Glass in the X-Y Plane at $z = h$ with a Stationary Sample

The spike in the temperature distribution is an indication that thermal runaway has started. The maximum temperature is 1450K, and 0.1 s prior to this time the maximum temperature was 1150K. In 0.1 s more, the solution will diverge which means the temperature is “running away”. The steep slope of the dielectric loss coefficient with temperature did lead to thermal runaway. Since temperatures were unstable with a stationary sample, the moving block was not studied.
The above results do not show that glass cannot be processed, but that glass with the dielectric loss properties shown in Fig. 5 cannot be processed. Some glasses may have a smaller slope of the dielectric loss coefficient with temperature for temperatures above 1200K and therefore, might have stable temperatures above 1200K.

Absorbed Power

The temperature could possibly be controlled by knowing the power absorbed by the block. The power absorbed could possibly be measured by subtracting the power reflected from the input power to the device. Some runs were performed to find a steady velocity which could produce a constant power absorbed which might lead to a constant temperature profile. A constant temperature profile meaning that the desired joining temperature can be maintained.

The total power was calculated by multiplying the volumetric heat generation for each node by the volume of that node and then summing up the power for all the nodes. The total power absorbed was also recorded for each time step as the beam moved 1 cm; the results are shown in Fig. 22.
Fig. 22 shows that for a velocity of 0.05 cm/s the power changes the least, and from previous temperature distributions, the peak temperature also changes the least for a velocity of 0.05 cm/s. This suggests that if the power absorbed is held constant then the peak temperature might also remain constant.
Discussion

Electric Field Strength Requirements

The thickness--0.03 to 0.1 cm--of the alumina sample did not significantly change the amount of time required to heat the sample to 1500K or affect the uniform temperature distribution through the depth of the material. For conventional heating on the surface, the alumina sample with a low thermal conductivity would have to be heated slowly to maintain a uniform temperature through the depth of the material, and therefore, the thicker the sample the more time it would take to heat it.

For a contact conductance of 100 W/m^2, the electric field strength to heat an alumina sample to 1500K in only 60 s is 165,000 V/m, and the minimum electric field strength to heat the same sample to 1500K in only 355 s is approximately 125,000 V/m. For both electric field strengths, the time required is only minutes, but when the model was modified for a constant heat flux at z=h in a circular shape of Gaussian distribution to simulate conventional heating techniques, the time was in hours. Even when the alumina sample was heated in 60 s, the temperature distribution through the depth of the sample was uniform, and for conventional heating, uniform temperatures like the ones obtained by microwave heating may never occur through the depth of the material. Finally, these times for microwave heating show that joining alumina could be done in minutes.
Contact Resistance Effects

A large contact conductance--1000 W/m^2--between the sample and the supporting material leads to nonuniform temperatures through the depth of the material. This implies that an insulating material should be used to support the sample. In the case of the coaxial microwave spot joining applicator where the applicator could be in contact with the sample, these contact resistance effects suggest that the contact resistance between the sample and the applicator should also be small to maintain uniform temperatures.

Temperature Distributions for a Moving Sample

For alumina, a velocity too large results in decrease of the peak temperature within the sample, and a velocity too small results in thermal runaway. However, there is a velocity where thermal runaway does not occur and the desired maximum temperature is maintained. For all velocities, the temperature distribution through the depth of the sample was found to be uniform. The uniform temperature distribution is important in eliminating thermal stresses along the joint. The snapshots of the temperature distribution as the sample was moved 1 cm showed the peak temperature initially increases. To maintain a constant peak temperature, the initial velocity could be increased, or the input power could be decreased.
For fused quartz glass, thermal runaway could not be avoided at temperatures above 1200K with available data. The reason for this is the large slope of the dielectric loss coefficient with temperature for temperatures above 1200K. However, this dielectric loss coefficient data is not the same for all glasses, so it may be possible to process some types of glass.

Absorbed Power

The power absorbed was found to be directly proportional to the peak temperature. For joining materials, being able to control the power absorbed by the sample might be one way to control the peak temperature within the sample. The other way to control the peak temperature would be to move the sample at a velocity which would keep the power absorbed constant.

Computational Requirements

The large number of nodes which are required for a 3-D mesh results in hours for a solution. For example, to run 100 timesteps of the thick case--$h = 0.1$ cm and 11 nodes in the $z$-direction--takes approximately 10 hours on a SGI Power Challenge. The thin case--$h = 0.03$ cm and 4 nodes in the $z$-direction--takes approximately 1 hour to run on a HP workstation. This is the main reason why more results were not obtained.
Conclusions and Recommendations

The following conclusions may be drawn from this study.

♦ A microwave spot heater appears to be an effective tool for the joining of ceramics.

♦ Alumina could be joined on a time scale of minutes rather than hours.

♦ The electric field strength required to join alumina is on the order of 100,000 V/m.

♦ When the contact resistance is not too large, a uniform temperature distribution through the depth of the material can be maintained even for a moving sample.

♦ To keep the contact conductance small—less than 100 W/m^2—, an insulating material should be used to support the sample being joined.

♦ For alumina, thermal runaway may be avoided by careful choice of the speed of sample movement.

♦ While moving two samples to join them, maintaining a constant absorbed power could be a way of maintaining the desired maximum temperature within the samples.

♦ For the dielectric loss coefficient data for glass, thermal runaway could not be avoided.

Based on the encouraging results of this study, a follow-on study is recommended to pursue the following areas.

♦ Research joining two different ceramics.

♦ Experimental results should be obtained to verify the computational results.

♦ Investigate methods to reduce computation time.
• Use model to compare with conventional techniques.

• With the model parameters, optimize the process.
Literature Cited


Appendix

/* The final 3-D code which was used to give the results */

/* Standard Libraries */
#include <stdio.h>
#include <math.h>
#include <ctype.h>

#define YES 1
#define NO 0

/* Header file for Sparse 1.3 */
#include "spmatrix.h"

#define NODES 37000 /* Upper Limit for the total Number of Nodes */

double velocity, deltat; /* Velocity and timestep variables */
char *Matrix=NULL;
void PrintMatrixErrorMessage( int Error );
int Error;n;
double *element[250000]; /* Pointer Array for Sparse 1.3 */
void elementslots( char *mat, double *slot[] );
void constants ( void ); /* Calculates constants--deltax, etc.. */

/* Calculates the coefficients for the matrix */
void coeff ( double *add[], double righthand[], int time );
void solvematrix( char *solmatrix, double right[]);
double thermakond( int condtheta, int plusminus, int r_or_z, int times); /* 1 is y & 1 is plus*/
double kfunction ( double ktheta );
double cp( int cptheta, int cptime, int plus );
double density( int dentheta, int dentime, int denplus );
double qdot ( int qtheta, int npx, int npx );
double heatgener( int heattheta , int heatime );
double atten( int attentheta ); /* Attenuation function */
double eppa( int eppatheta, int eppertime ); /* Dielectric Loss Coefficient */
double epa( int epatheta );
double guass( double maxvalue, double guassx, double guassy ); /*
double rhs[NODES], Solution[NODES], new [NODES][1], theta[NODES][1], power[500],
double deltai, delta, deltaz, delvol, length1, length2, length3, width, widthq, zlength, delxrat, delyrat, delzrat;
double convec, convecb, convect, theta0, sum = 0.0, emiss = 0.6, boltz = 5.6696e-8, emisboltz;
double frequency, electric_field, velocity1, zdepth, three8, three4, power_sum;
char *animation[12];
int nptx1, nptx2, nptxq, npty, nptyq, nptz, nptlay/* numbers of nodes on a layer*/;

48
int ntime_velocity, total_timesteps, velocity_flag,* 1 for moving beam */, anim_step, anim_flag,
anim_number = 0, anim_count = 0;
int g, gh[10];

main()
{
    int i,k,flag,j,q,y,z;double intererror;
    FILE *temp_out,*xzttemp_out, *temp_input, *power_out, *anim_out;
    Matrix = spCreate( 2, 0, &Error );
    if (Matrix == NULL)
    {
        fprintf(stderr, ": insufficient memory available.\n" );
        exit(1);
    }
    if( Error >= spFATAL ) printf("spfatal");
    constants();
elementsslots( Matrix, element ); /* Sets up the slots for the matrix for Sparse 1.3 */
    /* Start of the main loop for the total number of time steps */
    for ( n=1; n < total_timesteps; n++ )
    {
        coeff( element, rhs, i );
        solvematrix( Matrix, rhs );
        /* Start of the loop for inner iterations */
        for ( q=1; q < nptlay*nptz+1 ; q++ )
        {
            new [q][0]=Solution[q] ;
        }
        do /* Loop for the inner iteration ( for the temperature dependent props) */
        {
            flag=1;
            power_sum = 0.0;
            coeff( element, rhs, 1 );
            solvematrix( Matrix, rhs );
            for ( i=1; i < nptlay*nptz+1 ; i++ )
            {
                intererror = ( new[i][0]-Solution[i])/Solution[i];
                if ( intererror < 0 )
                    intererror = -intererror;
                if ( intererror > 0.001 )
                {
                    i = nptlay*nptz +1;flag=0;
                    for ( k = 1; k < nptlay*nptz +1 ; k++ )
                    {
                        new [k][0]=Solution[k];
                    }
                }
            }
        }
    }
}
while ( flag == 0 );
for ( j=1 ; j < nptlay*nptz + 1 ; j++ )
{
    theta[j][0]=Solution[j];
    if ( Solution[j] > 1500.0 )
        n = total_timesteps;
}

power[n] = power_sum;
if ( anim_flag == 1 )
{
    if ( n == 1 || n%anim_step == 0 )
    {
        if ( (anim_out=fopen(animation[anim_count], "a")) == NULL )
        {
            fprintf(stderr, "error");
            exit(1);
        }
        for ( y=(nptz-1)*nptlay + 1 ; y < nptz*nptlay + 1 ; y++ )
        {
            fprintf ( anim_out, "%f ", new[y][0] );
            if ( y%(npt) == 0 )
            {
                fprintf( anim_out, "\n" );
            }
        }
        anim_count++;
        fprintf( anim_out, "\n\n" );
        fclose( anim_out );
    }
}

/* Printing routines */
/* Prints the temperature for every node to "temp_in" file so the code can be rerun from the last timestep */
if ( (temp_input=fopen("temp_in", "w")) == NULL )
{
    fprintf(stderr, "error");
    exit(1);
}

for ( y=1 ; y < nptz*nptlay + 1 ; y++ )
{
    fprintf ( temp_input, "%f ", new[y][0] );
    if ( y%(npty) == 0 )
    {

fprintf( temp_input, "\n" );
}
}
fclose( temp_input );
/her* Just prints the temperatures for z = h to "temp" file */
if( ( temp_out = fopen("temp", "w")) == NULL )
{
    fprintf(stderr, "error");
    exit(1);
}
for ( y=(nptz-1)*nptlay + 1 ; y < nptz*nptlay + 1 ; y++ )
{
    fprintf( temp_out, "%f", new[y][0] );
    if( y%npty == 0 )
    {
        fprintf( temp_out, "\n" );
    }
}
fclose( temp_out );
/* Prints the temperatures for y = w/2 in the x-z plane to "xztemp" file */
if( ( xztemp_out = fopen("xztemp", "w")) == NULL )
{
    fprintf(stderr, "error");
    exit(1);
}
for ( y=((npty-1)*2) + 1; y < nptz*nptlay; y = y + npty )
{
    fprintf( xztemp_out, "%f ", new[y][0] );
    if( y != ((npty-1)/2) + 1 )
    {
        if( y%nptlay == ((npty-1)/2) + 1 )
            fprintf( xztemp_out, "\n" );
    }
}
fclose( xztemp_out );
/* Prints the total power absorbed to "power" file */
if( ( power_out = fopen("power", "w")) == NULL )
{
    fprintf(stderr, "error");
    exit(1);
}
for ( y=1; y < 1 + total_timesteps ; y++ )
{
    fprintf( power_out, "%f ", power[y] );
    fprintf( power_out, "\n" );
}
fclose( power_out );
printf("Solve time %ld \nCoeff time %ld \n", solve_total, coeff_total );
}


void constants ( void )
{
    int j; double temp_temp;
    FILE *input;FILE *temp_file;
    if ( (input = fopen("input", "r")) == NULL ) {
        fprintf(stderr, "error");
        exit(1);
    }
    /* Reads in all the data from the “input” file */
    fscanf( input, "%lf %lf %lf", &length3, &widthq, &zlength );
    fscanf( input, "%d %d %d %d", &nptx1, &nptx2, &npty, &nptz );
    fscanf( input, "%d %d", &nptxq, &nptyq );
    fscanf( input, "%lf %lf", &frequency, &electric_field );
    fscanf( input, "%lf %lf", &velocity, &velocity_flag );
    fscanf( input, "%d %d", &total_timesteps, &deltat );
    fscanf( input, "%d %d", &anim_flag, &anim_step );
    fclose(input);
    /*length3 = 0.01; nptxq = DELX; nptx1 = LEN1; nptx2 = NPT; */
    npty = NPTYC; nptyq = DELY;
    nptz = NPTZC; */
    nptlay = (nptx1+nptx2)*npty;
    emisboltz = emiss*boltz;
    three8 = 3.0/8.0;
    three4 = 3.0/4.0;
    /* widthq = 0.01;
    zlength = 0.005; */
    deltaz = length3 / (nptxq - 1);
    deltax = widthq / (nptyq - 1);
    deltae = zlength / (nptz-1);
    delvola = deltax*deltay*deltaz;
    length = deltax*(nptx1-1);
    width = deltax*(npty-1);
    theta0 = 300.0;
    /* Initializes all the temperatures to 300 K */
    if ( velocity_flag == 0 )
    {
        for ( j = 1 ; j < 1 + nptz*nptlay ; j++ )
        {
            theta [j][0]=300.0;
            new [j][0]=300.0;
        }
    }

    // ...
*/ Reads in the temperatures if not starting from time t = 0 */
else
{
   if ((temp_file = fopen("temp_in", "r")) == NULL)
   {
      fprintf(stderr, "error");
      exit(1);
   }
   for (j = 1 ; j < 1 + nptz*nptlay ; j++)
   {
      fscanf( temp_file, "%lf", &temp_temp);
      theta[j][0] = temp_temp;
      new[j][0] = theta[j][0];
   }
   fclose( temp_file );
}
convect = 10.0; convectb = 100.0; convect = 10.0;
deltax = (deltax*delta)/delta;
deltay = (deltay*delta)/delta;
deltaz = (deltaz*delta)/delta;
animation[0] = "anim_1";
animation[1] = "anim_2";
animation[2] = "anim_3";
animation[3] = "anim_4";
animation[4] = "anim_5";
animation[5] = "anim_6";
animation[6] = "anim_7";
animation[7] = "anim_8";
animation[8] = "anim_9";
animation[9] = "anim_10";
animation[10] = "anim_11";

/* Sparse 1.3 first wants you to setup the slots you want in the matrix and this is the function to do that */
void elementslots( char *mat, double *slot[])
{
   register int a,b; int y, q; /* All slots are opened starting from the center node then the top and then clockwise */
a=1;b=1;
/* Left hand side elements */
/* --------Bottom Layer------------------- */
/* (a,a) is the node being solved for */
slot[b]=spGetElement( mat, a, a ); b++; /* bottom */
slot[b]=spGetElement(mat, a, a+1) ; b++ ; /* Next node in the positive y-direction */
slot[b]=spGetElement(mat, a, a+nptyl) ; b++ ; /* Next node in the positive x-direction */
slot[b]=spGetElement(mat, a, a+nptyl) ; b++ ; /* Next node in the positive z-direction */
slot[b]=spGetElement(mat, a, a+2) ; b++ ;
slot[b]=spGetElement(mat, a, a+2+nptyl) ; b++ ;
slot[b]=spGetElement(mat, a, a+2*nptyl) ; b++ ;
for ( a = 2; a < nptyl ; a++ )
{
    slot[b]=spGetElement(mat, a, a) ; b++ ; /* middle */
    slot[b]=spGetElement(mat, a, a+1) ; b++ ;
    slot[b]=spGetElement(mat, a, a+nptyl) ; b++ ;
    slot[b]=spGetElement(mat, a, a-1) ; b++ ;
    slot[b]=spGetElement(mat, a, a+nptyl) ; b++ ;
    slot[b]=spGetElement(mat, a, a+2*nptyl) ; b++ ;
    slot[b]=spGetElement(mat, a, a+2*nptyl) ; b++ ;
}

a = nptyl;
slot[b]=spGetElement(mat, a, a) ; b++ ; /* top */
slot[b]=spGetElement(mat, a, a+nptyl) ; b++ ;
slot[b]=spGetElement(mat, a, a-1) ; b++ ;
slot[b]=spGetElement(mat, a, a+nptyl) ; b++ ;
slot[b]=spGetElement(mat, a, a-2*nptyl) ; b++ ;
slot[b]=spGetElement(mat, a, a+2*nptyl) ; b++ ;
for ( a = nptyl + 1 ; a < (nptxl + nptx2 - 1)*nptyl + 1 ; a++ )
{
    if ( a % nptyl == 0 ) /* top nodes */
    {
        slot[b]=spGetElement(mat, a, a) ; b++ ;
        slot[b]=spGetElement(mat, a, a+nptyl) ; b++ ;
        slot[b]=spGetElement(mat, a, a-1) ; b++ ;
        slot[b]=spGetElement(mat, a, a-nptyl) ; b++ ;
        slot[b]=spGetElement(mat, a, a+nptyl) ; b++ ;
        slot[b]=spGetElement(mat, a, a-2) ; b++ ;
        slot[b]=spGetElement(mat, a, a+2*nptyl) ; b++ ;
    }
else if ( a % nptyl == 1 ) /* bottom nodes */
{
    slot[b]=spGetElement(mat, a, a) ; b++ ;
    slot[b]=spGetElement(mat, a, a+1) ; b++ ;
    slot[b]=spGetElement(mat, a, a+nptyl) ; b++ ;
    slot[b]=spGetElement(mat, a, a-nptyl) ; b++ ;
    slot[b]=spGetElement(mat, a, a+nptyl) ; b++ ;
    slot[b]=spGetElement(mat, a, a+2) ; b++ ;
    slot[b]=spGetElement(mat, a, a+2*nptyl) ; b++ ;
}
54
else /* middle nodes */
{
    slot[b]=spGetElement( mat, a, a ) ; b++ ;
    slot[b]=spGetElement( mat, a, a+1 ) ; b++ ;
    slot[b]=spGetElement( mat, a, a+npty) ; b++ ;
    slot[b]=spGetElement( mat, a, a-1) ; b++ ;
    slot[b]=spGetElement( mat, a, a-npty) ; b++ ;
    slot[b]=spGetElement( mat, a, a+nptlay) ; b++ ;
    slot[b]=spGetElement( mat, a, a+2*nptlay); b++ ;
}

a = (nptx1 + nptx2 - 1)*npty + 1;
slot[b]=spGetElement( mat, a, a ) ; b++ ;
slot[b]=spGetElement( mat, a, a+1 ) ; b++ ;
slot[b]=spGetElement( mat, a, a-npty ) ; b++ ;
slot[b]=spGetElement( mat, a, a+nptlay) ; b++ ;
slot[b]=spGetElement( mat, a, a+2 ) ; b++ ;
slot[b]=spGetElement( mat, a, a-2*npty) ; b++ ;
slot[b]=spGetElement( mat, a, a+2*nptlay) ; b++ ;
for ( a = (nptx1 + nptx2 - 1)*npty + 2; a < (nptx1 + nptx2)*npty ; a++ )
{
    slot[b]=spGetElement( mat, a, a ) ; b++ ;
    slot[b]=spGetElement( mat, a, a+1 ) ; b++ ;
    slot[b]=spGetElement( mat, a, a-1) ; b++ ;
    slot[b]=spGetElement( mat, a, a-npty) ; b++ ;
    slot[b]=spGetElement( mat, a, a+nptlay) ; b++ ;
    slot[b]=spGetElement( mat, a, a-2*npty) ; b++ ;
    slot[b]=spGetElement( mat, a, a+2*nptlay) ; b++ ;
}

a = (nptx1 + nptx2)*npty;
slot[b]=spGetElement( mat, a, a ) ; b++ ;
slot[b]=spGetElement( mat, a, a-1) ; b++ ;
slot[b]=spGetElement( mat, a, a-npty) ; b++ ;
    slot[b]=spGetElement( mat, a, a+nptlay) ; b++ ;
    slot[b]=spGetElement( mat, a, a-2 ) ; b++ ;
    slot[b]=spGetElement( mat, a, a-2*npty) ; b++ ;
    slot[b]=spGetElement( mat, a, a+2*nptlay) ; b++ ;

/* --------Middle Layers ------------------------*/
for ( q = 1; q < nptz-1; q++ )
{
    a = 1 + nptlay*q;
    slot[b]=spGetElement( mat, a, a ) ; b++ ; /* bottom */
    slot[b]=spGetElement( mat, a, a+1) ; b++ ;
    slot[b]=spGetElement( mat, a, a+npty) ; b++ ;
    slot[b]=spGetElement( mat, a, a+nptlay) ; b++ ;

55
slot[b]=spGetElement( mat, a, a-nptlay) ; b++;
slot[b]=spGetElement( mat, a, a+2) ; b++;
slot[b]=spGetElement( mat, a, a+2*npty) ; b++;

for ( a = 2 + nptlay*q; a < npty + nptlay*q; a++ )
{
    slot[b]=spGetElement( mat, a, a ) ; b++ ; /* middle */
    slot[b]=spGetElement( mat, a, a+1 ) ; b++ ;
    slot[b]=spGetElement( mat, a, a+npty) ; b++ ;
    slot[b]=spGetElement( mat, a, a-1 ) ; b++ ;
    slot[b]=spGetElement( mat, a, a+nptlay) ; b++ ;
    slot[b]=spGetElement( mat, a, a-nptlay) ; b++ ;
    slot[b]=spGetElement( mat, a, a+2*npty) ; b++ ;
}

a = npty + nptlay*q;
slot[b]=spGetElement( mat, a, a ) ; b++ ; /* top */
slot[b]=spGetElement( mat, a, a+npty) ; b++ ;
slot[b]=spGetElement( mat, a, a-1) ; b++ ;
slot[b]=spGetElement( mat, a, a+nptlay) ; b++ ;
slot[b]=spGetElement( mat, a, a-nptlay) ; b++ ;
slot[b]=spGetElement( mat, a, a+2*npty) ; b++ ;
slot[b]=spGetElement( mat, a, a-2) ; b++ ;
for ( a = npty + 1 + nptlay*q; a < (nptxl + nptlx - 1)*npty + 1 + nptlay*q; a++ )
{
    if ( a % npty == 0 ) /* top nodes */
    {
        slot[b]=spGetElement( mat, a, a ) ; b++ ;
        slot[b]=spGetElement( mat, a, a+npty) ; b++ ;
        slot[b]=spGetElement( mat, a, a-1) ; b++ ;
        slot[b]=spGetElement( mat, a, a-npty) ; b++ ;
        slot[b]=spGetElement( mat, a, a+nptlay) ; b++ ;
        slot[b]=spGetElement( mat, a, a-nptlay) ; b++ ;
        slot[b]=spGetElement( mat, a, a-2) ; b++ ;
    }
    else if ( a % npty == 1 ) /* bottom nodes */
    {
        slot[b]=spGetElement( mat, a, a ) ; b++ ;
        slot[b]=spGetElement( mat, a, a+1 ) ; b++ ;
        slot[b]=spGetElement( mat, a, a+npty) ; b++ ;
        slot[b]=spGetElement( mat, a, a-npty) ; b++ ;
        slot[b]=spGetElement( mat, a, a+nptlay) ; b++ ;
        slot[b]=spGetElement( mat, a, a-nptlay) ; b++ ;
        slot[b]=spGetElement( mat, a, a-2) ; b++ ;
    }
    else /* middle nodes */
```c

```
```
for ( a = 2 + (nptz-1)*nptlay; a < npty + (nptz-1)*nptlay; a++ )
{
    slot[b]=spGetElement( mat, a, a ) ; b++ ; /* middle */
    slot[b]=spGetElement( mat, a, a+1 ) ; b++ ;
    slot[b]=spGetElement( mat, a, a+npty ) ; b++ ;
    slot[b]=spGetElement( mat, a, a-1 ) ; b++ ;
    slot[b]=spGetElement( mat, a+nptlay ) ; b++ ;
    slot[b]=spGetElement( mat, a+2*npty ) ; b++ ;
    slot[b]=spGetElement( mat, a-2*nptlay ) ; b++ ;
}

a = npty + (nptz-1)*nptlay;
slot[b]=spGetElement( mat, a, a ) ; b++ ; /* top */
slot[b]=spGetElement( mat, a, a+npty ) ; b++ ;
slot[b]=spGetElement( mat, a, a-1 ) ; b++ ;
slot[b]=spGetElement( mat, a-nptlay ) ; b++ ;
slot[b]=spGetElement( mat, a, a+2*npty ) ; b++ ;
slot[b]=spGetElement( mat, a, a-2 ) ; b++ ;
slot[b]=spGetElement( mat, a, a-2*nptlay ) ; b++ ;
for ( a = npty + 1 + (nptz-1)*nptlay; a < (nptx1 + nptx2 - 1)*npty + 1 + (nptz-1)*nptlay; a++ )
{
    if ( a % npty == 0 ) /* top nodes */
    {
        slot[b]=spGetElement( mat, a, a ) ; b++ ;
        slot[b]=spGetElement( mat, a, a+npty ) ; b++ ;
        slot[b]=spGetElement( mat, a, a-1 ) ; b++ ;
        slot[b]=spGetElement( mat, a-nptlay ) ; b++ ;
        slot[b]=spGetElement( mat, a, a+2*npty ) ; b++ ;
        slot[b]=spGetElement( mat, a, a-2 ) ; b++ ;
        slot[b]=spGetElement( mat, a, a-2*nptlay ) ; b++ ;
    }
    else if ( a % npty == 1 ) /* bottom nodes */
    {
        slot[b]=spGetElement( mat, a, a ) ; b++ ;
        slot[b]=spGetElement( mat, a, a+1 ) ; b++ ;
        slot[b]=spGetElement( mat, a, a+npty ) ; b++ ;
        slot[b]=spGetElement( mat, a, a-1 ) ; b++ ;
        slot[b]=spGetElement( mat, a, a+2*npty ) ; b++ ;
        slot[b]=spGetElement( mat, a, a-2*nptlay ) ; b++ ;
    }
    else /* middle nodes */
    {
        slot[b]=spGetElement( mat, a, a ) ; b++ ;
        slot[b]=spGetElement( mat, a, a+1 ) ; b++ ;
        slot[b]=spGetElement( mat, a, a+npty ) ; b++ ;
        slot[b]=spGetElement( mat, a, a-1 ) ; b++ ;
    }
}

58
slot[b]=spGetElement( mat, a, a-npty) ; b++;
slot[b]=spGetElement( mat, a, a-nptlay) ; b++;
slot[b]=spGetElement( mat, a, a-2*nptlay) ; b++;
}
}

a = (nptx1 + nptx2 - 1)*npty + 1 + (nptz-1)*nptlay;
slot[b]=spGetElement( mat, a, a ) ; b++;
slot[b]=spGetElement( mat, a, a+1 ) ; b++;
slot[b]=spGetElement( mat, a, a-npty ) ; b++;
slot[b]=spGetElement( mat, a, a-nptlay) ; b++;
slot[b]=spGetElement( mat, a, a+2 ) ; b++;
slot[b]=spGetElement( mat, a, a-2*nptlay) ; b++;
slot[b]=spGetElement( mat, a, a-2*npty) ; b++;
for ( a = (nptx1 + nptx2 - 1)*npty + 2 + (nptz-1)*nptlay; a < nptz*nptlay; a++ )
{
    slot[b]=spGetElement( mat, a, a ) ; b++;
    slot[b]=spGetElement( mat, a, a+1 ) ; b++;
    slot[b]=spGetElement( mat, a, a-1 ) ; b++;
    slot[b]=spGetElement( mat, a, a-npty) ; b++;
    slot[b]=spGetElement( mat, a, a-nptlay) ; b++;
    slot[b]=spGetElement( mat, a, a-2*nptlay) ; b++;
    slot[b]=spGetElement( mat, a, a-2*npty) ; b++;
}

a = nptz*nptlay;
slot[b]=spGetElement( mat, a, a ) ; b++;
slot[b]=spGetElement( mat, a, a-1) ; b++;
slot[b]=spGetElement( mat, a, a-npty) ; b++;
slot[b]=spGetElement( mat, a, a-nptlay) ; b++;
slot[b]=spGetElement( mat, a, a-2) ; b++;
slot[b]=spGetElement( mat, a, a-2*nptlay) ; b++;
slot[b]=spGetElement( mat, a, a-2*npty) ; b++;

/* Function to calculate the coefficients of the matrix and adds them to the slots which were
opened by elementslots */

void coeff( double *addf[], double righthandf[], int time )
{
    register int a,b, int y, x, q, zdepth,double value;
    a=1,b=1;zdepth=nptz;
    /* ----------------------------------------Bottom Layer--------------------------------------*/
    /* node on the most bottom left corner */
    value = (-three8*delyrat*thermalcond( a,4,1,time) + /* km+1 */
            three8*delyrat*thermalcond( a,2,1,time) + /* kn+1 */
            three8*delyrat*thermalcond( a,6,1,time) + /* kz+1 */
            convec*deltaz*((deltax + deltay)/4.0)) +

convecb*((deltax*deltay)/4.0) +
deltaz*((deltax + deltax)/4.0)*emisboltz*new[a][0]*new[a][0]*new[a][0] +
(delvol/8.0)*cp(a, time, 1)*(density(a, time, 1)/deltat); 
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = (delyrat/2.0)*thermalcond( a,2,1,time); /* kn+1 */
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = (delyrat/2.0)*thermalcond( a,4,1,time); 
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = (delyrat/2.0)*thermalcond( a,6,1,time); 
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = -(delyrat/8.0)*thermalcond( a,2,1,time); /* kn+1 */
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = -(delyrat/8.0)*thermalcond( a,4,1,time); 
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = -(delyrat/8.0)*thermalcond( a,6,1,time); 
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
righthand[a] = -(convec*deltaz*((deltax+deltay)/4.0) +
convec*deltax*(deltax/4.0))*(theta0 +
(delvol/8.0)*cp(a, time, 1)*(density(a, time, 1)/deltat)*theta[a][0] ) ;
for ( a = 2; a < npty; a++ )
{
value = -(three4*delyrat)*thermalcond( a,4,1,time) +
(delyrat/4.0)*thermalcond( a,1,1,time) +
(delyrat/4.0)*thermalcond( a,0,1,time) +
(three4*delyrat)*thermalcond( a,6,1,time) +
convec*deltax*(deltax/2.0) +
convec*deltax*(deltax/2.0) +
deltay*(deltaz/2.0)*emisboltz*new[a][9]*new[a][0]*new[a][0] +
(delvol/4.0)*cp(a, time, 1)*(density(a, time, 1)/deltat); 
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = (delyrat/4.0)*thermalcond( a,1,1,time); 
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = delyrat*thermalcond( a,4,1,time); 
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = (delyrat/4.0)*thermalcond( a,0,1,time); 
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = delzrat*thermalcond( a,6,1,time); 
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = -(delyrat/4.0)*thermalcond( a,4,1,time); 
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = -(delyrat/4.0)*thermalcond( a,6,1,time); 
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
righthand[a] = -(convec*deltay*(deltay/2.0) +
convec*deltay*(deltay/2.0))*(theta0 +
(delvol/4.0)*cp(a, time, 1)*(density(a, time, 1)/deltat)*theta[a][0] ) ;
}
a = npy; /* top */
value = -(three*delxrat*thermalcond(a,2,0,time)) + /* kn+1 */
   three*delxrat*thermalcond(a,4,1,time) + /* km+1 */
   three*delxrat*thermalcond(a,6,1,time) + /* kz+1 */
   convect*deltax*((deltax+deltay)/4.0) +
   convvecb*deltax*(deltax/4.0) +
   deltax*(deltax+deltay)/4.0)*emisboltz*new[a][0]*new[a][0]*new[a][0] +
   (delvol/8.0)*cp(a, time, 1)*(density(a, time, 1)/deltat)); /* node */
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = (delxrat/2.0)*thermalcond(a,4,1,time); /* right */
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = (delxrat/2.0)*thermalcond(a,2,0,time); /* bottom */ /* kn+1 */
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = (delxrat/2.0)*thermalcond(a,6,1,time); /* bottom */ /* kz+1 */
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(delxrat/8.0)*thermalcond(a,4,1,time); /* 2 right */ /* km+1 */
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(delxrat/8.0)*thermalcond(a,2,0,time); /* 2 bottom */ /* kn+1 */
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(delxrat/8.0)*thermalcond(a,6,1,time); /* kz+1 */
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
righthand[a] = -(convect*deltax*((deltax+deltay)/4.0) +
   convvecb*deltax*(deltax/4.0) )*theta0 +
   (delvol/8.0)*cp(a, time, 1)*(density(a, time, 1)/deltat)*theta[a][0] ;
x = 2; y = 1;
for ( a = npy + 1 ; a < (nptx1 + nptx2 - 1)*npy + 1 ; a++ )
{
   if ( a % npy == 0 ) /* top nodes */
   {
      value = -(delxrat/4.0)*thermalcond(a,1,0,time) +
      (delxrat/4.0)*thermalcond(a,0,0,time) +
      (three*delxrat*thermalcond(a,2,0,time) + /* kn-1 */
      (three*delxrat*thermalcond(a,6,1,time) + /* kz+1 */
      convect*deltax*(deltax/2.0) +
      convvecb*deltax*(deltax/2.0) +
      deltax*(deltax/2.0)*emisboltz*new[a][0]*new[a][0]*new[a][0] +
      velocity*deltax*(deltax/4.0)*cp(a, time, 1)*density(a, time, 1) +
      (delvol/4.0)*cp(a, time, 1)*density(a, time, 1)/deltat) ;
      spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
      value = (delxrat/4.0)*thermalcond(a,1,0,time);
      spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
      value = delxrat*thermalcond(a,2,0,time); /* kn-1 */
      spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
      value = delxrat*thermalcond(a,0,0,time); /* kn-1 */
      spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
      value = (delxrat/4.0)*thermalcond(a,0,0,time) + velocity*deltax*(deltax/4.0)*cp(a, time, 0) +
      (delvol/4.0)*cp(a, time, 0)*density(a, time, 0) ;
      spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
   }
value = delzrat * thermalcond(a,6,1,time);    /* kz+1 */
spADD_REAL_ELEMENT(add[b],value); b++;
value = -( (delxrat/4.0)*thermalcond(a,2,0,time)); /* kn-1 */
spADD_REAL_ELEMENT(add[b],value); b++;
value = -( (delzrat/4.0)*thermalcond(a,6,1,time)); /* kz+1 */
spADD_REAL_ELEMENT(add[b],value); b++;

righthand[a] = -( (convect*deltax*(deltaz/2.0) +
                convect*b*deltax*(deltay/2.0) )*theta0 +
                (delvol/4.0)*cp(a, time, 1 )*(density(a, time, 1 )/deltat)*theta[a][0] ) ;
x++; y = 1;
}
else if ( (a % npty == 1 ) /* bottom nodes */
{
    value = -( (delxrat/4.0)*thermalcond(a,1,0,time) +
            (deltax/4.0)*thermalcond(a,0,0,0,time) +
            (three4)*delxrat*thermalcond(a,2,1,time) +/* kn+1 */
            (three4)*delzrat*thermalcond(a,6,1,time) +/* kz+1 */
            convect*deltax*(deltaz/2.0) +
            convect*b*deltax*(deltay/2.0) +
            deltax*(deltaz/2.0)*emisboltz:new[a][0]*new[a][0]*new[a][0] +
            velocity*deltaz*(deltay/4.0)*cp(a, time, 1 )*(density(a, time, 1 )/deltat) );
spADD_REAL_ELEMENT(add[b],value); b++;
value = delxrat*thermalcond(a,2,1,time); /* kn+1 */
spADD_REAL_ELEMENT(add[b],value); b++;
value = (delxrat/4.0)*thermalcond(a,1,0,time);
spADD_REAL_ELEMENT(add[b],value); b++;
value = (deltax/4.0)*thermalcond(a,0,0,0,time) +
spADD_REAL_ELEMENT(add[b],value); b++;
value = -( (delxrat/4.0)*thermalcond(a,2,1,time)); /* kn-1 */
spADD_REAL_ELEMENT(add[b],value); b++;
value = -( (delzrat/4.0)*thermalcond(a,6,1,time)); /* kz+1 */
spADD_REAL_ELEMENT(add[b],value); b++;
righthand[a] = -( (convect*deltax*(deltaz/2.0) +
                convect*b*deltax*(deltay/2.0) )*theta0 +
                (delvol/4.0)*cp(a, time, 1 )*(density(a, time, 1 )/deltat)*theta[a][0] ) ;
y++;
}
else /* middle nodes */
{
    value = -( (delxrat/2.0)*thermalcond(a,1,0,time) +
            (deltax/2.0)*thermalcond(a,0,0,0,time) +
            (delzrat/2.0)*thermalcond(a,1,1,time) +
    62
(delxrat/2.0)*thermalcond( a,0,1,time) +
((3.0*delxrat/2.0)*thermalcond( a,6,1,time) +
ccnvecb*deltax*(deltay) +
velocity*(deltaz/2.0)*deltay*cp( a, time, 1)*density( a, time, 1 ) +
(delvol/2.0)*cp( a, time, 1 )*density( a, time, 1 )/deltat) );
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = (delxrat/2.0)*thermalcond( a,1,1,time);
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = (delyrat/2.0)*thermalcond( a,1,0,time);
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = (delxrat/2.0)*thermalcond( a,0,1,time);
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = (delyrat/2.0)*thermalcond( a,0,0,time) + velocity*(deltaz/2.0)*deltay*cp( a, time, 0 )
)*density( a, time, 0 ) ;
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = 2.0*delxrat*thermalcond( a,6,1,time);
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = -(3*delxrat/2.0)*thermalcond( a,6,1,time) ;
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
righthand[a] = -( (delvol/2.0)*qdot( a, x, y )
+ ccnvecb*deltax*(deltay)*theta0 +
(delvol/2.0)*cp( a, time, 1 )*density( a, time, 1 )/deltat)*theta[a][0] ) ;
y++;
}
}
a = (nptx1 + nptx2 - 1)*npty + 1;
value = -( three8*delxrat*thermalcond( a,4,0,time) + /* km+1 */
three8*delxrat*thermalcond( a,2,1,time) + /* kn+1 */
three8*delxrat*thermalcond( a,6,1,time) + /* kz+1 */
ccnvecb*deltax*((deltax+deltay)/4.0) +
ccnvecb*deltax*(deltax/4.0) +
deltaz*(deltax+deltay)/4.0)*emisbeltz*new[a][0]*new[a][0]*new[a][0] +
(delvol/8.0)*cp( a, time, 1 )*density( a, time, 1 )/deltat) );
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = (delxrat/2.0)*thermalcond( a,4,0,time); /* km+1 */
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = (delyrat/2.0)*thermalcond( a,4,0,time); /* km+1 */
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = (delxrat/8.0)*thermalcond( a,2,1,time); /* kn+1 */
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = -( (delxrat/8.0)*thermalcond( a,4,0,time) + /* km+1 */
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = -( (delxrat/8.0)*thermalcond( a,6,1,time) + /* kz+1 */
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = -( (delxrat/8.0)*thermalcond( a,6,1,time) + /* kz+1 */
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;

63
righthand[a] = -( (convec*deltaz*((deltax+deltay)/4.0) +
  convec*b*deltax*(deltay/4.0) ) ) * theta0 +
(dvol/8.0)*cp(a, time, 1) *(density(a, time, 1)/deltat)*theta[a][0] ;
for (a = (nptx1 + nptx2 - 1)*npty + 2; a < (nptx1 + nptx2)*npty; a++) {
  value = -( (three4*delyrat)*thermalcond(a,4,0,time) +
(delyrat/4.0)*thermalcond(a,1,1,time) ) +
(delyrat/4.0)*thermalcond(a,0,1,time) +
(three4*delyrat)*thermalcond(a,6,1,time) +
deltay*(deltaz/2.0)*convec +
deltay*(deltaz/2.0)*convecb +
deltay*(deltaz/2.0)*emisboltz*new[a][0]*new[a][0]*new[a][0] +
(dvol/4.0)*cp(a, time, 1) *(density(a, time, 1)/deltat) ) ;
spADD_REAL_ELEMENT(add[b],value) ; b++ ;
value = (delyrat/4.0)*thermalcond(a,1,1,time);
spADD_REAL_ELEMENT(add[b],value) ; b++ ;
value = (delyrat/4.0)*thermalcond(a,0,1,time);
spADD_REAL_ELEMENT(add[b],value) ; b++ ;
value = delyrat*thermalcond(a,4,0,time);
spADD_REAL_ELEMENT(add[b],value) ; b++ ;
value = delyrat*thermalcond(a,6,1,time);
spADD_REAL_ELEMENT(add[b],value) ; b++ ;
value = -( (delyrat/4.0)*thermalcond(a,4,0,time) ) ;
spADD_REAL_ELEMENT(add[b],value) ; b++ ;
value = -( (delyrat/4.0)*thermalcond(a,6,1,time) ) ;
righthand[a] = -( (deltay*(deltaz/2.0)*convec +
  deltay*(deltaz/2.0)*convecb ) ) * theta0 +
(dvol/4.0)*cp(a, time, 1) *(density(a, time, 1)/deltat)*theta[a][0] ;
}
a = (nptx1 + nptx2)*npty;
value = -( three8*delyrat*thermalcond(a,2,0,time) /* kn-1 */
  three8*delyrat*thermalcond(a,4,0,time) /* kn-1 */
  three8*delyrat*thermalcond(a,6,1,time) /* kn-1 */
  convec*deltaz*((deltax+deltay)/4.0) +
  convecb*(deltaz/4.0) ) +
deltaz*((deltax+deltay)/4.0)*emisboltz*new[a][0]*new[a][0]*new[a][0] +
(dvol/8.0)*cp(a, time, 1) *(density(a, time, 1)/deltat) ) ;
spADD_REAL_ELEMENT(add[b],value) ; b++ ;
value = (delyrat/2.0)*thermalcond(a,2,0,time) /* kn-1 */
spADD_REAL_ELEMENT(add[b],value) ; b++ ;
value = (delyrat/2.0)*thermalcond(a,4,0,time) /* kn-1 */
spADD_REAL_ELEMENT(add[b],value) ; b++ ;
value = (delyrat/2.0)*thermalcond(a,6,1,time) /* kn-1 */
spADD_REAL_ELEMENT(add[b],value) ; b++ ;
value = -( (delyrat/8.0)*thermalcond(a,2,0,time) ) /* kn-1 */
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = -( (delxrat/8.0)*thermalcond( a,4,0,time) ) ; /* km-1 */
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = -( (delxrat/8.0)*thermalcond( a,6,1,time) ) ; /* kz+1 */
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
righthand[a] = -( ( convec*deltaz*((deltax+deltay)/4.0) +
                   convec*(((deltax*deltay)/4.0) )*theta0 +
                   (deltvol/8.0)*cp( a, time, 1 )*(density( a, time, 1 )/deltat)*theta[a][0] ) ;

/* ---------------Middle Layers-----------------------------*/
for ( q = 1; q < nptz-1; q++ )
{
    a = 1 + nptlay*q; zdepth = nptz - q;
value = -( (three4)*delxrat*thermalcond( a,4,1,time) ) +/* km+1 */
       (three4)*delxrat*thermalcond( a,2,1,time) ) +/* kn+1 */
       (delxrat/4.0)*thermalcond( a,3,1,time) +
       (delxrat/4.0)*thermalcond( a,3,0,time) +
       convec*deltaz*((deltax + deltay)/2.0) +
       deltaz*((deltax + deltay)/2.0)*emisboltz*new[a][0]*new[a][0]*new[a][0] +
       (deltvol/4.0)*cp( a, time, 1 )*(density( a, time, 1 )/deltat) ) ;
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = delxrat*thermalcond( a,2,1,time) ; /* kn+1 */
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = delxrat*thermalcond( a,4,1,time) ;
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = (delxrat/4.0)*thermalcond( a,3,1,time) ;
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = (delxrat/4.0)*thermalcond( a,3,0,time) ;
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = -( (delxrat/4.0)*thermalcond( a,2,1,time) ) ; /* km+1 */
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = -( (delxrat/4.0)*thermalcond( a,4,1,time) ) ; /* km+1 */
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
righthand[a] = -( ( convec*deltaz*((deltax+deltay)/2.0)*theta0 +
                   (deltvol/4.0)*cp( a, time, 1 )*(density( a, time, 1 )/deltat)*theta[a][0] ) ;

for ( a = 2 + nptlay*q; a < npty + nptlay*q; a++ )
{
    value = -( ((3.0*delxrat)/2.0)*thermalcond( a,4,1,time) ) +
                   (delxrat/2.0)*thermalcond( a,1,1,time) ) +
                   (delxrat/2.0)*thermalcond( a,0,1,time) ) +
                   (delxrat/2.0)*thermalcond( a,3,1,time) ) +
                   (delxrat/2.0)*thermalcond( a,3,0,time) ) +
                   convec*deltay*deltaz +
                   deltay*deltaz*emisboltz*new[a][0]*new[a][0]*new[a][0] +
(delvol/2.0)*cp( a, time, 1 )*(density( a, time, 1 )/deltat ));
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = (delxrat/2.0)*thermalcond( a,1,1,time);
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = (2.0*delyrat)*thermalcond( a,4,1,time);
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = (delxrat/2.0)*thermalcond( a,0,1,time);
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = (delzrat/2.0)*thermalcond( a,3,1,time);
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = (delzrat/2.0)*thermalcond( a,3,0,time);
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = -( (delyrat/2.0)*thermalcond( a,4,1,time ));
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
rightness[a] = -( convect*deltay*deltaz*theta0 +
(delvol/2.0)*cp( a, time, 1 )*(density( a, time, 1 )/deltat)*theta[a][0] );
}
a = npty + nptlay*q;/* top */
value = -( (three4)*delxrat*thermalcond( a,2,0,time) + /* kn+1 */
(three4)*delyrat*thermalcond( a,4,1,time) + /* km+1 */
(delzrat/4.0)*thermalcond( a,3,1,time) +
(delzrat/4.0)*thermalcond( a,3,0,time) +
convect*deltaz*((deltax+deltay)/2.0) +
deltaz*((deltax+deltay)/2.0)*emisboltz*new[a][0]*new[a][0]*new[a][0] +
(delvol/4.0)*cp( a, time, 1 )*(density( a, time, 1 )/deltat ));/*/ node */
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = delyrat*thermalcond( a,4,1,time );/* right */
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = delxrat*thermalcond( a,2,0,time);/* bottom *//* kn+1 */
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = (delzrat/4.0)*thermalcond( a,3,1,time);
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = (delzrat/4.0)*thermalcond( a,3,0,time);
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = -( (delyrat/4.0)*thermalcond( a,4,1,time ));/* 2 right *//* km+1 */
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = -( (delxrat/4.0)*thermalcond( a,2,0,time));/* 2 bottom *//* kn+1 */
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
rightness[a] = -( convect*deltay*deltaz*(deltax+deltay)/2.0)*theta0 +
(delvol/4.0)*cp( a, time, 1 )*(density( a, time, 1 )/deltat)*theta[a][0] );
x = 2; y = 1;
for ( a = npty + 1 + nptlay*q; a < (npx1 + npx2 - 1)*npty + 1 + nptlay*q ; a++ )
{
if ( a % npty == 0 ) /* top nodes */
{
value = -( (delyrat/2.0)*thermalcond( a,1,0,time) +

(delyrat/2.0)*thermalcond( a,0,0,time) +
(3.0/2.0)*delxrat*thermalcond( a,2,0,time) +/* kn-1 */
(delzrat/2.0)*thermalcond( a,3,1,time) +
(delzrat/2.0)*thermalcond( a,3,0,time) +
convex*deltax*deltaz +
deltax*deltaz*emisboltz*new[a][0]*new[a][0]*new[a][0] +
velocity*deltax*(deltay/2.0)*cp(a, time, 1 )*density( a, time, 1 ) +
(delvol/2.0)*cp(a, time, 1 )*(density( a, time, 1 )/deltat));
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = (delyrat/2.0)*thermalcond( a,1,0,time) ;
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = (2.0*delxrat)*thermalcond( a,2,0,time) ; /* kn-1 */
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = (delyrat/2.0)*thermalcond( a,0,0,time) + velocity*deltax*(deltay/2.0)*cp(a, time, 0 )*density( a, time, 0 ) ;
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = (delzrat/2.0)*thermalcond( a,3,1,time) ;
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = (delzrat/2.0)*thermalcond( a,3,0,time) ; /* kn-1 */
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
righthand[a] = -( convex*deltax*deltaz*theta0 +
(delvol/2.0)*cp(a, time, 1 )*(density( a, time, 1 )/deltat)*theta[a][0] ) ;
x++;y = 1; }
else if ( a % npty == 1 ) /* bottom nodes */
{
value = -( (delyrat/2.0)*thermalcond( a,1,0,time) +
(delzrat/2.0)*thermalcond( a,0,0,time) +
(3.0/2.0)*delxrat*thermalcond( a,2,1,time) +/* kn+1 */
(delzrat/2.0)*thermalcond( a,3,1,time) +
(delzrat/2.0)*thermalcond( a,3,0,time) +
convex*deltax*deltaz +
deltax*deltaz*emisboltz*new[a][0]*new[a][0]*new[a][0] +
velocity*deltax*(deltay/2.0)*cp(a, time, 1 )*density( a, time, 1 ) +
(delvol/2.0)*cp(a, time, 1 )*(density( a, time, 1 )/deltat));
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = (2.0*delxrat)*thermalcond( a,2,1,time) ;/* kn+1 */
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = (delyrat/2.0)*thermalcond( a,1,0,time) ;
spADD_REAL_ELEMENT ( add[b],value ) ; b++ ;
value = (delyrat/2.0)*thermalcond( a,0,0,time) +
velocity*deltax*(deltay/2.0)*cp(a,time,0)*density(a,time,0);
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = (delzrat/2.0)*thermalcond(a,3,0,time);
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = ( (delxrat/2.0)*thermalcond(a,2,1,time) ) ; /* kn+1 */
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
righthand[a] = -( convect*deltax*deltaz*theta0 +
(delvol/2.0)*cp(a, time, 1 )*(density(a, time, 1 )/deltat)*theta[a][0] ) ;
y++;}
else /* middle nodes */
{
value = -( (delyrat)*thermalcond(a,1,0,time) +
( delxrat)*thermalcond(a,0,0,time) +
( delxrat)*thermalcond(a,1,1,time) +
( delzrat)*thermalcond(a,0,1,time) +
( delzrat)*thermalcond(a,3,1,time) +
( delzrat)*thermalcond(a,3,0,time) +
velocity*deltaz*(deltay)*cp(a, time, 1 )*density(a, time, 1 ) +
delvol*cp(a, time, 1 )*(density(a, time, 1 )/deltat) ) ;
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = (delxrat)*thermalcond(a,1,1,time);
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = (delyrat)*thermalcond(a,1,0,time);
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = (delxrat)*thermalcond(a,0,1,time);
spADD_REAL_ELEMENT ( add[b], value ) ; b++;
value = (delyrat)*thermalcond(a,0,0,time) +
righthand[a] = -( delvol*qdot(a, x, y) +
delvol*cp(a, time, 1 )*(density(a, time, 1 )/deltat)*theta[a][0] ) ;
y++;}
}
a = (nptx1 + nptx2 - 1)*npty + 1 + nptlay*q;
value = -( (three4)*delyrat*thermalcond(a,4,0,time) + /* km+1 */
(three4)*delxrat*thermalcond(a,2,1,time) + /* kn+1 */
(delzrat/4.0)*thermalcond(a,3,1,time) +
(delzrat/4.0)*thermalcond(a,3,0,time) +
convect*deltaz*((deltax+deltay)/2.0) +
deltaz*((deltax+deltay)/2.0)*emisboltz*new[a][0]*new[a][0]*new[a][0] +
(delvol/4.0)*cp(a, time, 1 )*(density(a, time, 1 )/deltat) ) ;

68
spADD_REAL_ELEMENT ( add[b].value ) ; b++;
value = delxrat*thermalcond( a,2,1,time); /* kn+1 */
spADD_REAL_ELEMENT ( add[b].value ) ; b++;
value = delyrat*thermalcond( a,4,0,time);
spADD_REAL_ELEMENT ( add[b].value ) ; b++;
value = (delzrat/4.0)*thermalcond( a,3,1,time);
spADD_REAL_ELEMENT ( add[b].value ) ; b++;
value = (delzrat/4.0)*thermalcond( a,3,0,time);
spADD_REAL_ELEMENT ( add[b].value ) ; b++;
value = -(delxrat/4.0)*thermalcond( a,2,1,time); /* kn+1 */
spADD_REAL_ELEMENT ( add[b].value ) ; b++;
value = -(delyrat/4.0)*thermalcond( a,4,0,time);
spADD_REAL_ELEMENT ( add[b].value ) ; b++;

righthand[a] = -(convect*deltaz*((deltax+deltay)/2.0)*theta0 +
(delvol/4.0)*cp(a, time, 1)*density(a, time, 1)/deltat)*theta[a][0] ;
for (a = (nptx1 + nptx2 - 1)*npty + 2 + nptlay*q; a < (nptx1 + nptx2)*npty + nptlay*q; a++)
{
value = -( (3*delxrat/2.0)*thermalcond( a,4,0,time) +
(delxrat/2.0)*thermalcond( a,1,1,time) +
(delxrat/2.0)*thermalcond( a,0,1,time) +
(delxrat/2.0)*thermalcond( a,3,1,time) +
(delxrat/2.0)*thermalcond( a,3,0,time) +
deltay*deltaz*convect +
deltay*deltaz*emisboltz*new[a][0]*new[a][0]*new[a][0] +
(delvol/2.0)*cp(a, time, 1)*density(a, time, 1)/deltat) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++;
value = (delxrat/2.0)*thermalcond( a,1,1,time);
spADD_REAL_ELEMENT ( add[b].value ) ; b++;
value = (delxrat/2.0)*thermalcond( a,0,1,time);
spADD_REAL_ELEMENT ( add[b].value ) ; b++;
value = (2.0*delxrat)*thermalcond( a,4,0,time);
spADD_REAL_ELEMENT ( add[b].value ) ; b++;
value = (delxrat/2.0)*thermalcond( a,3,1,time);
spADD_REAL_ELEMENT ( add[b].value ) ; b++;
value = (delxrat/2.0)*thermalcond( a,3,0,time);
spADD_REAL_ELEMENT ( add[b].value ) ; b++;
value = -(delxrat/2.0)*thermalcond( a,4,0,time));
spADD_REAL_ELEMENT ( add[b].value ) ; b++;
righthand[a] = -(deltay*deltaz*convect*theta0 +
(delvol/2.0)*cp(a, time, 1)*density(a, time, 1)/deltat)*theta[a][0] ;
}
a = (nptx1 + nptx2)*npty + nptlay*q;
value = -(three4)*delxrat*thermalcond( a,2,0,time) +/* kn-1 */
(three4)*delyrat*thermalcond( a,4,0,time) +/* km-1 */
(delzrat/4.0)*thermalcond( a,3,1,time) +
(delzrat/4.0)*thermalcond( a,3,0,time) +

69
\[ \text{convect}\ast \text{deltaz}\ast((\text{deltax}+\text{delay})/2.0) + \]
\[ \text{deltax}\ast((\text{deltax}+\text{delay})/2.0)\ast\text{emisboltz}\ast\text{new}[a][0]\ast\text{new}[a][0]\ast\text{new}[a][0] + \]
\[ (\text{deltvol}/4.0)\ast\text{cp}(a, \text{time}, 1)\ast(\text{density}\ast(1, 1)/\text{deltat})\];
\]
\[ \text{spADD_REAL_ELEMENT}(\text{add}[b], \text{value}) ; b++ ; \]
\[ \text{value} = \text{deltax}\ast\text{thermalcond}(a, 2, 0, \text{time}) \ast / \text{kn-1} * / \]
\[ \text{spADD_REAL_ELEMENT}(\text{add}[b], \text{value}) ; b++ ; \]
\[ \text{value} = \text{deltax}\ast\text{thermalcond}(a, 4, 0, \text{time}) ; \]
\[ \text{spADD_REAL_ELEMENT}(\text{add}[b], \text{value}) ; b++ ; \]
\[ \text{value} = (\text{deltax}/4.0)\ast\text{thermalcond}(a, 3, 1, \text{time}) ; \]
\[ \text{spADD_REAL_ELEMENT}(\text{add}[b], \text{value}) ; b++ ; \]
\[ \text{value} = (\text{deltax}/4.0)\ast\text{thermalcond}(a, 3, 0, \text{time}) ; \]
\[ \text{spADD_REAL_ELEMENT}(\text{add}[b], \text{value}) ; b++ ; \]
\[ \text{value} = -(\text{deltax}/4.0)\ast\text{thermalcond}(a, 2, 0, \text{time}) ; \ast / \text{kn-1} * / \]
\[ \text{spADD_REAL_ELEMENT}(\text{add}[b], \text{value}) ; b++ ; \]
\[ \text{value} = -(\text{deltax}/4.0)\ast\text{thermalcond}(a, 4, 0, \text{time}) ; \ast / \text{kn-1} * ; \]
\[ \text{spADD_REAL_ELEMENT}(\text{add}[b], \text{value}) ; b++ ; \]
\[ \text{righthand}[a] = -(\text{convect}\ast\text{deltaz}\ast((\text{deltax}+\text{delay})/2.0)\ast\text{theta}_0 + \]
\[ (\text{deltvol}/4.0)\ast\text{cp}(a, \text{time}, 1)\ast(\text{density}\ast(1, 1)/\text{deltat})\ast\text{theta}[a][0]) ; \]

/* ------ Top Layer ------------------------- */
\[ a = 1 + (\text{nptz}-1)\ast\text{nptlay} ; \text{zdepth} = \text{nptz} - 1 ; \]
\[ \text{value} = -(\text{three8}\ast\text{deltax}\ast\text{thermalcond}(a, 4, 1, \text{time}) + / \text{kn+1} * / \]
\[ \text{three8}\ast\text{deltax}\ast\text{thermalcond}(a, 2, 1, \text{time}) + / \text{kn+1} * / \]
\[ \text{three8}\ast\text{deltax}\ast\text{thermalcond}(a, 6, 0, \text{time}) + / \text{kn-1} * / \]
\[ \text{convect}\ast\text{deltax}\ast((\text{deltax}+\text{delay})/4.0) + \]
\[ \text{convect}\ast((\text{deltax}+\text{delay})/4.0) + \]
\[ (\text{deltax}\ast((\text{deltax}+\text{delay})/4.0) + (\text{deltax}\ast\text{deltay}/4.0) \ast \text{emisboltz}\ast\text{new}[a][0]\ast\text{new}[a][0]\ast\text{new}[a][0] + \]
\[ (\text{deltvol}/8.3)\ast\text{cp}(a, \text{time}, 1)\ast(\text{density}\ast(1, 1)/\text{deltat}) ; \]
\[ \text{spADD_REAL_ELEMENT}(\text{add}[b], \text{value}) ; b++ ; \]
\[ \text{value} = (\text{deltax}/2.0)\ast\text{thermalcond}(a, 2, 1, \text{time}) ; / \text{kn+1} * / \]
\[ \text{spADD_REAL_ELEMENT}(\text{add}[b], \text{value}) ; b++ ; \]
\[ \text{value} = (\text{deltax}/2.0)\ast\text{thermalcond}(a, 4, 1, \text{time}) ; \]
\[ \text{spADD_REAL_ELEMENT}(\text{add}[b], \text{value}) ; b++ ; \]
\[ \text{value} = (\text{deltax}/2.0)\ast\text{thermalcond}(a, 6, 0, \text{time}) ; \]
\[ \text{spADD_REAL_ELEMENT}(\text{add}[b], \text{value}) ; b++ ; \]
\[ \text{value} = -(\text{deltax}/8.0)\ast\text{thermalcond}(a, 2, 1, \text{time}) ; / \text{kn+1} * / \]
\[ \text{spADD_REAL_ELEMENT}(\text{add}[b], \text{value}) ; b++ ; \]
\[ \text{value} = -(\text{deltax}/8.0)\ast\text{thermalcond}(a, 4, 1, \text{time}) ; \]
\[ \text{spADD_REAL_ELEMENT}(\text{add}[b], \text{value}) ; b++ ; \]
\[ \text{value} = -(\text{deltax}/8.0)\ast\text{thermalcond}(a, 6, 0, \text{time}) ; \]
\[ \text{spADD_REAL_ELEMENT}(\text{add}[b], \text{value}) ; b++ ; \]
\[ \text{righthand}[a] = -(\text{convect}\ast\text{deltaz}\ast((\text{deltax}+\text{delay})/4.0) + \]
\[ \text{convect}\ast((\text{deltax}+\text{delay})/4.0) \ast \text{theta}_0 + \]
\[ (\text{deltvol}/8.0)\ast\text{cp}(a, \text{time}, 1)\ast(\text{density}\ast(1, 1)/\text{deltat})\ast\text{theta}[a][0]) ; \]

for (a = 2 + (nptz-1)\ast\text{nptlay} ; a < \text{npty} + (nptz-1)\ast\text{nptlay} ; a++)
\[
\text{value} = -(3\text{three}4*\text{delyrat})*\text{thermalcond}(a,4,1,\text{time}) + \\
(\text{delyrat}/4.0)*\text{thermalcond}(a,1,1,\text{time}) + \\
(\text{delyrat}/4.0)*\text{thermalcond}(a,0,0,\text{time}) + \\
(\text{three}4*\text{delyrat})*\text{thermalcond}(a,6,6,\text{time}) + \\
\text{convect}^{*}\text{delray}^{*}(\text{deltaz}/2.0) + \\
\text{convect}^{*}\text{delray}^{*}(\text{deltaz}/2.0) + \\
(\text{deltaz}/2.0) + ((\text{deltax}^{*}\text{deltay}/2.0) \\
*\text{emisboltz}^{*}\text{new}[a][0]^{*}\text{new}[a][0]^{*}\text{new}[a][0]^{*} \\
(\text{delvol}/4.0)^{*}\text{cp}(a,\text{time},1)^{*}(\text{density}(a,\text{time},1)/\text{deltat})) ;
\]

spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = (\text{delyrat}/4.0)^{*}\text{thermalcond}(a,1,1,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = \text{delyrat}^{*}\text{thermalcond}(a,4,4,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = (\text{delyrat}/4.0)^{*}\text{thermalcond}(a,0,1,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = \text{delyrat}^{*}\text{thermalcond}(a,6,0,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,4,1,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,6,0,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,4,1,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,6,0,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,4,1,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,6,0,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,4,1,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,6,0,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,4,1,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,6,0,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,4,1,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,6,0,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,4,1,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,6,0,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,4,1,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,6,0,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,4,1,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,6,0,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,4,1,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,6,0,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,4,1,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,6,0,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,4,1,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,6,0,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,4,1,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,6,0,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,4,1,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,6,0,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,4,1,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,6,0,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,4,1,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,6,0,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,4,1,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,6,0,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,4,1,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,6,0,\text{time}) ;
spADD_REAL_ELEMENT ( add[b].value ) ; b++ ;
value = -(\text{delyrat}/4.0)^{*}\text{thermalcond}(a,4,1,\text{
spADD_REAL_ELEMENT ( add[b].value ) ; b++;
value = -( (deltat/8.0)*thermalcond(a,6,0,time) ); /* kz-1 */
spADD_REAL_ELEMENT ( add[b].value ) ; b++;
righthand[a] = -( (convect*delzax*(deltax*deltay)/4.0) +
             convect*((deltax*deltay)/4.0) )*theta0 +
             (deltvol/8.0)*cp(a, time, 1)*(density(a, time, 1)/deltat)*theta[a][0] ;
x = 2; y = 1;
for ( a = nptya + 1 + (nptya-1)*nptlay; a < (nptxa + nptxa2 - 1)*nptya + 1 + (nptya-1)*nptlay; a++ )
{
  if ( a % nptya == 0 ) /* top nodes */
  {
    value = -( (delyrat/4.0)*thermalcond(a,1,0,time) +
              (delyrat/4.0)*thermalcond(a,0,0,time) +
              (three4)*deltax*thermalcond(a,2,0,time) ) / kn-1 */
    (three4)*deltax*thermalcond(a,6,0,time) + /* kz-1 */
    convect*deltax*(deltay/2.0) +
    convect*deltax*(deltax/2.0) +
    (deltax*(deltax/2.0) + ((deltax*deltay)/2.0) )
    *emisboltz[new[a][0]*new[a][0]*new[a][0] + velocity*deltax*(deltay/4.0)*cp(a, time, 1) *density(a, time, 1) +
    (deltvol/4.0)*cp(a, time, 1) *density(a, time, 1) /deltat) ;
    spADD_REAL_ELEMENT ( add[b].value ) ; b++;
    value = (delyrat/4.0)*thermalcond(a,1,0,time) ;
    spADD_REAL_ELEMENT ( add[b].value ) ; b++;
    value = delzax*thermalcond(a,2,0,0); /* kn-1 */
    spADD_REAL_ELEMENT ( add[b].value ) ; b++;
    value = -( (delyrat/4.0)*thermalcond(a,2,0,time) ); /* kn-1 */
    spADD_REAL_ELEMENT ( add[b].value ) ; b++;
    value = -( (delyrat/4.0)*thermalcond(a,6,0,time) ); /* kz-1 */
    spADD_REAL_ELEMENT ( add[b].value ) ; b++;
    righthand[a] = -( (convect*deltax*(deltax/2.0) +
                     convect*deltax*(deltay/2.0) )*theta0 +
                 (deltvol/4.0)*cp(a, time, 1) *density(a, time, 1) /deltat)*theta[a][0] ) ;
x++; y = 1;
  }
else if ( a % nptya == 1 ) /* bottom nodes */
{
  value = -( (delyrat/4.0)*thermalcond(a,1,0,time) +
             (delyrat/4.0)*thermalcond(a,0,0,time) +
             (three4)*delzax*thermalcond(a,2,1,time) + /* kn+1 */
             (three4)*delzax*thermalcond(a,6,0,time) + /* kz-1 */

convect*deltax*(deltaz/2.0) +
convect*deltax*(deltay/2.0) +
( deltaz*(deltax/2.0) + ((deltax*deltay)/2.0) )
*emisboltz*[new[0][0]*new[0][0]*new[0][0]
 + velocity*deltaz*(deltay/4.0)*cp( a, time, 1 )*density( a, time, 1 ) +
(delvol/4.0)*cp( a, time, 1 )*density( a, time, 1 )/deltat )
); spADD_REAL_ELEMENT ( add[b],value ) ; b++;
value = delxrat*thermalcond( a,2,1,time); /* kn+1 */
spADD_REAL_ELEMENT ( add[b],value ) ; b++;
value = (delyrat/4.0)*thermalcond( a,1,0,time);
spADD_REAL_ELEMENT ( add[b],value ) ; b++;
value = (delyrat/4.0)*thermalcond( a,0,0,time) + velocity*deltaz*(deltay/4.0)*cp( a, time, 0 )
)*density( a, time, 0 );
spADD_REAL_ELEMENT ( add[b],value ) ; b++;
value = delzrat*thermalcond( a,6,0,time); /* kz-1 */
spADD_REAL_ELEMENT ( add[b],value ) ; b++;
value = -( (delxrat/4.0)*thermalcond( a,2,1,time) ); /* kn+1 */
spADD_REAL_ELEMENT ( add[b],value ) ; b++;
value = -( (delyrat/4.0)*thermalcond( a,6,0,time) ); /* kz-1 */
spADD_REAL_ELEMENT ( add[b],value ) ; b++;
righthand[a] = -( (convect*deltax*(deltaz/2.0) +
convect*deltax*(deltay/2.0) )*theta0 +
(delvol/4.0)*cp( a, time, 1 )*density( a, time, 1 )/deltat)*theta[a][0] );
y++;
}
else /* Top: middle nodes */
{
value = -( (delyrat/2.0)*thermalcond( a,1,0,time) +
(delyrat/2.0)*thermalcond( a,0,0,time) +
(delxrat/2.0)*thermalcond( a,1,1,time) +
(delxrat/2.0)*thermalcond( a,0,1,time) +
(3.0*deltaz/2.0)*thermalcond( a,6,0,time) + /* kz-1 */
convect*deltax*deltay +
deltaz*deltay*emisboltz*[new[0][0]*new[0][0]*new[0][0]
 + velocity*deltaz*(deltay/2.0)*cp( a, time, 1 )*density( a, time, 1 ) +
(delvol/2.0)*cp( a, time, 1 )*density( a, time, 1 )/deltat )
); spADD_REAL_ELEMENT ( add[b],value ) ; b++;
value = (delyrat/2.0)*thermalcond( a,1,1,time);
spADD_REAL_ELEMENT ( add[b],value ) ; b++;
value = (delyrat/2.0)*thermalcond( a,1,0,time);
spADD_REAL_ELEMENT ( add[b],value ) ; b++;
value = (delyrat/2.0)*thermalcond( a,0,1,time);
spADD_REAL_ELEMENT ( add[b],value ) ; b++;
value = (delyrat/2.0)*thermalcond( a,0,0,time) + velocity*deltaz*(deltay/2.0)*cp( a, time, 0 )
)*density( a, time, 0 );
spADD_REAL_ELEMENT ( add[b],value ) ; b++;

73
value = (2.0*delzrat)*thermalcond( a,6,0,time);
spADD_REAL_ELEMENT( add[b],value ) ; b++;
value = -( (delzrat/2.0)*thermalcond( a,6,0,time) ;
spADD_REAL_ELEMENT( add[b],value ) ; b++;
righthand[a] = -( (delvol/2.0)*qdot( a, x, y ) +
deltax*deltay*convect +
( delvol/2.0)*cp( a, time, 1 )* (density( a, time, 1 )/deltat)*theta[a][0] ) ;
y++;
}
/* Top : Rightside */
a = (nptx1 + nptx2 - 1)*npty + 1 + (nptz-1)*nptlay;
value = -( three8*delyrat*thermalcond( a,4,0,time) + /* km+1 */
three8*delzrat*thermalcond( a,2,1,time) + /* kn+1 */
three8*delzrat*thermalcond( a,6,0,time) + /* kz-1 */
convect*deltaz*((deltax+deltay)/4.0) +
convect*((deltaz*deltay)/4.0) +
( deltaz*((deltax+deltay)/4.0) + ((deltax*deltay)/4.0) )
*emisboltz*new[a][0]*new[a][0]*new[a][0] +
(delvol/8.0)*cp( a, time, 1 )*(density( a, time, 1 )/deltat ));
spADD_REAL_ELEMENT( add[b],value ) ; b++;
value = (delzrat/2.0)*thermalcond( a,2,1,time); /* kn+1 */
spADD_REAL_ELEMENT( add[b],value ) ; b++;

value = (delzrat/2.0)*thermalcond( a,6,0,time);
spADD_REAL_ELEMENT( add[b],value ) ; b++;
value = -( (delzrat/8.0)*thermalcond( a,2,1,time) ;/* kn+1 */
spADD_REAL_ELEMENT( add[b],value ) ; b++;
value = -( (delzrat/8.0)*thermalcond( a,4,0,time) ;
spADD_REAL_ELEMENT( add[b],value ) ; b++;
value = -( (delzrat/8.0)*thermalcond( a,6,0,time) ;
spADD_REAL_ELEMENT( add[b],value ) ; b++;
righthand[a] = -( (convect*deltaz*((deltax+deltay)/4.0) +
convect*((deltaz*deltay)/4.0)) *theta0 +
( delvol/8.0)*cp( a, time, 1 )*(density( a, time, 1 )/deltat)*theta[a][0] ) ;
/* Top : Middle Rightside */
for ( a = (nptx1 + nptx2 - 1)*npty + 2 + (nptz-1)*nptlay; a < nptz*nptlay; a++ )
{

value = -( (three4*delyrat)*thermalcond( a,4,0,time) +
(delxrat/4.0)*thermalcond( a,1,1,time) ) +
(delxrat/4.0)*thermalcond( a,0,1,time) +
(three4*delzrat)*thermalcond( a,6,0,time) +
deltay*(deltaz/2.0)*convect +
deltay*(deltay/2.0)*convect +
( delxrat*(deltax/2.0) + ((deltax*deltay)/2.0) )

74
*emisboltz*new[a][0]*new[a][0]*new[a][0] +
(dolv/4.0)*cp(a, time, 1)*(density(a, time, 1/deltat));
spADD_REAL_ELEMENT (add[b],value) ; b++;
value = (delxrat/4.0)*thermalcond(a,1,1,time);
spADD_REAL_ELEMENT (add[b],value) ; b++;
value = (delxrat/4.0)*thermalcond(a,0,1,time);
spADD_REAL_ELEMENT (add[b],value) ; b++;
value = delxrat*thermalcond(a,4,0, time);
spADD_REAL_ELEMENT (add[b],value) ; b++;
value = delxrat*thermalcond(a,6,0, time);
spADD_REAL_ELEMENT (add[b],value) ; b++;
value = -(delxrat/4.0)*thermalcond(a,4,0, time);
spADD_REAL_ELEMENT (add[b],value) ; b++;
value = -(delxrat/4.0)*thermalcond(a,6,0, time);
spADD_REAL_ELEMENT (add[b],value) ; b++;
righthand[a] = -(deltay*(deltaz/2.0)*convec +
deltay*(deltax/2.0)*convec)*theta0 +
(dolv/4.0)*cp(a, time, 1)*(density(a, time, 1/deltat)*theta[a][0]);
/* Top : Rightside */
a = nptz*nptlay;
value = -(three8*delxrat*thermalcond(a,2,0,time)) +/* kn-1 */
three8*delxrat*thermalcond(a,4,0, time) /* km-1 */
three8*delxrat*thermalcond(a,6,0, time) /* kz-1 */
convec*deltaz*((deltax+deltay)/4.0) +
convec*((deltax*deltay)/4.0) +
(deltay*((deltax+deltay)/4.0) +
(deltay*(deltax*deltay)/4.0) )
*emisboltz*new[a][0]*new[a][0]*new[a][0] +
(dolv/8.0)*cp(a, time, 1)*(density(a, time, 1/deltat));
spADD_REAL_ELEMENT (add[b],value) ; b++;
value = (delxrat/2.0)*thermalcond(a,2,0, time); /* kn-1 */
spADD_REAL_ELEMENT (add[b],value) ; b++;
value = (delxrat/2.0)*thermalcond(a,4,0, time);
spADD_REAL_ELEMENT (add[b],value) ; b++;
value = (delxrat/2.0)*thermalcond(a,6,0, time);
spADD_REAL_ELEMENT (add[b],value) ; b++;
value = -(delxrat/8.0)*thermalcond(a,2,0, time); /* kn-1 */
spADD_REAL_ELEMENT (add[b],value) ; b++;
value = -(delxrat/8.0)*thermalcond(a,4,0, time); /* km-1 */;
spADD_REAL_ELEMENT (add[b],value) ; b++;
value = -(delxrat/8.0)*thermalcond(a,6,0, time); /* kz-1 */;
spADD_REAL_ELEMENT (add[b],value) ; b++;
righthand[a] = -(convec*deltaz*((deltax+deltay)/4.0) +
convec*((deltax*deltay)/4.0) )*theta0 +
(dolv/8.0)*cp(a, time, 1)*(density(a, time, 1/deltat)*theta[a][0]);
void solvematrix ( char *solmatrix, double right[] )
{
    /*Error = spOrderAndFactor( Matrix, right, 0.001 , 0 , 1);
     PrintMatrixErrorMessage( Error );
     if( Error >= spFATAL )
         exit(1);*/
    Error = spFactor( solmatrix );
    if( Error != spOKAY ) PrintMatrixErrorMessage( Error );
    if( Error >= spFATAL ) exit(1);
    spSolve( solmatrix, right, Solution);
    /*for ( n=1; n < TOTALNODES; n++ )
     theta[n][timeinterval] = Solution [n];*/
    spClear ( solmatrix );
}

/* times is the solving time step */
double thermalcond ( int condtheta, int plusminus , int r_or_z, int times )
{
    double kf=1.0.0,ktemp,knode,kother;
    knode = kfunction ( new [condtheta][times-1] );
    if( plusminus == 1 )
    {
        if( r_or_z == 1 )
            kother= kfunction( new [condtheta + 1 ][times-1] );
        else
            kother= kfunction( new [condtheta + npty][times-1] );
        ktemp = (knode + kother)/2.0;
    }
    else if( plusminus == 0 )
    {
        if( r_or_z == 1 )
            kother= kfunction( new [condtheta - 1 ][times-1] );
        else
            kother= kfunction( new [condtheta - npty][times-1] );
        ktemp = (knode + kother)/2.0;
    }
    else if( plusminus == 2 ) /* For k(n+1) */
    {
        if( r_or_z == 1 )
            ktemp = kfunction( new [condtheta + 1][times-1] );
        else
            ktemp = kfunction( new [condtheta - 1][times-1] );
    }
    else if( plusminus == 4 )/* For k(m+1) */
}
{  
    if ( r_or_z == 1 )
        ktemp = kfunction( new [condtheta + npty][times-1] );
    else
        ktemp = kfunction( new [condtheta - npty][times-1] );
}
else if ( plusminus == 3 )/* For k(z-between) */
{
    if ( r_or_z == 1 )
        kother = kfunction( new [condtheta + nptlay][times-1] );
    else
        kother = kfunction( new [condtheta - nptlay][times-1] );
    ktemp = (knode + kother)/2.0;
}
else if ( plusminus == 6 )/* For k(z+1) */
{
    if ( r_or_z == 1 )
        ktemp = kfunction( new [condtheta + nptlay][times-1] );
    else
        ktemp = kfunction( new [condtheta - nptlay][times-1] );
}
else
{
    kother = kfunction( theta0 );
    ktemp = (knode + kother)/2.0;
}
/* ktemp = 50.0; */
return ktemp ;

/* Thermal Conductivity function for Alumina */
double kfunction ( double ktheta )
{
    double kvalue,cal;
    cal = 0.55*(6.788.0/( ktheta - 125.0))+(3.562e-33)*pow( ktheta , 10.0 );
    kvalue = cal + ((1.6/3.0)*pow( 1.2, 2.0))*(5.669e-8)*pow( ktheta , 3.0 ))/1.0e7;
    return kvalue;
}

/* Specific Heat function for Alumina */
double cp( int cptheta, int cptime, int plus )
{
    double specific,cpvalue, ctemp, sum, cons[7];int h; specific = 675.0;
    //
if (plus == 1)
{
    cptemp = new [ctheta][ctime-1];
}
else if (plus == 0)
{
    cptemp = new [ctheta-npt][ctime-1];
}
else
{
    cptemp = theta0;
}
sum = 0.0;
for (h = 1; h < 6; h++)
{
    sum = cptemp*(sum + cons[6-h]);
}
cpvalue = sum + cons[0];

/*
if ((new [ctheta][ctime-1]+tempsur) <= 400)
{
    cpvalue = 64.5 + 2.0389*(new [ctheta][ctime-1]+tempsur);
}
else if((new [ctheta][ctime-1]+tempsur) > 400 && (new [ctheta][ctime-1]+tempsur) <= 900)
{
    cpvalue = 358.179 + 1.66146*(new [ctheta][ctime-1]+tempsur) -
    0.000854589*(new [ctheta][ctime-1]+tempsur)*(new [ctheta][ctime-1]+tempsur);
}
else
{
    cpvalue = 964.389 + 0.230072*(new [ctheta][ctime-1]+tempsur);
} */
return cpvalue;

/* For Alumina */
double density( int dentheta, int dentime, int denplus)
{
    double denvalue, dens, q, z;
    q=1.92;
    z=1.07e-03;
    if (denplus == 1)
    {
if (new[dentheta][dentime-1] <= 1872.0)
dens=q+z*new[dentheta][dentime-1];
else

dens=3.92;
}
else
{

if (new[dentheta - npty][dentime-1] <= 1872.0)
dens=q+z*new[dentheta - npty][dentime-1];
else

dens=3.92;
}
denvalue = 3000;
dens = dens/0.001;
return dens;
;

double qdot ( int qtheta, int npx, int npy )
{

double qvalue, xdist, ydist;

xdist = fabs( length1 - deltax*(npx - 1) );
ydist = fabs( width/2.0 - deltax*(npy - 1) );
/* if ( xdist < deltax*((npxq - 1)/2)+0.000001 && ydist < deltax*((nptqy - 1)/2)+0.000001 ) */
if ( (xdist*xdist + ydist*ydist) < (deltax*deltax*((npxq-1)/2.0)*((nptq-1)/2.0)+ 0.0000001) )
{

qvalue = heatgener( qtheta , 1 );
qvalue = guass( qvalue, xdist, ydist );
if ( zdepth == nptz || zdepth == nptz - 1 )
	power_sum = power_sum + qvalue*(delvol/2.0);
else
	power_sum = power_sum + qvalue*delvol;
}
else

qvalue = 0.0;
return qvalue;
;
}

double heatgener( int heattheta , int heatime )
{

double heat,freq=frequency/*245000000.0*/0.0000000000008854,electf_o=electric_field, electf=

= electf_o*atten( heattheta );
heat = 2.0*3.14159*freq*eppa( heattheta, heatime)*electf*electf;
return heat;
;
79
void

double atten(int attenthteta)
{
    int atop, f; double attenvalue, sum, litan, mu, attfreq, sp_light = 3.0e8;
    attfreq = frequency*2.0*3.1415926;
    atop = attenthteta + (zdepth-2)*nptlay;
    sum = 0.0;
    for ( f = 1; f < zdepth; f++)
    {
        litan = eppa(atop, 1)/epa(atop);
        mu = sqrt(0.5*(sqrt(1+litan*litan)-1.0));
        sum = sum + mu*deltaz;
        atop = atop - nptlay;
    }
    attenvalue = exp(-(attfreq/sp_light)*sum);
    return attenvalue;
}

double eppa(int epatheta)
{
    double epavalue;
    epavalue = 8.5 + 8.33e-4*(new[epatheta][0] - 273.0);
    return epavalue;
}

/* Calculates the dielectric loss */
double eppa(int epatheta, int epptime)
{
    double epp, test;
    test = 0.03;
    epp = -0.1039 + (3.114e-4)*new[epatheta][epptime-1];
    if ( epp < test )
        epp = test,
    return epp;
}

double guass(double maxvalue, double guassx, double guassy)
{
    double guassvalue;
    guassvalue = maxvalue*exp(-(guassx*guassx + guassy*guassy)/
        (deltax*deltax*((nptxq-1)/2.0)*((nptxq-1)/2.0)));
}
return guassvalue;

/* Error messages from Sparse 1.3 */

void PrintMatrixErrorMessage( int Error )
{
    int Row, Col;

    /* Begin 'PrintMatrixErrorMessage'. */

    if (Error == spOKAY)
        return;
    if (Error >= spFATAL)
    {
        fprintf(stderr, ": fatal error detected in file ", "n");
    }
    else fprintf(stderr, ": warning in ", "n");

    if (Error == spNO_MEMORY)
        fprintf(stderr, " Insufficient memory available.
"");
    else if (Error == spSINGULAR)
    {
        spWhereSingular( Matrix, &Row, &Col );
        printf(" Singular matrix (detected at row %d, column %d).\n", Row, Col);
    }
    else if (Error == spZERO_DIAG)
    {
        spWhereSingular( Matrix, &Row, &Col );
        printf(" Zero diagonal detected at row %d, column %d.\n", Row, Col);
    }
    else if (Error == spPANIC)
        fprintf(stderr, " Matrix routines being used improperly.\n"");
    else if (Error == spSMALL_PIVOT)
        fprintf(stderr, " A pivot was chosen that is smaller than the absolute threshold.\n"");
    return;
    /* End of Code */
Sample Input File

0.01 0.01 0.003 Diameter of the beam, Diameter of the beam, Thickness of the sample
16 15 21 4 Number of Nodes before the beam(x-direction), Number of Nodes after the beam(x-direction), Number of Nodes in the y-direction, Number of Nodes in the z-direction
11 11 Number of Nodes for the beam(x-direction), Number of Nodes for the beam(y-direction)
2450000000.0 155000.0 Frequency, Electric Field Strength
0.0 0 Velocity, Flag to determine if the code should start at time t = 0
3 1.0 Total Number of timesteps, Timestep
0 10 Flag for writing animation files, Number of animation files to write
Program Flowchart for Solving Temperatures

1. Begin Timestep Loop
2. Construct Coefficient Matrix with $T(p)$
3. Solve for $T(p+1)_{old}$ Temperatures
4. Construct Coefficient Matrix with $T(p+1)_{old}$
5. Solve for $T(p+1)_{new}$ Temperatures
6. Are $T(p+1)_{old}$ and $T(p+1)_{new}$ the same value within the tolerance?
   - No: Set $T(p+1)_{new}$ equal to $T(p+1)_{old}$
   - Yes: Set $T(p+1)_{new}$ equal to $T(p)$
Vita

The author was born on August 30, 1971 and continues to breathe.......

Rollin M. Roos