A THERMAL ANALYSIS TOOL FOR THREE-DIMENSIONAL MODELS OF MULTILAYER MICROELECTRONICS

by

Kenneth E. Creel

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

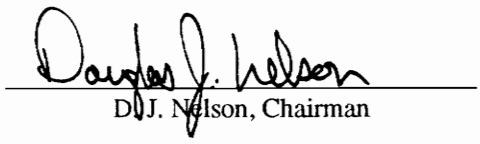
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

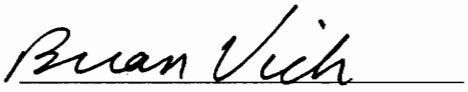
in

Mechanical Engineering

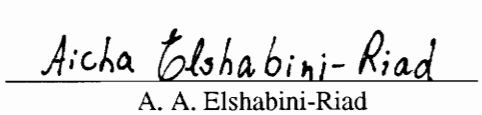
APPROVED:



D. J. Nelson, Chairman



B. Vick



A. A. Elshabini-Riad

July, 1994
Blacksburg, Virginia

A THERMAL ANALYSIS TOOL FOR THREE-DIMENSIONAL MODELS OF MULTILAYER MICROELECTRONICS

by
Kenneth E. Creel

Committee Chairman: D. J. Nelson
Mechanical Engineering

(ABSTRACT)

This work details a computer-based modeling tool for predicting temperatures in three-dimensional multilayer microelectronic packages. It is capable of modeling surface connections (e.g., wire bonds and pins), edge connections (e.g., leads), and thermal vias. A three-dimensional control-volume finite difference method is used, permitting transient as well as steady solutions. Numerical behavior is examined with respect to the device geometry and external environment. The features of this tool are demonstrated on a sample multilayer package. The effects of the modeling scheme are discussed.

An alternate version of the program removes a layer from the numerical model to simplify the solution of the problem. To compensate for the removal of the layer a contact resistance is added. This replaces the thermal resistance the removed layer provided in the z direction. The x-y conductivity of the adjacent layers are modified based on the removed layer thickness and conductivity. This measure imitates the spreading resistance or conductance that the removed layer provided. The effect of removing a layer in the model is studied, documenting the relationship between layer thickness and conductivity and the error introduced by removing the layer. A simple relationship is sought which can indicate the instances in which the computer model can be simplified. The results are applicable to any method including finite element and series-analytical methods.

Acknowledgments

I would like to express my appreciation to Doug Nelson for his untiring and good-natured support of my efforts. His insight into my problems, tolerance of my mistakes, and expertise at solving them was invaluable. Without his help, I would not have graduated, much less learned how to apply my education.

I would like to give thanks to my mother for her steadfast support throughout my life. I credit my father for imbuing me with some fraction of the intangible qualities that made him so competent. I would also like to recognize my sister for her acerbic yet genuine encouragement.

I would also like to thank Dr. Vick and Dr. Riad for serving on my committee.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

1.1 Problem Definition

The design of microelectronic packages must account for the thermal behavior of the device. Proper thermal design insures that the device will not exceed its intended range of operation and malfunction due to thermal failure. To predict a physical system's behavior, it is necessary to calculate the heat distribution throughout the device, with particular attention to the high temperature areas, which are at the highest risk of failure.

The objective of this computer program is to thermally model a three-dimensional microelectronic package using numerical techniques. Methods are sought to model typical microelectronic elements such as heat sources, wire connections, and thermal vias. The program offers the option of simplifying the numerical model to produce faster solutions.

1.2 Numerical Solutions

The advent of modern computing has been a boon to the study of mathematically complex engineering problems such as heat transfer. Although many heat transfer problems can be solved analytically, practical problems quickly approach the level of complexity where an analytical solution is either impractical or impossible. Here is where the use of numerical methods and computers yields the greatest benefit. A physical model and set of governing equations can be converted into a numerical model with a corresponding collection of algebraic equations. This set of equations can then be solved, showing the behavior of the system.

A numerical model is actually an approximation of the physical model. However, if it is constructed in a mathematically consistent manner, the numerical solution should approach the exact solution at sufficient resolution. The exception to this is cases in which the formulation of the problem is such as to cause inconsistencies within the numerical model.

1.3 Previous Work

TAMS (Thermal Analyzer for Multilayer Structures) is a well-established tool used to thermally model electronic equipment (1). It uses a Fourier-series method to obtain a solution for multilayer devices. Anisotropic conductivities can be used for each layer. Heat sources and multiple thermal resistances can be specified for the top and bottom surface, in addition to the ambient and source temperatures.

Sweet, et al.(2) examines a three-dimensional multichip-module with a heat sink acting on the bottom surface. The maximum temperature is measured experimentally and compared to the results obtained from a full finite element model and from TAMS. Attention is directed towards finding regions of excessive thermal resistance in the device. The results suggest that the die attach region is responsible for much of the thermal resistance.

Arbeitman (3) addresses the issue of node spacing versus accuracy. Using ASTAP, a thermal modeling tool, he seeks to obtain the coarsest spacing which will give suitable accuracy. The solutions are compared to that obtained using a finite element

program. He then proposes guidelines for mesh selection, based on the characteristics of the device and the environment.

Lee, et al. (4) develops an analytical approach to modeling via networks. Using closed form expressions, he is able to calculate the complete thermal resistance between the die and substrate. He calculates this resistance by combining the thermal conductivity of the via network in a series with the adjacent layers. This permits thermal spreading through the via network and the associated layers such as epoxy and thermal grease. He examines a set of via islands in a four-layer polyamide dielectric. Predictions using the method compare favorably to both numerical and experimental results.

Nelson and Sayers (5) look at numerical approximation in a thermal model. They determine what conditions are necessary to accurately model a three-dimensional system using a two-dimensional model. Spreading resistances are incorporated into the two-dimensional model to more accurately mimic the three-dimensional system. Examination of the two-dimensional model finds that it is only accurate for thin substrates and low external thermal resistance.

A recent review of thermal modeling of electronic structures is given by Ellison (6). Ellison also describes a computer implementation of TAMS for a four layer structure in reference 7. Other recent work on multilayer structures includes Chien et al. (8) and Hussein et al. (9).

2. Method

2.1 Basis of the Program

This code is based on an extension of CONDUCT, a control-volume, finite-difference heat transfer program created by Suhas Patankar (10). The original program has been modified and extended in several ways. It has been expanded to three dimensions and modified to allow the use of orthotropic properties. The capability to add a contact resistance between layers in the z direction has been added to the code. Other additions allow the modeling of various microchip-specific geometries, such as surface and edge connections and thermal vias. The code has two versions, one which models the device fully and one which replaces a layer with a collection of approximations intended to mimic the effects of that layer. Removing the layer from the numerical model results in a faster solution at the possible expense of accuracy

2.2 Principle of the Control Volume Method

One of the most physically meaningful and easiest to understand numerical methods is the control volume method. This method takes a continuous domain and breaks it up into a discrete number of volumes, each with an associated grid point. The properties attached to each grid point are assumed to prevail over the control volume surrounding that grid point. That is to say, the physical system is broken into a discrete number of domains, in which the physical characteristics of each domain are assumed to be constant. The problem can now be expressed as a set of algebraic discretization equations

which describe a mathematical model analogous to the continuous physical system. A solution of this discrete model approximates the exact solution, and as the number of control volumes is increased, the numerical solution will approach the exact solution. If the system of equations is linear, it can be solved as a matrix. However, this is not efficient for large problems, so an iterative method is typically used.

Expressed in rectangular coordinates the general three-dimensional heat conduction equation is:

$$\rho c \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(k_x \frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_y \frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(k_z \frac{\partial T}{\partial z}\right) + S' \qquad (1)$$

where ρ is the density, c is the specific heat, and k is the conductivity. S' is the volumetric source over the domain of the equation. T, ρ, c, and S' may depend on x, y, z, and t. The conductivity is orthotropic. This equation can be used to describe any physical system. However, to do so it must be accompanied by both boundary conditions and initial conditions.

A more general differential equation can be expressed as:

$$\lambda \frac{\partial \Phi}{\partial t} = \frac{\partial}{\partial x}\left(\Gamma_x \frac{\partial \Phi}{\partial x}\right) + \frac{\partial}{\partial y}\left(\Gamma_y \frac{\partial \Phi}{\partial y}\right) + \frac{\partial}{\partial z}\left(\Gamma_z \frac{\partial \Phi}{\partial z}\right) + S' \qquad (2)$$

The heat conduction equation can be expressed in this form as well. λ is equivalent to the heat capacity ρc, Γ is equivalent to the conductivity, and Φ is equivalent to the temperature. Integrating this equation over a control volume (see Fig. 1) yields the

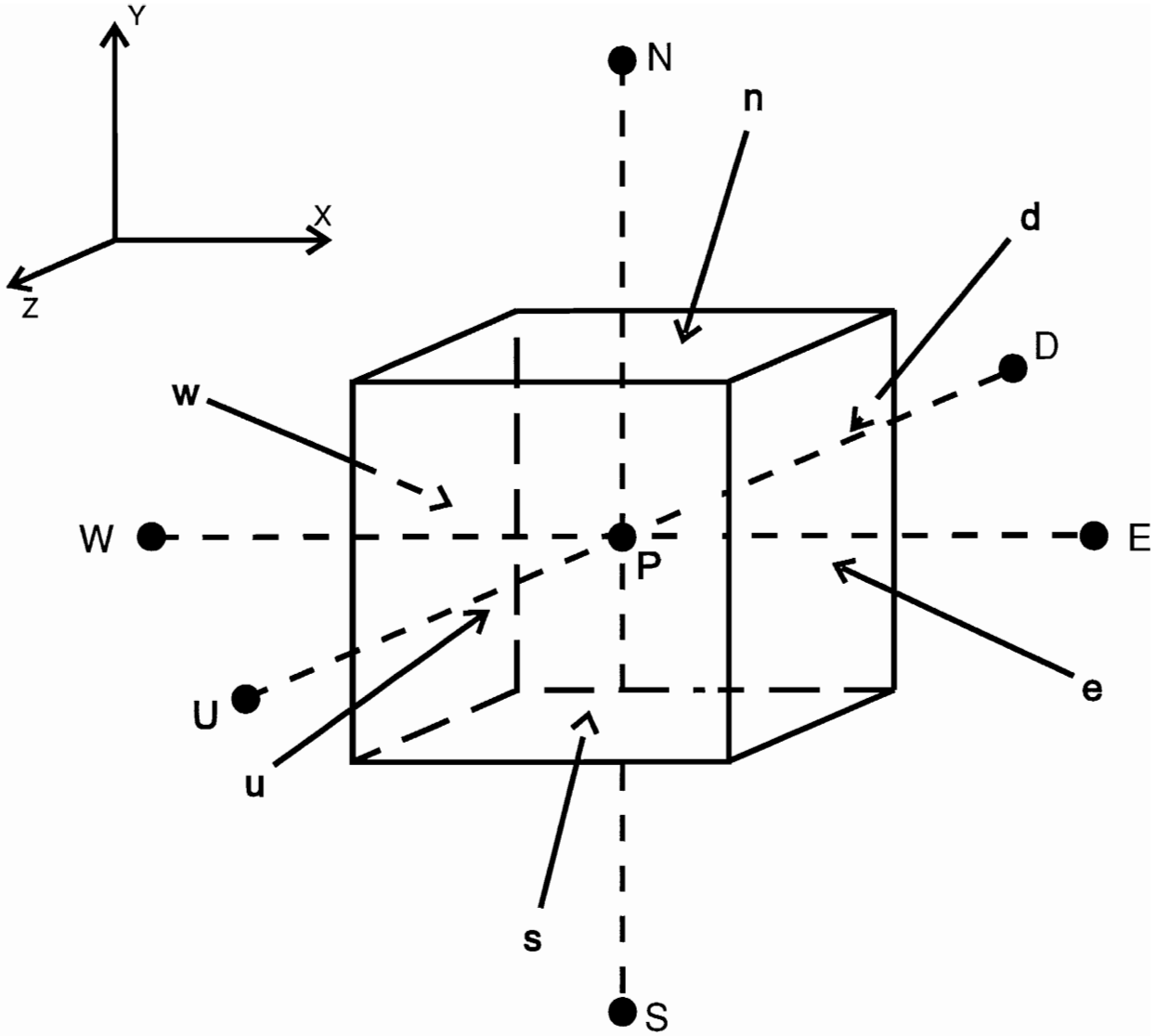


Figure 1: Control Volume Element and Neighboring Grid Points

three-dimensional discretization equation. (Note that the upper-case subscripts refer to the grid points and the lower-case subscripts refer to the control volume faces.)

$$\lambda_P \frac{\Delta V}{\Delta t}(\Phi_P - \Phi_P^0) = J_w A_w - J_e A_e + J_s A_s - J_n A_n + J_d A_d - J_u A_u + S\,\Delta V \qquad (3)$$

S is the average value of the source over the control volume. J represents the diffusion fluxes at the control volume faces:

$$J_e A_e = D_e(\Phi_P - \Phi_E) \qquad (4)$$
$$J_w A_w = D_w(\Phi_W - \Phi_P) \qquad (5)$$
$$J_n A_n = D_n(\Phi_P - \Phi_N) \qquad (6)$$
$$J_s A_s = D_s(\Phi_S - \Phi_P) \qquad (7)$$
$$J_u A_u = D_u(\Phi_P - \Phi_U) \qquad (8)$$
$$J_d A_d = D_d(\Phi_D - \Phi_P) \qquad (9)$$

where D is the diffusion conductance at the control volume face. This value is an average of the values of Γ at the grid points on either side of the control volume. For example,

$$D_u = A_u\left(\frac{(\delta x)_{u-}}{\Gamma_{zP}} + \frac{(\delta x)_{u+}}{\Gamma_{zU}}\right) \qquad (10)$$

$(\delta x)_{u-}$ is the distance between grid point P and control volume face u. $(\delta x)_{u+}$ is the distance between grid point U and control volume face u. This average yields an interface conductivity that is sensitive to the geometry of the control volume. The five other diffusion conductances are expressed in the same manner.

It is useful to define the source in terms of linear dependence on Φ:

$$S = S_C + S_P \Phi_P \qquad (11)$$

Using this definition of S the discretization equation can then be expressed as:

$$a_P\Phi_P = a_E\Phi_E + a_W\Phi_W + a_N\Phi_N + a_S\Phi_S + a_U\Phi_U + a_D\Phi_D + b \tag{12}$$

where

$$a_E=D_e \tag{13}$$
$$a_W=D_w \tag{14}$$
$$a_N=D_n \tag{15}$$
$$a_S=D_s \tag{16}$$
$$a_U=D_u \tag{17}$$
$$a_D=D_d \tag{18}$$

$$b = S_c\Delta V + a_P^0\Phi_p^0 \tag{19}$$

$$a_P^0 = \frac{\lambda_P\Delta V}{\Delta t} \tag{20}$$

$$a_P = a_E + a_W + a_N + a_S + a_U + a_D + a_P^0 - S_p\Delta V \tag{21}$$

This is the discretization for just one grid point. To describe a numerical model, a discretization equation is needed for each grid point that has an associated volume, in addition to boundary conditions and initial conditions. This produces a system of equations that can be solved for the dependent variable.

2.2.1 Source and Flux Terms

Only interior grid points have a source term (since they have an associated volume). Grid points on the boundary have a flux term (since they have an associated area). A source term is expressed as in the discretization equation:

$$S = S_C + S_P \cdot T_P \tag{22}$$

where S_C and S_P are constants and T_P is a grid point in the interior of the domain where the source term is defined. Fluxes are also expressed as a linear function of temperature:

$$F = F_C + F_P \cdot T_P \tag{23}$$

where F_C and F_P are constants and T_P is a grid point on the control-volume boundary where the flux term is defined. This manner of expressing the sources and fluxes easily handles such common situations as a constant heat input or a convective boundary condition. However, non-linear conditions would have to first be linearized, then applied to the numerical problem. Limiting the expression of source terms and fluxes to linear functions makes the iterative solution easier.

2.3 Layer Approximation

An alternate version of the code approximates a layer of the device (i.e., a three-layer physical model would become a two-layer numerical model). By removing a layer, the complexity of the numerical model is reduced, since there are less grid points included in the solution. This speeds the solution of the problem, but the removal of the layer must be compensated for. The thickness and conductivity of the removed layer are specified in the same manner as the other layers. However, the effect of the removed layer is replaced in the numerical model by a set of approximations. The conductivities of the adjacent layers are modified to reflect the role of the removed layer has in heat spreading in the x-y plane. A contact resistance is added to approximate the resistance the of the removed layer in the z-direction. These approximations can speed the solution of the problem, at the possible expense of accuracy. The approximations work best when the removed layer is thin with low relative conductivity. Details of this approximation scheme are discussed in chapter 6.

Much of the following discussion deals with modifications involving particular layers of the model. It is assumed that the user has accounted for any layer approximations being used. In some cases, the use of layer approximation may conflict with the placement of connections (e.g., connections to the removed layer). A decision must be made to either forego the approximation, or to modify the placement of the connections.

2.4 Structure of the Program

The program retains the original structure created by Patankar. It is organized into two main parts to simplify its use. The *invariant* part is responsible for calculating the solution to the model. The *adaptation* part is modified by the user to define the specifics of the problem to be solved. The adaptation subroutine in the program is divided into four sections: GRID, BEGIN, OUTPUT, and PHI. These are separated within the subroutine using the ENTRY command, which allows the invariant part of the code to call on the particular sections at appropriate times (see Fig. 2). GRID and BEGIN are called only once during execution, OUTPUT and PHI are called every iteration.

2.4.1 GRID

This part of the adaptation subroutine defines the dimensions of the model and the size of the grid. Here the user defines the number of control volumes in the x and y directions (the number of control volumes in the z direction is defined as NL using a

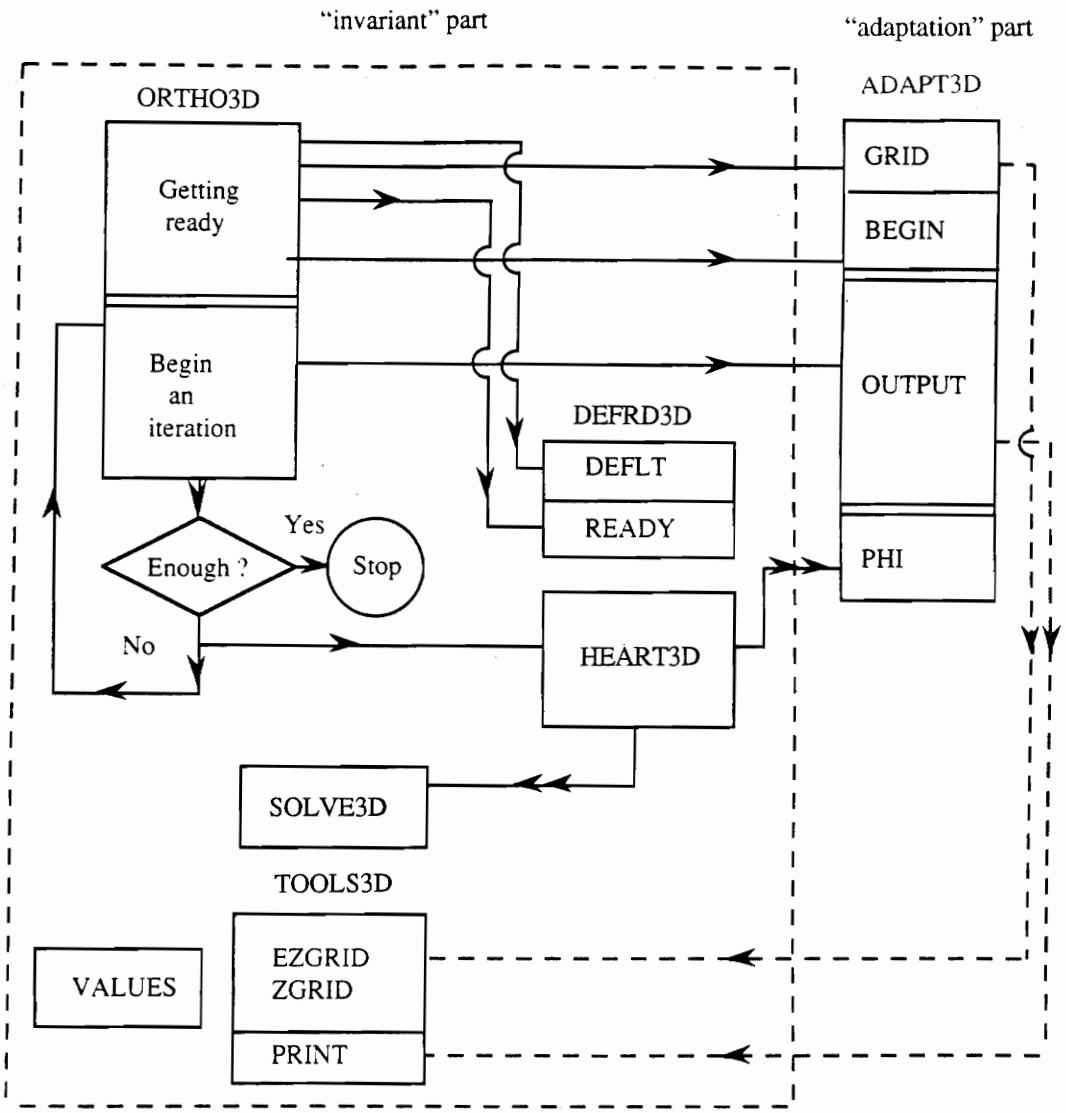


Figure 2: Structure of the Program

parameter statement at the beginning of the adaptation subroutine), the convergence criterion, the maximum number of iterations to use, and the conductivity and thickness of the layers. (Actually, the conductivity and thickness of the layers is defined in the PHI section. However, it is usually easier to assign variables to make changes easier.) The x and y dimensions are specified using XL and YL. A zoned grid is then defined, which allows the user to specify finer spacing in some regions and coarser in others. For instance, in the x direction, NZX defines the number of zones, then one-dimensional arrays XZONE and NCVX are used to define the size and number of control volumes in each zone. If a uniform grid spacing is desired in one or more directions, the number of zones is set to one, the size of zone 1 is set to the domain size in that direction, and the number of control volumes is that direction is defined.

2.4.2 BEGIN

BEGIN is only called once during the execution of the code. It is used to define the external and internal thermal characteristics of the model. The external environment is specified by defining the top and bottom convective cooling coefficients, HTOP and HBOT, and the associated temperatures, TTOP and TBOT. (The convective conditions applied to the top also act on the edges of the model.) The average temperatures of each layer are defined. The conductivities and thicknesses of the layers are also defined here.

To model more specific devices, the user has the option of defining specific heat sources and known temperatures. The geometry of the heat sources and surface, edge and via connections are specified within BEGIN. The code then calculates the corresponding

points in the numerical model used to define the user-specified heat sources and connections.

2.4.2.1 Heat Sources

Heat generation in the model is defined by specifying an area over which the source acts, which plane of z nodes to use (i.e. K=1..N1), and a total heat input. If the source acts on the boundary of the device then it is implemented in the code as a flux on the same boundary. If the source is within the device, it is modeled as a source term.

2.4.2.2 Surface Connections

Surface connections to some known temperature are implemented by defining the area of contact, the layer (using K), the overall thermal resistance of the connection, and the known temperature. A typical example of a surface connection would be mounting pins on a chip. If the connection occurs on the boundary (i.e. the top or bottom surface), it is modeled as a flux. If it occurs on the surface of one of the interior layers, then it is modeled as a source term within that layer.

2.4.2.3 Edge Connections

Edge connections to a known temperature through a thermal resistance can be added to any edge in the x-y plane. These could be used to model mounting wires or the effect caused by contact with a case. The edge connections are specified by defining which edge, the layer to which the connection is made, and the initial point and length of

the connection. In addition, the thermal resistance of the connection and the known temperature must be given. The connection is assumed to cover the entire thickness of the layer (i.e., the height of the layer in the z direction is equivalent to the layer thickness). Since the edge connections occur on boundaries, they are always modeled as fluxes.

2.4.2.4 Thermal Vias

Thermal vias differ from the above features in that they are point to point connections, affecting only two specific nodes, instead of any nodes in an area. They are not limited to adjacent nodes, but instead can connect any two nodes in the domain. The user specifies a via by defining two pairs of x,y points, and a pair of corresponding K (layer number) values. In addition, the thermal resistance between the two vias must be defined. The x,y points are then converted to their corresponding I,J values and linked to each other using source and flux terms. Once again, if the point is on the boundary, then the thermal effect is modeled as a flux. If it is in the interior, source terms are used. Since the temperature of the points will change from iteration to iteration, the value of the flux or source term will change also. This introduces nonlinearity into the solution and convergence must be monitored carefully.

2.4.2.5 Point Lists

Source, surface and edge connections occur over a certain area of the model and hence one connection may affect many nodes, depending on the size of the area and the fineness of the grid. Connections are defined using a set of one-dimensional arrays for each type of connection. One array variable corresponds to each connection, so the size

of the array must be larger or equal to the number of connections to be defined. After the geometric dimensions are defined, a set of grid points that corresponds to each connection is determined by testing for them in an iterative loop. A grid point is included if it occupies a position that is a part of a connection (e.g., a boundary node that is located within the area defined to have a surface connection). If a grid point corresponds to some connection, then its value is added to that connection's point list. Each point list consists of four one-dimensional arrays. Three of the variables are used to store the I, J, and K variables for the points. The fourth is a integer reference variable which refers to the particular connection of which the point is a part(see Fig. 3). (For heat sources, NSRC(NSCPT) is used; for surface connections, NSURF(NSFPT) is used; for edge connections, NEDGE(NEPT) is used.) These reference variables are necessary so that the correct temperature and resistance (or heat input in the case of heat sources) is used when calculating the effect on the control volume.

This variable structure means that there can be many points corresponding to one surface connection, but each point is linked to only one connection. That is not to say that the points *values* (which describe the location) cannot be repeated in another part of a list, only that the point list *variables* match to only one connection. Depending on the grid resolution, a grid point can be a part of more than one connection, although this may often be impracticable in the physical device.

Since vias are considered to be point-to-point connections, the closest node to each end of the via is determined. There are no point lists for each via connection, merely

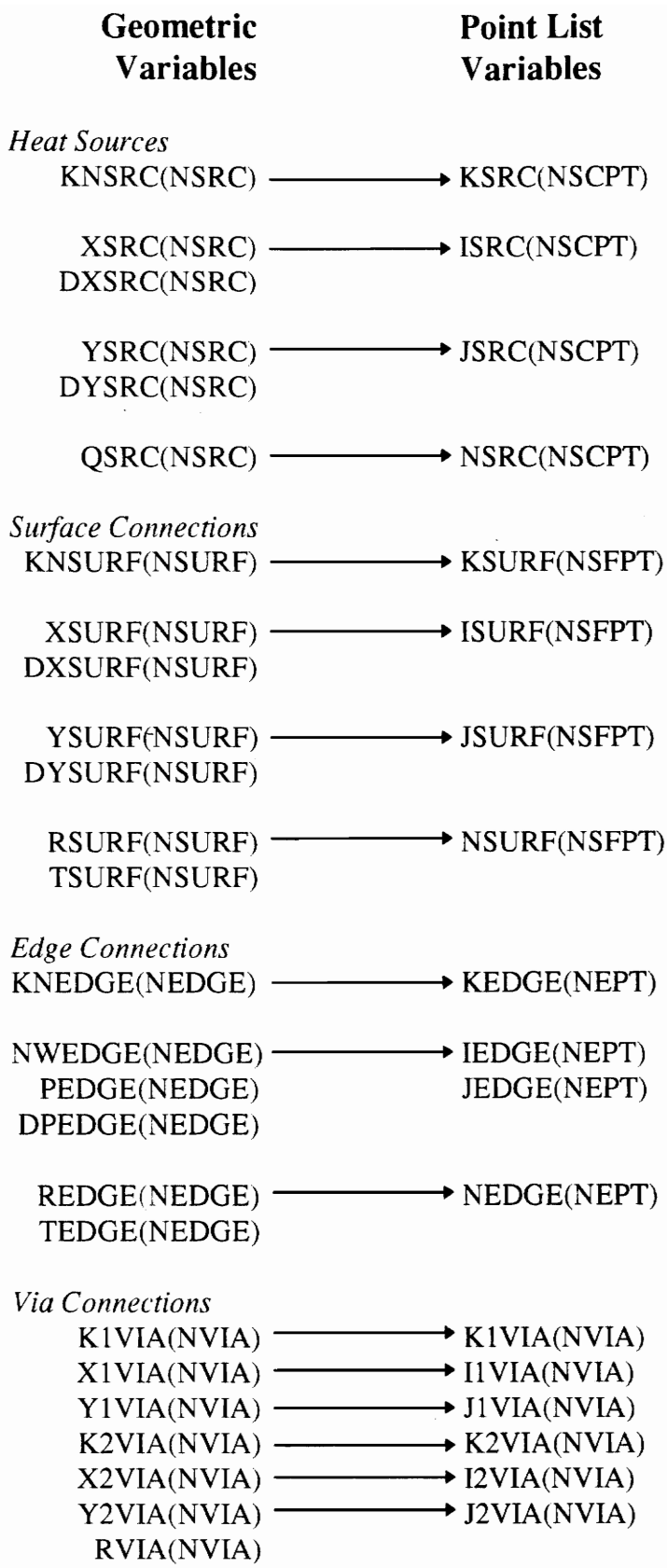


Figure 3: Variable Relationships

a set of I and J variables corresponding to the set of X and Y variables and the K node layer location given by the user. Each set of user-specified geometric variables leads to only *one* set of point variables used by the program, unlike the other types of connections in which each connection may have many sets of point variables associated with it. These nodes are used in PHI to assign the correct sources and fluxes.

2.4.3 OUTPUT

OUTPUT is used to check convergence during the solution of the problem. Convergence is checked by performing an overall energy balance. As the solution converges, the energy balance approaches zero. Depending on the problem, an additional method of checking the solution may be desired, such as convergence to steady-state for a transient problem.

OUTPUT is also used for post-processing. The boundary temperatures are extrapolated. These include the values for the top layer, which typically has the highest temperatures in the model. The average and maximum temperature for the top surface is then calculated. The corner temperatures, which are not used in computation, are extrapolated. Output is written to both the screen and/or to a file. To write output to the screen set the output index, KOUT, to 1 for screen, 2 for file, and 3 for both.

2.4.4 PHI

PHI is the most important part of the program. It is here that the physical properties of the model are specified. Conductivity, fluxes, and source terms are all defined for each grid point in the domain. Using the point lists calculated in BEGIN, the

effect of vias, connections, and source terms are all accounted for by the modification of fluxes and source terms within the domain. The effect of the external environment is incorporated by defining the boundary conditions and fluxes. Each grid point's source term or flux term is modified by adding the new effect to the old value. This means that each grid point can model the effects of multiple connections or boundary conditions.

The user can easily account for holes or cavities within a physical model by setting the conductivity of the appropriate section to zero. This essentially disconnects this section of the numerical model from the rest of the domain, since there is no mechanism (conductivity) to transfer energy into or out of the void. The solution produced by the code will include values for the irrelevant section, but these numbers are meaningless.

2.5 Iterative Solution

The structure of the code contains two nested iterative loops. The outer loop calculates the coefficients of the discretization equation and the inner loop solves the set of equations defined by these coefficients.

The iterations in the outer loop are counted by ITER. The convergence criteria for this loop include an energy balance and LAST, a maximum number of iterations to perform. The loop terminates when either criterion is reached.

The inner loop is contained within the solution subroutine (SOLVE3D), which iteratively solves the set of discretization equations. The solution subroutine contains an independent set of convergence criteria, including NTIMES(NF) a value for the maximum number of iterations to perform. The number of inner iterations is counted by NTC(NF).

NF represents the appropriate dependent variable. In typical conduction problems, only one dependent variable, temperature, is used. For these cases, NF=1.

It is important to remember that whether a linear or non-linear problem is being solved, the set of discretization equations sent to SOLVE3D is linear. The non-linearity of problems is handled by the updating of the coefficients performed by the outer loop. If the problem is linear, a sufficiently large value of NTIMES would produce a correct solution. However, if the problem is nonlinear, the coefficients must be updated by the outer loop. It is therefore more efficient to limit the number of inner iterations to a relatively small value, allowing the coefficients to be updated and then returning to SOLVE3D to recalculate the temperatures. Early in nonlinear problems, typically the repetitions in the solution subroutine reach the maximum value allowed by NTIMES. As more outer iterations are performed and the problem approaches convergence, the number of inner iterations drops to some value less than NTIMES. Although a problem can still converge and produce accurate results with NTC=NTIMES, this behavior could indicate an improper formulation.

2.6 Unsteady Problems

Time-dependent problems also fall within the scope of this code. To solve an unsteady problem, several additional parameters must be given. The storage capacity of each control volume and time step are specified as ALAM(I,J,K) and DT. Also, the initial temperature distribution now represents the temperatures at time = 0 instead of guesses. In these problems, each outer iteration represents a time increment. If the outer loop is

used to increment the time step, it cannot be used to explicitly update the coefficients as required by a non-linear problem. This would seem to rule out the solution of time-dependent, non-linear problems. However, most time steps would be small enough to allow the satisfactory solution of an unsteady non-linear problem. For unsteady problems, it is essential to insure that the solver reaches a satisfactory solution before returning to the outer iteration. Otherwise, the solution at the next time step will be based upon inaccurate values.

3. Notation

3.1 Coordinate Systems

The code can be used with either of two coordinate systems. The first system is the standard right-handed rectangular system. It consists of an origin located at one corner of the domain and x, y, and z axes. This coordinate system produces control volumes that are box-like in shape. The appropriate areas and volumes are easily calculated as shown in the first lines of the equations in Fig. 4. The rectangular coordinate system can be specified by setting MODE=1 in the GRID section of the user subroutine. This is, however, the default value, so MODE does not need to be defined.

The second coordinate system is the cylindrical coordinate system. The X coordinate acts as Θ, the angular coordinate, and the z coordinate indicates the length of the cylinder (see Fig. 5). The y coordinate is equivalent to the radial coordinate of the cylinder plus a constant. This constant is defined using R(1). The cylindrical coordinate system is invoked by setting MODE=3.

The calculation of areas and volumes using the cylindrical coordinate system is somewhat more complicated than in the rectangular system. The use of X as an angle measurement necessitates the use of the correct radius to calculate the appropriate geometric quantities. Two radius variables are used in the calculations. R(J) is the value of the radius at grid point J, and RV(J) is the value of the radius at the Y interface associated with grid point J.

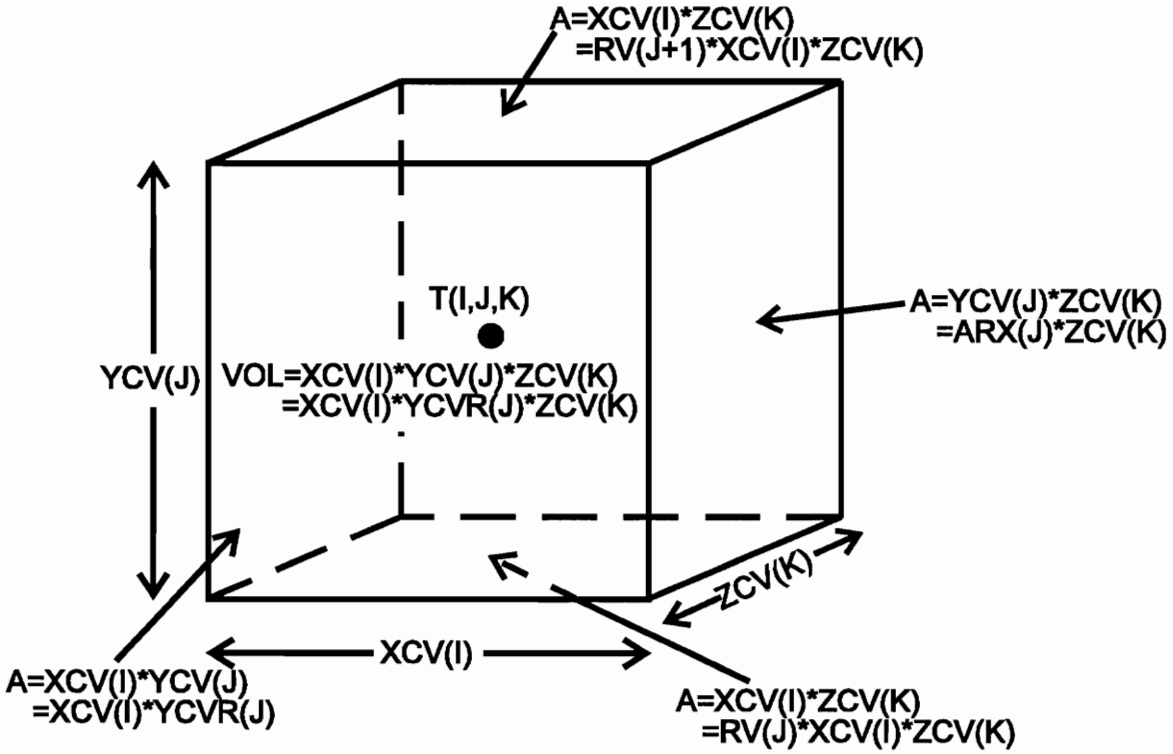


Figure 4: Control Volume Element in the Rectangular Coordinate System

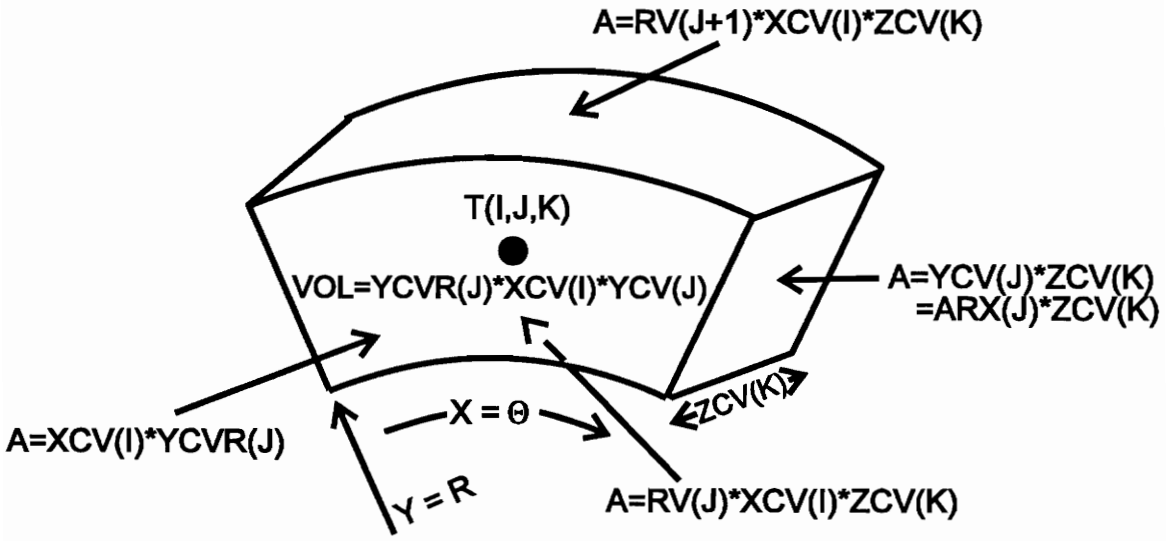


Figure 5: Control Volume Element in the Cylindrical Coordinate System

To allow a consistent treatment of the two coordinate systems, R(J) and RV(J) are defined to be unity for MODE=1 operation. YCVR(J) is calculated and stored for each control volume. This value is equivalent to R*ΔY for the control volume. The use of YCVR(J) is necessary because when calculating the volume or areas in the X-Y plane the appropriate radius varies. Equivalent to R(J) is SX(J), a scale factor. This variable allows the length between two X grid points to be defined as SX(J)*(X(I+1) - X(I)) which gives the correct length regardless of the coordinate system used. The use of these variables allows the user to define geometric quantities that work for both rectangular and cylindrical systems. The second lines of the equations in Figs. 4 and 5 show the areas and volumes defined in these consistent variables.

Since most applications of this program will be best served by use of the rectangular coordinate system, this mode will be used in further discussion of features of the program. If use of the cylindrical coordinate system is desired, the user may find the examples given in reference 10 useful.

3.2 Grid Points

The type of grid used (called Practice B by Patankar) first divides the domain into control volumes and then places grid points at the center of these control volumes. For a uniform grid this practice results in equal sized control volumes and equal spacing between grid points (except between the boundary grid points and the first interior points) (see Fig. 6). The control volume nodes are numbered starting at 1 with limits of L1,M1,N1 for the

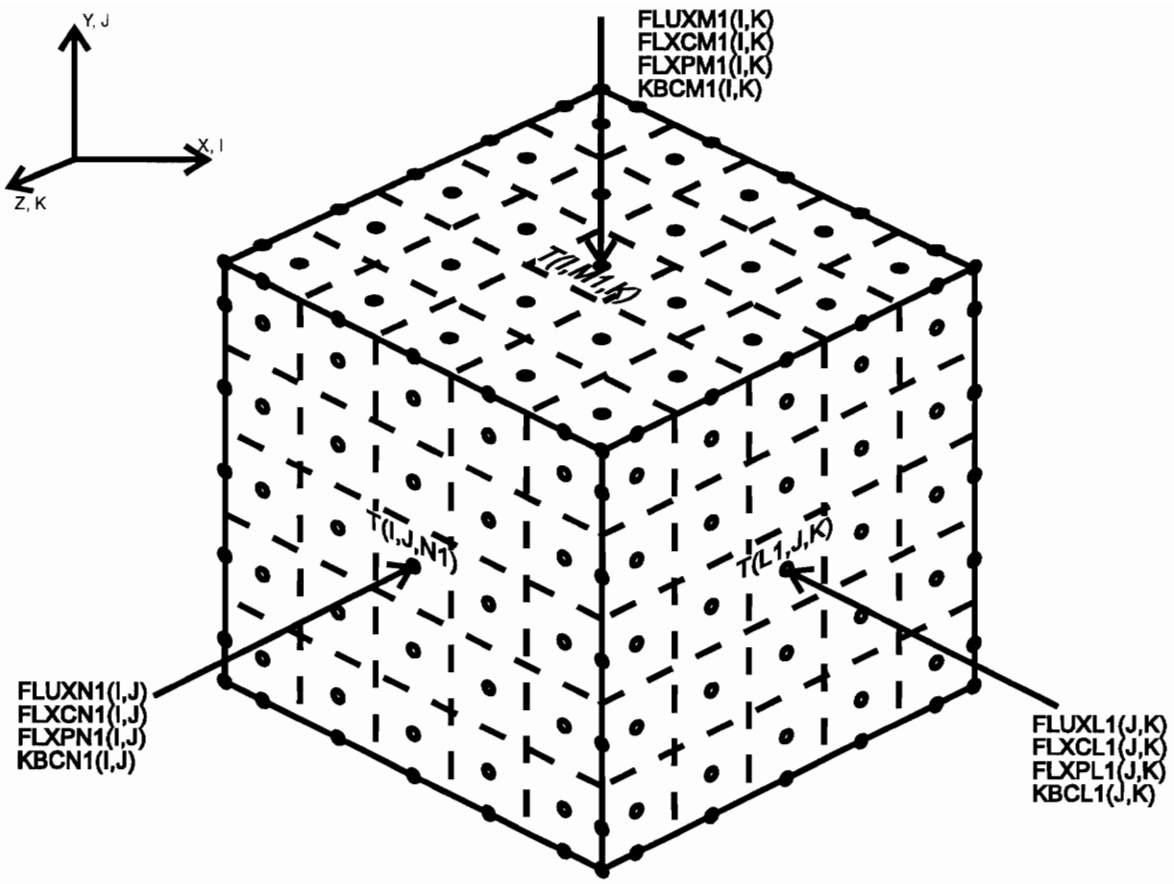


Figure 6: Isometric View of the Numerical Model and Control Volumes

x, y, and z directions respectively (see Figs. 7-12). Because of their frequent use, some grid points near the boundaries are defined:

L2 = L1 -1 M2 = M1 -1 N2 = N1 -1

L3 = L1 - 2 M3 = M1 - 2 N3 = N1 -2

Due to the nature of the solution, the values stored at the grid points along the edges (meaning the intersection of two faces) or the corners (meaning the intersection of three faces) of the domain have no role to play in the solution. There is no volume associated with them and they often are a point of discontinuity between the boundary conditions on two faces. However, in some problems, the maximum temperature may occur at an edge or corner. In this case, the user may extrapolate the edge or corner value after the solution is obtained.

3.3 Control Volumes

Each control volume has dimensions XCV(I) x YCV(J) x ZCV(K) where I, J, and K are the coordinates of the node in that control volume (see Fig 4). The Cartesian coordinates for node(I,J,K) are given by X(I), Y(J), Z(K). The control volume grid point has six neighbors. Its relationship to these neighbors is dictated by the physical values of the control volume (see Fig. 13). The conductivities in each direction can be defined individually. The source term is specified using SC(I,J,K) and SP(I,J,K). For unsteady problems, ALAM(I,J,K) is given.

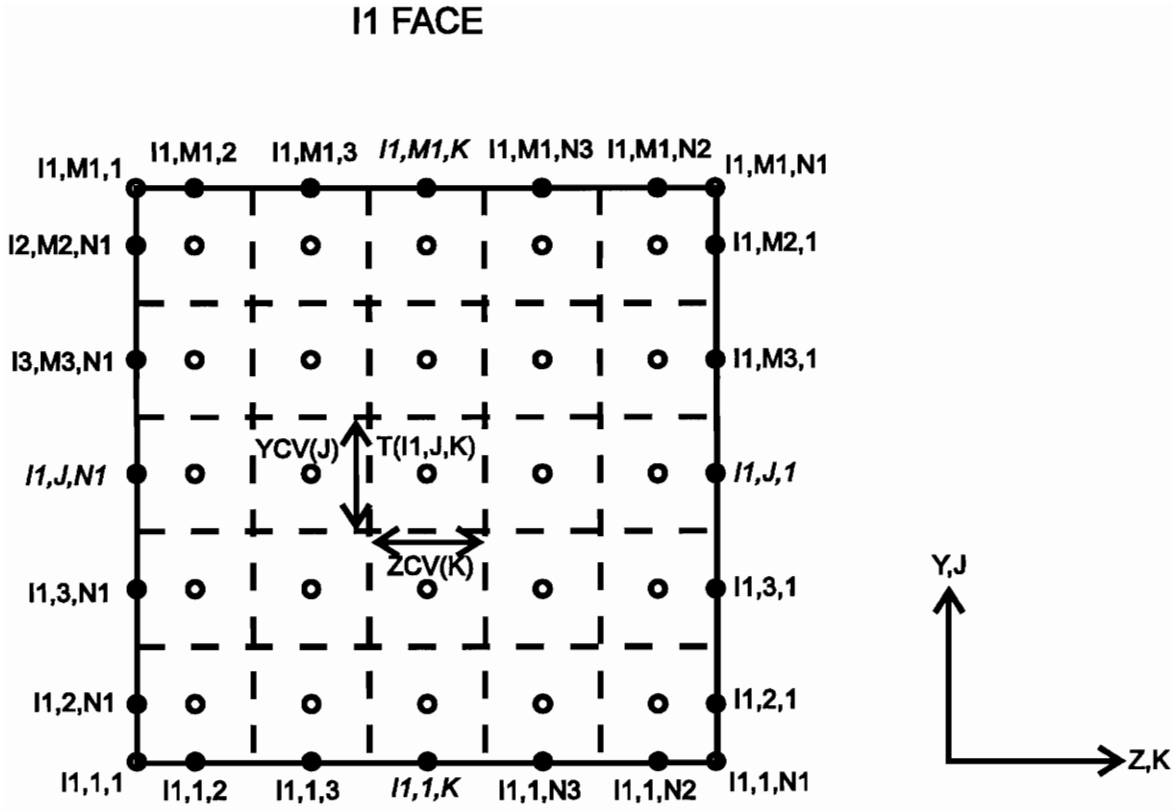


Figure 7: I1 Face of the Numerical Model

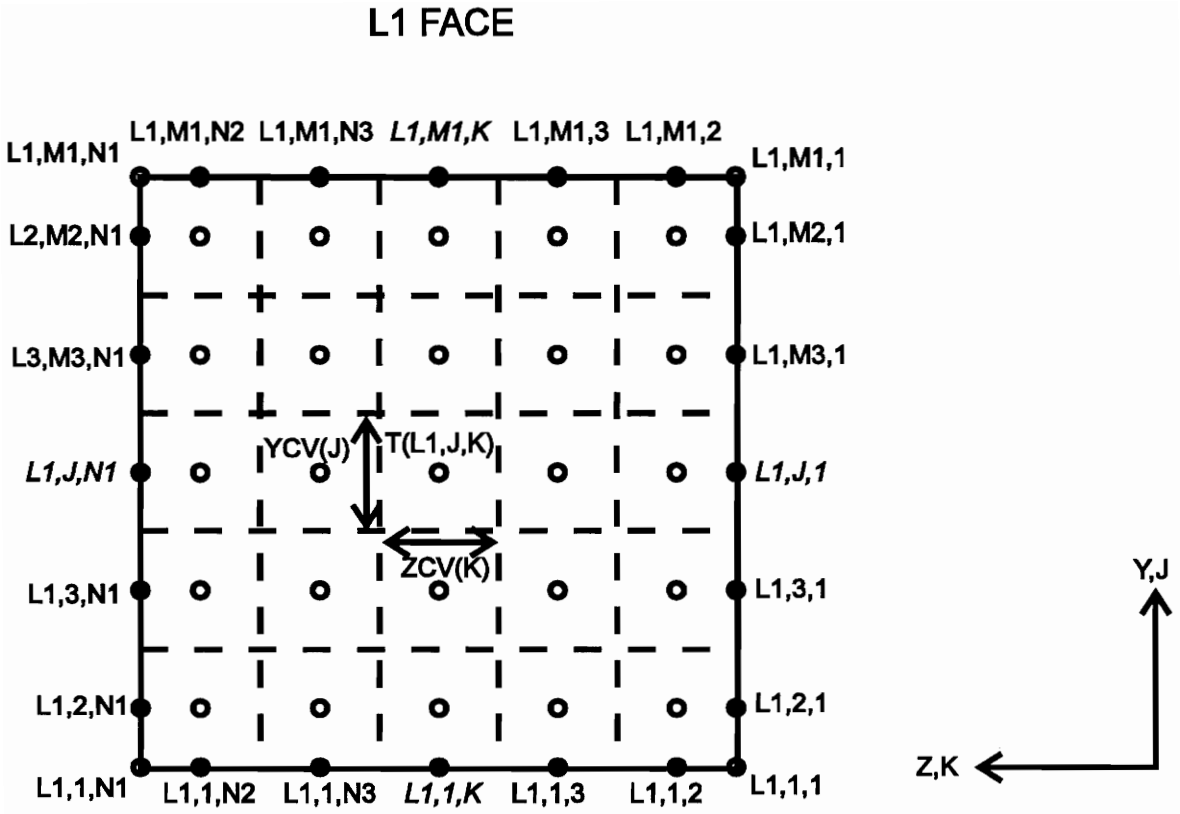


Figure 8: L1 Face of the Numerical Model

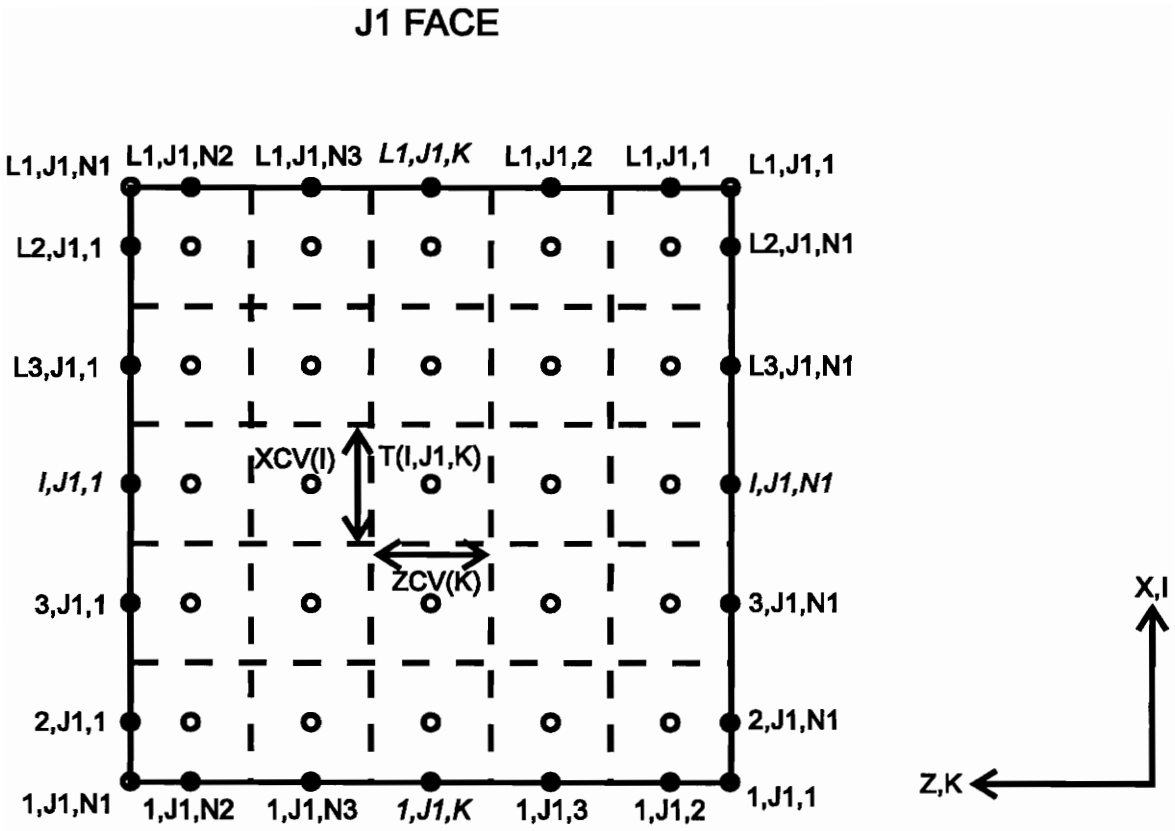


Figure 9: J1 Face of the Numerical Model

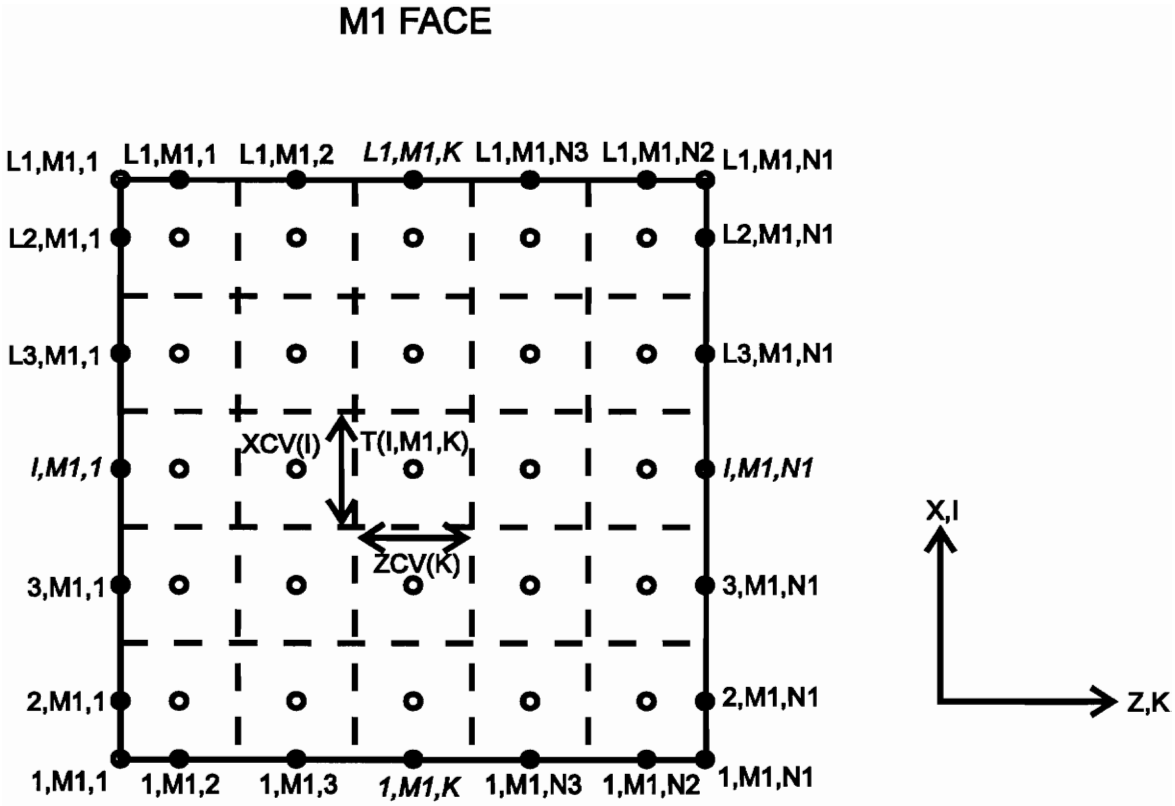


Figure 10: M1 Face of the Numerical Model

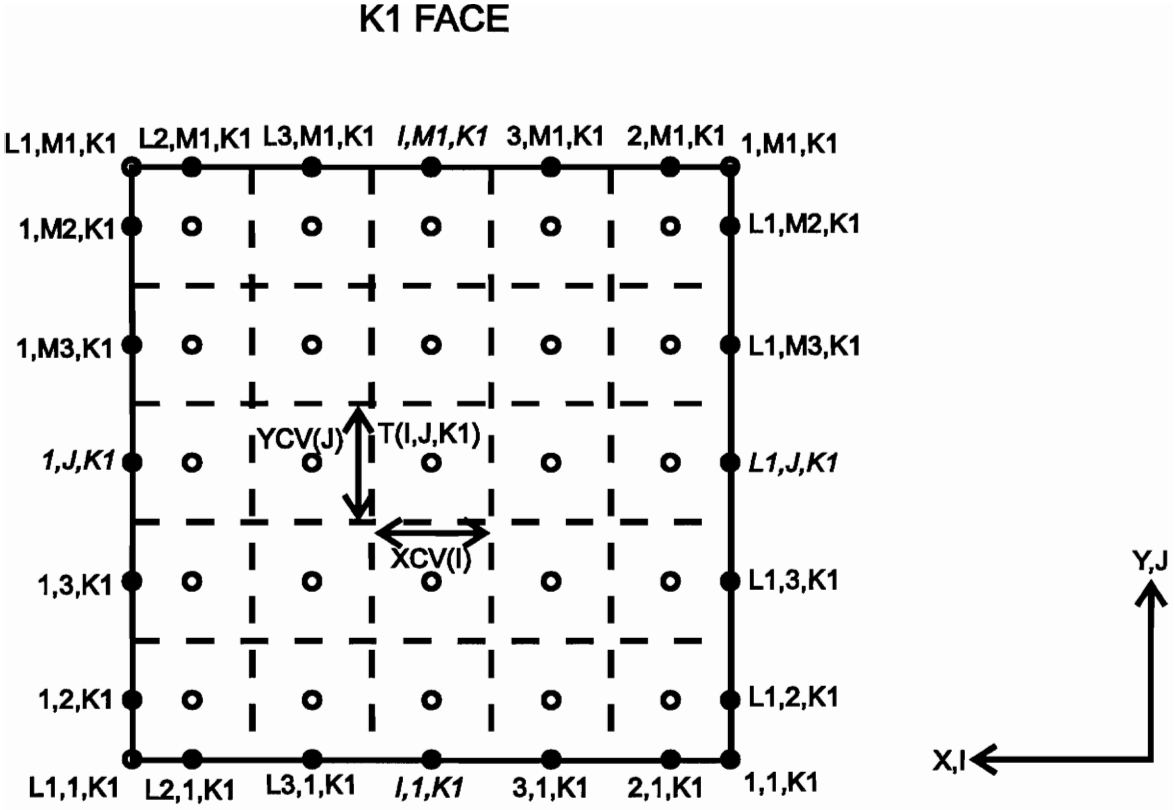


Figure 11: K1 Face of the Numerical Model

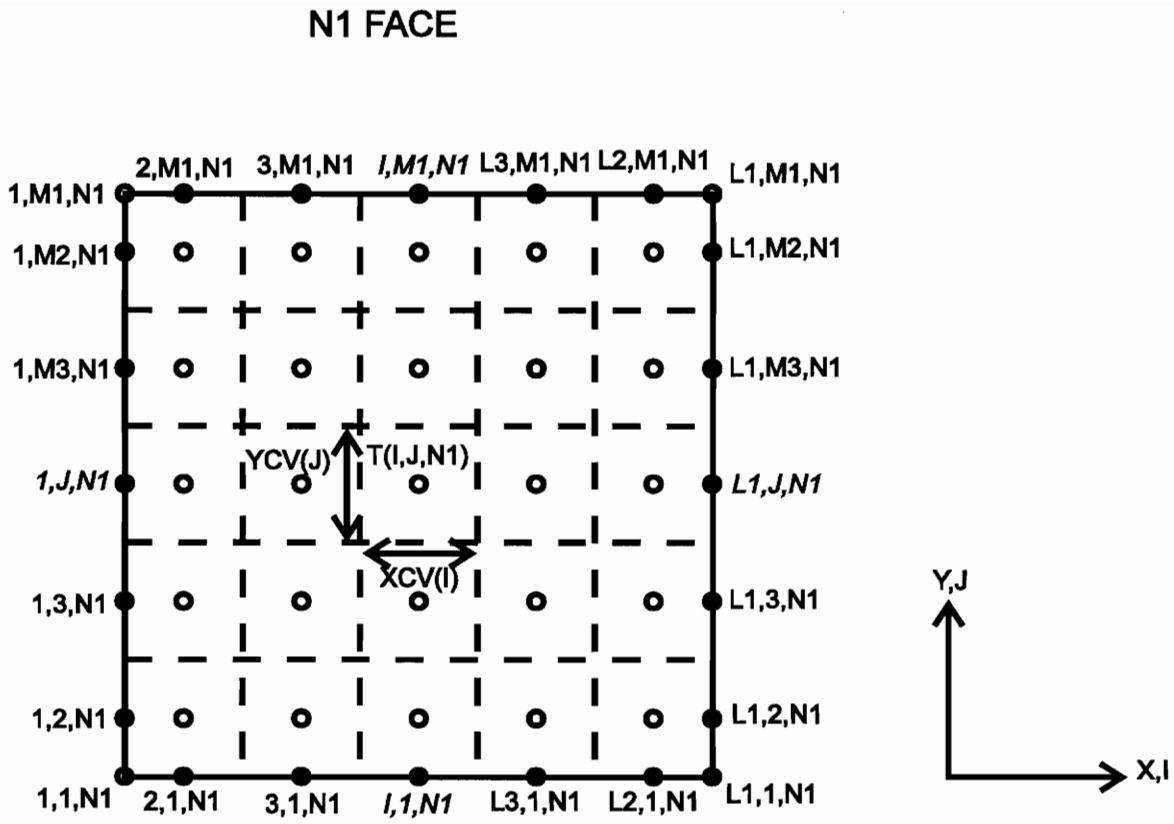


Figure 12: N1 Face of the Numerical Model

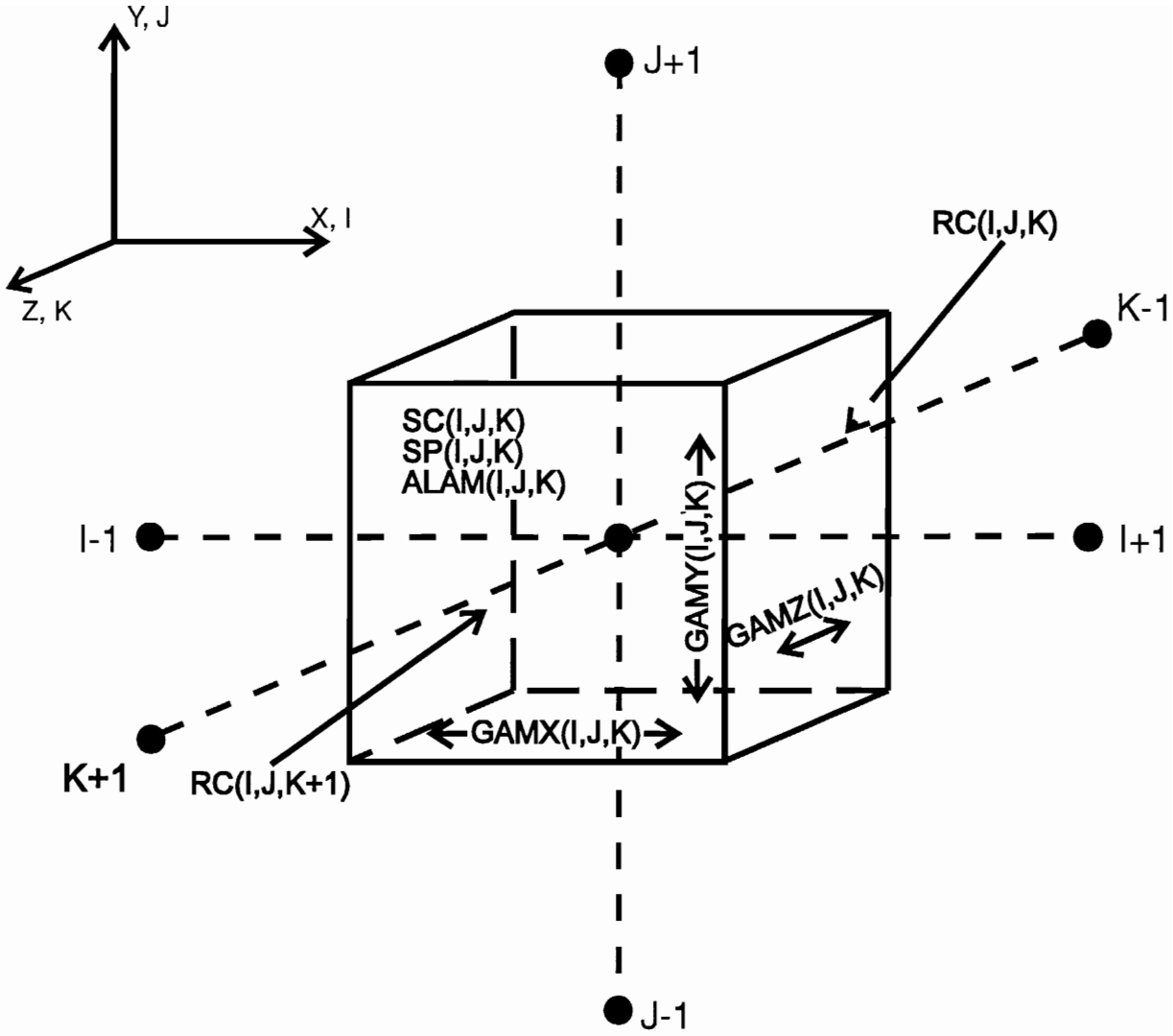


Figure 13: Grid Point Values and Relationships to Neighboring Grid Points

3.4 Source Terms

Each interior grid point (i.e., each grid point that has an associated volume) carries a source term. They are specified in a linear fashion:

$$S(I,J,K) = [SC(I,J,K) + SP(I,J,K) \cdot T(I,J,K)] \cdot VOL(I,J,K) \qquad (24)$$

where S is the total source at some grid point, SC and SP are constants, VOL is the volume of the control volume, and T is the temperature of the grid point. SC and SP are volumetric variables used to define the source term. The code calculates the correct value of VOL when solving the problem.

3.5 Flux Terms

In addition to the discretization equations, the solution also depends on the boundary conditions. Either the temperature or the flux must be specified at each boundary point.

The boundaries of the model are identified using I1, J1, K1, L1, M1, or N1 (see Fig. 14) This notation is then used to describe the boundary-related variables. All of these boundary variables are two dimensional arrays in the plane over which they are applied. For example, on the I1 boundary, a two-dimensional array in the J and K directions is used.

KBCx (where x refers to the particular boundary e.g., I1) is a boundary condition indicator with value of 1 or 2. The default value of 1 means that the temperature on the boundary at this point is known and will be assigned. A KBC value of 2 indicates that the

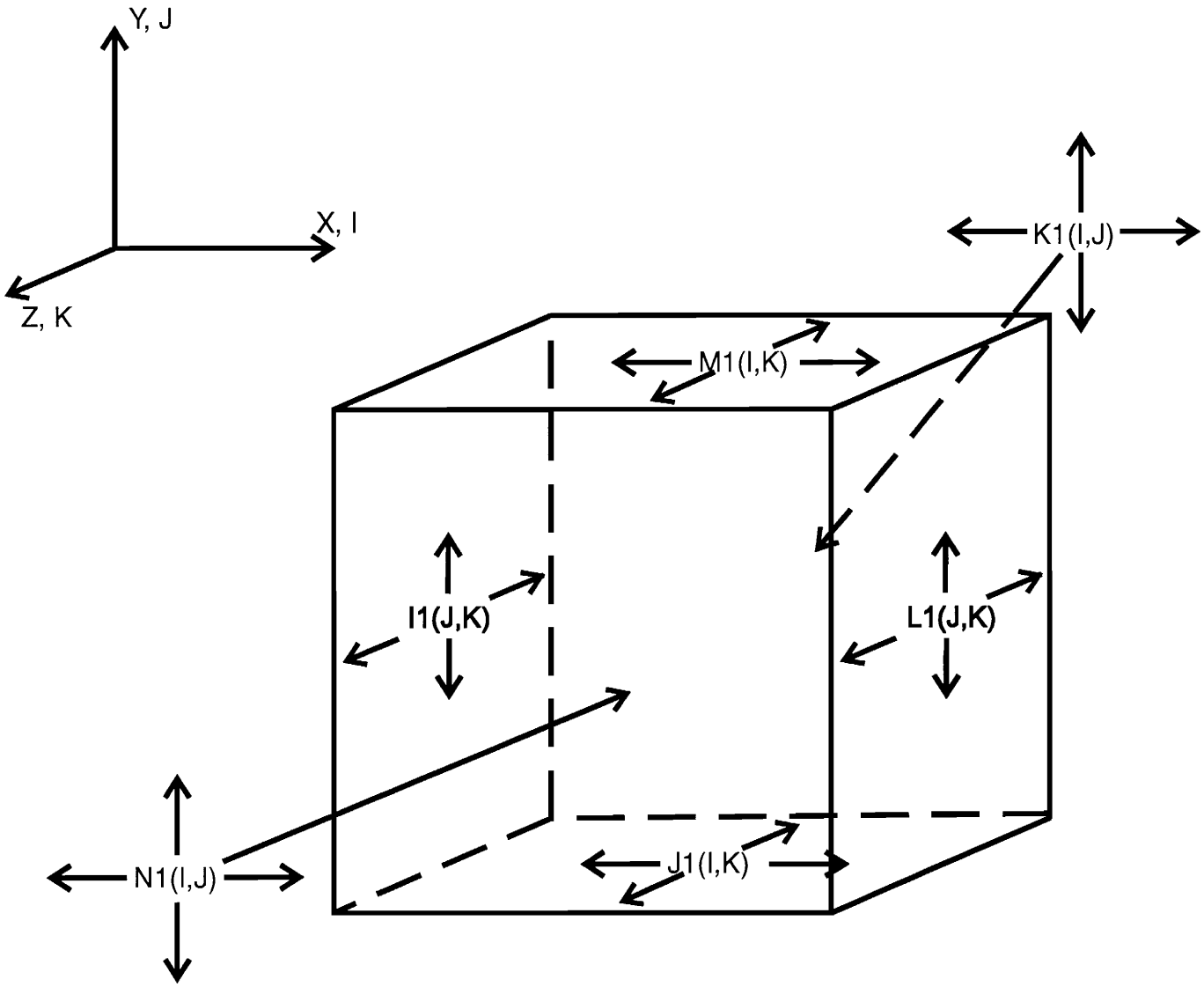


Figure 14: Domain Boundaries and Notation

boundary condition will be specified in the form of a flux, with the default flux equal to zero (insulated). Boundary fluxes are specified using the same notation, that is FLXCx and/or FLXPx, as a linear function of the form:

$$flux = FLXCx + FLXPx \cdot T(I,J,K) \tag{25}$$

FLXCx and FLXPx only need to be defined when there is a known flux on the boundary (i.e., KBCx=2 on a non-insulated boundary). It is important to remember that the heat input at the boundary is specified as a flux instead of a heat transfer rate. The correct area is then used within the code to convert it to a total heat transfer rate specific to that control volume face.

3.6 Layer Dimensions

Sometimes it may be desirable make the top layer's x and y dimensions smaller than the device dimensions. For example, this could be used to model a microchip mounted on a base. If this is desired, the dimensions can be specified using XLAY, YLAY, DXLAY, and DYLAY (see Fig. 15). This sets the conductivity of the top layer outside the boundary to zero and limits the effect of the external environment to the surface of the layer within the boundary. It also applies the effect of the external environment to the exposed surface of the second layer from the top. This could be accomplished manually by using DO and IF loops within PHI. In the same manner, holes and voids in the domain may be modeled.

The program will still incorporate the grid points outside of the top layer boundary into the solution. However, the conductivity of these grid points is zero which effectively

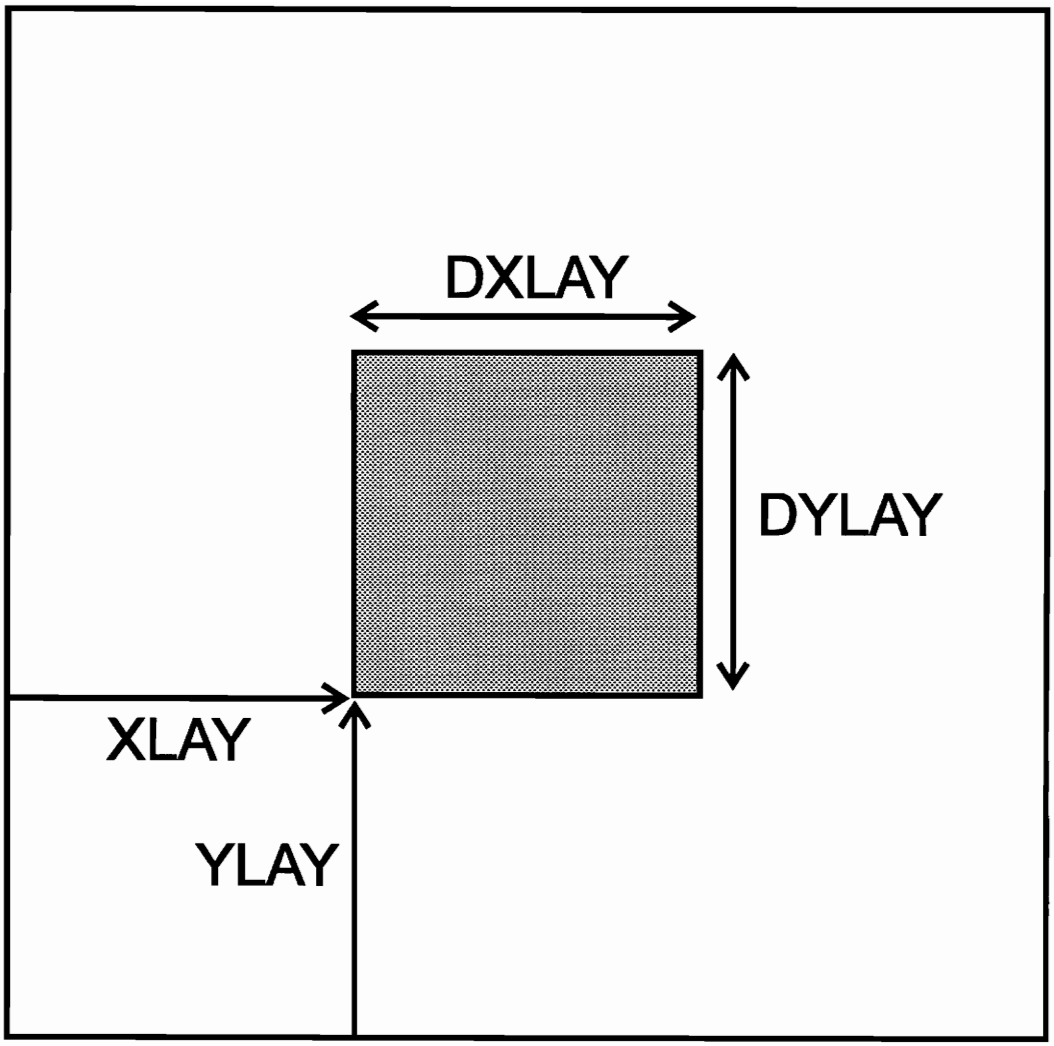


Figure 15: Definition of Top Layer Dimensions

disconnects the grid points from the rest of the domain, preventing them from having an effect on the solution. The numbers returned by the code for these grid points are physically meaningless.

4. Implementation

4.1 Physical System

To accurately calculate the thermal characteristics of a physical system, it is necessary to model various device features such as heat sources, thermal vias, and edge and surface connections in addition to the three-dimensional layer structure. (Examples of edge and surface connections would include pins, wire bonds, and device casings.) These can all be interpreted as either a constant heat input or a connection to a known temperature (convective boundary condition). Heat input from a connection to a known temperature would be expressed as:

$$Q = \frac{1}{R_a} \cdot (T_a - T) \tag{26}$$

where T_a is the known temperature and R_a is the effective resistance of the connection. T is the temperature of the device at the point of connection.

The effect on the physical device is implemented by applying the equivalent effect in the corresponding control volumes in the numerical model. The exact procedure for translating the effect of the heat input in eqn. 26 to the numerical model varies depending upon the type of numerical model, but the general principle is the same. The effect of the connection acts over a defined area or volume in the physical model. In the discretized numerical model, this area is composed of many elements, each of which is affected by the connection. The correct effect on each element is determined by equating the equations

used in the particular numerical implementation to the expression for the overall effect on the physical model (in this case a connection to a known temperature).

This general principle can be better understood if illustrated by its use on an actual numerical model. In this code, the effects of connections and heat sources are modeled as source and flux terms. On the boundary, fluxes are used. In the interior, source terms are used. To ensure that the physical effect is correctly converted into the appropriate source or flux variables, the related areas and volumes must be compared.

4.2 Source Terms

If the heat input into control volume (I,J,K) is the product of a connection to a known temperature (see Fig. 16), then it can be expressed in a form similar to eqn. 26:

$$Q_{cv} = \frac{1}{R_{cv}} \cdot (T_a - T(I,J,K)) \tag{27}$$

where T_a is the known temperature, R_{cv} is the total resistance of the connection, and T is the grid point temperature. The heat generated by a volumetric source term S(I,J,K) in the control volume is:

$$Q_{cv} = [SC(I,J,K) + SP(I,J,K) \cdot T(I,J,K)] \cdot XCV(I) \cdot YCV(J) \cdot ZCV(K) \tag{28}$$

where XCV(I), YCV(J), and ZCV(K) are the control volume dimensions. To obtain the proper expression for the source term, the two expressions are equated, yielding:

$$\begin{aligned} &[SC(I,J,K) + SP(I,J,K) \cdot T(I,J,K)] \cdot XCV(I) \cdot YCV(J) \cdot ZCV(K) \\ &= \frac{1}{R_{cv}} \cdot [T_a - T(I,J,K)] \end{aligned} \tag{29}$$

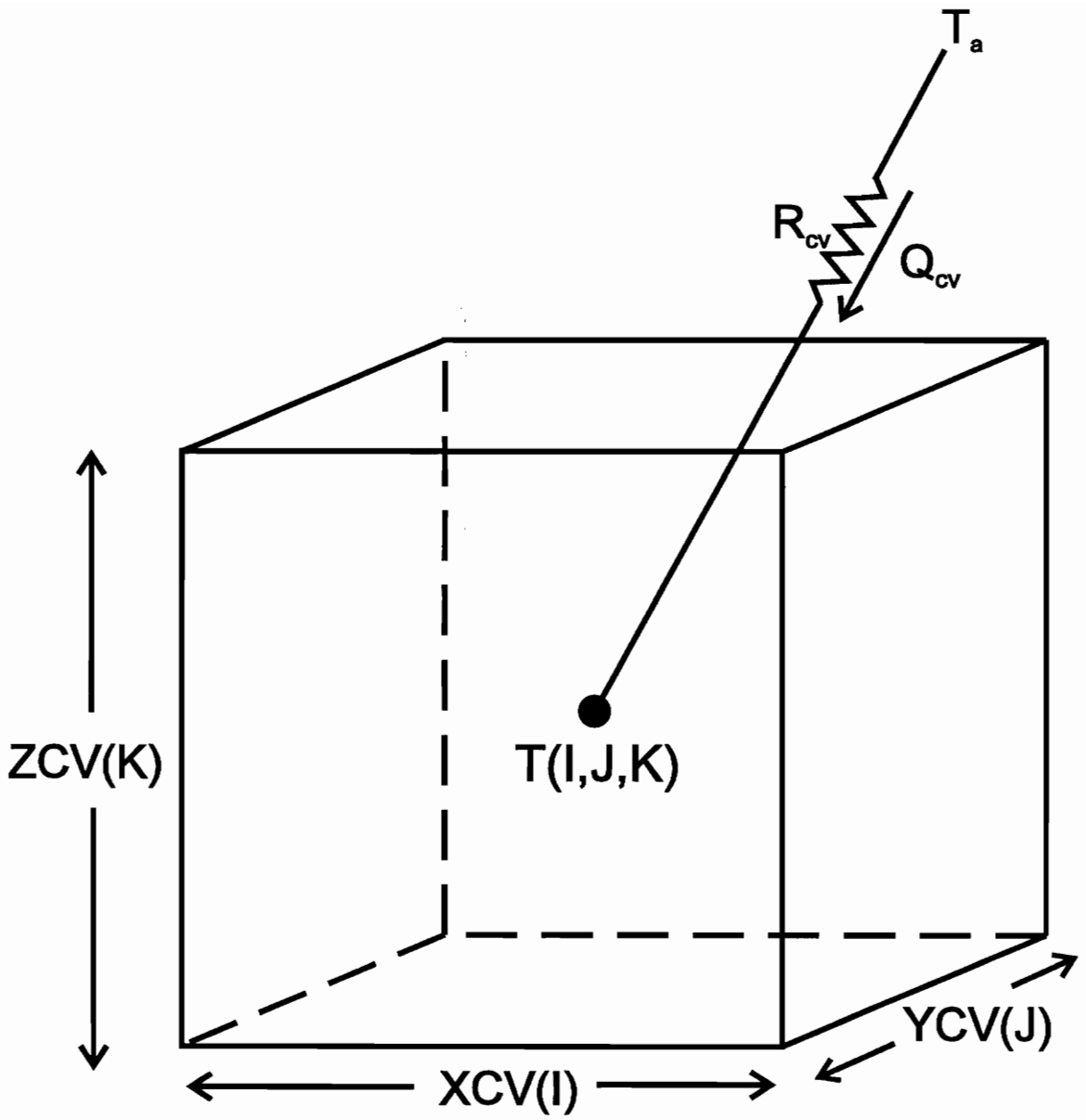


Figure 16: Control Volume Connected to an External Known Temperature

The values of SC and SP can then be obtained:

$$SC(I,J,K) = \frac{\frac{1}{R_{cv}} \cdot T_a}{XCV(I) \cdot YCV(J) \cdot ZCV(K)} \tag{30}$$

$$SP(I,J,K) = -\frac{\frac{1}{R_{cv}}}{XCV(I) \cdot YCV(J) \cdot ZCV(K)} \tag{31}$$

Using these values of SC and SP essentially performs the connection to the known temperature within the context of the code.

4.3 Fluxes

Heat input into a boundary node takes the same form as eqn. 27. The heat input from the flux is expressed as:

$$Q_{cv} = (FLXCx + FLXPx \cdot T(I,J,K)) \cdot area \tag{32}$$

where *area* represent the area over which the flux acts (e.g., XCV*YCV for faces K1 and M1). Equating eqn. 27 and eqn. 32 and solving for the linear constants yields:

$$FLXCx(A,B) = \frac{\frac{1}{R_{cv}} \cdot T_a}{area} \tag{33}$$

$$FLXPx(A,B) = -\frac{\frac{1}{R_{cv}}}{area} \tag{34}$$

where *area* is the area of the control volume face and A and B represent the dimensional coordinates necessary to describe the location of the control volume on the boundary. On the I1 or L1 boundaries, *area=YCV(J)*ZCV(K), (A,B) = (J,K);* on the J1 or M1 boundaries, *area=XCV(I),ZCV(K), (A,B) = (I,K);* and on the K1 or N1 boundaries, *area=XCV(I)*YCV(J), (A,B) = (I,J).*

4.4 Distribution Among Control Volumes

Source surface and edge connections are defined by the user to act over a certain area of the physical model. In the numerical model, however, the equivalent area usually contains many control volumes. In this case, then the effects of the connection must be divided accordingly. Equations 26-34 apply to a heat input acting on one control volume, but the following equations account for the transition from the physical to numerical model.

4.5 Heat Sources

Heat sources are defined to act at a certain layer of z grid points (see Fig. 17). This is done by specifying a value for the K coordinate, which means the user must be aware of how the grid spacing in the numerical model corresponds to the layers in the physical system. The rectangular area over which the source term acts is defined by specifying the lower left hand corner with XSRC and YSRC and the height and width with DXSRC and DYSRC. The K coordinate is given by KNSRC and the heat input is QSRC. Depending on grid spacing, this area could contain one or many control volumes (see Fig.

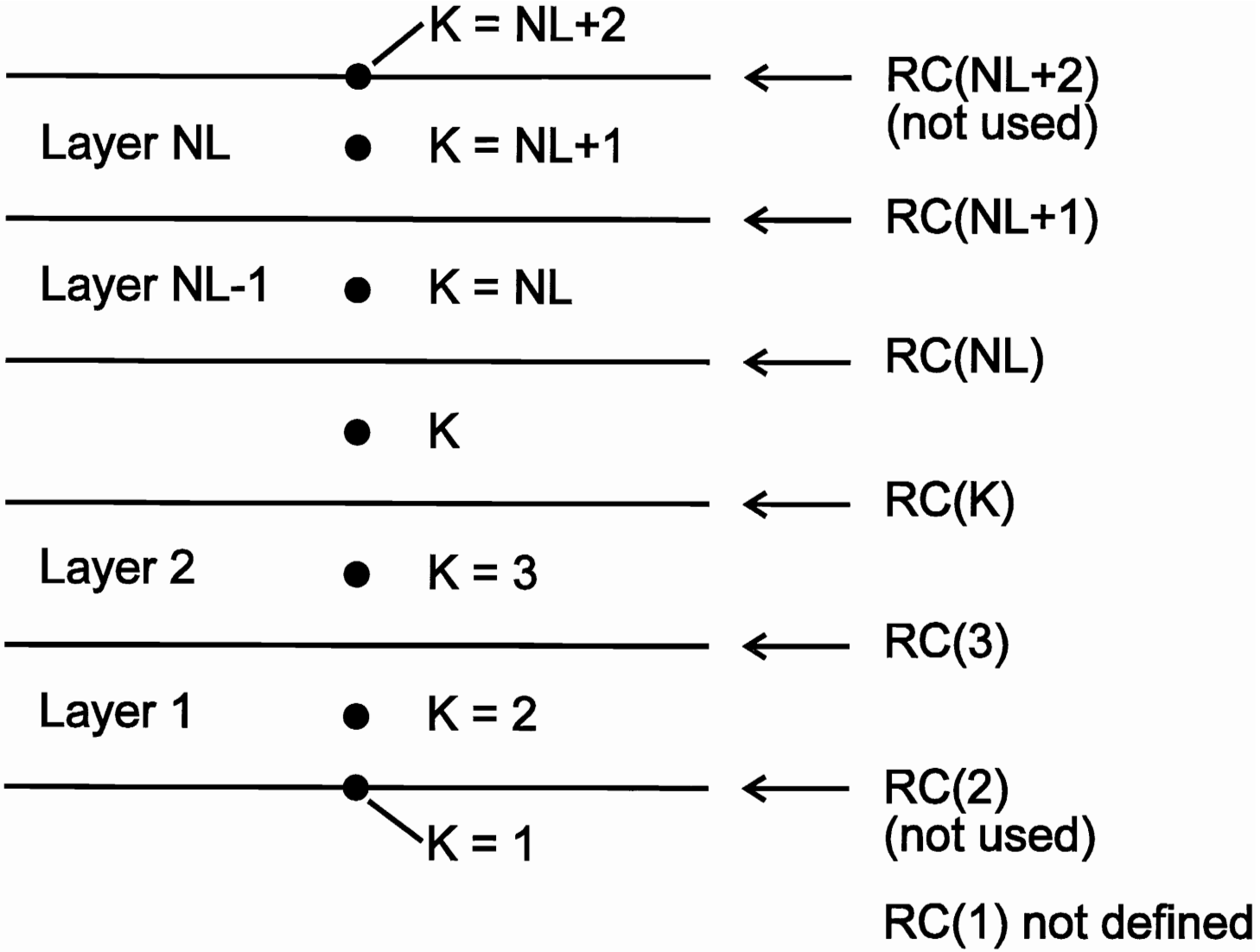


Figure 17: Relationship of Contact Resistances to Nodes in the Z direction

18). If multiple control volumes are used, the total heat input must be correctly appropriated among them. It is divided so:

$$Q_{cv} = Q_{total} \cdot \frac{XCV(I) \cdot YCV(J)}{DXSRC \cdot DYSRC} \tag{35}$$

where Q_{cv} is heat input to the control volume (with dimensions XCV(I) and YCV(J)) and Q_{total} = QSRC.

4.5.1 Source Terms

If the heat source is defined to act on a interior grid points (i.e., points not on a boundary) the heat source is expressed as source terms in a group of control volumes. Using Q_{cv} in as defined in equation 35 and following the same steps shown in eqns. 27, 28, and 29 gives:

$$SC(I,J,K) = \frac{QSRC}{ZCV(K) \cdot DXSRC \cdot DYSRC} \tag{36}$$

Since heat sources are defined as a constant heat input, QSRC, the resulting source term is specified only in terms of SC.

4.5.2 Flux Terms

Similarly, if the heat source is defined to act on a boundary grid point then it is expressed as a flux. The expression of flux also consists only of the constant term:

$$FLXCx = \frac{QSRC}{DXSRC \cdot DYSRC} \tag{37}$$

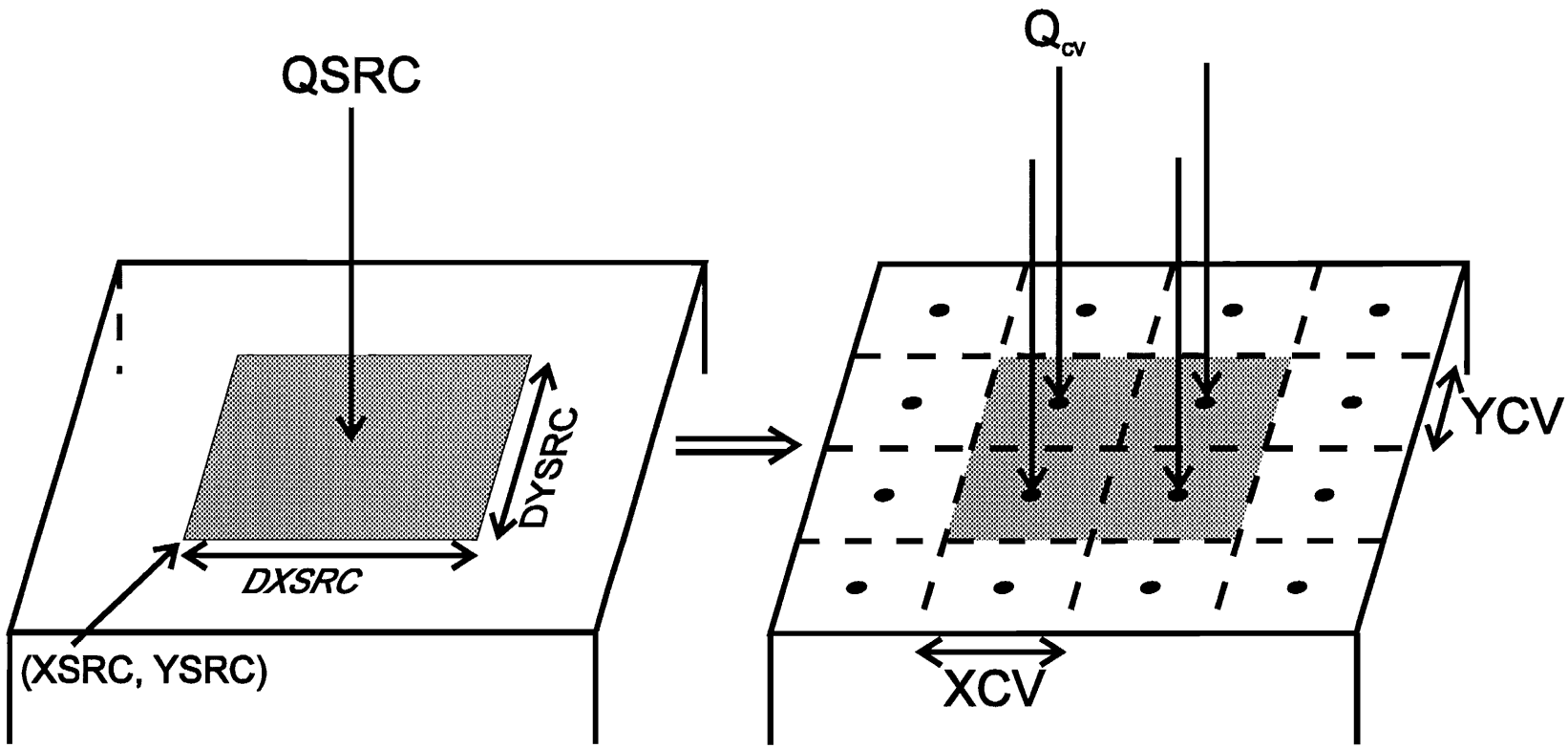


Figure 18: Definition of a Heat Source and Associated Control Volumes

Note that the expressions for the flux and source terms are independent of the control volume x-y dimensions, but that the source term relies on ZCV.

4.6 Surface Connections

In the same manner as sources, surface connections are defined to act at some z layer of control volumes by specifying a K value, KNSURF. A connection is defined to be a rectangular area located at XSURF and YSURF with dimensions DXSURF and DYSURF. Typically, this area will contain multiple control volume faces (see Fig. 19). The distinction is that the heat input from a surface connection is the result of a temperature difference (eqn. 26). The heat input used to calculate the source or flux terms for each control volume is a fraction of the total heat input. This is calculated according to the control volumes boundary area relative to the area of the connection. Surface connections are restricted to the x-y plane, so the area of the control volumes can be expressed using XCV and YCV.

$$Q_{cv} = Q_{cv} \cdot \frac{XCV(I) \cdot YCV(J)}{DXSURF \cdot DYSURF} \tag{38}$$

Now, evaluating eqn. 35 with $Q_{total} = Q$ from eqn. 27,

$$Q_{cv} = \frac{1}{R_a} \cdot [T_a - T(I,J,K)] \cdot \frac{XCV(I) \cdot YCV(J)}{DXSURF \cdot DYSURF} \tag{39}$$

This gives the correct heat input for a control volume within a known temperature connection.

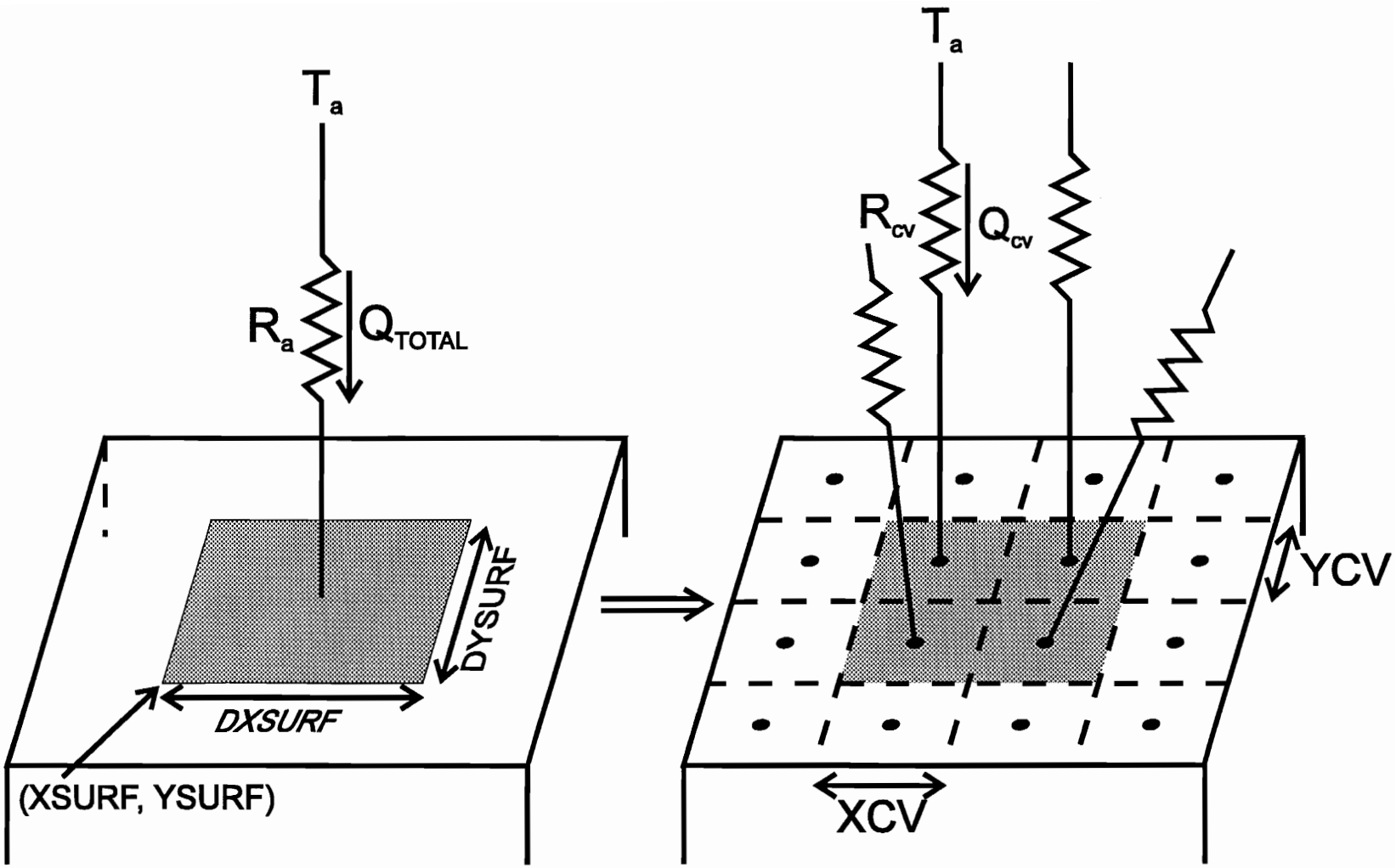


Figure 19: Definition of a Surface Connection and Associated ControlVolumes

4.6.1 Source Terms

To obtain the source term expression equate eqn. 39 with eqn. 28:

$$[SC(I,J,K)+SP(I,J,K)\cdot T(I,J,K)]\cdot XCV(I)\cdot YCV(J)\cdot ZCV(K) = \frac{1}{R_a}\cdot[T_a - T(I,J,K)]\cdot\frac{XCV(I)\cdot YCV(J)}{DXSURF\cdot DYSURF} \quad (40)$$

SC and SP for the control volume can now be determined:

$$SC(I,J,K) = \frac{\frac{1}{R_a}\cdot T_a}{ZCV(K)\cdot DXSURF\cdot DYSURF} \quad (41)$$

$$SP(I,J,K) = -\frac{\frac{1}{R_a}}{ZCV(K)\cdot DXSURF\cdot DYSURF} \quad (42)$$

When surface connections are modeled as source terms, the user may wish to account for the additional resistance caused by the distance between the surface of the layer and the node in the middle of the layer. This is done by adding the an appropriate amount to R_a.

4.6.2 Flux Terms

If the surface connection is to be expressed as a flux, then a similar procedure is used, equating eqn. 39 to eqn. 32:

$$(FLXCx + FLXPx\cdot T(I,J,K))\cdot area = \frac{1}{R_a}\cdot(T_a - T(I,J,K)\cdot\frac{XCV(I)\cdot YCV(J)}{DXSURF\cdot DYSURF} \quad (43)$$

Since a surface connection expressed as a flux occurs on either the K1 or N1 boundaries, then *area* = *XCV*YCV* and *x* = *K1* or *N1* This leads to the definition of the flux:

$$FLXCx(I,J) = \frac{\frac{1}{R_a} \cdot T_a}{DXSURF \cdot DYSURF} \tag{44}$$

$$FLXPx(I,J) = -\frac{\frac{1}{R_a}}{DXSURF \cdot DYSURF} \tag{45}$$

Note that both flux equations are independent of control volume size. This is a result of the way sources and fluxes are expressed in the code and the way surface connections are specified. Source terms, although independent of the control volume x and y dimensions, are dependent on the control volume z dimension.

4.7 Edge Connections

Edge connections are defined to act over some rectangular boundary area *perpendicular* to the x-y plane (i.e., on the I1, J1, L1, and M1 boundaries). The connection is specified by defining a layer, boundary, and length of the connection. The boundary is defined using NWEDGE(NEDGE) for each connection. This variable can be equal to 1, 2, 3, or 4. These integers represent the I1, L1, J1, and M1 edges respectively. Edge connections always occur on the boundary and so are always expressed using fluxes. The total area of the edge connections is the product of the layer thickness and the length of the connection. This area will usually contain multiple control volume faces (see Fig. 20). The heat input used to calculate the flux terms for each control volume is a fraction

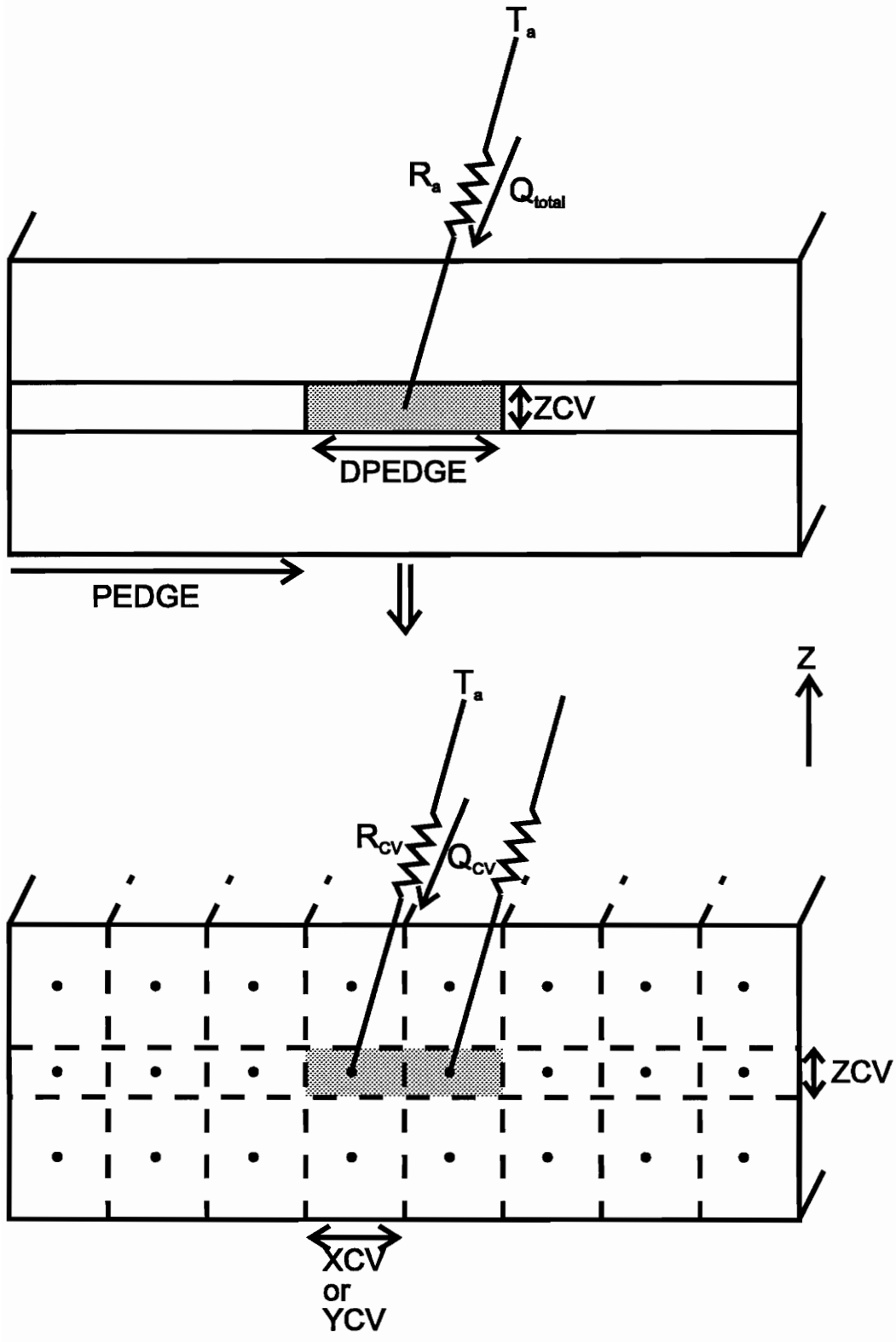


Figure 20: Definition of an Edge Connection and Associated Control Volumes

of the total heat input calculated according to the control volumes size relative to the size of the connection. For clarity, eqns. 46-50 apply only to edge connections on the J1 and M1 boundaries. The derivation for edge connections on the I1 and L1 boundaries are nearly identical, with XCV(I) replaced by YCV(J).

$$Q_{cv} = Q_{total} \cdot \frac{XCV(I) \cdot ZCV(K)}{ZCV(K) \cdot DPEDGE} \tag{46}$$

Now, evaluating eqn. 33 with $Q_{total} = Q$ from eqn. 26,

$$Q_{cv} = \frac{1}{R_a} \cdot [T_a - T(I,J,K)] \cdot \frac{XCV(I) \cdot ZCV(K)}{ZCV(K) \cdot DPEDGE} \tag{47}$$

This gives the correct heat input for a control volume within a known temperature connection.

To obtain the flux term expression equate eqn. 47 with eqn. 32 with *area* = *XCV(I)*ZCV(K)*:

$$\begin{aligned} &(FLXCx + FLXPx \cdot T(I,J,K)) \cdot XCV(I) \cdot ZCV(K) \\ &= \frac{1}{R_a} \cdot [T_a - T(I,J,K)] \cdot \frac{XCV(I) \cdot ZCV(K)}{ZCV(K) \cdot DPEDGE} \end{aligned} \tag{48}$$

FLXC and FLXP for the control volume can now be determined:

$$FLXCx(I,K) = \frac{\frac{1}{R_a} \cdot T_a}{ZCV(K) \cdot DPEDGE} \tag{49}$$

$$FLXPx(I,K) = -\frac{\frac{1}{R_a}}{ZCV(K) \cdot DPEDGE} \tag{50}$$

Here x represents $J1$ or $M1$, depending on the particular edge to which the connection is made.

The derivation for connections to the I1 and L1 faces is obtained by replacing XCV(I) with YCV(J). Repeating eqns. 46 through 48 with this substitution yields:

$$FLXCx(J,K) = \frac{\frac{1}{R_a} \cdot T_a}{ZCV(K) \cdot DPEDGE} \tag{51}$$

$$FLXPx(J,K) = -\frac{\frac{1}{R_a}}{ZCV(K) \cdot DPEDGE} \tag{52}$$

The resulting equations are identical to eqn. 49 and eqn. 50 except that the arrays are (J,K) instead of (I,K) and x represents $I1$ or $L1$, depending on the particular edge to which the connection is made.

Note that the right side of the flux equations are not edge specific. The only difference among the equations is that the variable names for the fluxes and the variable subscripts differ for different edges.

4.8 Thermal Vias

Thermal vias connect two grid points to each other through a given resistance, unlike the previously discussed features that connect one grid point to a known temperature. The connection is similar in that from the perspective of one grid point, it is connected to a known temperature. However, in this case, the temperature (previously defined as a constant T_a in surface and edge connection equations) changes from iteration

to iteration because it is another grid point in the model. This variation in temperature introduces an appparent non-linearity into the solution, increasing the number of iterations required to obtain a solution.

The effect of the connection of the grid points is modeled using sources and fluxes in the same manner as the other connections. Given two temperatures at grid points 1 and 2, the effect of T_2 on T_1 can be described by modifying the source or flux terms at grid point 1 using equations eqn. 30 and eqn. 31 if T_1 is an interior grid point or eqn. 33 and 34 if T_1 is a boundary grid point. In these equations, T_a assumes the value of T_2. The effect of T_1 on T_2 would be modeled in the reverse fashion, modifying the source or flux terms at grid point 2 with the same equations using $T_a = T_1$.

Since the effect of via connections only involves one control volume, it is not necessary to divide the effect among many control volumes as was done for the other types of connections.

It is important to remember that early in the numerical solution, the temperatures involved in a via connection are being modified using other temperatures within a solution domain that is not converged. They are therefore not accurate. Since these temperatures are linked to each other, the effect is perpetuated in a sense, making an otherwise linear problem appear to be non-linear. This change usually has profound effects on the convergence speed. Cases involving via connections can make the selection of accurate initial temperatures especially important.

5. Numerical Behavior

5.1 Grid Refinement

Increasing the grid resolution can easily produce a numeric system with so many equations that it overwhelms the computational resources. Therefore, it is prudent to use the coarsest grid refinement possible, while maintaining numerical accuracy. The resolution of the numerical model depends not only on the grid refinement, which changes the resolution in the x-y plane, but also on the number of layers used. The user may choose to use greater or fewer layers in the numerical model than are present in the physical model. This would affect the resolution in the z-direction. Chapter 6 discusses the removal of a layer from the model in greater detail.

5.2 Convergence

The solution of the model must be monitored carefully to insure that it is converging. Failure to converge indicates an improperly formulated problem or unattainable convergence criteria. Convergence speed is affected by both the physical properties of the problem and by the numerical configuration.

The convergence of the solution is monitored in the OUTPUT part of the user subroutine. Typically, an energy balance on the entire device is the best method to check the solution's progress. A general energy balance is defined by:

$$E_{total} = E_{in} + E_{out} + E_{gen} \tag{53}$$

Convergence is reached when $E_{total} = 0$. In practice, a value of zero cannot actually be obtained (due to computational accuracy), so when the energy balance is very small, the solution is considered converged.

Within the context of the code, the energy balance is calculated by summing the effects of all source terms, fluxes (meaning FLUX*x*, the values returned by the code, not the values defined by the user). It is important to remember these terms are multiplied by the appropriate geometric quantities (volume for sources, area for fluxes).

An additional method for checking convergence can provide assurance that the solution is converged. A good procedure is to monitor the change in temperature values from iteration to iteration. When the temperatures show very little variation, it is an indication that the solution is no longer changing. By itself, this is *not* an acceptable convergence test, but it does provide a useful adjunct to the energy balance.

It is important to remember that convergence is only an indication that the numerical scheme has arrived at a solution. It does not mean that the solution itself is a correct indication of the behavior of the physical device.

5.2.1 Effects of the External Environment

The external environment can have a profound effect on the solution of the problem. The convective cooling parameters can either emphasize conduction within the device, or render it less important.

A common situation (and one encountered in the example problem) is one in which the majority of the heat transfer occurs through the base of the device, with

negligible heat transfer through the top and edges of the device. With a relatively large heat source positioned on the top surface, the behavior of the model resembles the corresponding one-dimensional solution. In this situation, the spreading resistance of the device is subordinate to the vertical resistance.

In problems where the top external environment has a significant impact on the heat transfer and the size of the heat source is smaller, the role of spreading is important. The spreading serves to increase heat loss through the top surface.

5.2.2 Effects of Nonlinearity

Although non-linear effects are often encountered in heat transfer, the structure of the code is such that most non-linear effects must be linearized to be included (e.g., boundary conditions and fluxes). However, using via connections introduces an element of non-linearity into the solution.

At each outer iteration, a set of linearized equations is sent to the SOLVE subroutine, regardless of the linear or non-linear nature of the problem. If the problem is indeed linear, then the subroutine SOLVE can arrive at the correct solution, given that the convergence criteria is sufficiently stringent and that the NTIMES value is high enough to allow enough inner iterations (here inner means refers to the iterations of the SOLVE subroutine, not the outer iterations of the code). If the set of equations is non-linear, many outer iterations may be needed to arrive at the correct solution to the set of equations. In these cases, it is not usually numerically efficient to allow SOLVE to fully converge because outer iterations will change the values of the discretization equations.

Instead, it is better to limit the inner iterations to a relatively low number (e.g., NTIMES=10) an then allow the outer iterations to bring the solution closer to the correct one. As the outer iteration approaches convergence, the number of iterations required by SOLVE approaches one. The structure of the outer and inner iterative loops and their role in the solution is discussed in greater detail in sections 2.5 and 2.6.

5.3 Physical Example

A typical application of the code would consist of the analysis of the heat distribution within a multichip module (MCM). A simple example of an MCM consists of four chips positioned on a two layer substrate consisting of polyimide and alumina (see Fig 21). Each chip is positioned over a set of thermal vias in the polyimide layer (see Fig 22). This is necessitated by the relatively low conductivity of polyimide. Wire bonds extend from the edges of each chip connecting it to the layer below. The assembly has mounting pins attached to the perimeter of the alumina layer (see Fig. 23). Through connections are present on the periphery of the MCM. The module is cooled through convection on the top and sides and from contact with a heat sink on the bottom. Note that the model is symmetric in two planes so it is only necessary to obtain a solution of one-quarter of the MCM (see Fig. 24).

To numerically model this device, the physical attributes of the MCM are converted into their numerically analogous counterparts. The heat input from the chip (14 W) is input as a heat source. The limits of the top layer dimensions are defined (in this case identical to the area of the heat source).

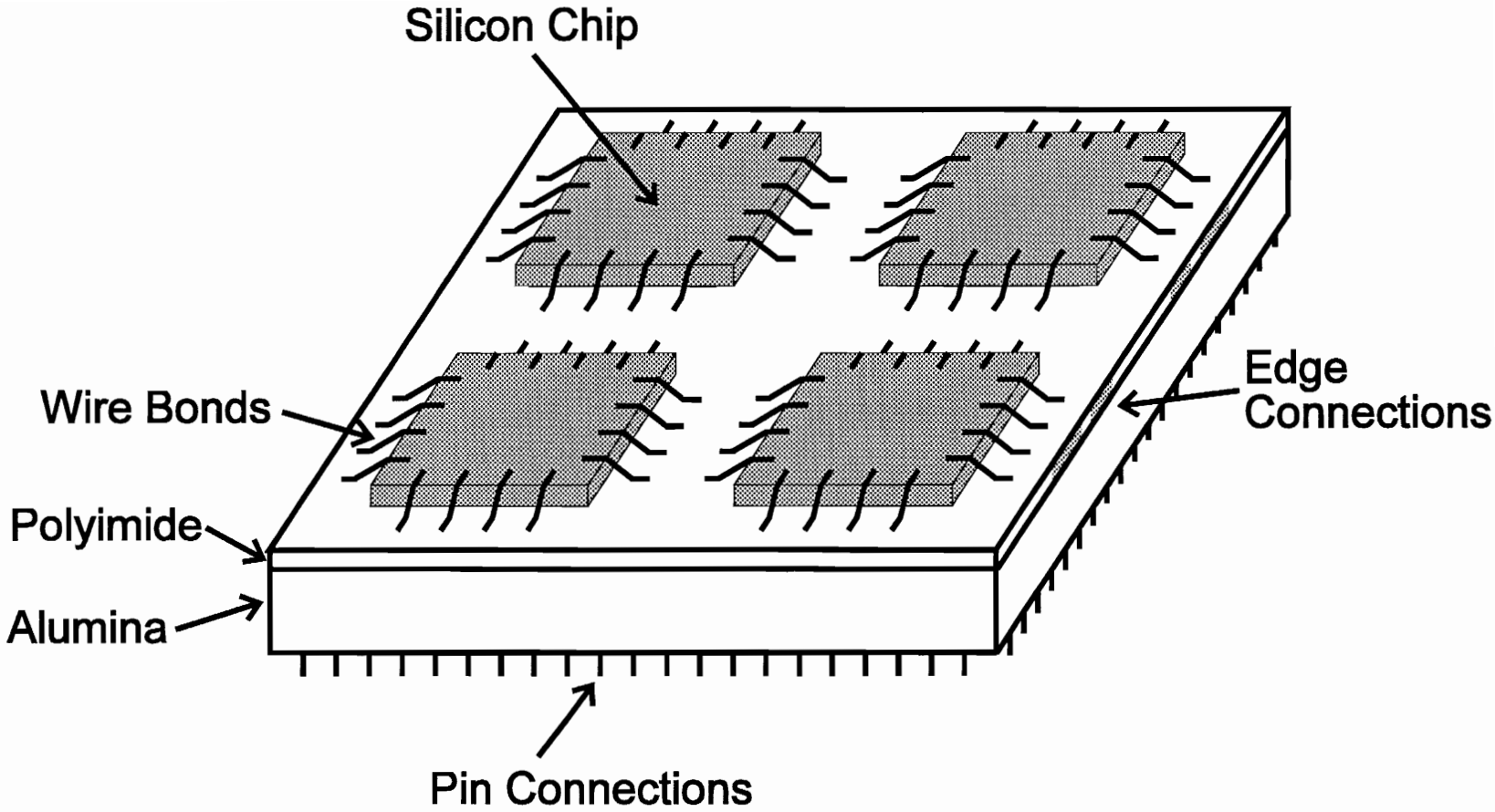


Figure 21: Physical Model

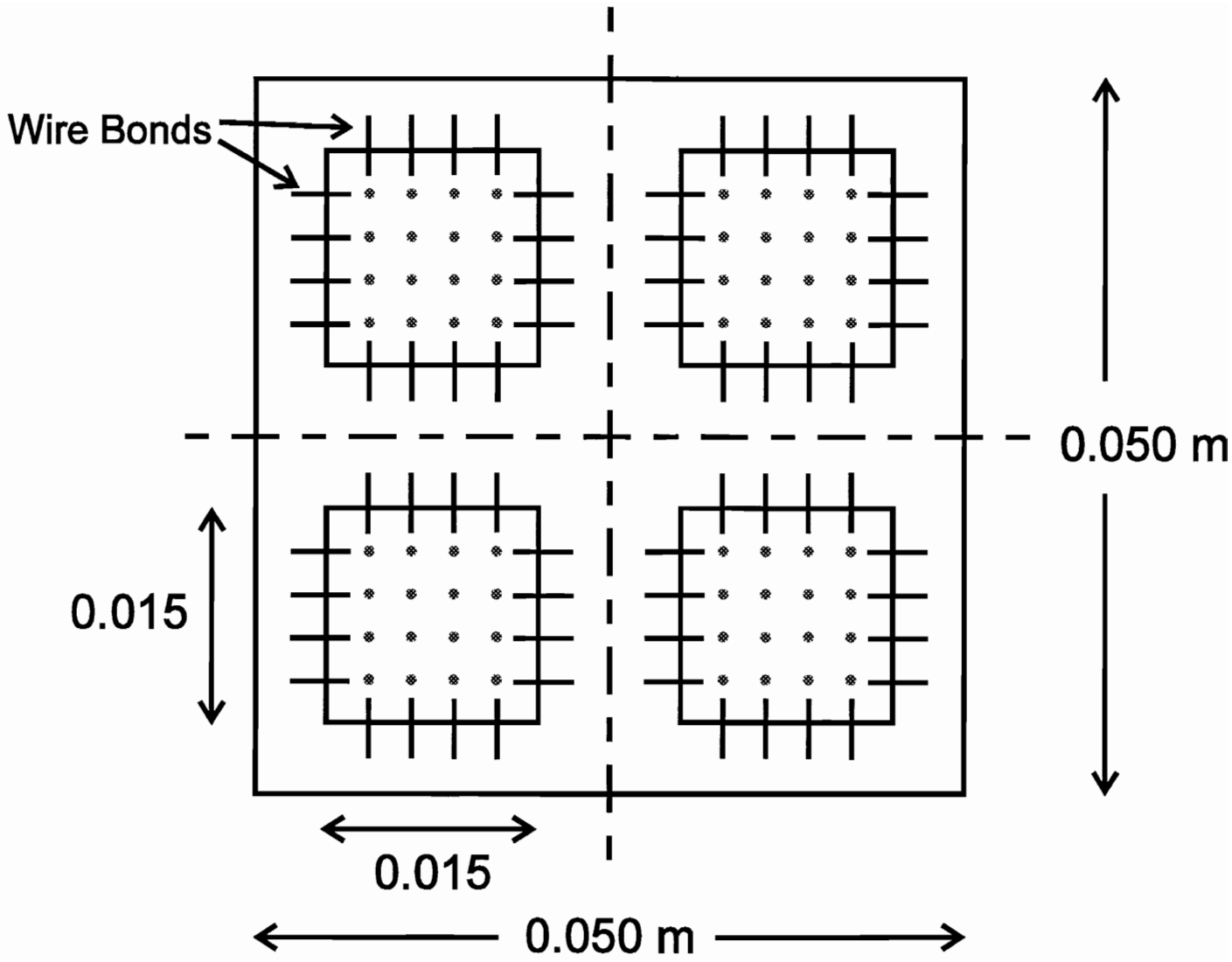


Figure 22: Top View of the Physical Model

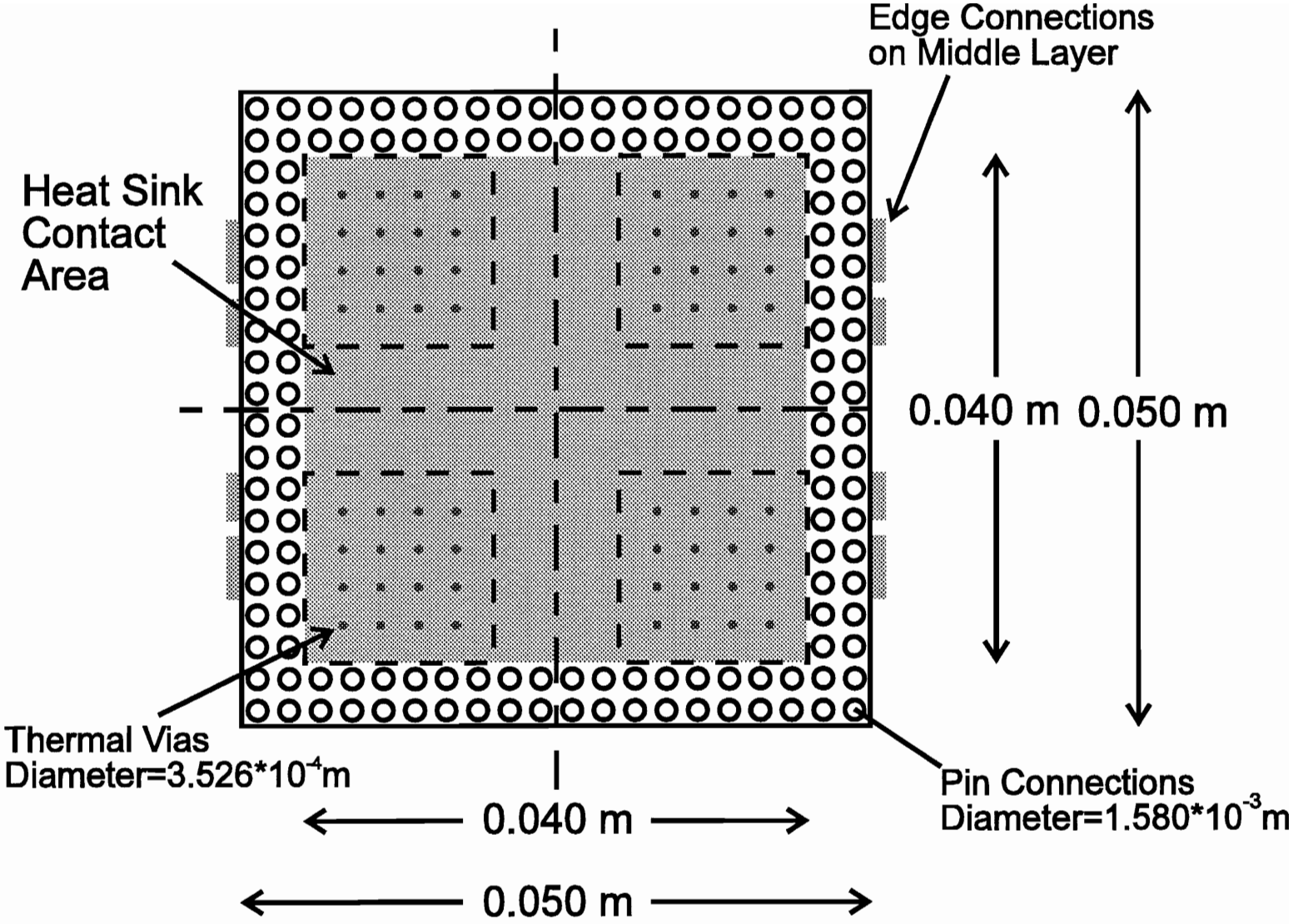


Figure 23: Bottom View of the Physical Model

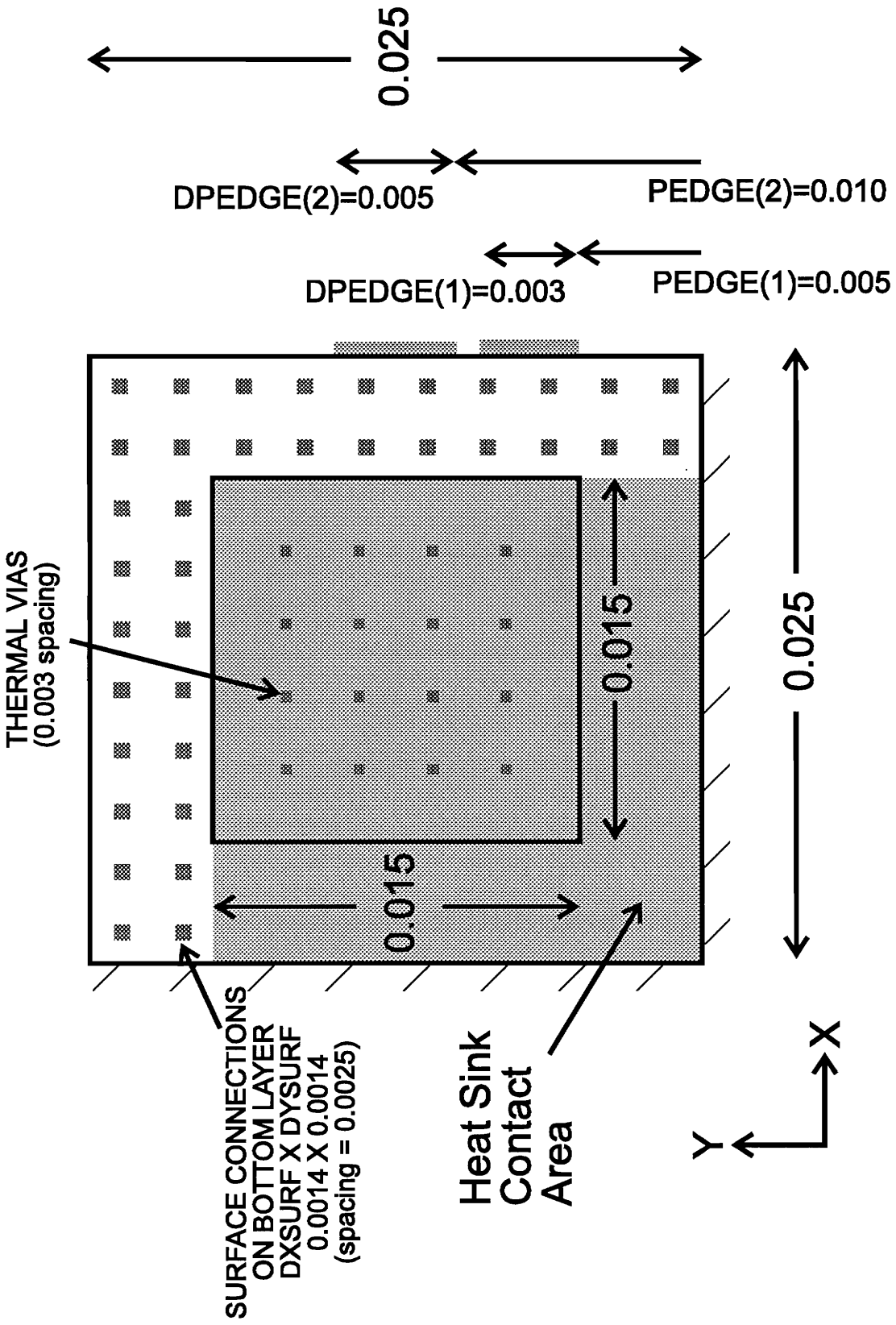


Figure 24: Top View of the Numerical Model

5.3.1 Thermal Vias

Each thermal via in the polyimide layer (node K=3) is modeled as a via connection from node K=2 (the alumina layer) to node K=3 and a connection from node K=3 to node K=2 (the chip layer). The vias, which have a circular cross-section in the physical device, are defined as node-to-node connections in the numerical model. These numerical vias can then be thought of as having a rectangular cross-section, since the control volume cross-sections are rectangular. Depending on the grid size used, the size of this rectangular cross-section may vary, however, the numerical effect will be the same. The strength of the source terms used to model the via is dependent on the size of the control volume. A larger control volume will have a weaker term; a smaller control volume will have a stronger term. The effect on the solution will be similar as long as the control volume cross-sectional area is not absurdly small or large. For most problems a grid size producing the latter would be too coarse to resolve the problem and a grid size producing the former would be so small as to be unnecessary and inefficient. The resistance used to model the thermal vias are determined by the physical properties of both the vias and the adjoining layers. In the physical device, there is only one path through which the heat transfer can follow in the vias. In the numerical model, there exists two paths of heat transfer between nodes that are at either end of a via connection. The heat transfer over one path is governed by the coefficient relating grid points to their neighbors, which is determined by the conductivities associated with the grid points and the distance separating them. The heat transfer through the second path is determined by the

resistance of the via connection. These two paths for heat transfer are in parallel, which means that the total heat transfer is governed by the path of least resistance, which in this numerical model would be the via connection. (Actually not just *this* numerical model, but *any* numerical model, since the point of vias is to lower the thermal resistance.) It is therefore important that the appropriate via resistance is used. The resistance in the numerical model is a sum of the resistance of the via and the resistance of the layer through which the connection must be made (see Fig. 25). This is calculated from the physical properties of the via and the adjacent layers. The resistance for the entire via is given by:

$$R_{via} = \frac{d_{via}}{k_{via} \cdot A_{via}} = \frac{0.001\,m}{100\ W/m{\cdot}K\ \cdot\ 9.766 \cdot 10^{-8} m^2} = 102.4\ K/W \qquad (54)$$

where d_{via} is the length of the via, k_{via} is the conductivity of the via, and A_{via} is the area of the via. Similarly, the resistance in half of the alumina layer is:

$$R_{Al} = \frac{d_{Al}}{k_{Al} \cdot A_{cv}} = \frac{0.005\,m}{29\ W/m{\cdot}K\ \cdot\ 9.766 \cdot 10^{-8} m^2} = 1765\ K/W \qquad (55)$$

and the resistance of half the chip is:

$$R_{Si} = \frac{d_{Si}}{k_{Si} \cdot A_{cv}} = \frac{0.00025\,m}{153\ W/m{\cdot}K\ \cdot\ 9.766 \cdot 10^{-8} m^2} = 16.73\ K/W \qquad (56)$$

Note that $A_{cv} = A_{via}$ for an 80 X 80 grid.

Since node K=3 is situated in the center of the control volume, half of R_{via} would be used to calculate the resistance for the via connection to the layer above (K=4) and half

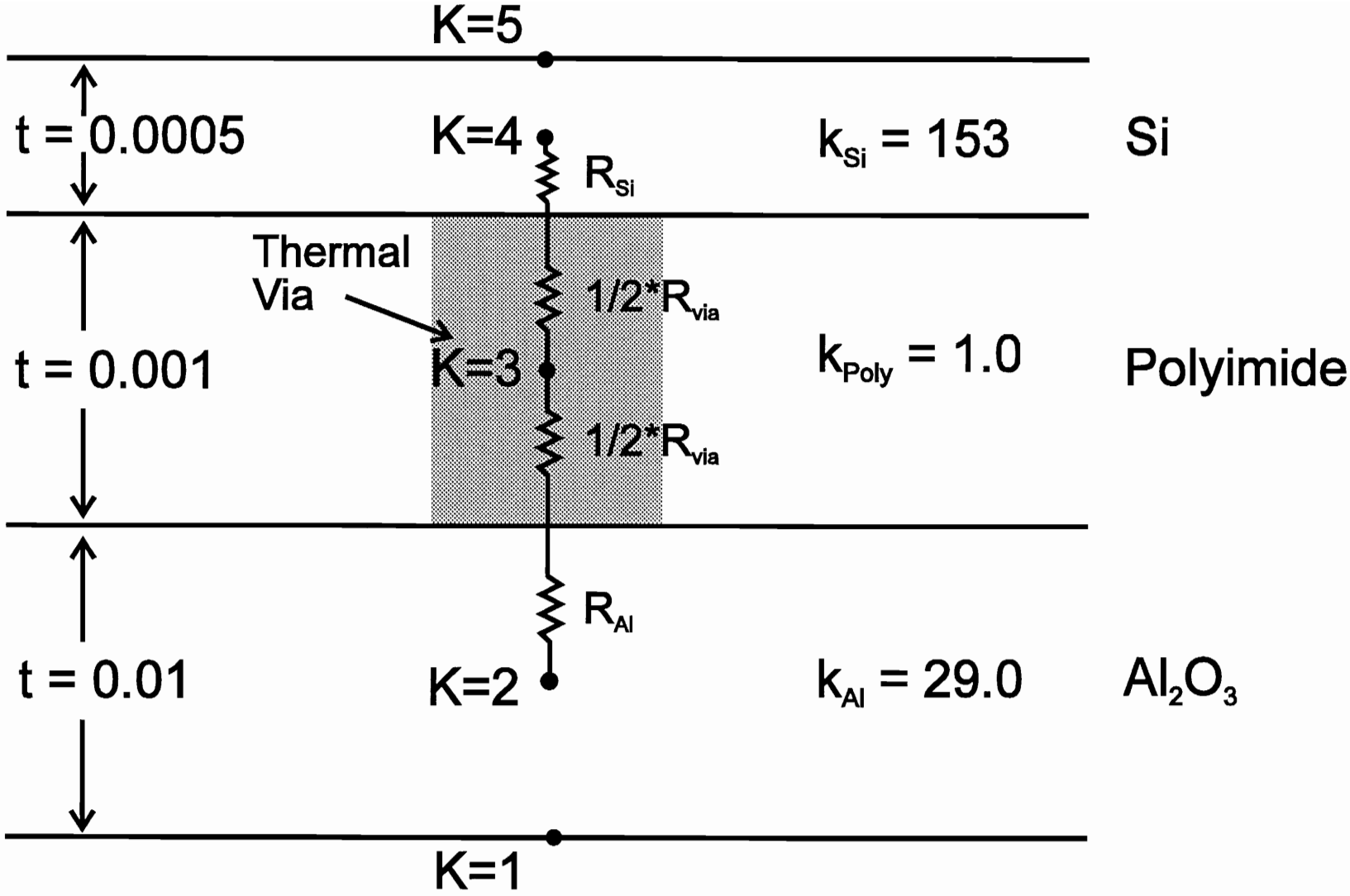


Figure 25: Side View of a Thermal Via and Associated Resistances

would be used for the connection to node below (K=2).The appropriate resistance for each via connection from node K=3 to node K=2 would then be:

$$R_{3\leftrightarrow 2} = \frac{R_{via}}{2} + R_{Al} = 1816\ K/W \tag{57}$$

and for node K=3 to node K=4:

$$R_{3\leftrightarrow 4} = \frac{R_{via}}{2} + R_{Si} = 67.93\ K/W \tag{58}$$

The suitability of the method used to numerically model the thermal vias can be verified by manually modifying the conductivity of the model within PHI. To do this, the conductivities of the appropriate control volumes within the middle layer are changed to reflect the higher conductivities of the thermal vias. In effect, this change creates thermal vias in the shape of the altered control volumes. Therefore, the grid spacing is of major importance in determining the effect of the new conductivities. In this particular example, a uniform 80 X 80 grid produces control volumes with the same cross-sectional area as that of the thermal vias in the physical model. A comparison of the maximum temperatures in Table 1 shows little difference, demonstrating that the method used to model the thermal vias is a valid implementation within the context of the code.

Table 1: Comparison of Maximum Temperature in the Model

	Via Connections	Altered Conductivity
Vias Alone	92.761	92.995
Vias with other connections	83.136	83.206

The slightly lower temperatures obtained using the via connections can be attributed to the presence of the two paths for heat transfer. In addition to the avenue provided by the via connections, heat transfer still occurs through the mechanism that relates each grid point to its neighbors (the base polyimide material). Even though this path has a much higher resistance, enough heat transfer occurs to cause the slightly lower temperatures calculated by the via connection method.

5.3.2 Wire Bonds

The wire bonds are also modeled as vias from the point of connection in the chip to the point of connection on the substrate. In the physical model, these are surface-to-surface connections. However, in the numerical model they must be node-to-node connections (in this case, interior grid points are used). Since both ends of these via connections are modeled as source terms, it is necessary to account for the added resistance caused by the layer thickness between the physical model's connection point and the numerical model's connection point (see Fig. 26). To calculate the resistance caused by the layers, and appropriate area must be used. In this situation, the correct area is that of the control volume in which the connection is made. Since an 80 X 80 uniform grid is used, the area of a control volume is easily found:

$$A_{cv} = \left(\frac{0.025\,m}{80}\right)^2 = 9.766 \cdot 10^{-8}\ m^2 \tag{59}$$

This is then used in the resistance calculations:

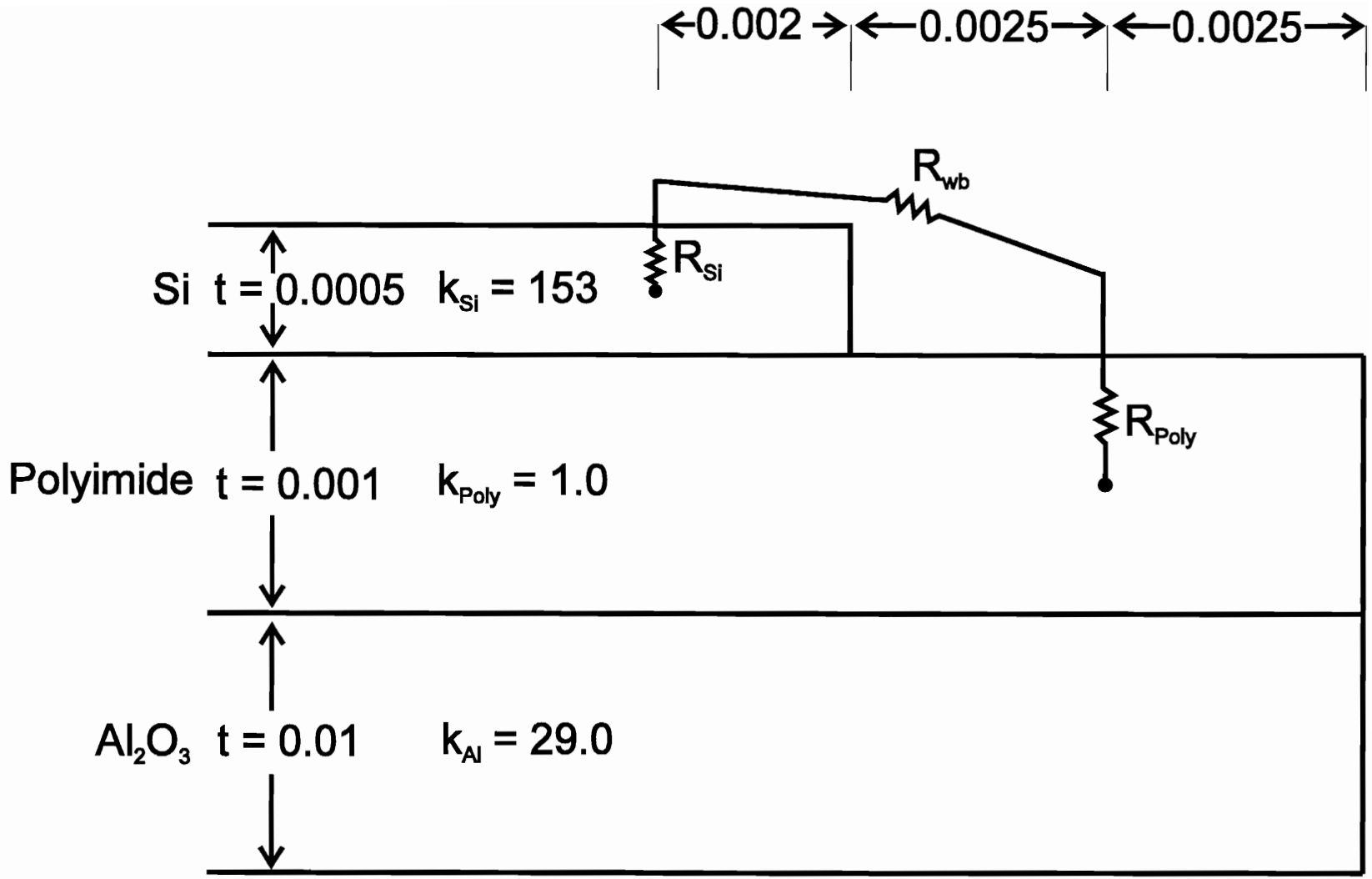


Figure 26: Side View of a Wire Bond and Associated Resistances

$$R_{Si} = \frac{\frac{1}{2} \cdot t_{Si}}{k_{Si} \cdot A_{cv}} = \frac{0.00025\,m}{153\ W/mK \cdot 9.766 \cdot 10^{-8}\,m^2} = 16.73\ K/W \tag{60}$$

$$R_{Poly} = \frac{\frac{1}{2} \cdot t_{Poly}}{k_{Poly} \cdot A_{cv}} = \frac{0.0005\,m}{1.0\ W/mK \cdot 9.766 \cdot 10^{-8}\,m^2} = 5120\ K/W \tag{61}$$

These resistances are then added to the resistance of the wire bond,

$$R_{wb} = \frac{d}{k_{wb} \cdot A_{wb}} = \frac{0.0045\,m}{150\ W/mK \cdot 1.824 \cdot 10^{-8}\,m^2} = 1640\ K/W \tag{62}$$

to obtain the overall resistance of the via representing the wire bond:

$$R_{wbvia} = R_{Si} + R_{Poly} + R_{wb} = 6780\ K/W \tag{63}$$

5.3.3 Pin Connections

The pins on the bottom layer are modeled as surface connections. The physical pin connection is circular, however the method of defining surface connection makes it necessary for the connection to be rectangular in the numerical model. The resistance of the pin connections is considered to be R_{pin}=62.5 K/W.

5.3.4 Through Connections

The through connections are modeled as edge connections. Two edge connections are defined to act on middle (K=3) layer. For comparison purposes, the resistance of the edge connections is considered to be the same as the pin connections, R_{edge}=62.5 K/W.

5.3.5 Results

The solutions for each layer and the top and bottom surfaces are plotted using the same scale in Fig. 27. Figures 28-32 show these same results in more detail. The x and y coordinate variables represent control volume indices I and J. The values for the edge grid points are not included (recall that they are not part of the solution and must be extrapolated). In the case of Figures 28 and 29, only the grid point values for the chip are plotted (i.e, the values outside of the defined boundaries of the chip are not included). The z coordinate represents temperature. Note that the limits on the z axis vary in range from plot to plot. For the top layer and surface, the range of temperature is quite small so that the temperatures plotted to three significant figures appear "stepped". It these were plotted on a larger scale, they would appear flat. All of the surface plots have the same view, with the exception of the plots for K=1, where the view is rotated 180°. The contour plots all have the same orientation as the diagrams of the numerical model.

Inspection of Figures 28 and 29 reveals their similarity. The K=5 plot is essentially identical to the K=4 plot with incrementally higher temperatures. The higher temperatures are a result of the extrapolation from the lower layer to a convective boundary condition (remember that the K=5 values are boundary points). The maximum temperature occurs at nearly the center of layer 5, as would be expected in light of the low convective cooling on the top of the device and the large size of the heat source. However, the position of the maximum temperature might be expected to occur closer to the insulated perimeter

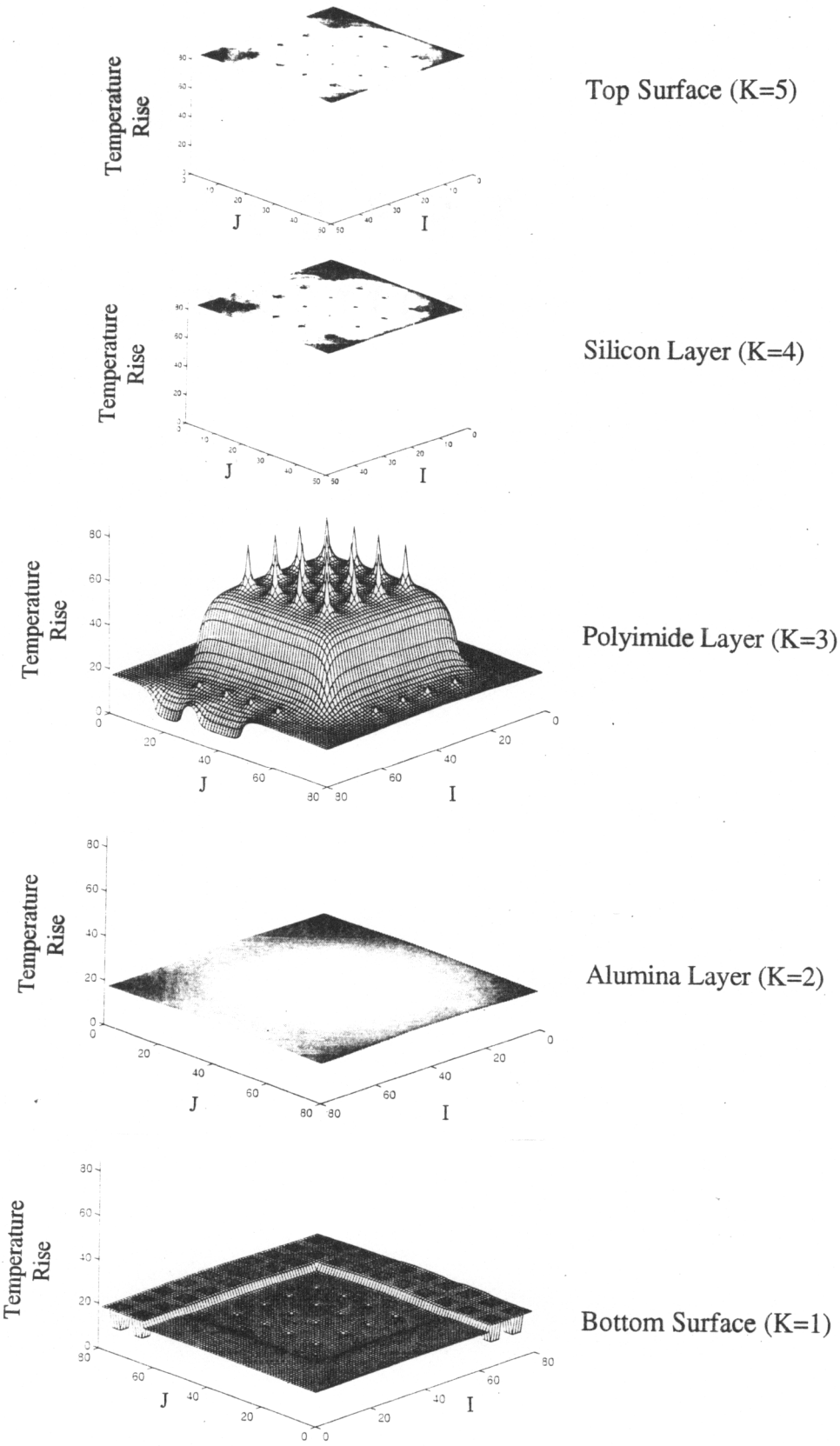


Figure 27: Grid Point Temperature Rise in the Complete Model

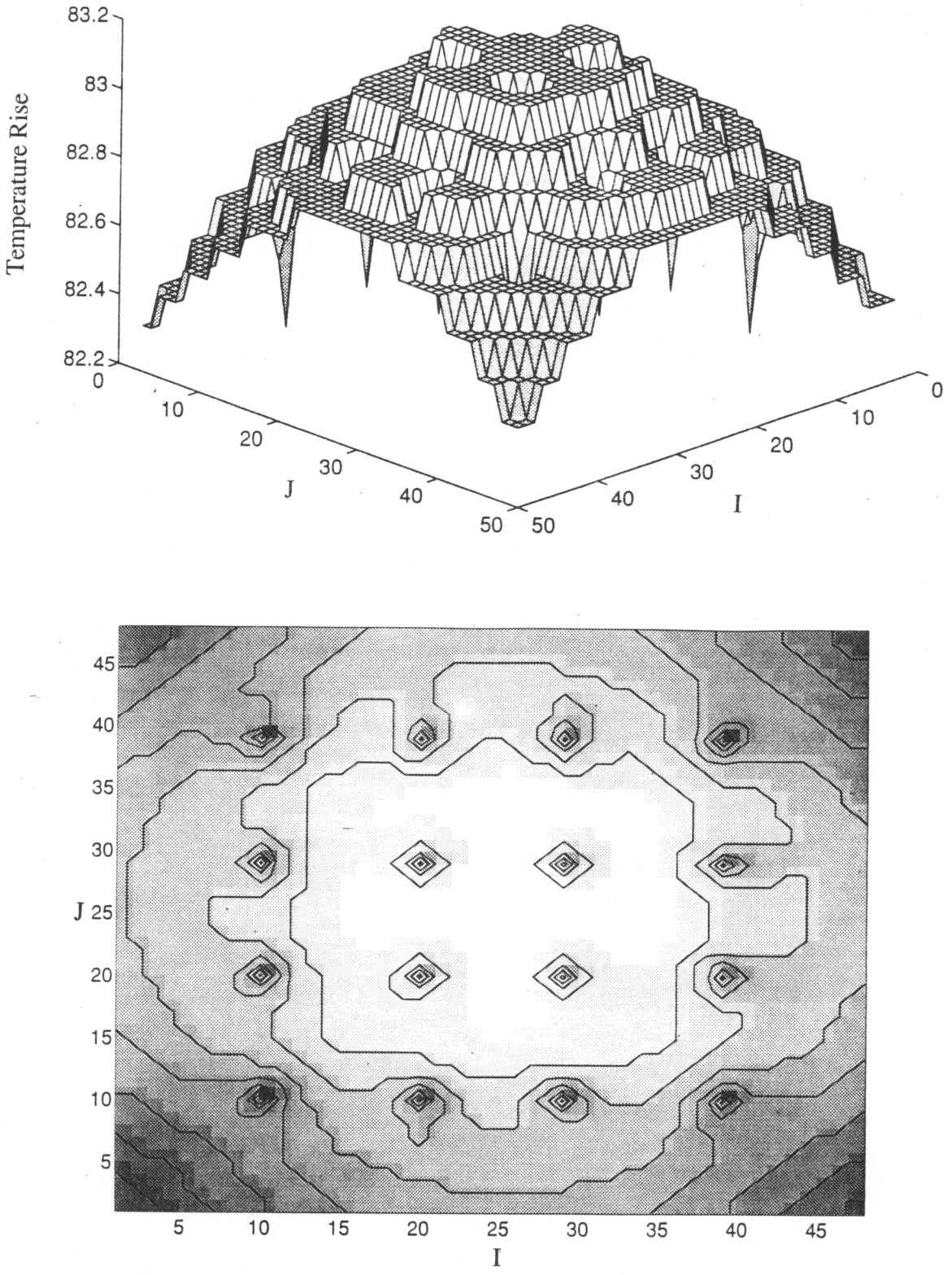


Figure 28: Temperature Rise on the Top Surface (K=5) for the Complete Model

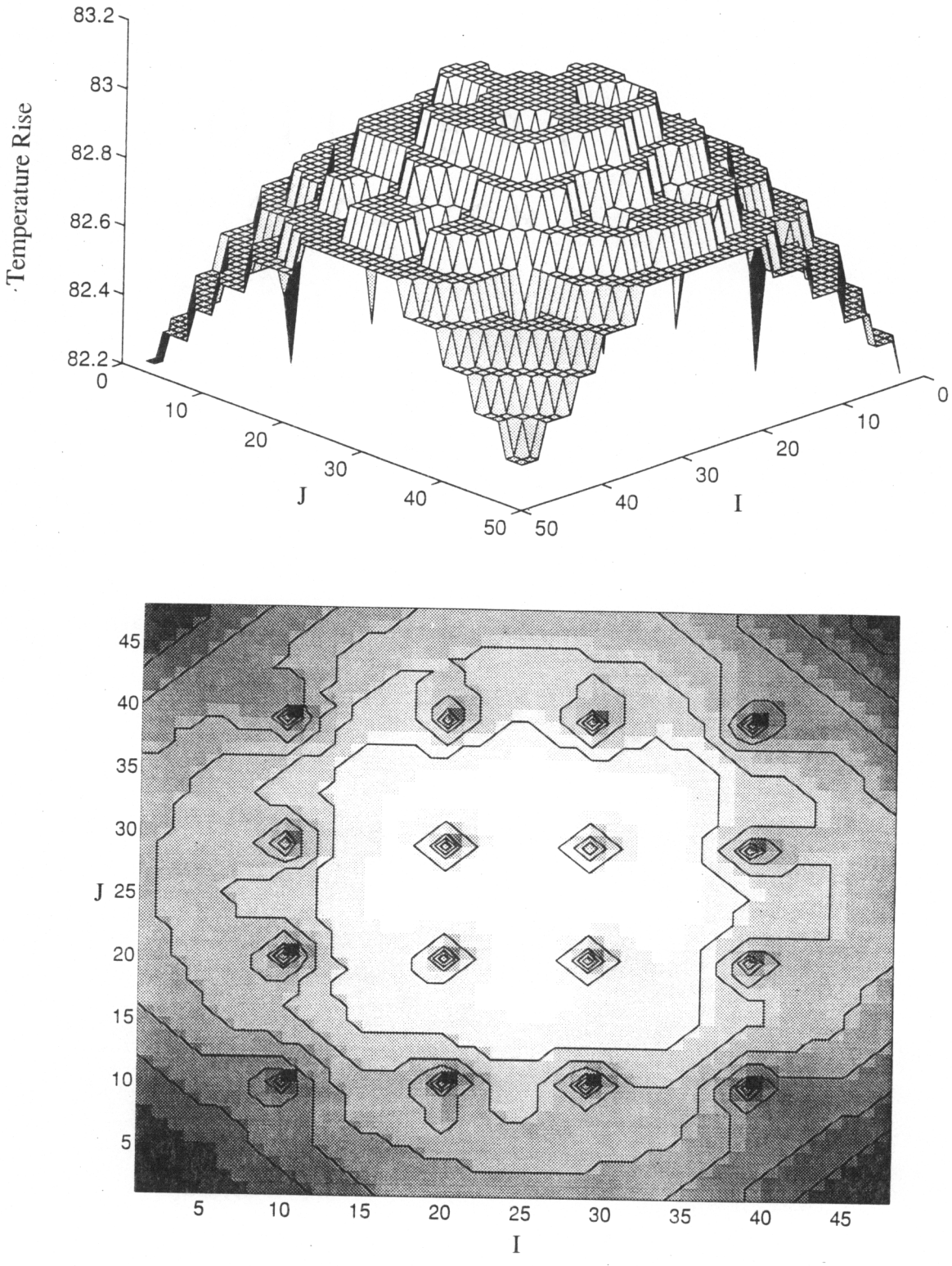


Figure 29: Temperature Rise in the Silicon Layer (K=4) for the Complete Model

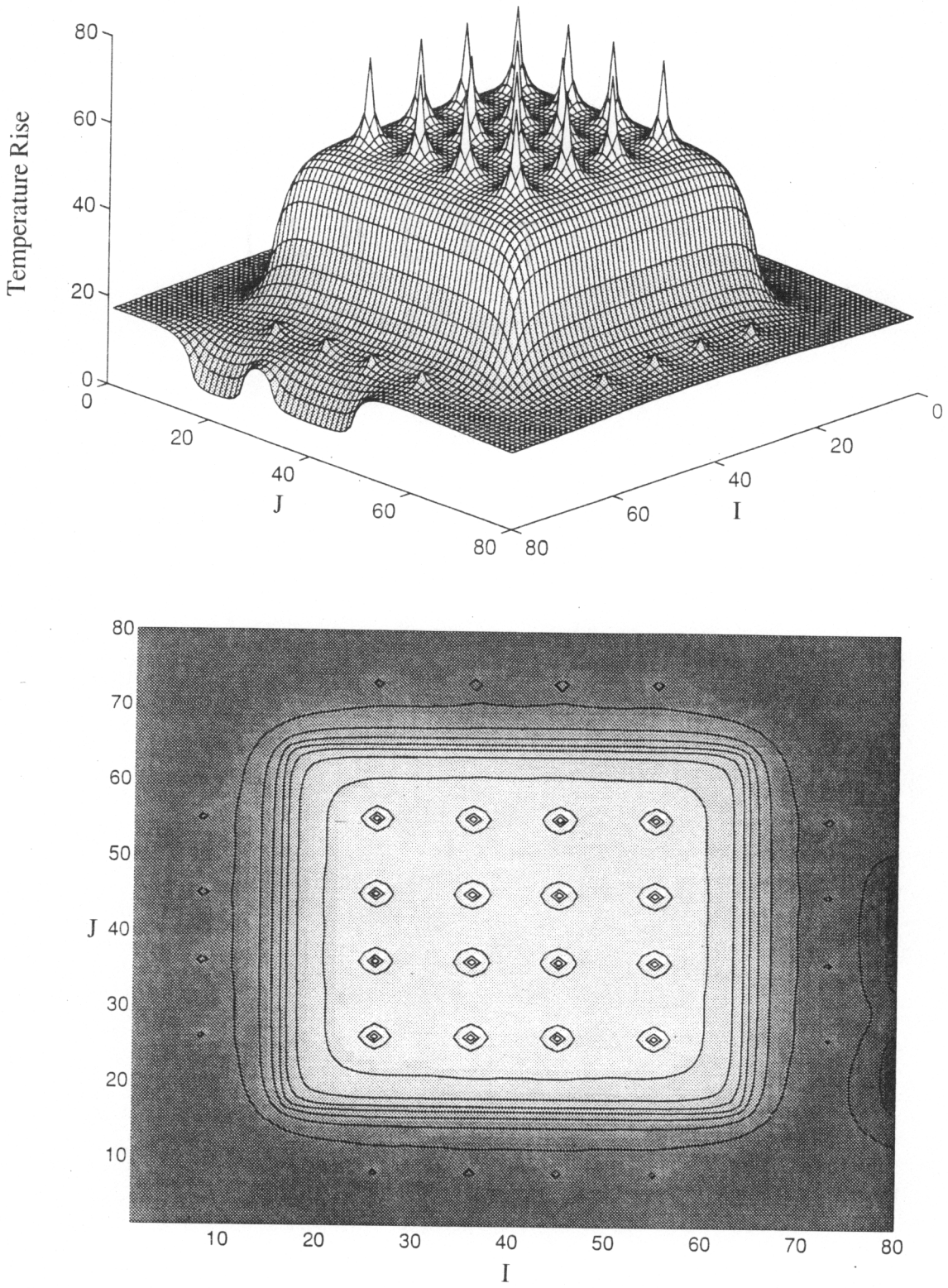


Figure 30: Temperature Rise in the Polyimide Layer (K=3) for the Complete Model

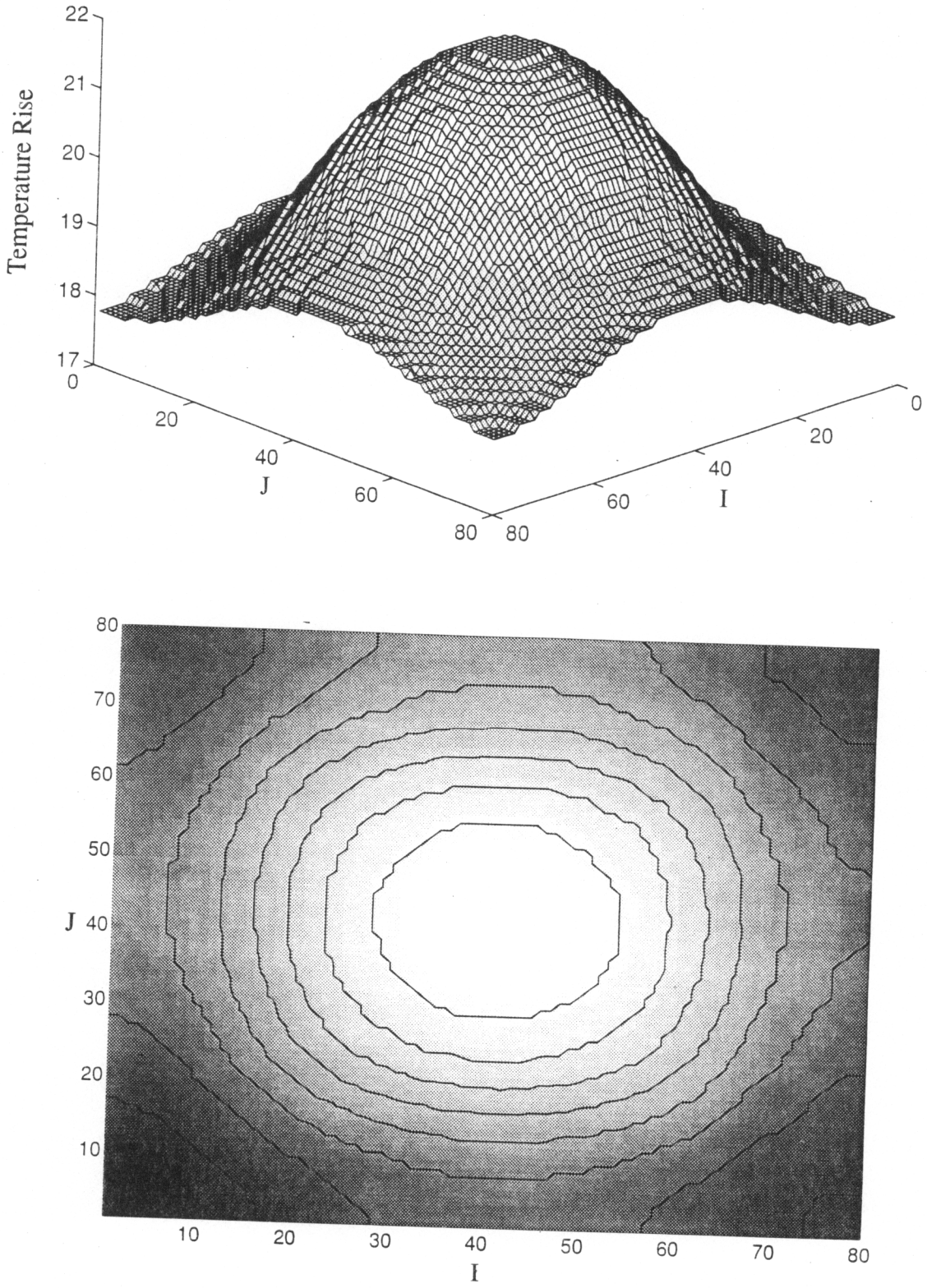


Figure 31: Temperature Rise in the Alumina Layer (K=2) for the Complete Model

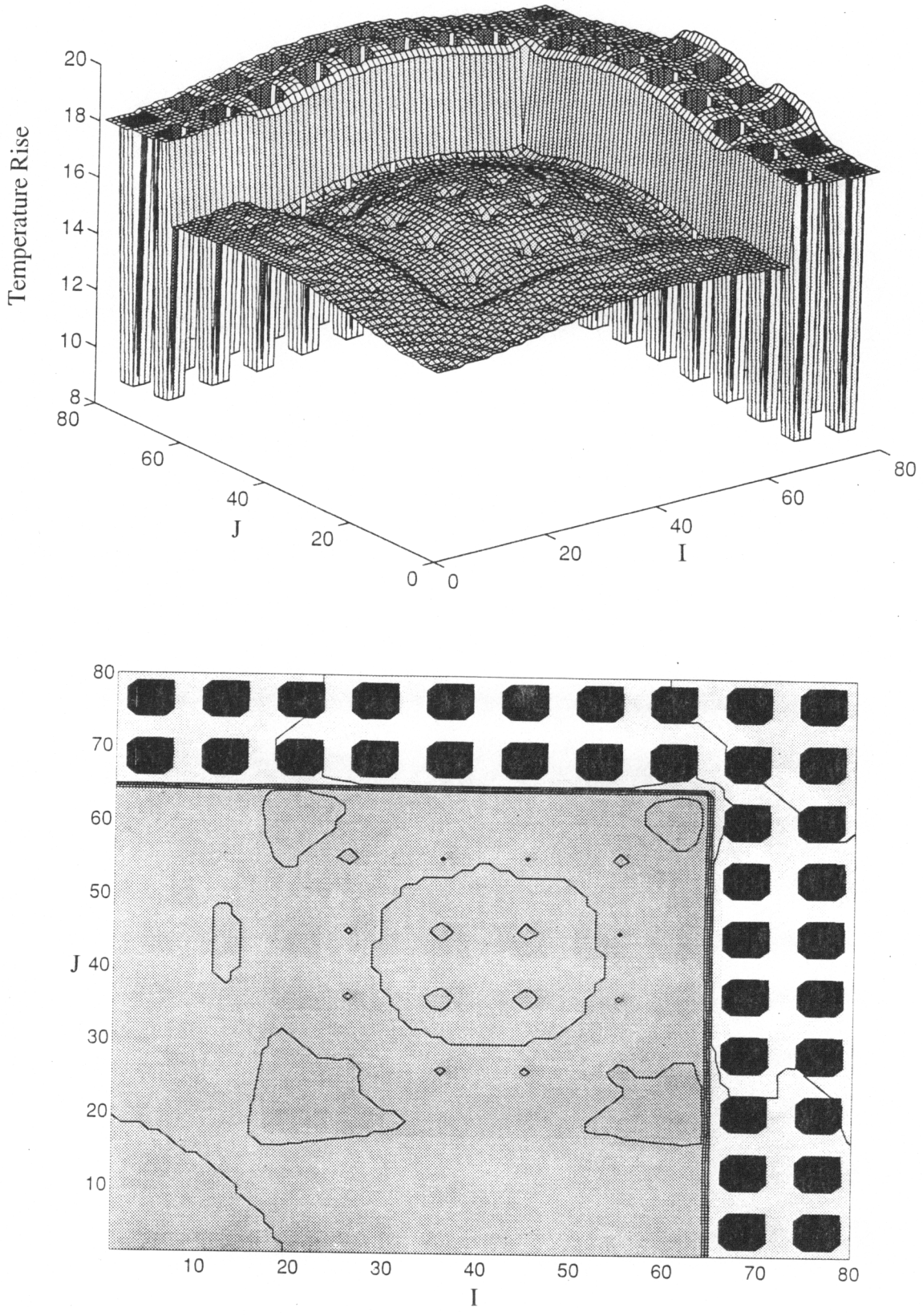


Figure 32: Temperature Rise on the Bottom Surface (K=1) for the Complete Model

containing the pin connection. As will become apparent, the addition of the connections to the model serve to shift the location of the maximum temperature.

Figure 30 shows the K=3 layer. Here the effects of the thermal vias can clearly be seen. The via from layer 4 to layer 3 connects the relatively high temperatures in layer 4 to the cooler temperatures in layer 3, raising the temperatures at the points of connection in layer 3 and lowering the temperatures in layer 4. Looking again at Figs. 28 and 29 shows the effect of the vias in the form of peaks on the underside of the plots.

Figure 31 shows the K=2 layer. Here via connections are not apparent, although they are present in the form of a connection to layer 3. The high resistance of these connections render them ineffective in affecting the temperature of layer 2. The plot represents the spreading of the centered heat source.

Figure 32 shows the temperatures on the bottom, or K=1 layer. The discontinuity in temperature is a product of the extrapolation method. Since the extrapolation depends on the boundary conditions, the calculated temperatures in the area of the heat sink are lower than those extrapolated to the insulated perimeter (with the exception of the location of the pin connections, where, of course, an insulated boundary condition does not exist). This behavior is not only mathematically consistent, but physically sensible, as the temperature over the heat sink should be lower. However, the extrapolation does not account for the spreading that occurs near the bottom surface. In the physical model, this would serve to somewhat smooth the transition from the heat sink to the insulated perimeter.

In the heat sink area, there are small dimples corresponding to the placement of the thermal vias. The code uses a second-order method to extrapolate the temperature on the boundaries. This means that the extrapolated temperature is a product of the boundary condition and the two adjacent interior layers. In this case, the extrapolated temperatures depend on the temperatures in layers 2 and 3. This accounts for the dimples, as they are essentially reflections of the temperature peaks seen in layer 3.

In the insulated perimeter, the pin connections have a drastic effect on the temperatures. Since they are strong connections to a low temperature in an otherwise insulated region, the temperatures at the pin connections are much lower than the surrounding area. Here again, since these are extrapolated temperatures, the effect of spreading is not apparent.

It is interesting to compare the effect of each element of the model on the maximum temperature (expressed as rise above ambient) and the convergence.

Table 2: Results for the Solution of the Problem and Effects of Individual Connections

Type of Connection:	None	Pins Only	Edge Only	Thermal Vias Only	Wire Bonds Only	Full Model
Maximum Temperature	97.252	88.429	96.749	92.761	96.563	83.136
Difference vs. No C.	-----	-8.823	-0.503	-4.491	-0.689	-14.116
Location (I,J,K)	50,50,5	43,43,5	49,50,5	51,51,5	49,49,5	42,42,5
Iterations to Convergence	6	3	7	46	7	41

The pin connections and the thermal vias have a large impact on the overall maximum temperature of the model. The edge connections have little effect because of their connection area and placement in the model. Because of the wire bonds high resistance, they only minimally change the maximum temperature.

Note that the sum of the changes in temperature produced by each connection alone (-14.506) is close to that obtained when the connections are implemented simultaneously. This behavior is favorable, as it allows the user to estimate the effects of additions to the model without having to obtain a complete solution. However, this will not always be the case. Variations in the environment, heat source size, and geometry could disrupt this trend.

The position of the maximum temperature varies considerably with the configuration. (A value of (0,0,5) would refer to the intersection of the I1 and J1 boundaries on the top surface.) In the solution with none of the connections, the maximum temperature occurs closer to the insulated perimeter. However, when the pin connections are added to the insulated perimeter, the location moves towards the insulated boundaries. The addition of the pin connections reduces the area of the insulated perimeter. The use of the edge connections moves the location of the maximum temperature away from the edge connection by one control volume. The thermal vias move the location slightly towards the insulated perimeter, whereas the wire bonds have the same effect in the opposite direction.

The addition of via connections hinders the convergence considerably because of the non-linear effect previously discussed. However, the via connection must have a fairly low resistance to increase the number of iterations. The wire bond vias had little effect on convergence speed because of their high resistance.

It is enlightening to examine the plots for the solution of the model with none or only one set of connections present.

No Connections

The solution of the model without any of the surface, edge, or via connections demonstrates typical conduction behavior (see Figs. 33-37). (The heat sink on the bottom and convection from the top are still active.) The K=5 and K=4 plots are smooth, showing none of the irregularities caused by the thermal vias in the complete model. (Remember that if the K=3 scale were used to plot layers 4 and 5, they would appear nearly flat.) The maximum temperature occurs towards the insulated perimeter. The only discontinuity in the plots is in layer 1, where the transition from the heat sink to the insulated perimeter occurs.

Edge Connections Only

The results using the edge connections strongly resemble the results obtained with no connections (see Figs. 38-42), differing only at the connection site. In the K=5, K=4, and K=2 plots, the maximum temperature occurs towards the insulated perimeter of the device, away from the center of the heat sink. Note that the variation in temperature apparent at the source of the edge connection in layer 3 is reflected into layer 1 by the extrapolation. Layer 1 still shows the sharp discontinuity in temperature corresponding to

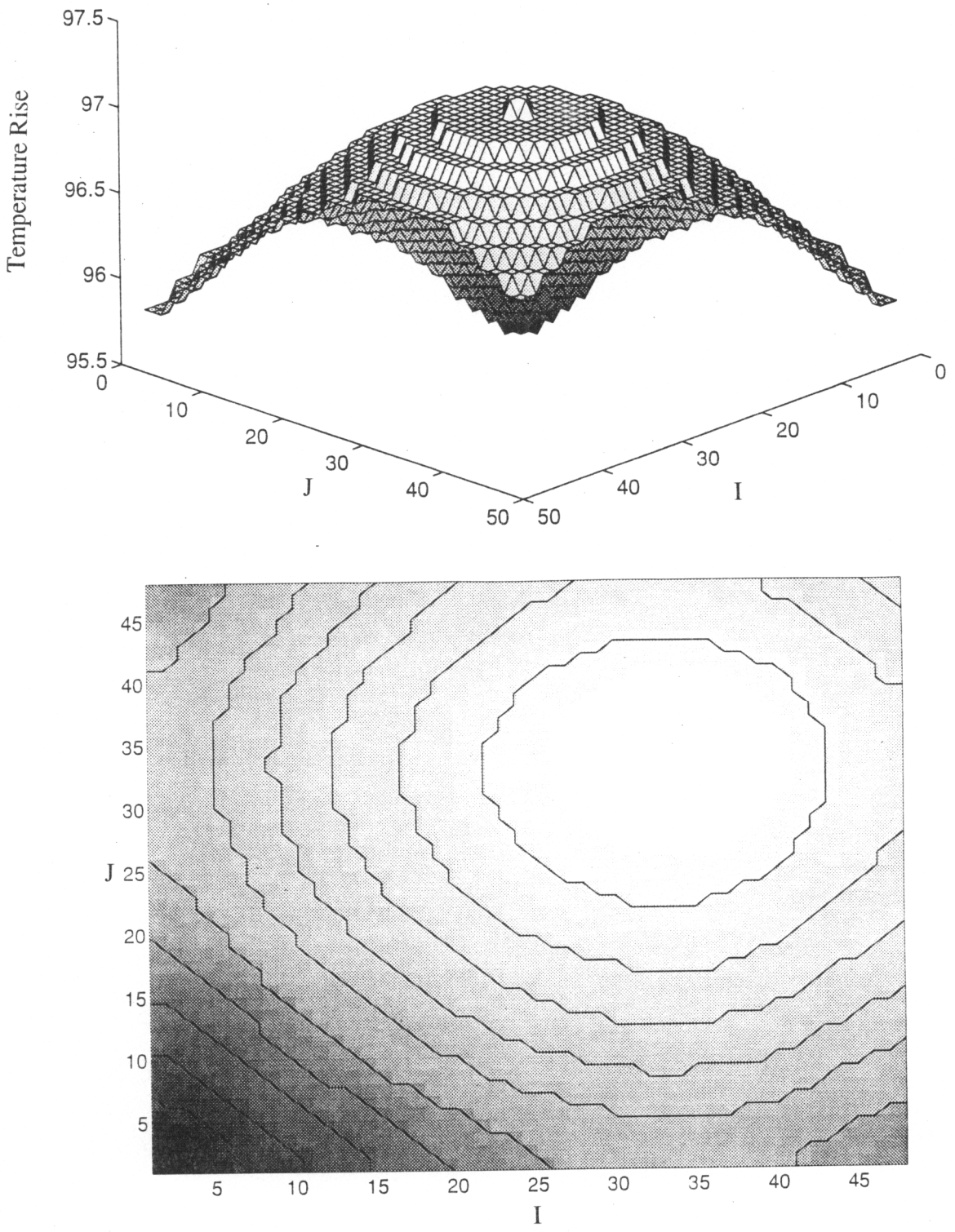


Figure 33: Temperature Rise on the Top Surface (K=5) with No Connections

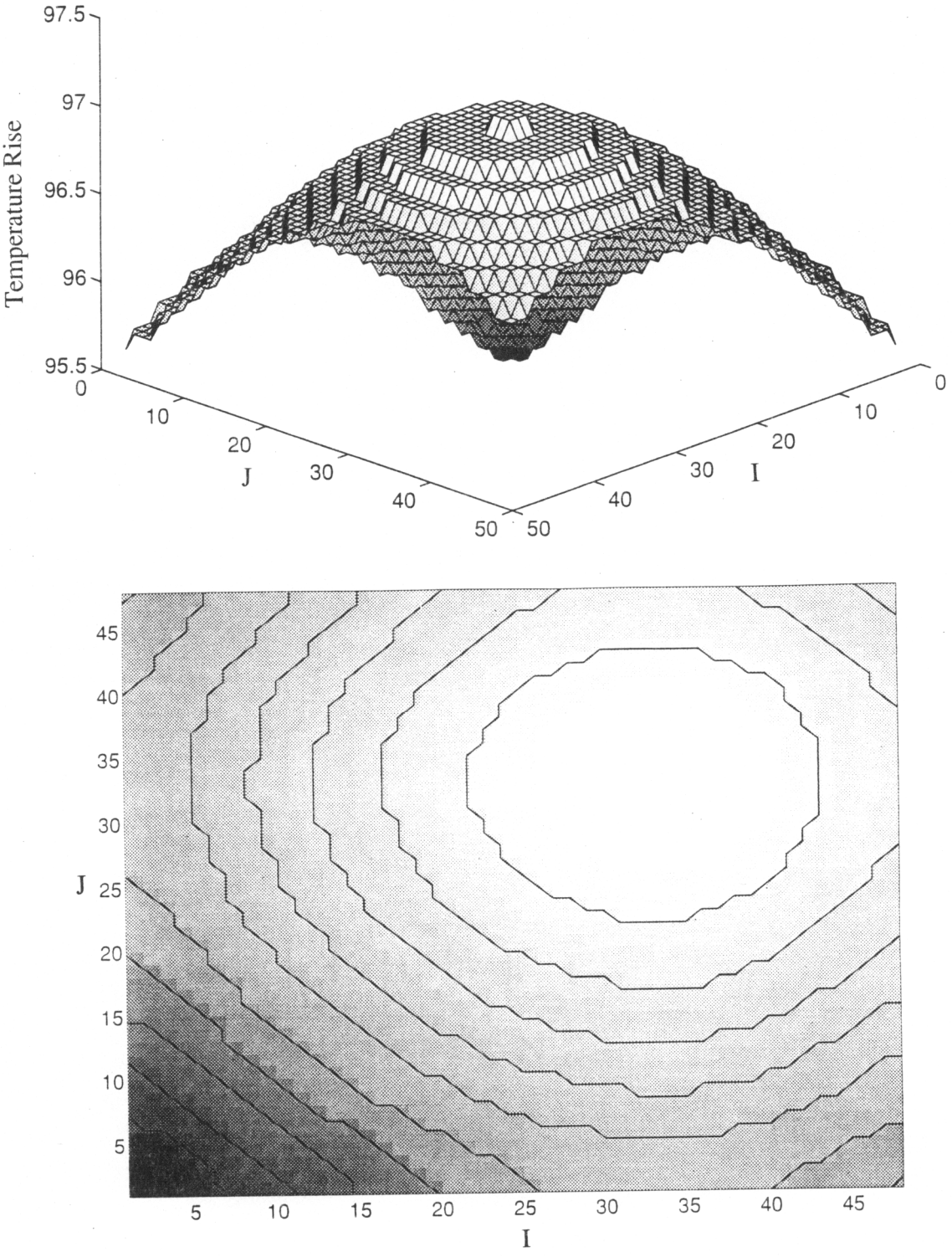


Figure 34: Temperature Rise in the Silicon Layer (K=4) with No Connections

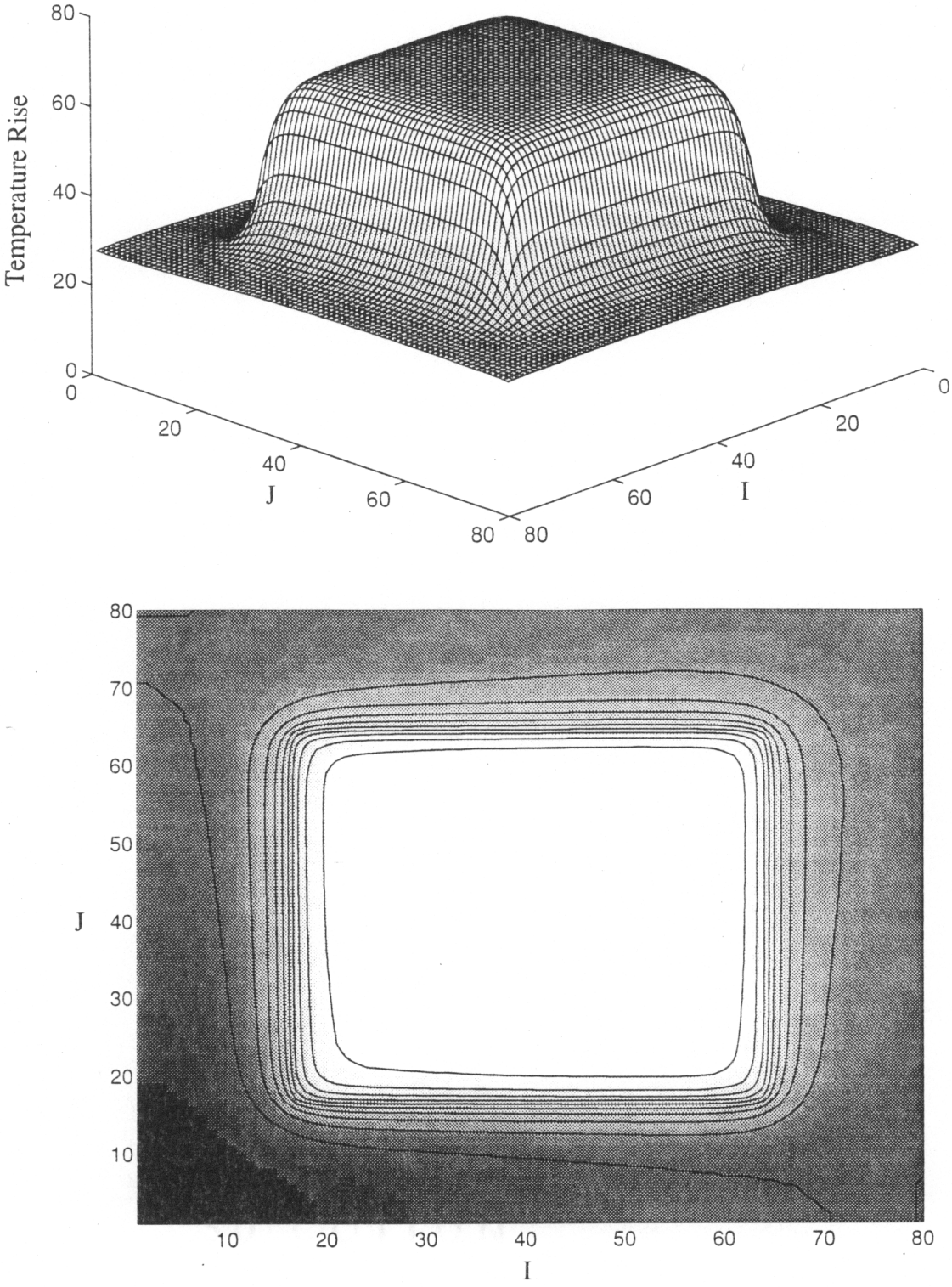


Figure 35: Temperature Rise in the Polyimide Layer (K=3) with No Connections

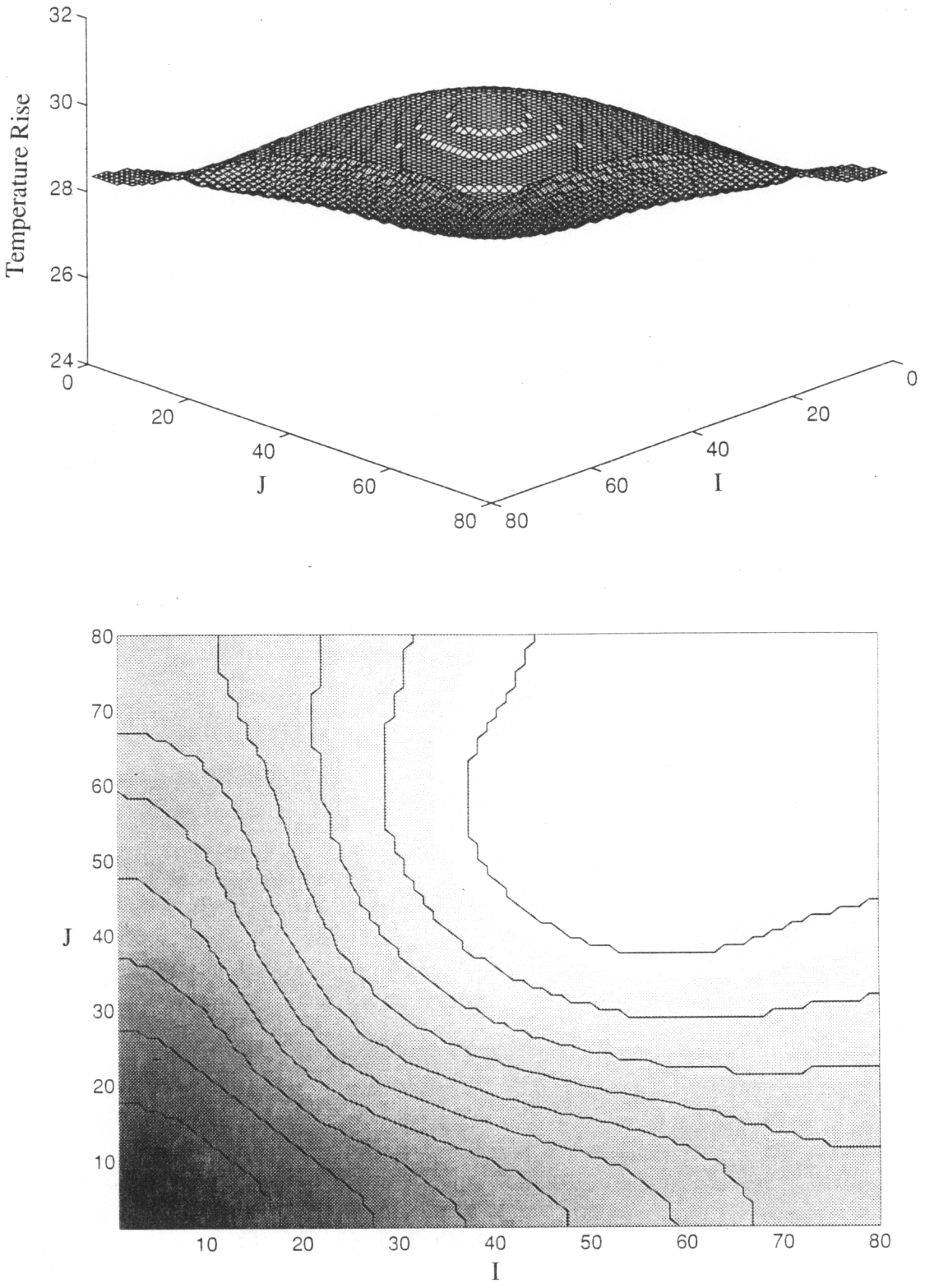


Figure 36: Temperature Rise in the Alumina Layer (K=2) with No Connections

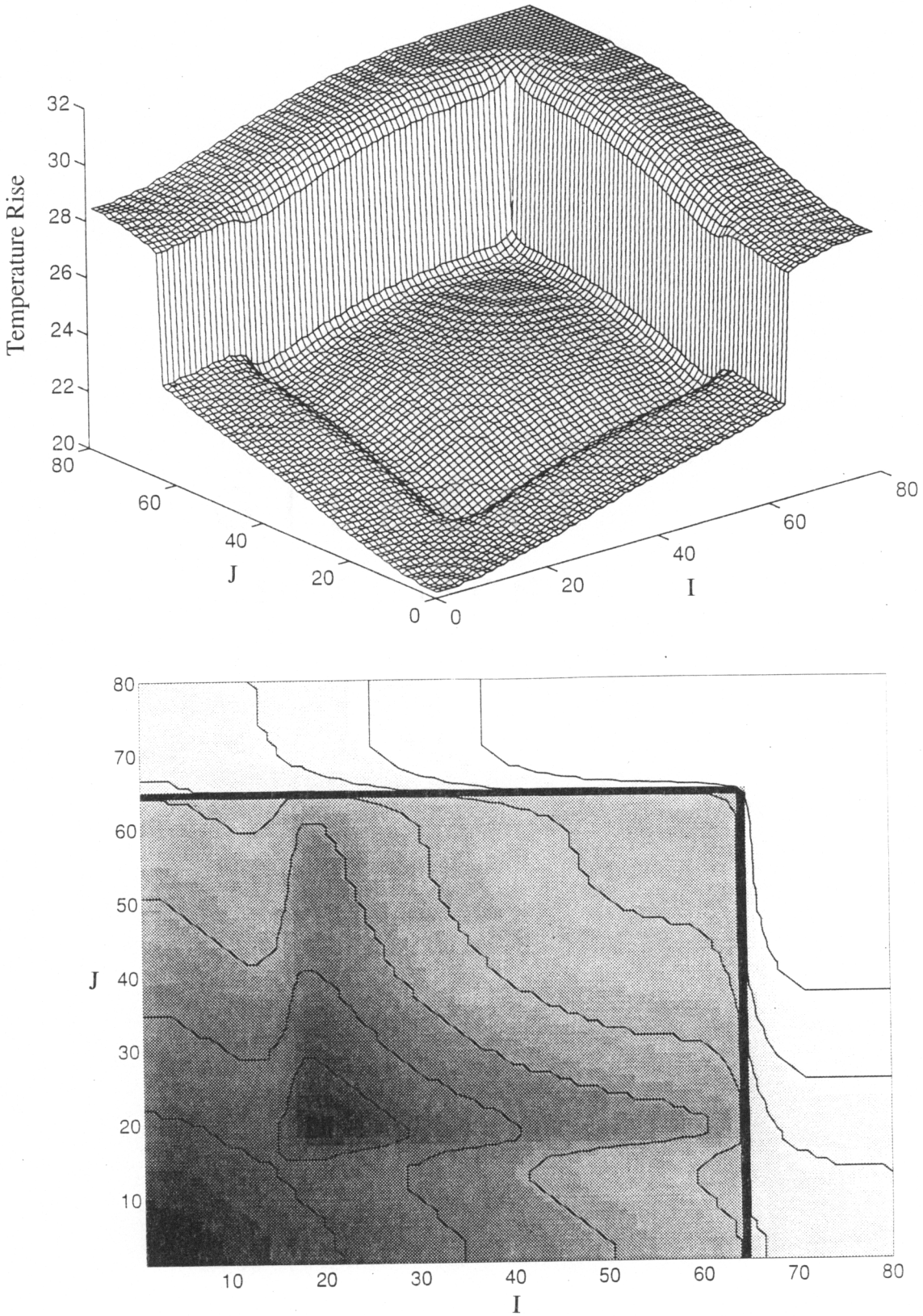
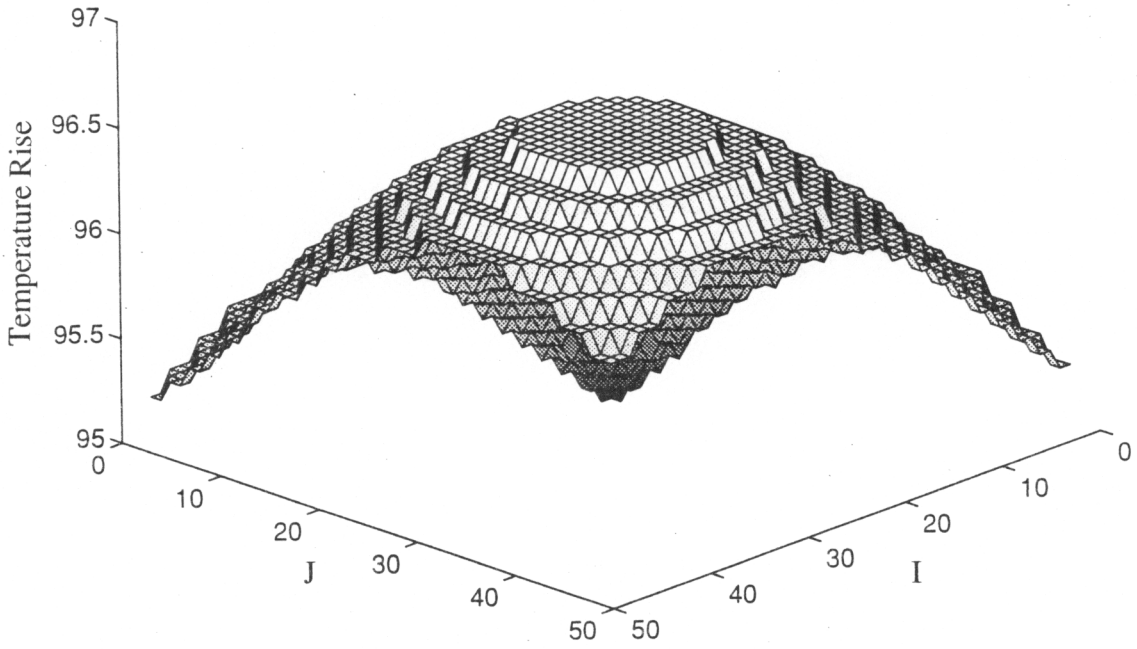


Figure 37: Temperature Rise on the Bottom Surface (K=1) with No Connections



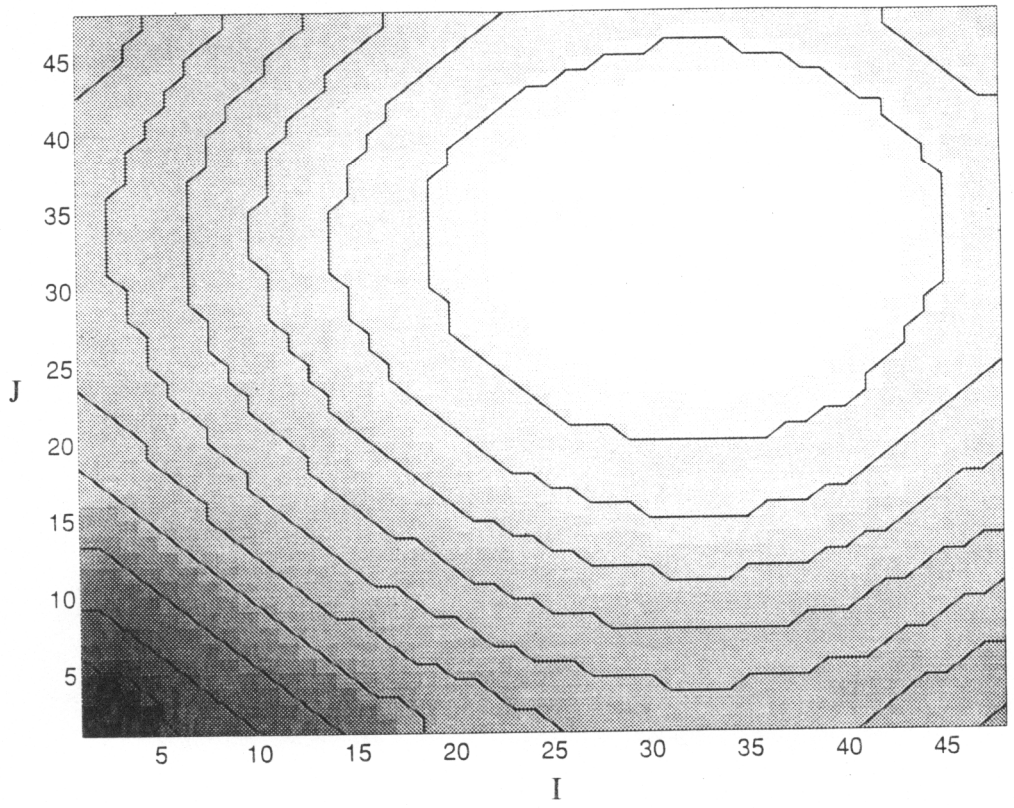


Figure 38: Temperature Rise on the Top Surface (K=5) with Edge Connections Only

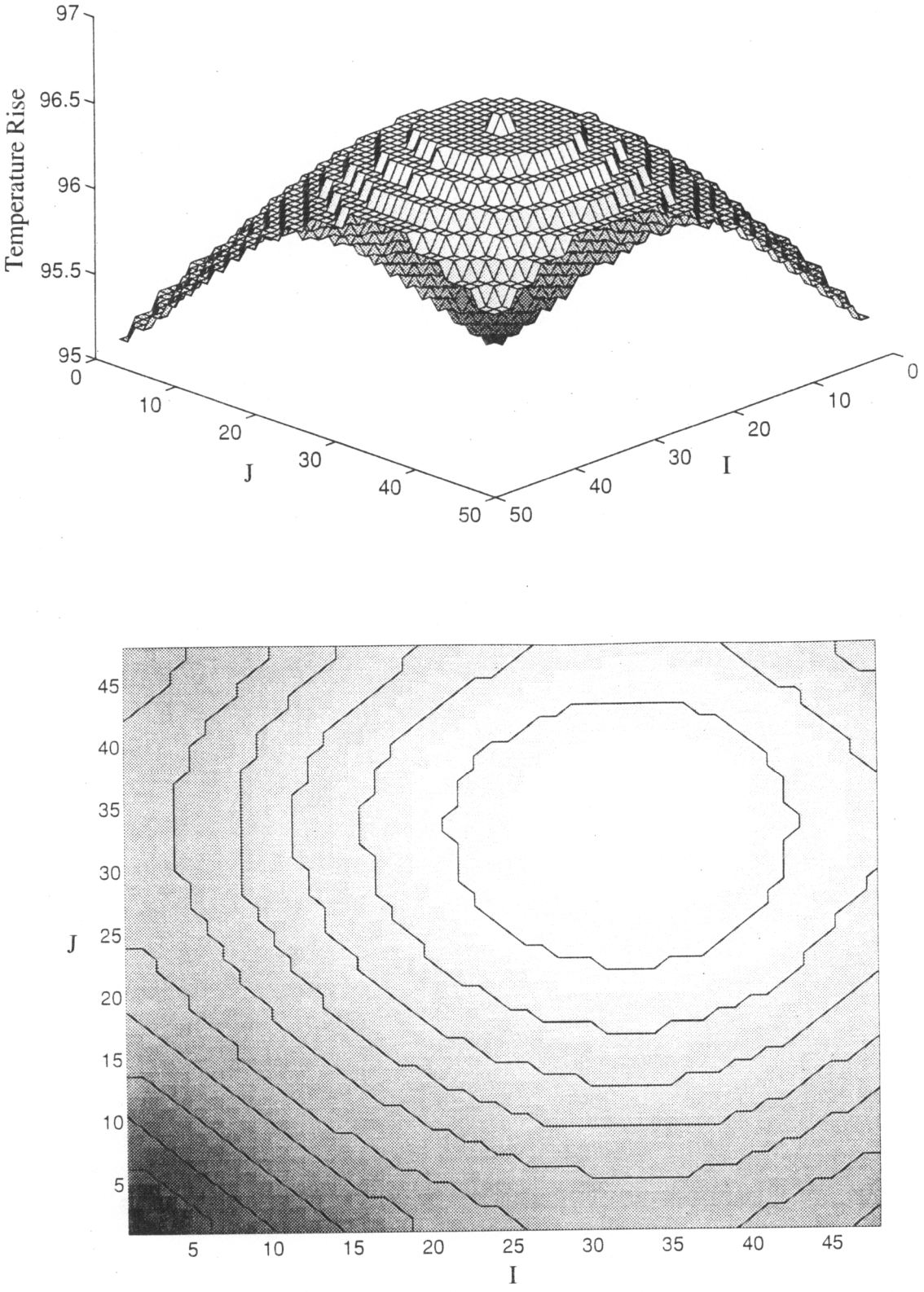


Figure 39: Temperature Rise in the Silicon Layer (K=4) with Edge Connections Only

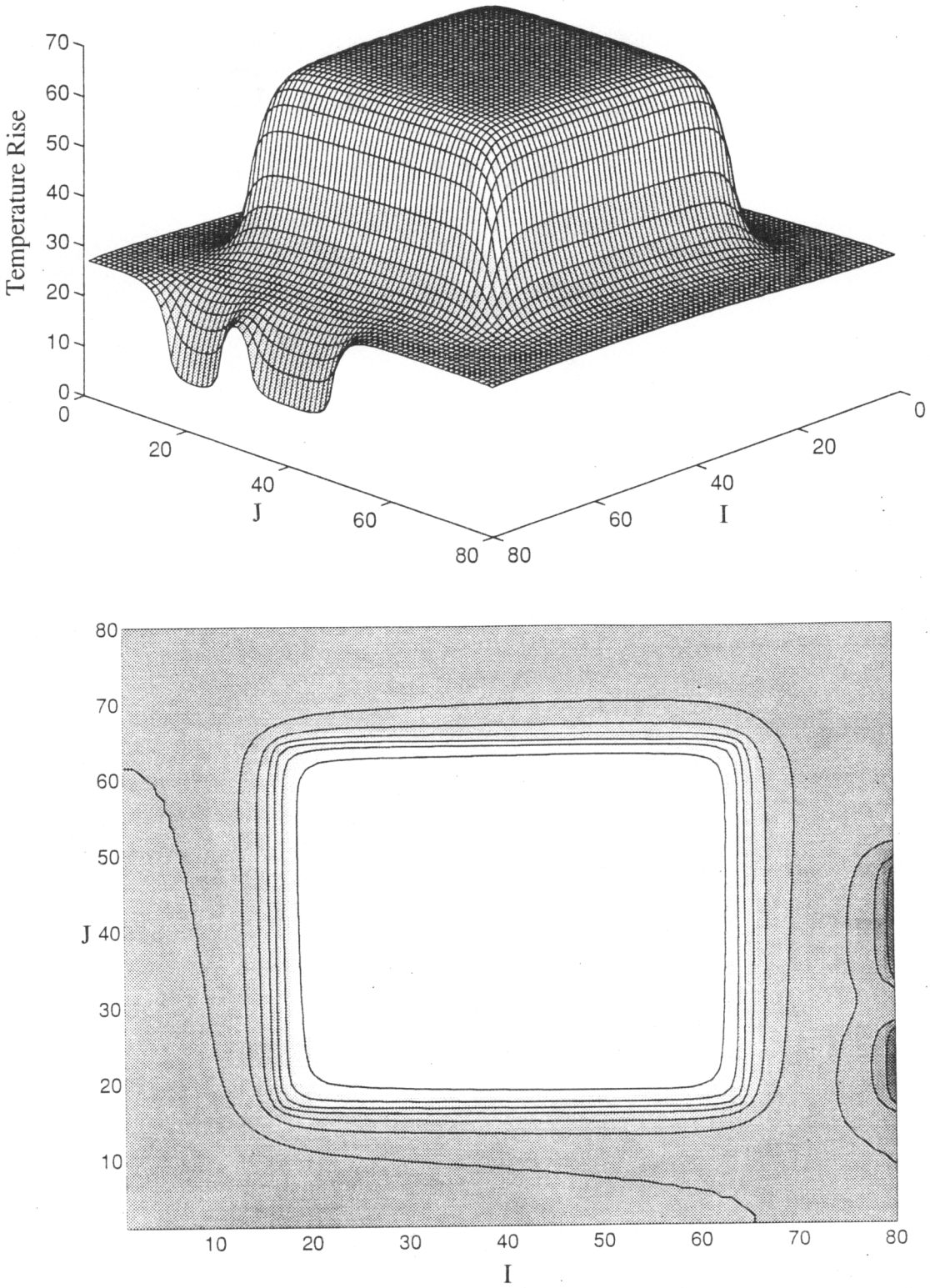


Figure 40: Temperature Rise in the Polyimide Layer (K=3) with Edge Connections Only

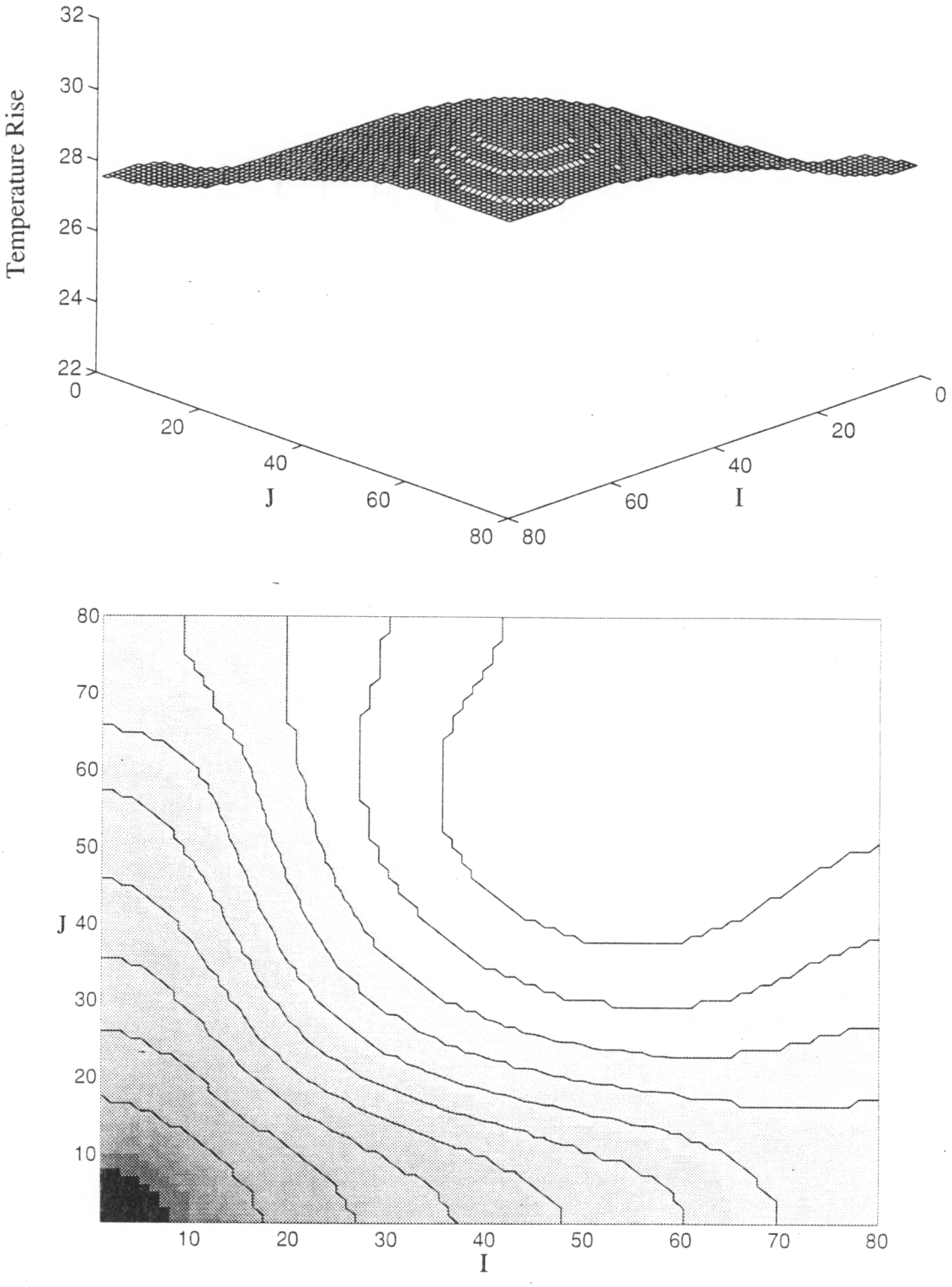


Figure 41: Temperature Rise in the Alumina Layer (K=2) with Edge Connections Only

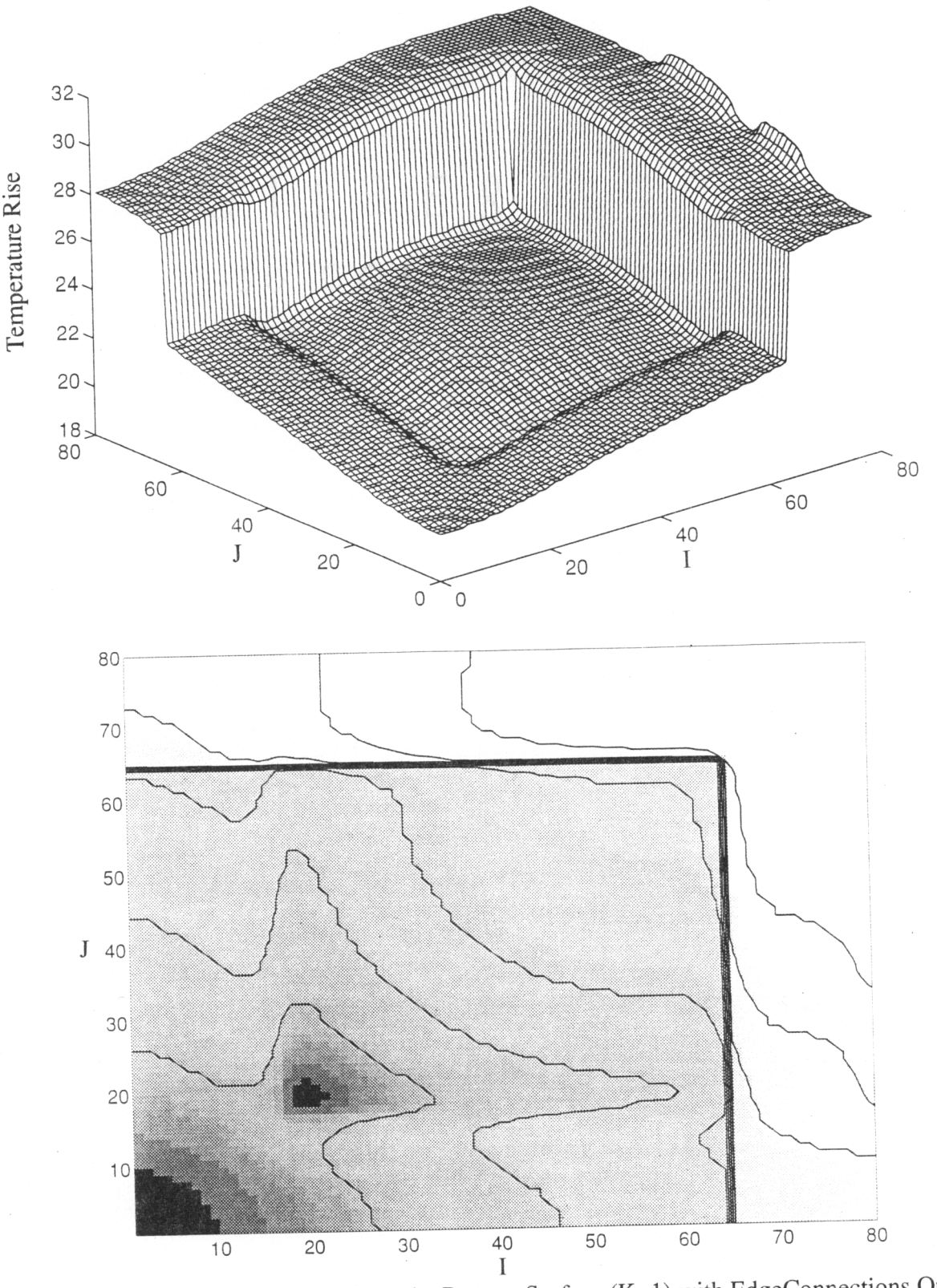


Figure 42: Temperature Rise on the Bottom Surface (K=1) with EdgeConnections Only

the heat sink. However, the absence of the pin connections creates a nearly constant temperature in the perimeter area.

Wire Bonds Only

Addition of the wire bonds to the model only slightly affect the solution (see Figs. 43-47). The plots differ little from those obtained using no connections. Except for the slight difference in temperature, the only other evidence of the presence of the wire bonds is the bumps seen in the K=3 plot.

Thermal Vias Only

The results obtained using the thermal vias only shows the marked effect that the vias have on the solution (see Figs. 48-52). Here, as in the plot of the complete model, layer 3 shows the sharp peaks caused by connection with the hotter layers above. Layers 4 and 5 show the downward peaks caused by the vias, but the location of the maximum temperature is close to that of the no connections case, rather than the centered location obtained using the complete model.

Pin Connections Only

The implementation of only the pin connections produces plots for the K=1 layer similar in appearance to those of the full model (see Figs. 53-57). The sharp discontinuity in layer 1 is present accompanied by the low temperatures at the sites of the connections. These connections are responsible for the largest temperature drop of any set of connections (see Table 2). The number and size of the pin connections are responsible for the large drop in temperature.

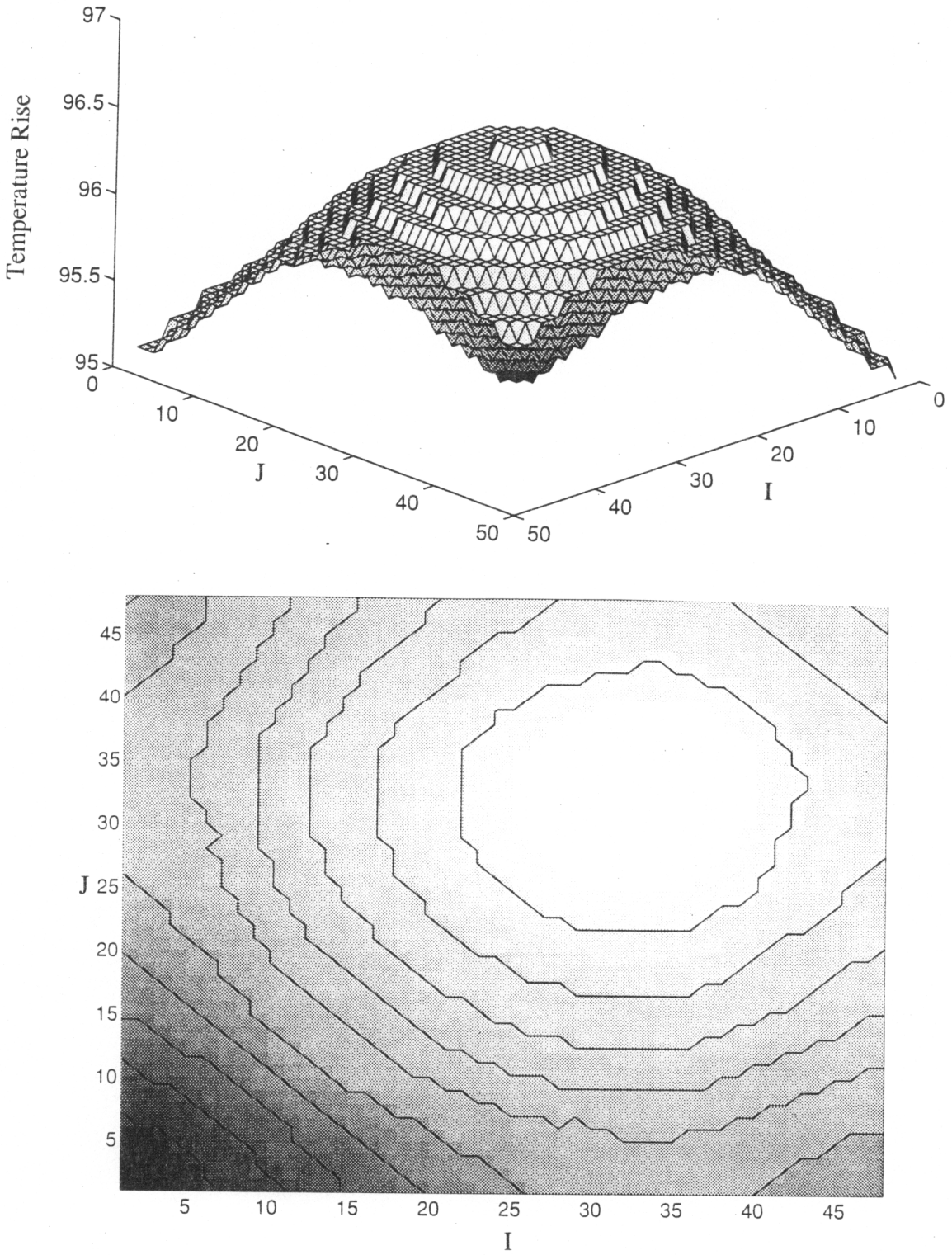


Figure 43: Temperature Rise on the Top Surface (K=5) with Wire Bonds Only

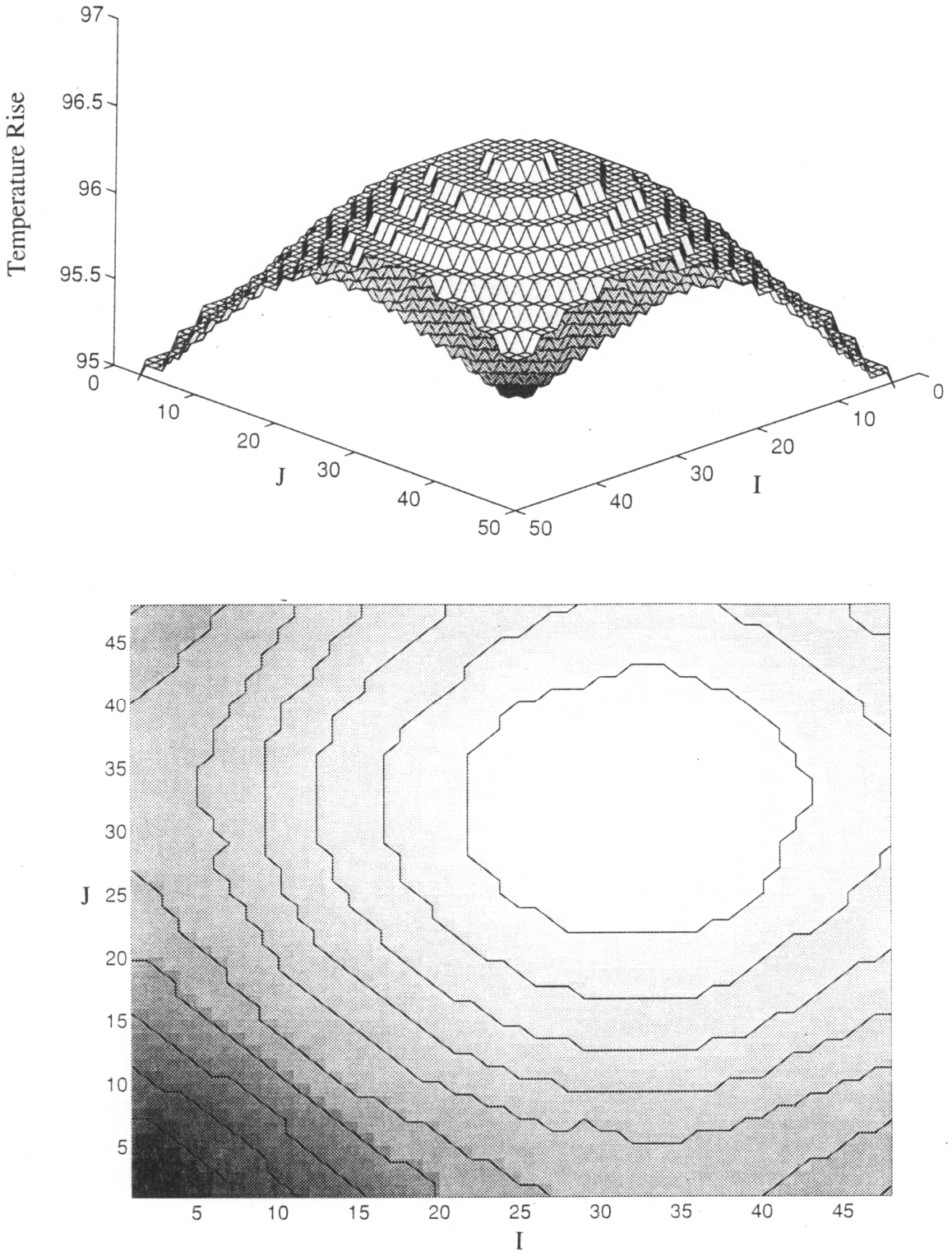


Figure 44: Temperature Rise in the Silicon Layer (K=4) with Wire Bonds Only

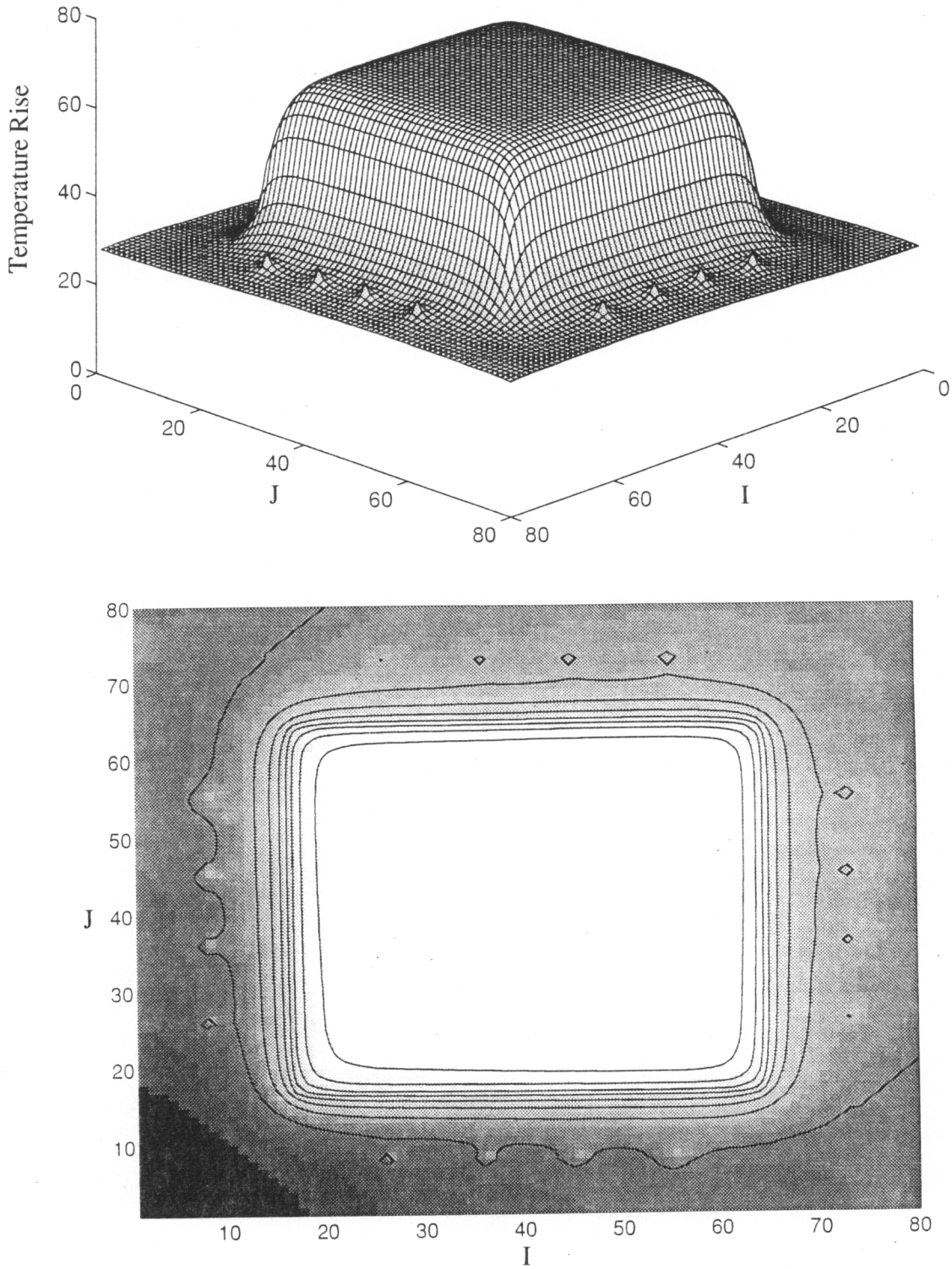


Figure 45: Temperature Rise in the Polyimide Layer (K=3) with Wire Bonds Only

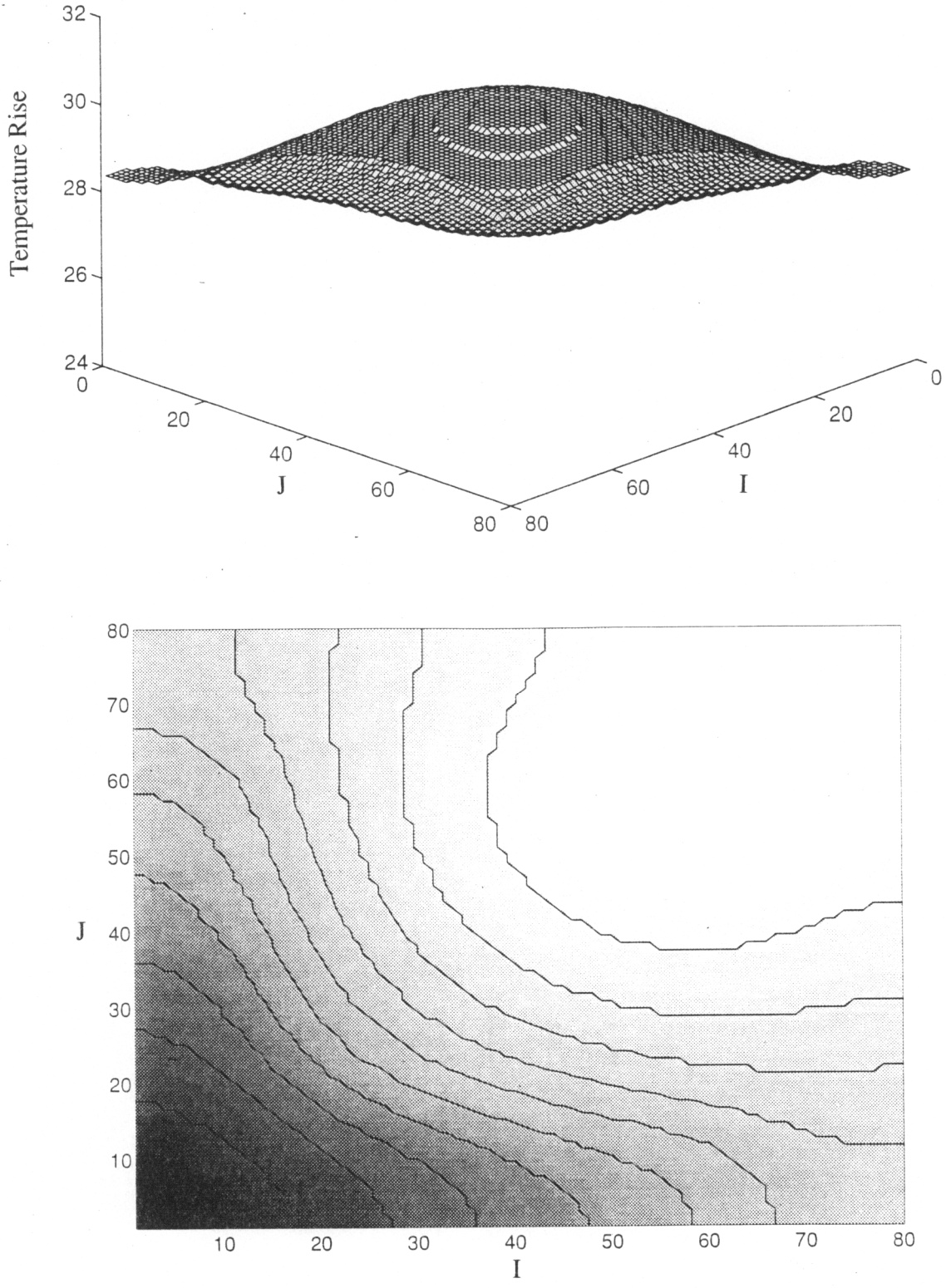


Figure 46: Temperature Rise in the Alumina Layer (K=2) with Wire Bonds Only

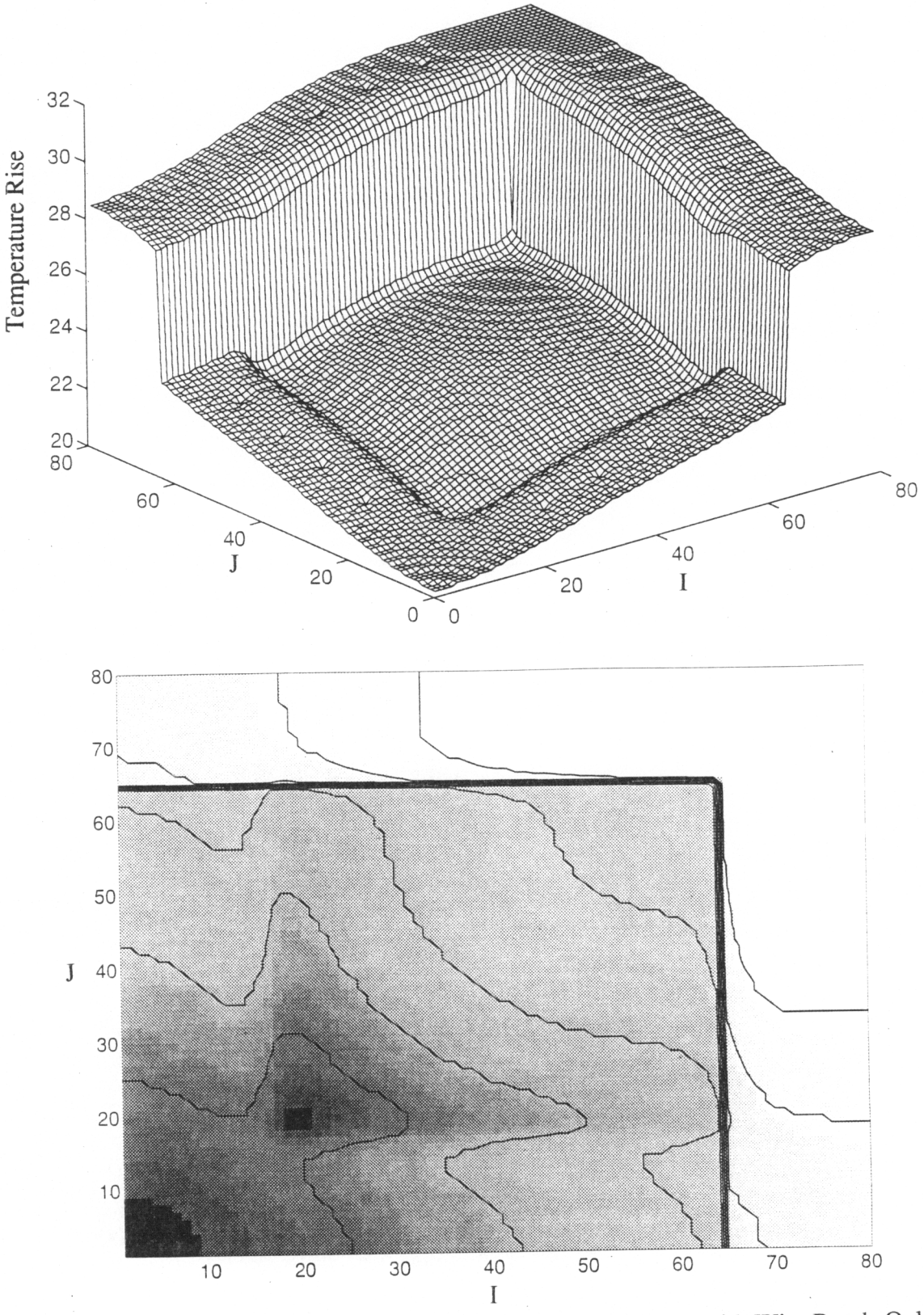
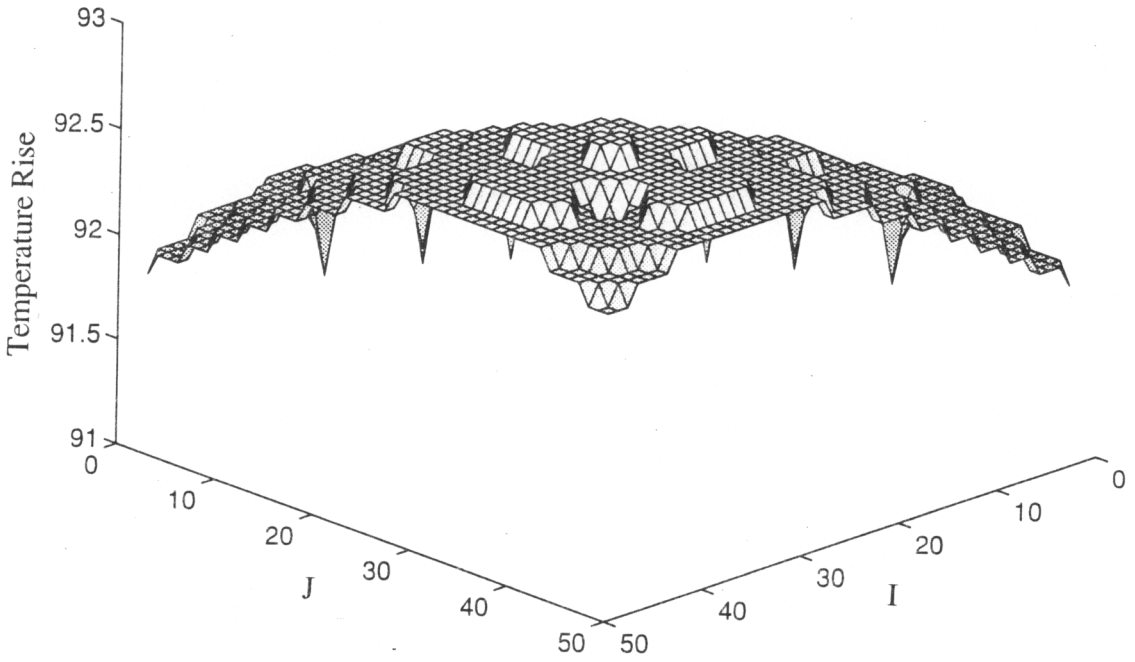


Figure 47: Temperature Rise on the Bottom Surface (K=1) with Wire Bonds Only



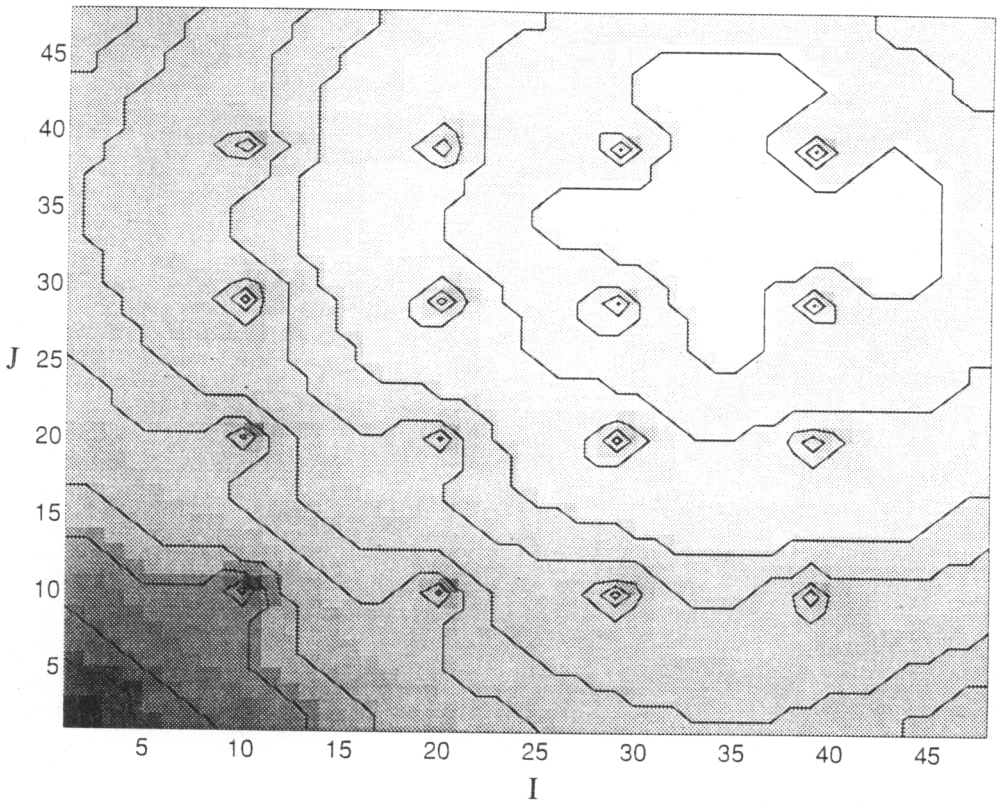


Figure 48: Temperature Rise on the Top Surface (K=5) with Thermal Vias Only

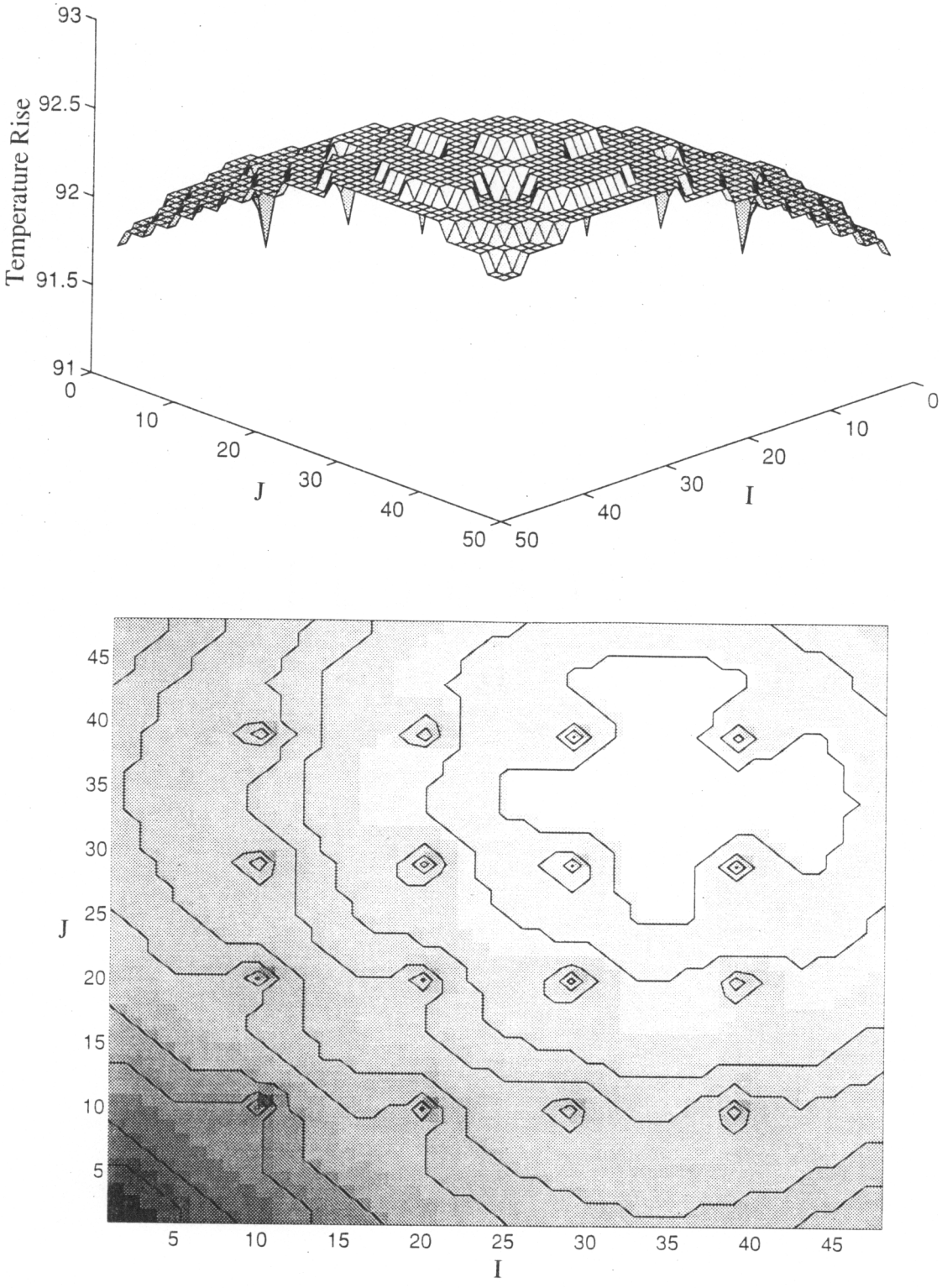


Figure 49: Temperature Rise in the Silicon Layer (K=4) with Thermal Vias Only

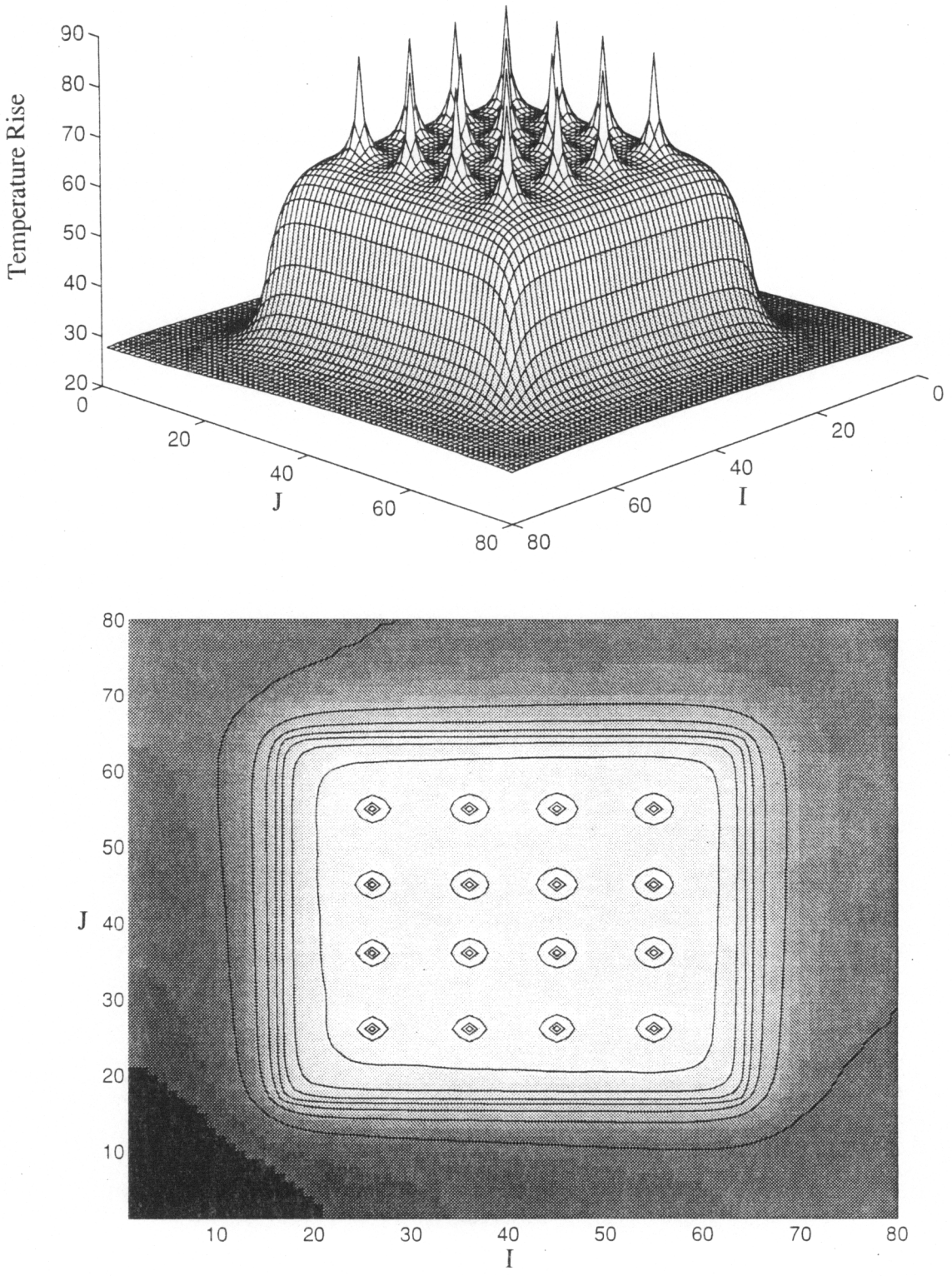


Figure 50: Temperature Rise in the Polyimide Layer (K=3) with Thermal Vias Only

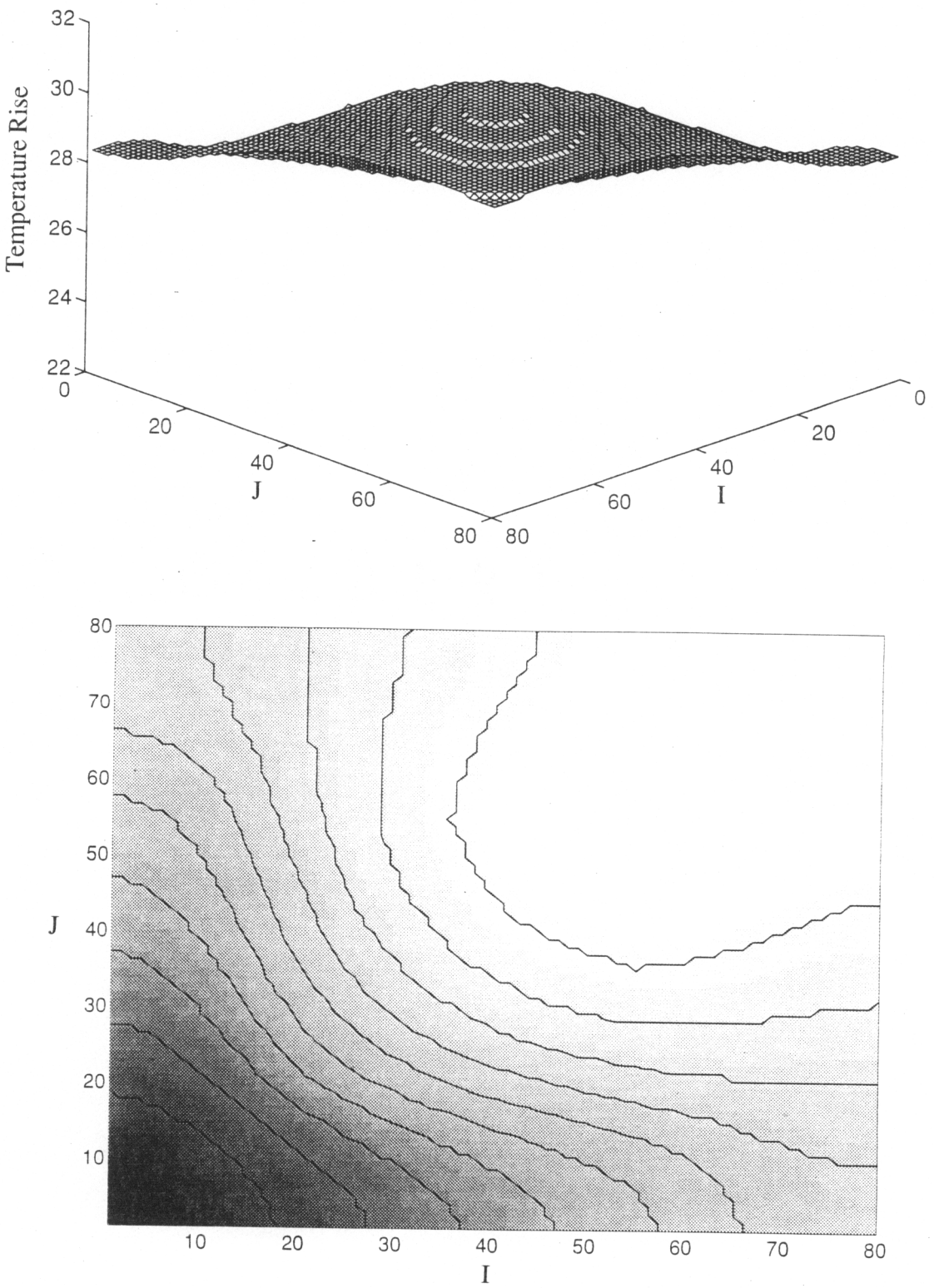


Figure 51: Temperature Rise in the Alumina Layer (K=2) with Thermal Vias Only

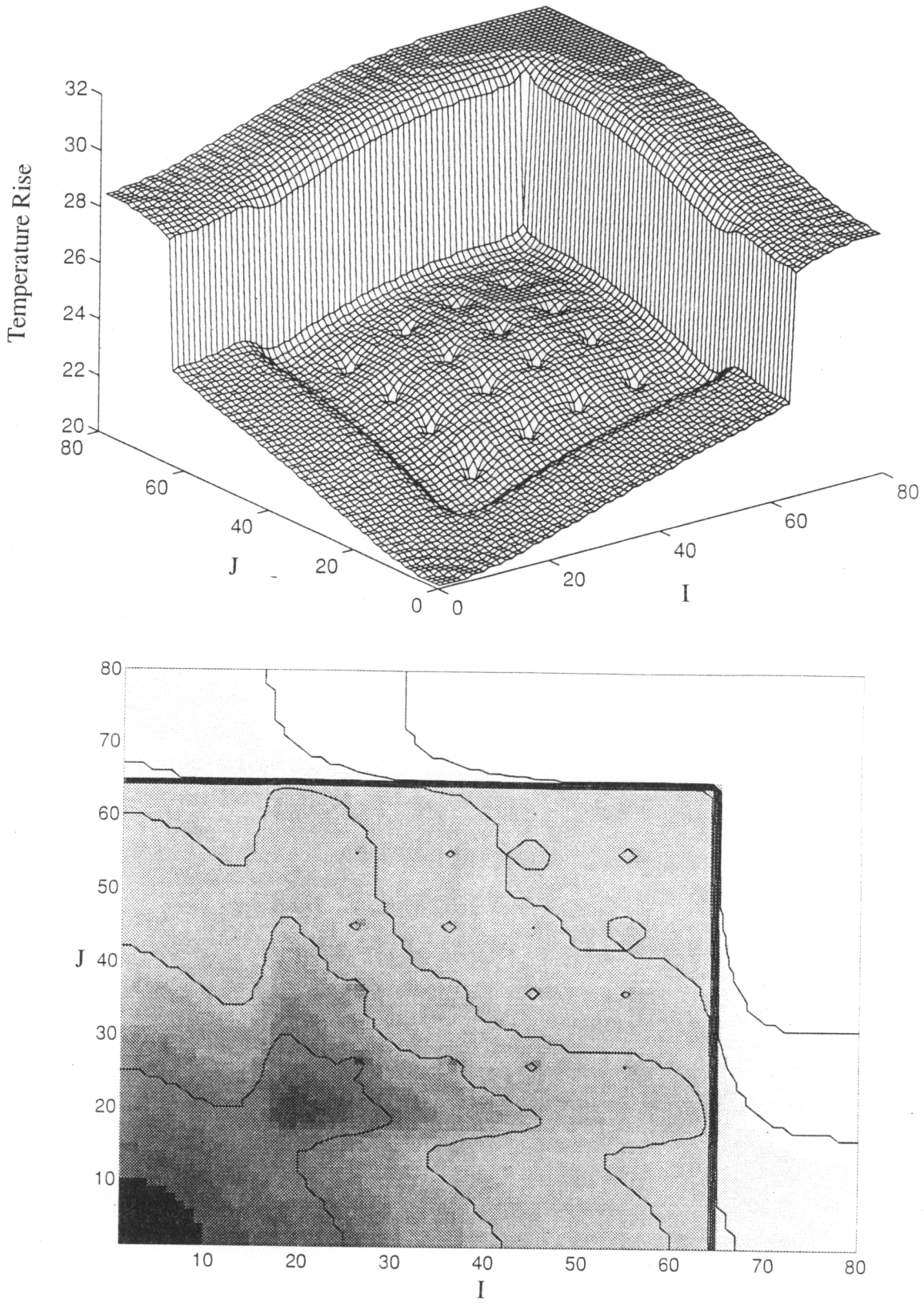


Figure 52: Temperature Rise on the Bottom Surface (K=1) with Thermal Vias Only

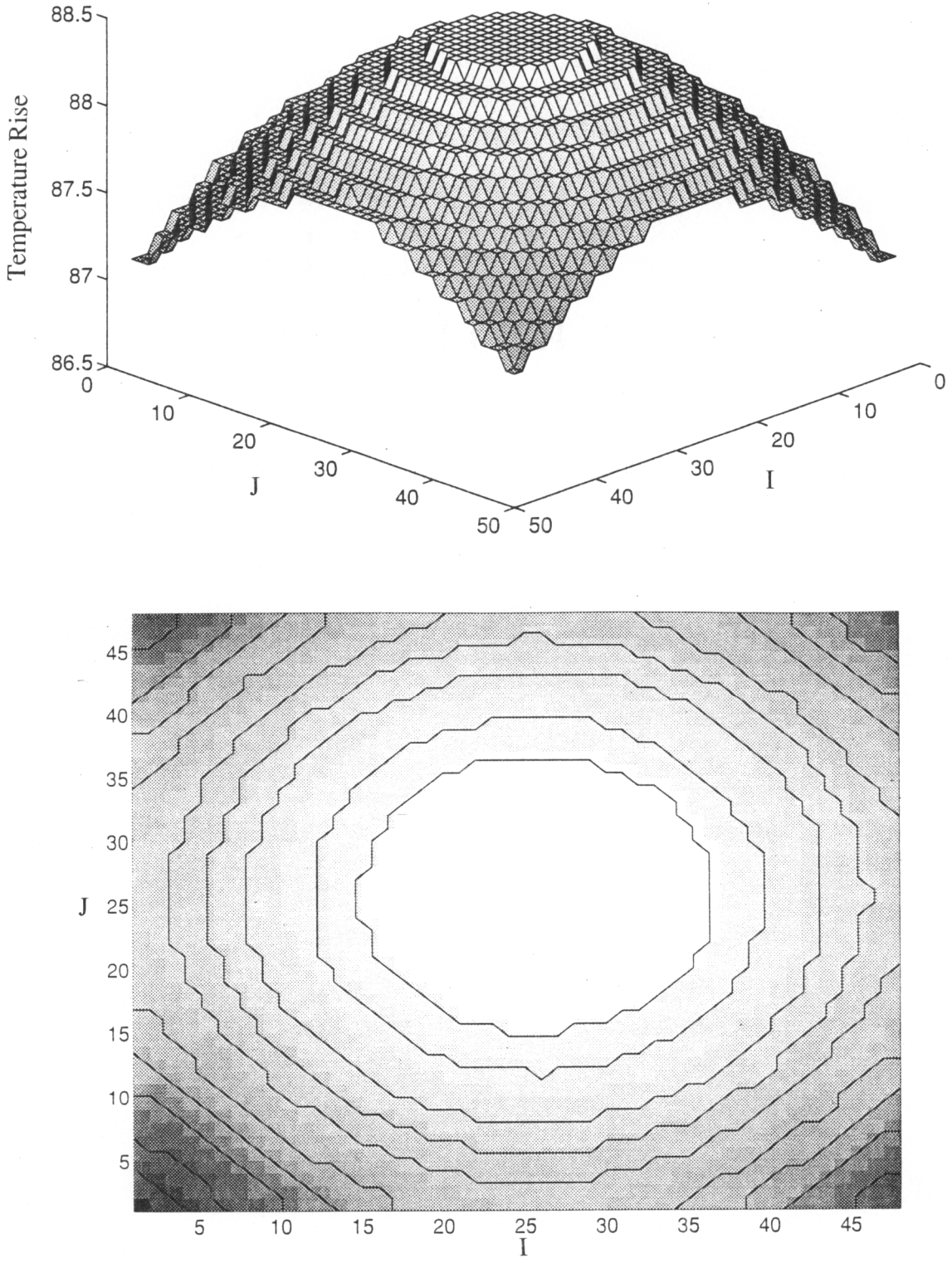


Figure 53: Temperature Rise on the Top Surface (K=5) with Pin Connections Only

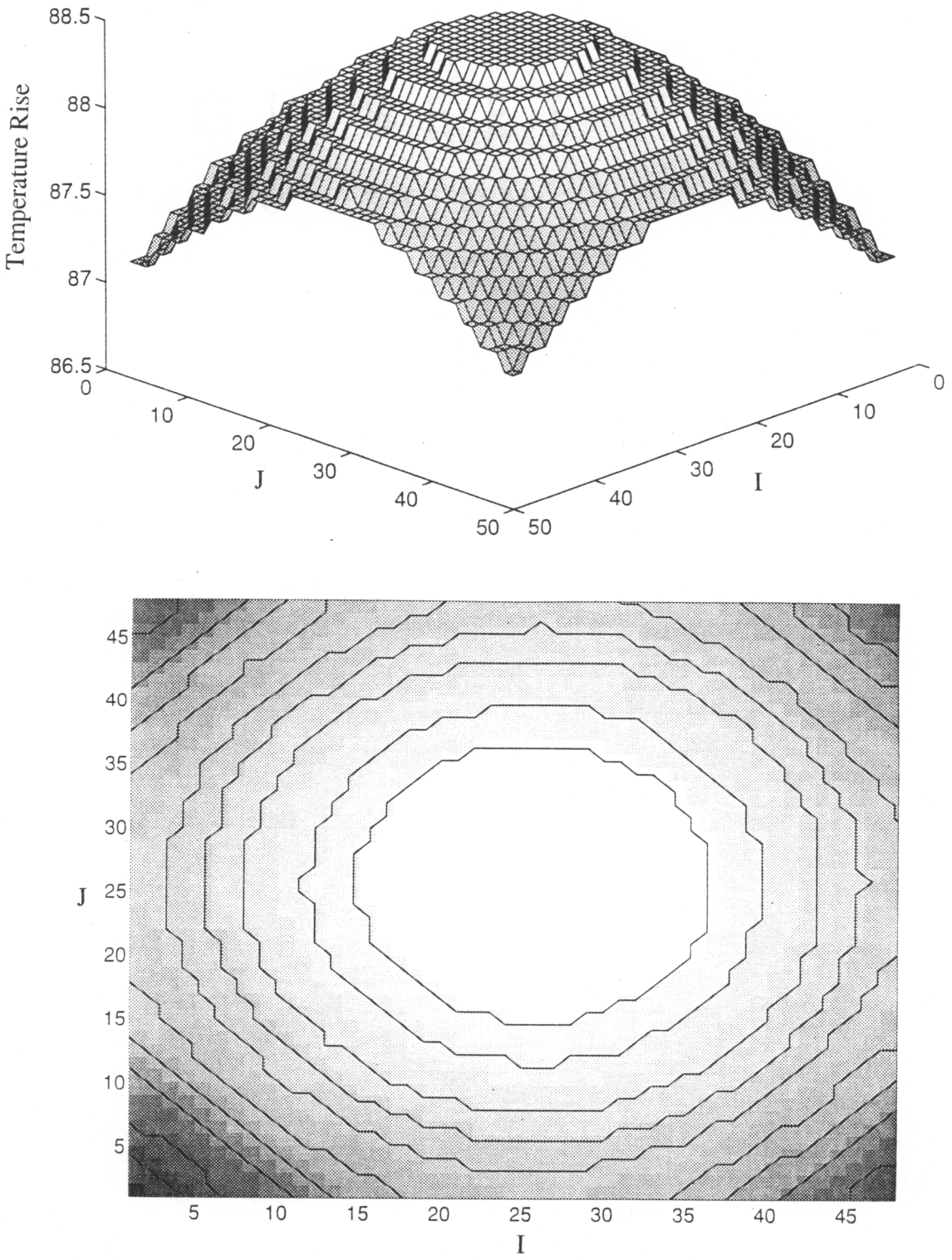


Figure 54: Temperature Rise in the Silicon Layer (K=4) with Pin Connections Only

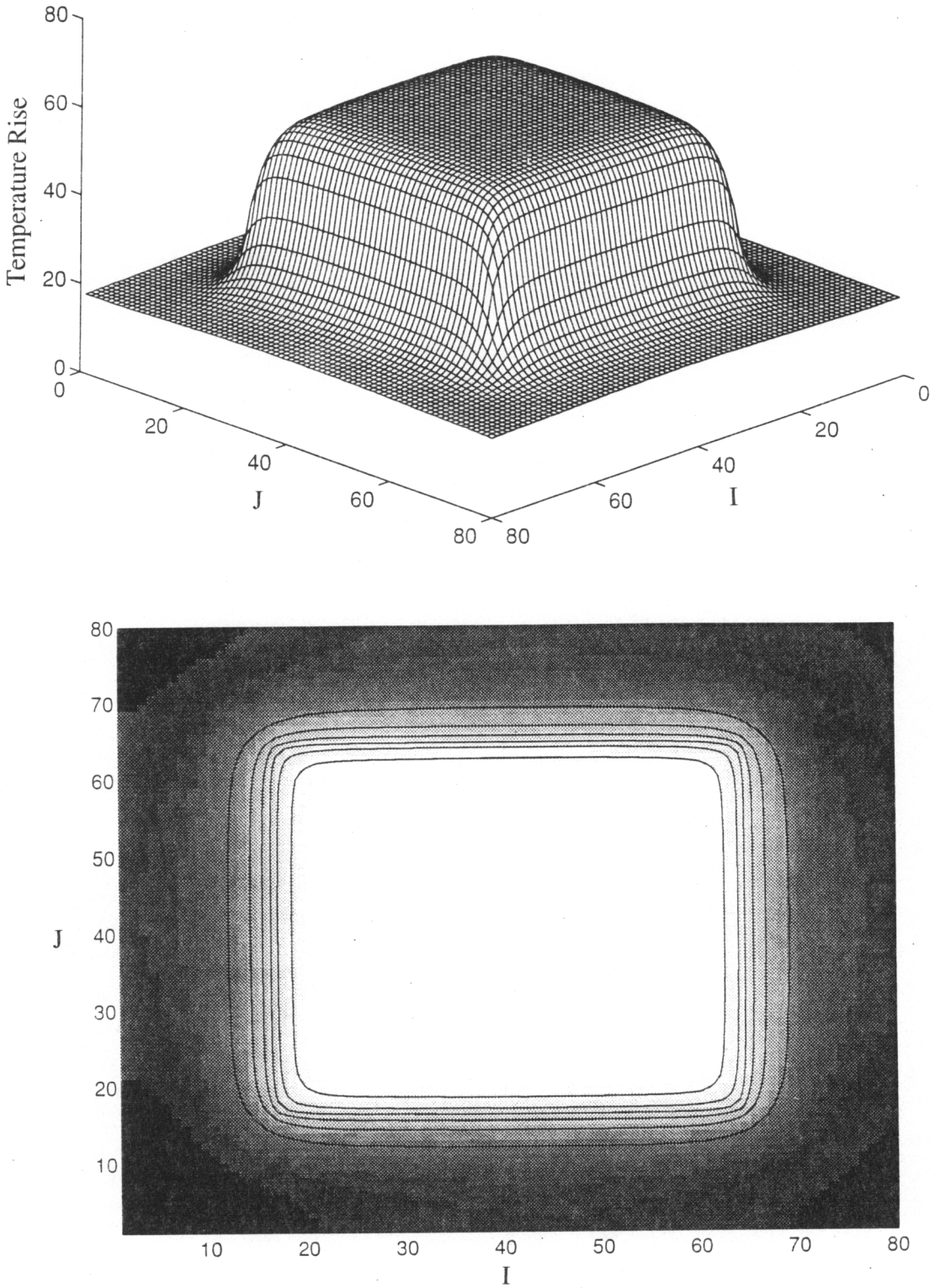


Figure 55: Temperature Rise in the Polyimide Layer (K=3) with Pin Connections Only

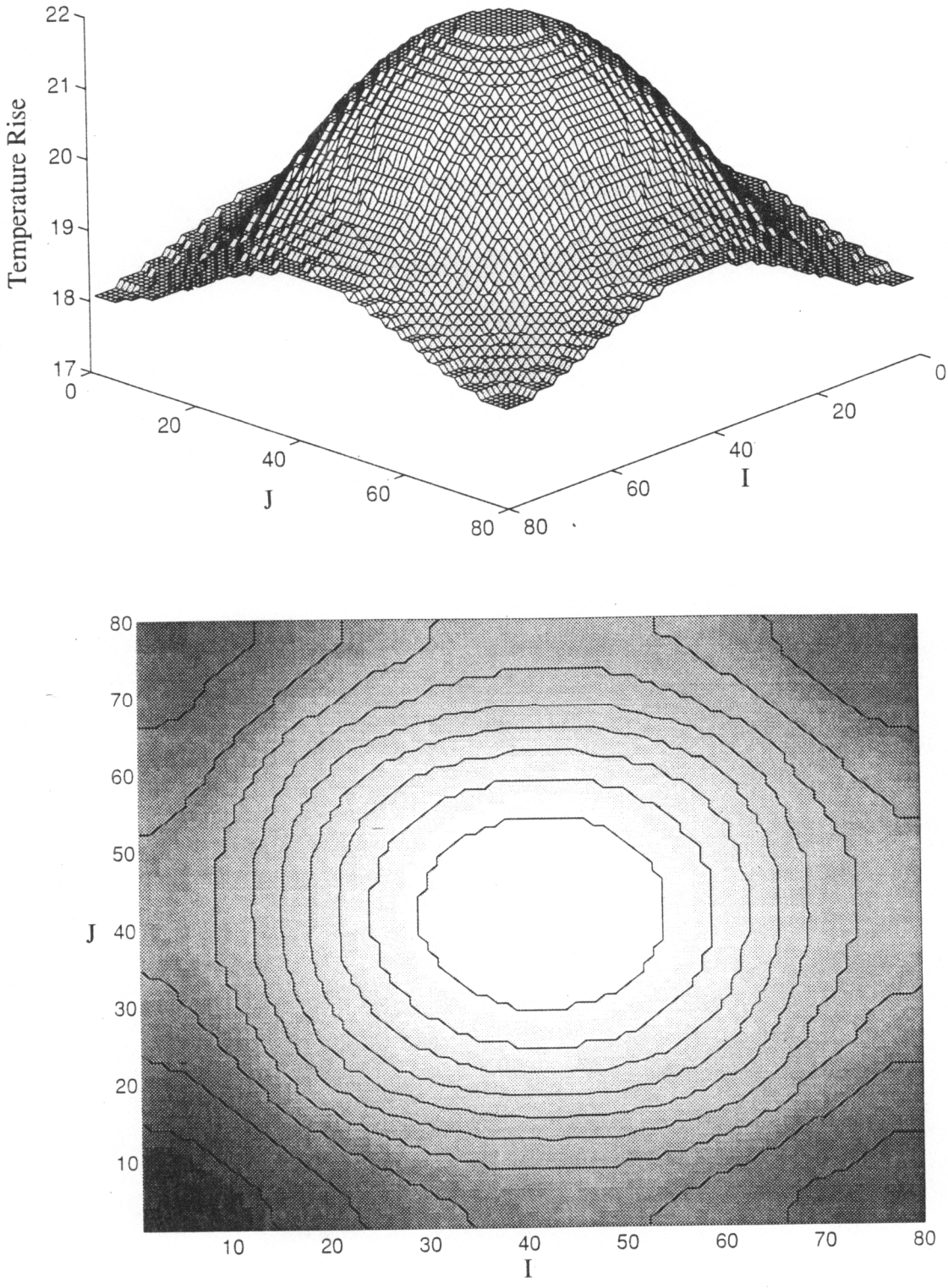


Figure 56: Temperature Rise in the Alumina Layer (K=2) with Pin Connections Only

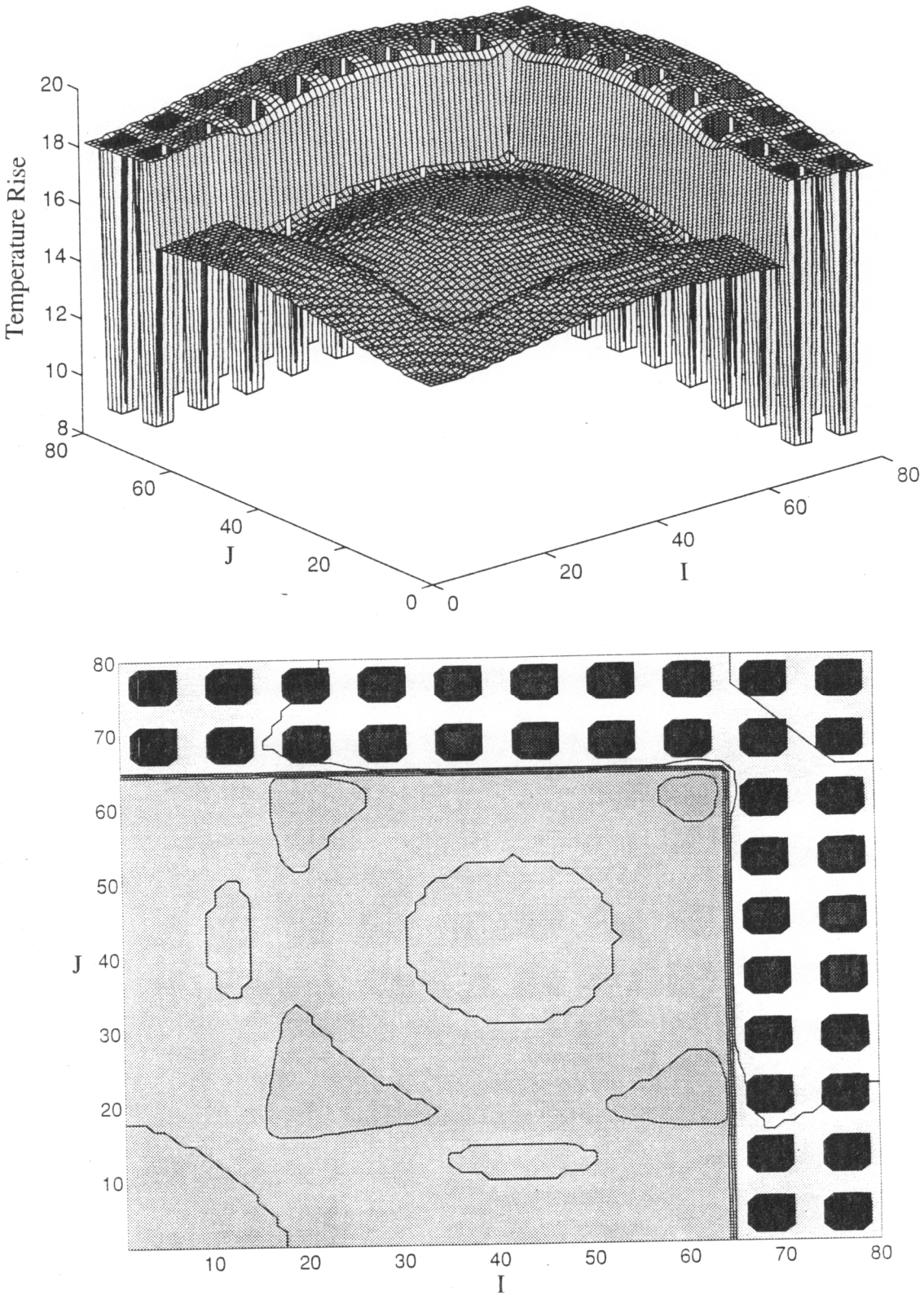


Figure 57: Temperature Rise on the Bottom Surface (K=1) with Pin Connections Only

6. Simplification of the Numerical Model

6.1 Objective

The proper design of microelectronic packages achieves a balance between performance, reliability, and cost. One aspect of the overall design is thermal management. To be effective, a thermal analysis must rapidly consider many design alternatives, including changes in geometry, materials, and environment. These changes may be driven by thermal management considerations, or by other design criteria and constraints. Since thermal analysis can be a costly process, in terms of both time and money, the computational effort required should be minimized. One way to accomplish this is to selectively eliminate parts of the model. Of course, changes of this sort must be accompanied by other measures to compensate for the missing elements.

In analysis of multilayer microelectronics, it would often be helpful to neglect a thin layer for computational purposes. However, the resulting analysis may be in error if the layer is crucial to the behavior of the model. A basic model is studied to find the conditions under which the model can be simplified without unreasonable error. A simple relationship is sought which can indicate criteria when the thermal model can be simplified.

6.2 Physical Model

The physical system consists of a multilayer microelectronic device, specifically one of three layers, two with equal thickness and conductivity and the third with variable

attributes. The model is examined with the third layer placed in the top, middle and bottom positions. This simplified multilayer substrate is mounted on a heat sink which is modeled as an effective external resistance. The substrate can also be cooled convectively from the top surface and exposed edges (see Fig. 58). Two heat sources are attached to the top surface symmetrically (see Fig. 59). It is assumed that the total heat input of the devices goes into the substrate, meaning that convection does not occur over the area on which the heat sources are mounted. Using symmetry, the model area can be reduced to one-fourth the physical size. Non-dimensional parameters are used, based on the thickness and conductivity of the two constant layers. The model is run with the variable layer in the middle, on the top, and on the bottom (see Figs. 60-62). In each of the cases, the relative conductivity is varied from 0.1 to 100 by multiples of 10. The relative thickness is also varied from 0.01 to 1.0. The convective environments are varied on both the top and the bottom, from $Bi_{top} = 1.0$, $Bi_{bot} = 1.0$ to $Bi_{top} = 0.0$,$Bi_{bot} = 0.1$.

These results are compared to those from a simplified model with the variable layer removed. This is accompanied by the addition of approximations designed to compensate for the absence of the layer.

6.3 Thermal Model

The steady-state solution of the model is accomplished using a modified version of the code. The user still defines the thickness and conductivity of the layer to be removed (the conductivity is considered to be isotropic). The code then modifies the numerical model in the manner detailed below.

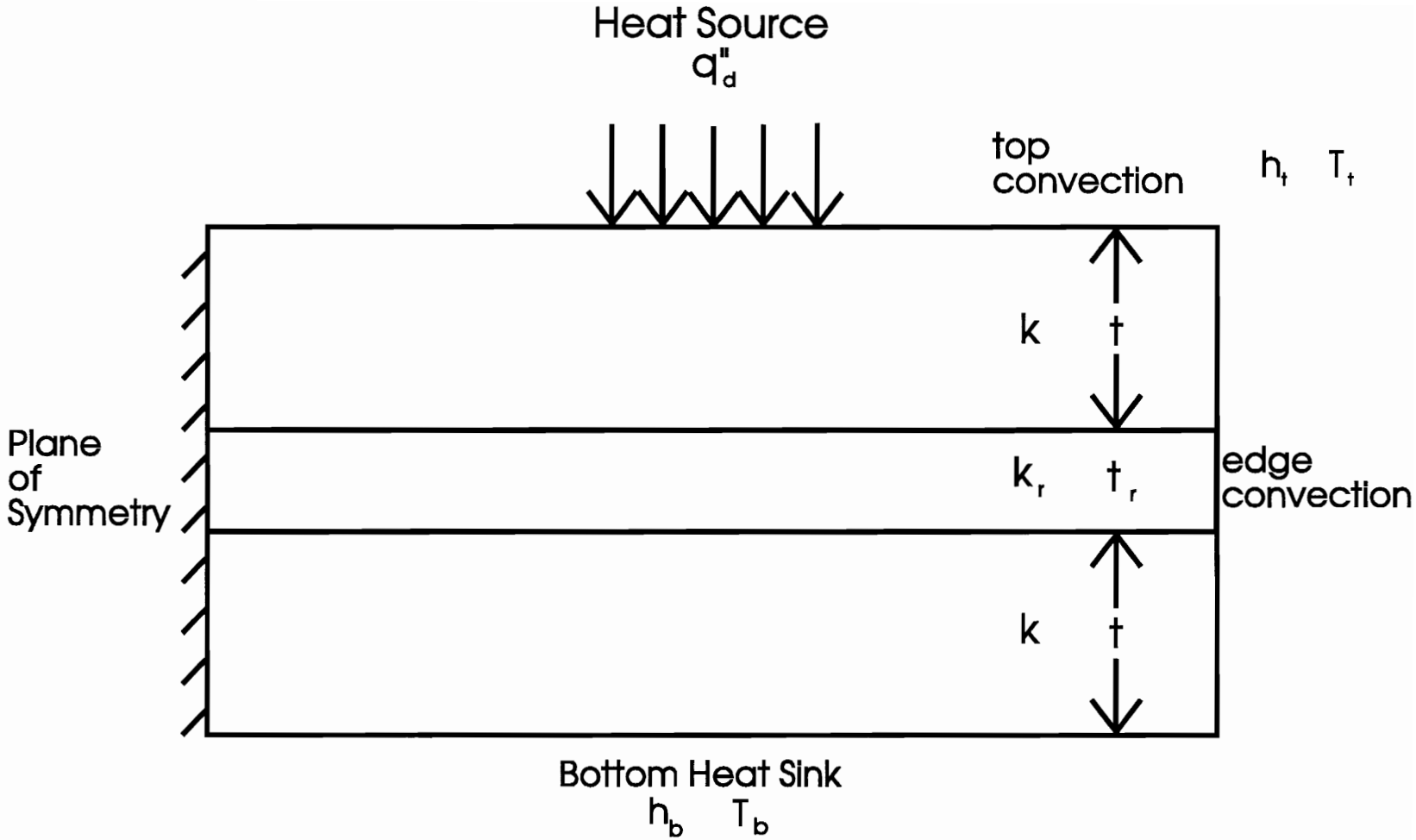


Figure 58: Side View of Model with Convective Cooling on the Top and Heat Sink on the Bottom

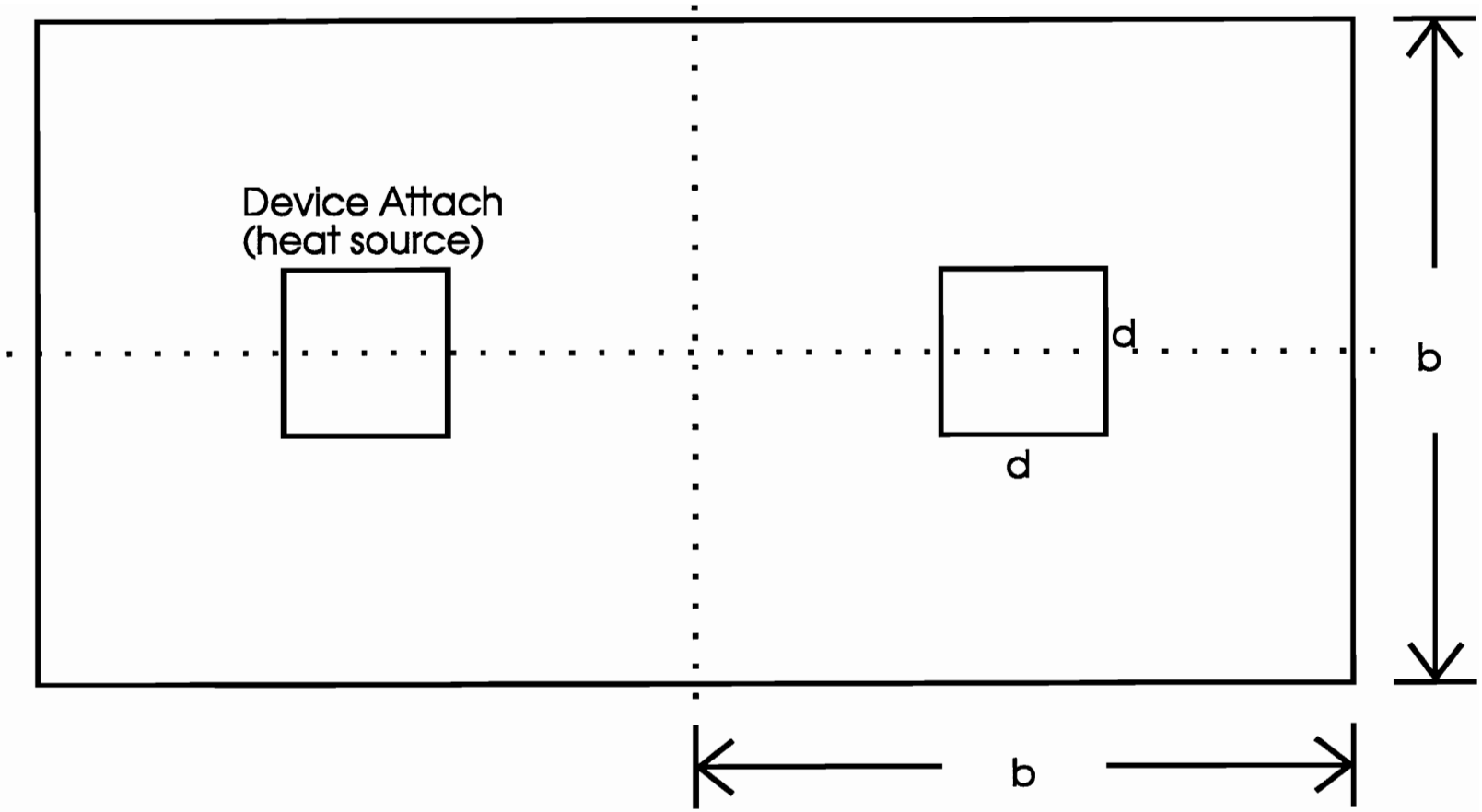


Figure 59: Top View of Model with Symmetrically Placed Heat Sources

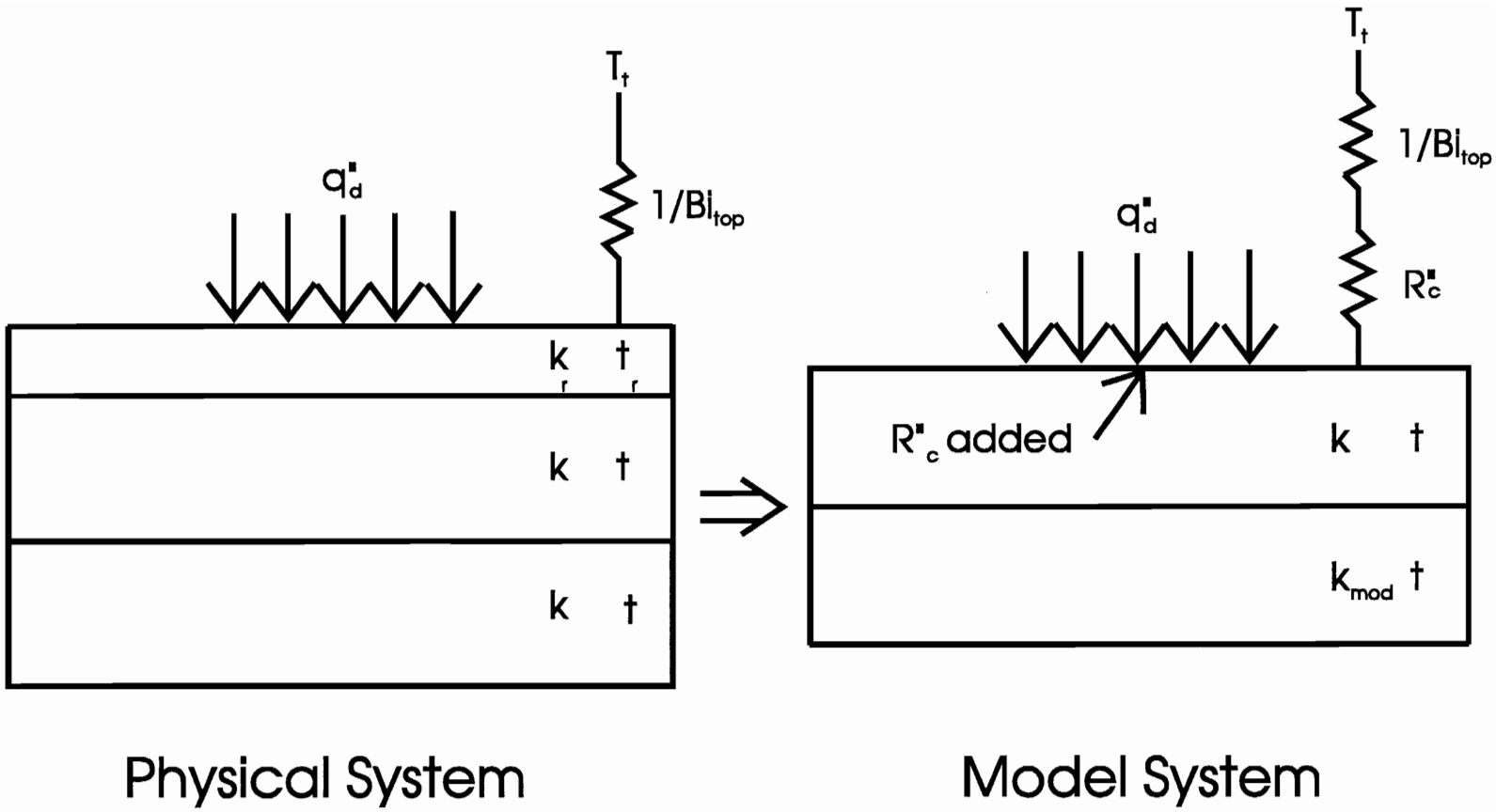


Figure 60: Approximations for Removal of Top Layer

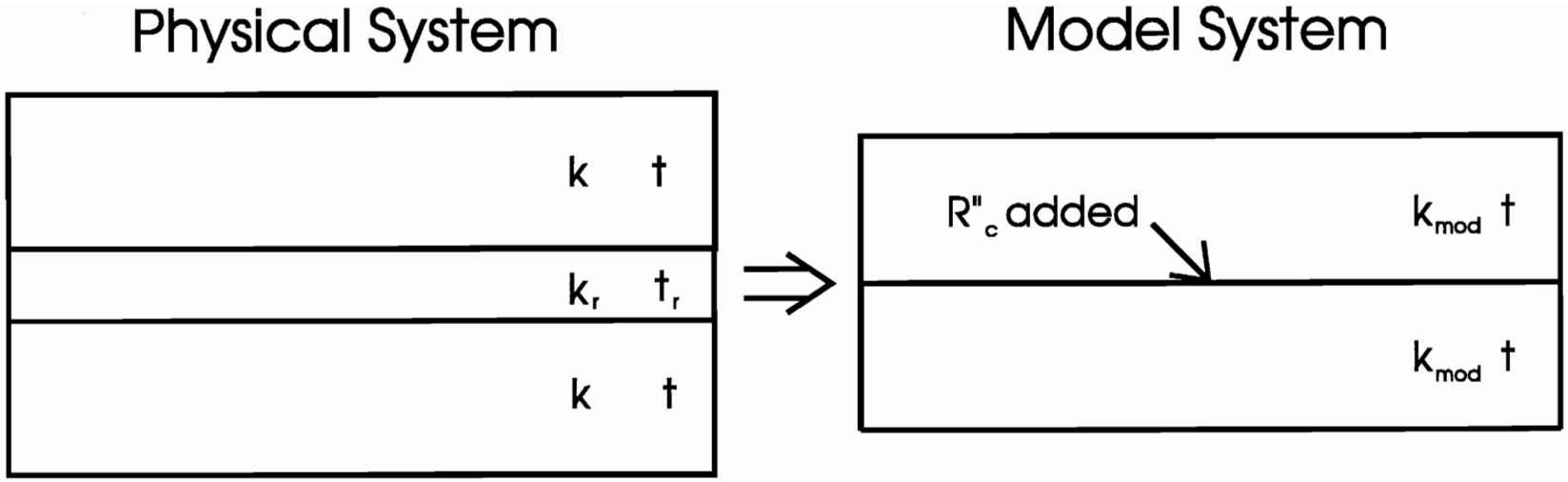


Figure 61: Approximations for Removal of Middle Layer

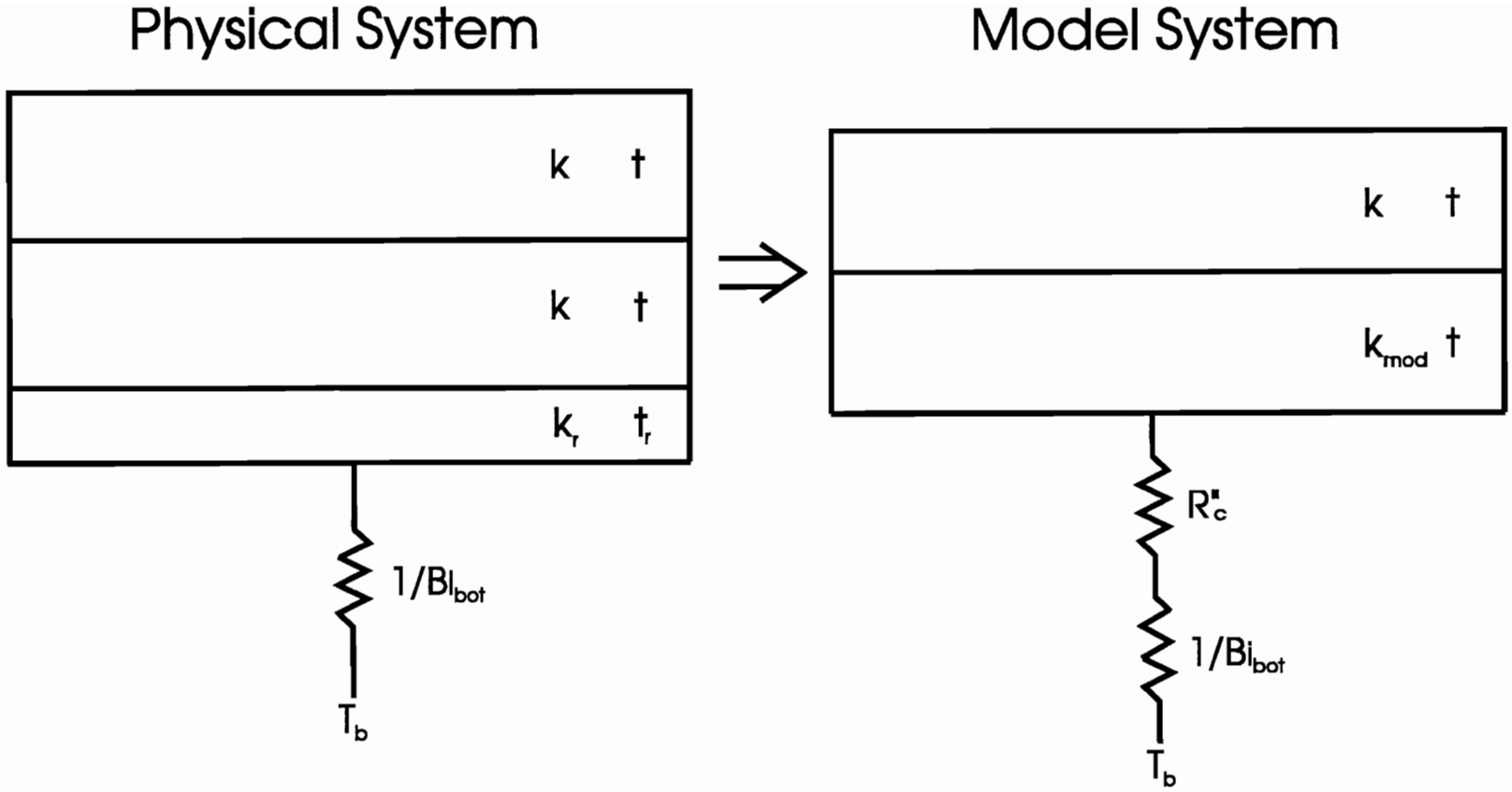


Figure 62: Approximations for Removal of Bottom Layer

To test just the grid, the model is run with both three layers and five layers. Comparison of these results shows that the three layer model provides sufficient grid resolution in the z-direction. Similar comparisons between x-y grid spacings of 16 X 8, 40 X 20 and 80 X 40 show that the 40 X 20 grid is adequate to provide three significant figures in the results.

The model was then run for three layers and compared to an equivalent two layer case. In the two-layer case, the layer with the variable attributes was removed and several approximations were made to imitate the physical characteristics of the missing layer.

To compensate for the thermal resistance the removed layer provided in the z direction, a non-dimensional contact resistance is added to the model as:

$$R_c'' = \frac{\tau_r}{K_r} \tag{64}$$

This contact resistance is added as a modification of the conductances expressed in the discretization equation (see equation 10).

A similar effect could be obtained by modifying the z-direction conductivity of adjacent layers. The contact resistance is applied either between two layers or between a layer and the environment (see Figs. 60-62). If the removed layer is on the bottom the contact resistance can be added to the non-dimensional heat sink resistance:

$$R''_{bot} = \frac{1}{Bi_{bot}} \cdot R''_c \tag{65}$$

The top case is similar in the addition of the contact resistance for convection:

$$R''_{top} = \frac{1}{Bi_{top}} + R''_{c} \tag{66}$$

but the approximate surface temperature must be obtained by extrapolation through the contact resistance:

$$T_{\max} = T_{top} + R''_{c} \cdot q''_{d} \tag{67}$$

The x-y conductivity of the adjacent layers are modified based on the removed layer thickness and conductivity:

$$k_{equiv} = \frac{\frac{1}{k \cdot t} + \frac{c}{k_r \cdot t_r}}{\frac{c \cdot t_s}{k \cdot t \cdot k_r \cdot t_r}} \tag{68}$$

where c = 2 if the removed layer is between two layers (i.e. two layers have the conductivity adjusted) and c = 1 if the removed layer is on the top or bottom of the device (i.e. only one layer conductivity is adjusted.) This measure imitates the spreading resistance or conductance that the removed layer provided in the lateral directions. The conductivity correction tended to overestimate the actual spreading of the removed layer. Trial and error showed that two-thirds of the gave the best results. The modified conductivity was then calculated using:

$$k_{\mathrm{mod}} = \frac{2}{3} \cdot (k_{equiv} - k) + k \tag{69}$$

6.4 Results

The error between the two-layer and three-layer model was calculated by comparing the maximum temperatures over a range of conductivity and thickness values. The relative error is compared to the product of the non-dimensional conductivity and the non-dimensional thickness $K_r \tau_r$. The non-dimensional conductivity-thickness product is defined by

$$K_r \tau_r = \frac{k_r}{k} \cdot \frac{t_r}{t} \tag{70}$$

and is representative of the conductance (or inverse of thermal resistance) of the removed layer in the x-y plane.

This model was initially run for $Bi_{top} = 1.0$ and $Bi_{bot} = 1.0$. With the removed layer at the bottom of the model, the two-layer model gave reasonable results throughout the range of thickness-conductivity products (see Fig. 63.). The accuracy of the two-layer model began to deteriorate at a $K_r\tau_r$ value of 10 and sharply declines at a value of 100.

When the removed layer is positioned in the middle, the error behaves similarly (see Fig. 64). The small upturn at a $K_r\tau_r$ value of 1 causes the error at $K_r\tau_r = 10$ to be slightly smaller than in the bottom case. At $K_r\tau_r = 100$, the error is nearly the same, but the error declines more sharply to that point.

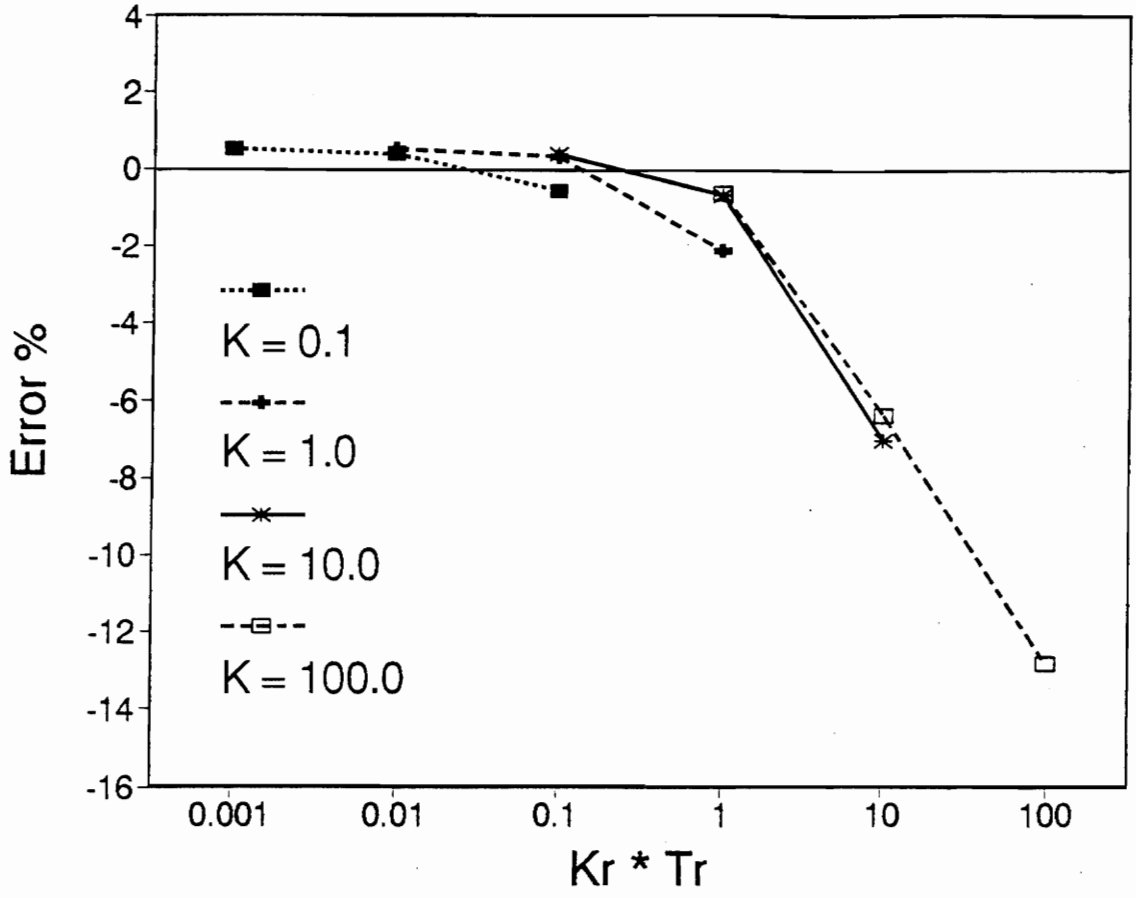


Figure 63: Relative Error vs. $K\tau_r$ for Bottom Case with $Bi_{top} = 1.0$, $Bi_{bot} = 1.0$

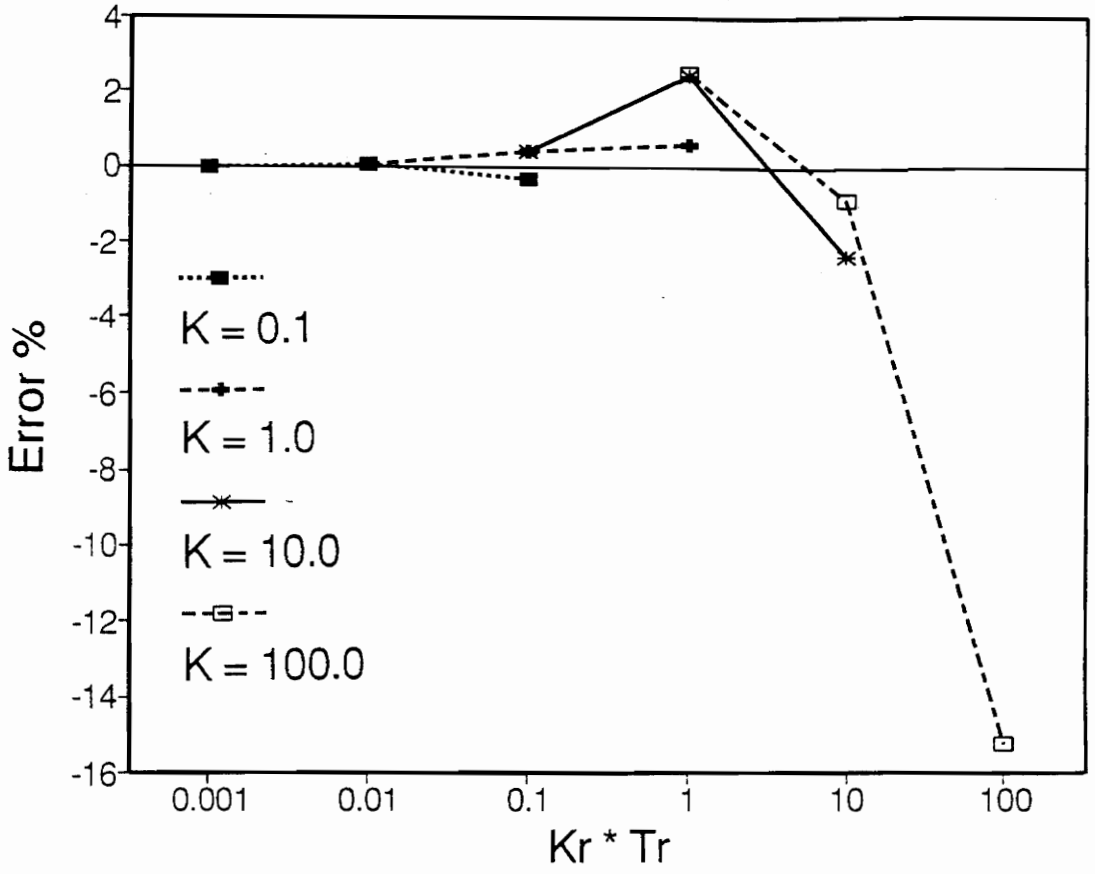


Figure 64: Relative Error vs. $K\tau_r$ for Middle Case with $Bi_{top} = 1.0$, $Bi_{bot} = 1.0$

With the removed layer in the top position, the error behaves markedly different (see Fig. 65). The error begins a sharp upturn at a $K_r\tau_r$ value of 1 and becomes unacceptably high at larger $K_r\tau_r$ values.

Comparing Figs. 63-65 shows that the approximations used work well for low $K_r\tau_r$ values. At a conductivity-thickness product of 10, the error does not exceed 3 percent for the middle case and 8 percent for the bottom case. For the top case, at a thickness conductivity value of 1, the error does not exceed 30 percent. As the values become higher, the absolute error increases. This reflects the strong influence high values of thickness and conductivity have on the behavior of the model. This influence is particularly evident when the removed layer is positioned on the top. Here it plays a crucial role in convection and spreading from the device. The approximations tend to overestimate the contribution of the removed layer when it is in the middle or on the bottom. However, when the removed layer is on the top, the approximations tend to underestimate the layer's importance. This can be attributed to the proximity of the layer to the heat source. For high values of the thickness-conductivity product, the spreading caused by the top layer is so large that the approximations fail to account for the complete effect.

The effect of variation of the external environment on the top case bears further inspection. The errors with $Bi_{top} = 0.1 = Bi_{bot} = 0.1$ are less than those for $Bi_{top} = Bi_{bot} =$

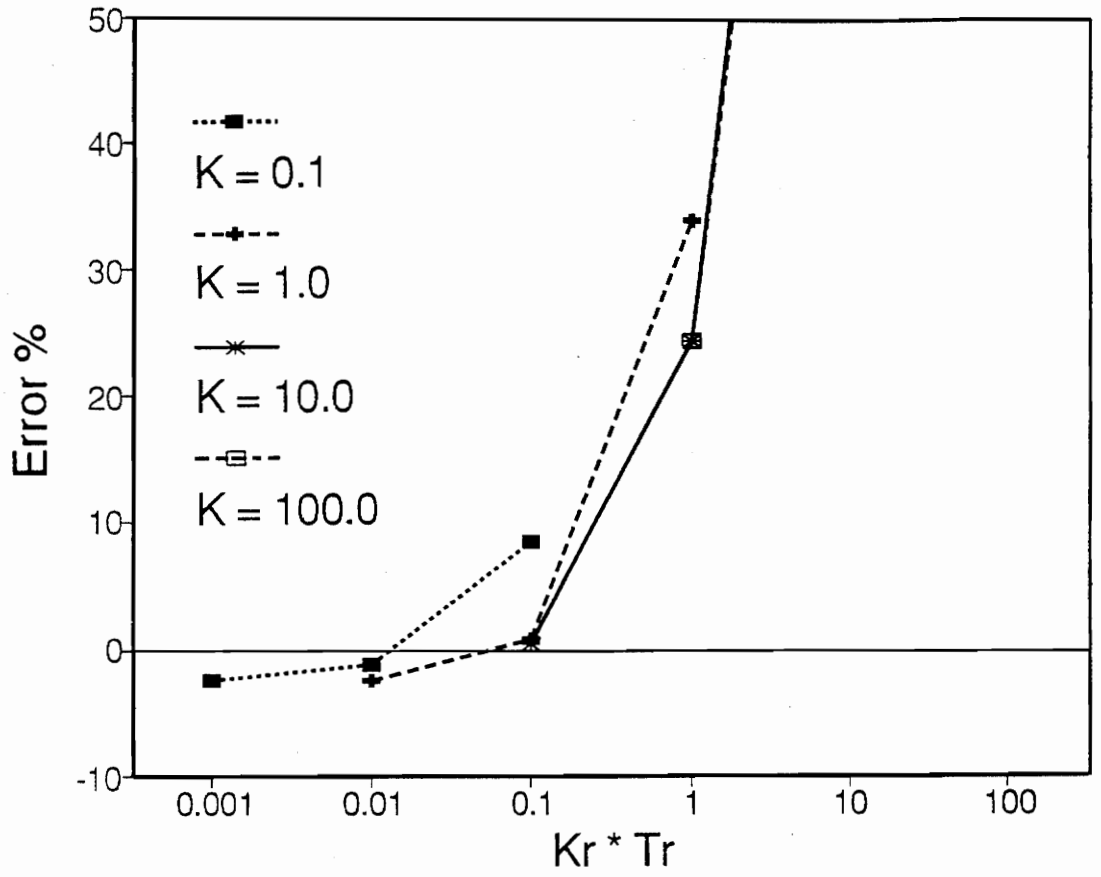


Figure 65: Relative Error vs. $K\tau_r$ for Top Case with $Bi_{top} = 1.0$, $Bi_{bot} = 1.0$

1.0 (see Fig. 66). The error is further reduced with an insulated top ($Bi_{top} = 0.0$) and $Bi_{bot} = 0.1$ (see Fig. 67). The lower Biot numbers allow the uniform approximation of spreading and z-direction resistance to work when adjacent to the heat source.

All of the cases are affected by changes in the external environment. Table 3 summarizes these results for various $K_r\tau_r$ values. For the top case with $K_r\tau_r = 10$ and $K_r\tau_r = 100$, the errors were larger than 40%, and so they are not included in the table.

Table 3: Summary of Relative Error Results

$K\tau_r = 1$	$Bi_{top} = 1.0$ $Bi_{bot} = 1.0$	$Bi_{top} = 0.0$ $Bi_{bot} = 1.0$	$Bi_{top} = 0.1$ $Bi_{bot} = 0.1$	$Bi_{top} = 0.0$ $Bi_{bot} = 0.1$
top	34.0	25.5	28.5	22.9
middle	-2.5	1.3	3.5	3.3
bottom	-2.1	-2.1	1.8	1.8
$K\tau_r = 10$				
middle	-2.4	-12.7	-10.1	-8.1
bottom	-7.0	-7.0	-4.5	-4.2
$K\tau_r = 100$				
middle	-15.2	-40.9	-36.5	-27.3
bottom	-12.7	-12.8	-11.3	-9.7

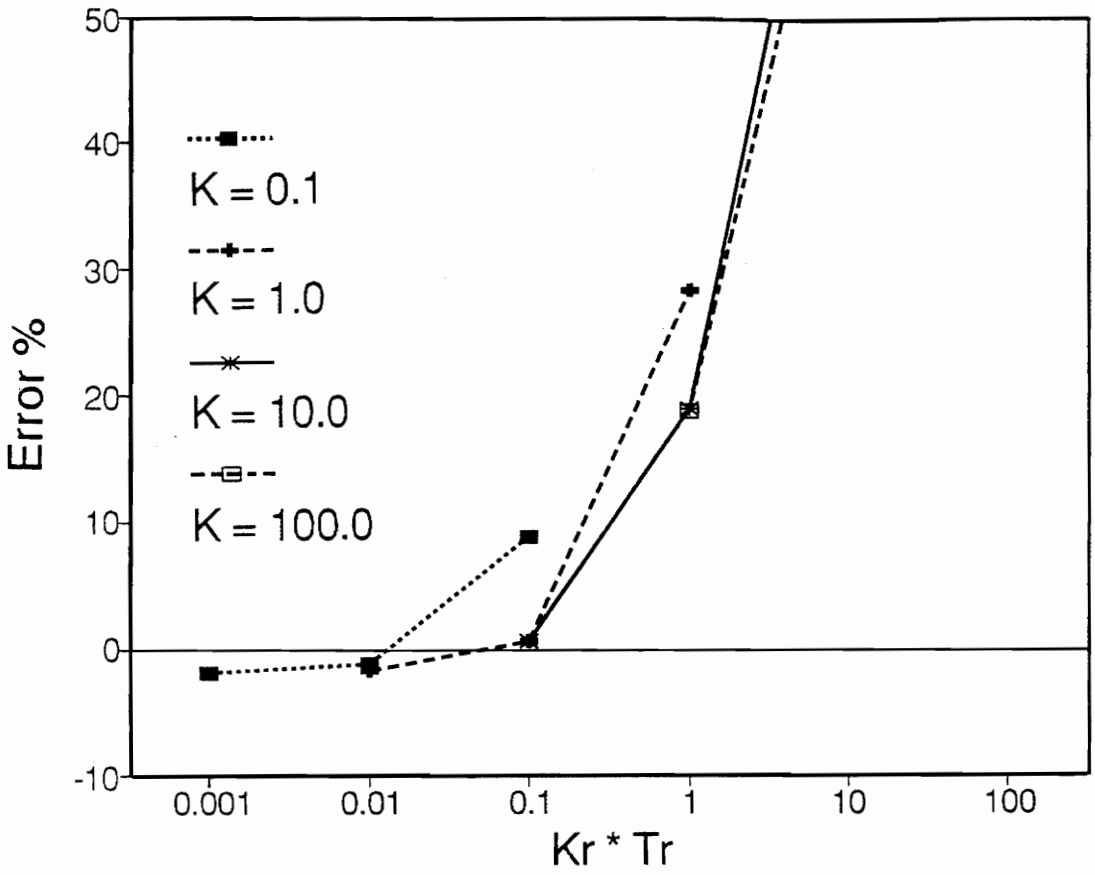


Figure 66: Relative Error vs. $K\tau_r$ for Top Case with $Bi_{top} = 0.1$, $Bi_{bot} = 0.1$

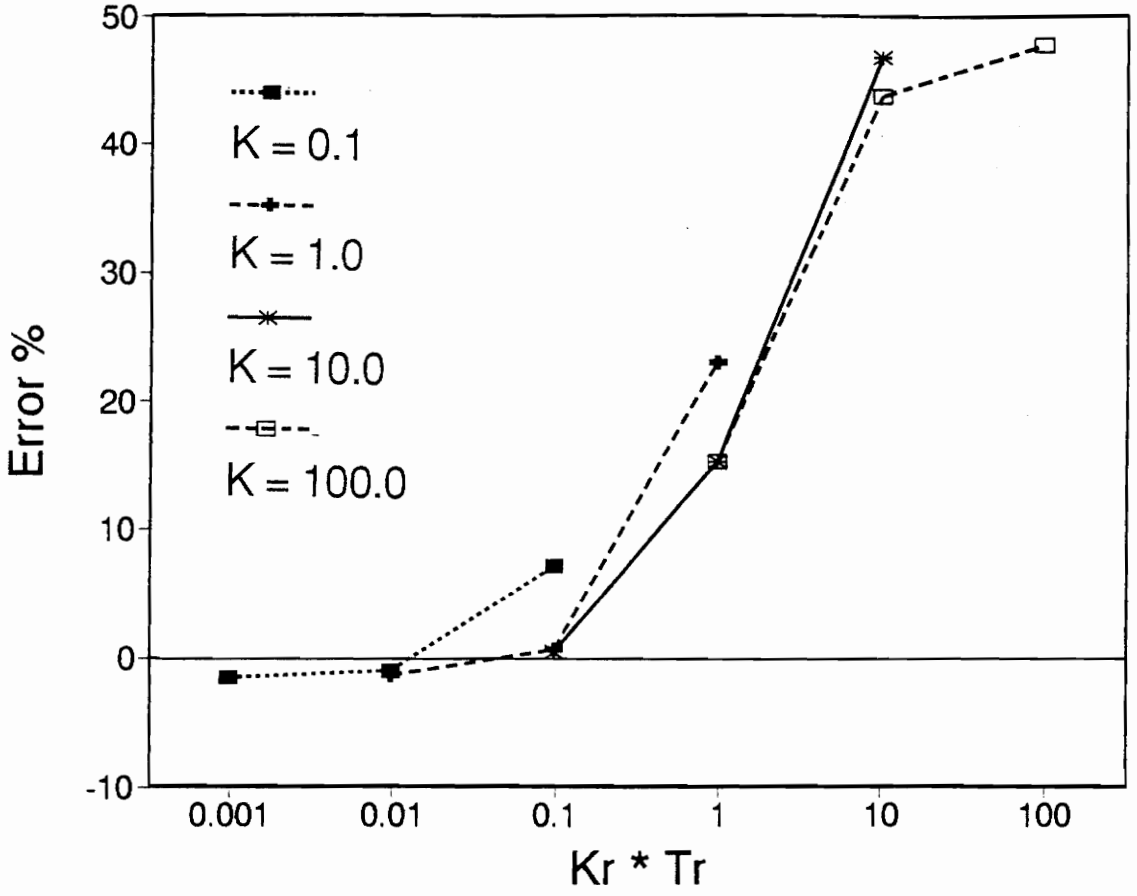


Figure 67: Relative Error vs. $K\tau_r$ for Top Case with $Bi_{top} = 0.0$, $Bi_{bot} = 0.1$

6.5 Application Example

This example of a simple multilayer system is taken from reference 9. Since the temperature rise is linearly proportional to the device heat dissipation, these results are expressed as thermal resistance values or temperature rise per unit power dissipation (K/W). The actual temperature rise for any power level can then be obtained by multiplying the thermal resistance times the device total heat dissipation ($=q\ d^2$). Consider then two square devices of size d = 3.18 mm placed on a copper metallization layer with thickness t_r mounted on a 0.635 mm thick alumina substrate with dimension b = 63.5 mm.

Substrate thermal conductivity:

$$k = 25 \text{ W/m K}$$

Metallization layer conductivity:

$$k_r = 400 \text{ W/m K}$$

Thus, the nondimensional metallization layer conductivity is:

$$K_r = 16$$

Equivalent heat transfer coefficient for heat sink thermal resistance:

$$h = 4000 \text{ W/m K}$$

The nondimensional device size is:

$$D = \frac{3.18}{0.635} = 5.0$$

And the nondimensional substrate size is:

$$B = \frac{63.5}{0.635} = 100$$

The substrate thickness is modeled in two layers, each having nondimensional thickness of $t = 0.5$.

The Biot number for the heat sink is:

$$B_{i_{bot}} = \frac{ht}{k} = 0.1$$

The thermal resistance can be expressed as:

$$R_{th} = \frac{T_{max} - T_o}{q''_d d^2} = \frac{\theta_{max}}{Dk_d} \quad (K/W)$$

For a relatively thin copper layer thickness of 0.00635 mm ($\tau_r = 0.01$), the solution with the copper layer included is $R_{th} = 9.35$ K/W. The approximate 2-layer solution yields $R_{th} = 9.95$ K/W. This results in a relative error of 6.4% when the copper layer is eliminated from the model. The $K_r\tau_r$ product is 0.16 for these conditions.

When the copper thickness is increased to 0.0635 mm ($\tau_r = 0.1$), the full model gives $R_{th} = 5.38$ K/W. The solution using the layer approximations gives $R_{th} = 7.97$ K/W, a relative error of 48%. An error this high would be unacceptable for most applications. The $K_r\tau_r$ product has increased to 1.6.

The proximity of the metallization layer to the heat source heightens its importance in the role of spreading. Thus, in this example, only the approximation of thin copper layers is possible.

6.6 Conclusions

To help speed the thermal analysis used for design of multilayer microelectronic packages, the possibility of neglecting some layers (to reduce the number of nodes) is investigated. By considering a model three layer system, a set of criteria have been developed to relate the error caused by approximating a layer to the thickness and conductivity of the layer relative to adjacent layers.

The criteria can be summarized as follows:

For a relative thickness-conductivity product

$$\frac{t_r}{t} \cdot \frac{k_r}{k} \leq 10$$

the error in the top heat source temperature rise will be less than 8% when an interior layer is replaced by an equivalent contact resistance (see eqn. 43) and the x-y conductivity of the adjacent layers is modified as in eqn. 48. If the layer is adjacent to the heat source, then the relative thickness-conductivity product should be less than 0.1 to keep the error below 10%.

These criteria indicate that, for example, a low conductivity epoxy bond between a substrate and heat sink can usually be approximated. However, a high conductivity metalization layer adjacent to a device must be very thin if it is to be approximated with acceptable accuracy. The user must weigh increased computational speed against the error that is induced by the simplification of the model.

7. Conclusions and Recommendations

A numerical model has been devised which is capable of accurately modeling microelectronic devices. Comparison with established numerical techniques has shown the validity of the model.

A layer approximation scheme was devised to allow the simplification of the numerical model while retaining accuracy. The conditions under which the layer approximation is valid were determined and an approximation criterion was defined.

Although the capability of the program to model relatively simple physical systems has been demonstrated, the effectiveness on more complex systems has not been investigated. A device with many connections could prove to be unwieldy, both in defining the geometry correctly, and in computational requirements. To better model complex geometries, an optimization of the program would be necessary. Dynamic allocation of memory would also make the code more efficient.

This program provides a solid basis for a more complete thermal analysis program. Additional analytical and programming work is needed to produce a more versatile and efficient tool. However, the current code would be useful in a wide spectrum of problems.

References

1. G. N. Ellison (1989), Thermal Computations for Electronic Equipment, Malabar, FL: Robert E. Krieger Publishing Company, Inc.

2. J. N. Sweet, D. W. Peterson, D. Chu, B. L. Bainbridge, R. A. Gassman, and C. A. Reber (1993), "Analysis and Measurement of Thermal Resistance in A Three-Dimensional Silicon Multichip Module Populated with Assembly Test Chips," Proceedings, Ninth IEEE Semiconductor Thermal Measurement and Management Symposium, pp. 1-7.

3. K. J. Arbeitman (1993), Temperature Sensitivity to Node Spacing in ASTAP Finite Difference Modeling For Flat Cap Single- and Multi-Chip Modules," Proceedings, Ninth IEEE Semiconductor Thermal Measurement and Management Symposium, pp. 81-87.

4. S. Lee, T. F. Lemczyk, and M. M. Yovanovich (1992), "Analysis of Thermal Vias in High Density Interconnect Technology," Proceedings, Eighth IEEE Semiconductor Thermal Measurement and Management Symposium, pp. 55-61.

5. D. J. Nelson and W. A. Sayers (1992), "A Comparison of Two-Dimensional Planar, Axisymmetric and Three-Dimensional Spreading Resistances," Proceedings, Eighth IEEE Semiconductor Thermal Measurement and Management Symposium, pp. 62-68.

6. G. N. Ellison (1993), Advances in Thermal Modeling of Electronic Components and Systems, vol. 3, chapter 3, Ed. A. Bar-Cohen, A. D. Kraus, ASME, New York: IEEE Press.

7. G. N. Ellison (1990), "TAMS, A Thermal Analyzer for Multilayer Structures", Electrosoft, vol. 1, pp. 85-97.

8. D. H. Chien, C. T. Wang and C. C. Lee (1992), "Temperature Solution of Five-Layer Structure with a Circular Embedded Source and Its Applications," IEEE Components, Hybrids, and Manufacturing Technology, vol. 15, No.5, pp. 707-714.

9. M. M. Hussein, D. J. Nelson, and A. Elshabini-Riad (1992), "Thermal Interaction of Semiconductor Devices on Copper Clad Ceramic Substrates," IEEE Components, Hybrids, and Manufacturing Technology, vol. 15, no. 5, pp. 651-657.

10. S. V. Patankar (1991), Computation of Conduction and Duct Flow Heat Transfer, Maple Grove, MN: Innovative Research, Inc.

Appendix A - Nomenclature

A	area used in thermal resistance calculation (m^2)
A_{via}	cross-sectional area of a thermal via (m^2)
(A,B)	represent subscripts (I, J, or K)
area	area on the boundary (m^2)
d	length used in thermal resistance calculation
d_{via}	length of a thermal via (m)
E_{gen}	heat generated in the model (W)
E_{in}	heat into the model (W)
E_{out}	heat out of the model (W)
E_{total}	sum of the heat transfer on the entire model (W)
F	flux
F_c	constant part of linearized flux term
F_p	coefficient of T in the linearized flux term
k_{via}	conductivity of a thermal via (W/mK)
Q	heat transfer
Q_{cv}	heat transfer into a control volume (W)
Q_{total}	total heat transfer through a connection
R_a	thermal resistance of a connection (K/W)
R_{cv}	thermal resistance of a connection to a control volume (K/W)
R_{via}	thermal resistance of a via connection (K/W)
S	source
S(I,J,K)	source term for control volume (I,J,K)
S_c	constant part of linearized source term
S_p	coefficient of T in the linearized source term
T_a	known temperature of a connection through a thermal resistance (K)
T_1	temperature at one end of a via connection
T_2	temperature at one end of a via connection
$T_{boundary}$	boundary grid point temperature (K)
T_p	temperature at some point in the model
VOL(I,J,K)	volume of control volume (I,J,K)
x	represents a boundary variable (I1, J1, K1, L1, M1, N1)

Chapter 6

b	substrate size (m)
B	non-dimensional substrate size = b / t
Bi_{top}	Biot number for the top and edges = $h\ t / k$
Bi_{bot}	Biot number for the bottom heat sink = $t / k\ R^2_{bot}$
d	device size (m)
D	non-dimensional device size = d / t
h	convective heat transfer coefficient ($W/m^2\ K$)
k_r	conductivity of the removed layer (W/m K)
k	substrate conductivity (W/m K)
K_r	non-dimensional conductivity in removed layer = k_r / k
K	non-dimensional conductivity in substrate = 1.0
q_d''	heat flux from the device
t_r	thickness of the removed layer (m)
t	substrate thickness (m)
R''_{bot}	effective resistance per unit area of the bottom heat sink ($K\ m^2 / W$)
τ_r	non-dimensional removed-layer thickness = t_r / t
τ	non-dimensional substrate thickness = 1.0
T	temperature (°C)

Program Variables

ALAM(I,J,K)	storage capacity per unit volume (for unsteady problems)
CVX	number of control volume widths in the x direction
CVY	number of control volume widths in the y direction
CVZ	number of control volume widths in the z-direction (layers)
DPEDGE(NEDGE)	defines length of edge connection
DT	time step (for unsteady problems)
DXLAY	x-direction length of the top layer boundary
DXSRC(NSRC)	x-direction length of heat source area
DXSURF(NSURF)	x-direction length of surface connection area
DYLAY	y-direction length of the top layer boundary
DYSRC(NSRC)	y-direction length of heat source area
DYSURF(NSURF)	y-direction length of surface connection area
FLUXI1(J,K)	flux into I1 boundary
FLUXJ1(I,K)	flux into J1 boundary
FLUXK1(I,J)	flux into K1 boundary
FLUXL1(J,K)	flux into L1 boundary
FLUXM1(I,K)	flux into M1 boundary
FLUXN1(I,J)	flux into N1 boundary
FLXCI1(J,K)	constant for flux expression on I1 boundary
FLXCJ1(I,K)	constant for flux expression on J1 boundary
FLXCK1(I,J)	constant for flux expression on K1 boundary
FLXCL1(J,K)	constant for flux expression on L1 boundary
FLXCM1(I,K)	constant for flux expression on M1 boundary
FLXCN1(I,J)	constant for flux expression on N1 boundary
FLXPI1(J,K)	coefficient for flux expression on I1 boundary
FLXPJ1(I,K)	coefficient for flux expression on J1 boundary
FLXPK1(I,J)	coefficient for flux expression on K1 boundary
FLXPL1(J,K)	coefficient for flux expression on L1 boundary
FLXPM1(I,K)	coefficient for flux expression on M1 boundary
FLXPN1(I,J)	coefficient for flux expression on N1 boundary
HBOT	convective cooling coefficient for the top suface and edges
HTOP	convective cooling coefficient for the top suface and edges
I1	boundary where I = 1
I2	I1 - 1
I1VIA(NVIA)	I coordinate of one end of a via connection
I2VIA(NVIA)	I coordinate of one end of a via connection
IEDGE(NEPT)	I subscript of point in edge connection point list
ISRC(NSCPT)	I subscript of point in heat source point list
ISURF(NSFPT)	I subscript of point in surface connection point list
ITER	iteration counter for the outer loop
J1	boundary where J = 1

J2	J1 - 1
J1VIA(NVIA)	J coordinate of one end of a via connection
J2VIA(NVIA)	J coordinate of one end of a via connection
JEDGE(NEPT)	J subscript of point in edge connection point list
JSRC(NSCPT)	J subscript of point in heat source point list
JSURF(NSFPT)	J subscript of point in surface connection point list
K1	boundary where K = 1
K2	K1 - 1
K1VIA(NVIA)	K coordinate of one end of a via connection
K2VIA(NVIA)	K coordinate of one end of a via connection
KBCI1(J,K)	boundary condition indicator for I1 boundary
KBCJ1(I,K)	boundary condition indicator for J1 boundary
KBCK1(I,J)	boundary condition indicator for K1 boundary
KBCL1(J,K)	boundary condition indicator for L1 boundary
KBCM1(I,K)	boundary condition indicator for M1 boundary
KBCN1(I,J)	boundary condition indicator for N1 boundary
KEDGE(NEPT)	layer on which an edge connection acts
KNEDGE(NEDGE)	layer on which an edge connection acts
KNSRC(NSRC)	layer on which a source term acts
KNSURF(NSURF)	layer on which a surface connection acts
KPRINT(NF)	enables printing of F(NF)
KSOLVE(NF)	enables solution of dependent F(NF)
KSRC(NSCPT)	layer on which a source term acts
KSURF(NSFPT)	layer on which a surface connection acts
KSTOP	when nonzero, computation stops
L1	boundary where I = L1
L2	L1 - 1
LAST	maximum number of outer iterations
M1	boundary where J = M1
M2	M1 - 1
MODE	indicator for coordinate system (1 for rectangular, 3 for cylindrical)
N1	boundary where K = N1
N2	N1 - 1
NCVX(NZ)	number of x-direction control volumes widths in a zone
NCVY(NZ)	number of y-direction control volumes widths in a zone
NDV	maximum number of vias that can be defined
NED	maximum number of edge connections that can be defined
NEDGE	edge connection index
NEDGE(NEPT)	reference pointer for edge connection point list
NEPT	edge connection point index
NL	number of layers
NPT	maximum number of points in a list
NSC	maximum number of heat sources that can be defined

NSCPT	heat source point index
NSF	maximum number of surface connections that can be defined
NSFPT	surface connection point index
NSRC	heat source index
NSRC(NSCPT)	reference pointer for heat source point list
NSURF	surface connection index
NSURF(NSFPT)	reference pointer for surface connection point list
NTC	iteration counter in SOLVE3D
NTIMES	maximum number of iteration in SOLVE3D
NVIA	via connection index
NWEDGE(NEDGE)	defines the edge on which an edge connection acts
NZ	subscript for a zone
NZX	number of zones in the x direction
NZY	number of zones in the y direction
PEDGE(NEDGE)	defines initial point of edge connection
QBASE	base heat transfer, used in energy balance
QEDGE	edge heat transfer, used in energy balance
QGEN	heat generated, used in energy balance
QTOP	top heat transfer, used in energy balance
QTOL	energy balance on the model
QSRC(NSRC)	heat input of a heat source (W)
R(J)	radius for grid point
REDGE(NEDGE)	thermal resistance of an edge connection
RSURF(NSURF)	thermal resistance of a surface connection
RV(J)	radius at the control volume face
RVIA(NVIA)	thermal resistance of a via connection
SC(I,J,K)	constant for source expression for control volume (I,J,K)
SP(I,J,K)	coefficient for source expression for control volume (I,J,K)
SX(J)	scale factor for the x direction at grid locations Y(J)
TBOT	ambient temperature in the bottom environment
TEDGE(NEDGE)	temperature of a edge connection
TSURF(NSURF)	temperature of a surface connection
TTOP	ambient temperature in the top environment
X(I)	value of x at grid location I
X1VIA(NVIA)	x coordinate of the via connection area
X2VIA(NVIA)	x coordinate of the via connection area
XCV(I)	x-direction width of control volume
XL	x-direction size of the numerical model
XLAY	x-coordinate of lower-left corner of top layer boundary
XSRC(NSRC)	x coordinate of the lower left corner of the heat source area
XSURF(NSURF)	x coordinate of the lower left corner of the surface connection area
XZONE(NZ)	x-direction length of a zone
Y(J)	value of y at grid location J

Y1VIA(NVIA)	y coordinate of the via connection are
Y2VIA(NVIA)	y coordinate of the via connection are
YCV(J)	y-direction width of control volume
YCVR(J)	area r•Δy for a control volume
YL	y-direction size of the numerical model
YLAY	y-coordinate of lower-left corner of top layer boundary
YSRC(NSRC)	y coordinate of the lower left corner of the heat source area
YSURF(NSURF)	y coordinate of the lower left corner of the surface connection area
YZONE(NZ)	Y-direction length of a zone
Z(K)	value of z at grid location K
ZCV(K)	z-direction width of control volume

Appendix B - ORTHO3D.F

```
************************************************************************
       PROGRAM ORTHO3D
*
* Control Volume Method solution of three-dimensional, orthotropic
* heat conduction
*
* User portion of code contained in layer*.f
* layer3d.f - standard user subroutine
* layermod.f - subroutine with layer approximation modification
*
* Contains additions in layer*.f specifically designed to model
* multilayer microelectronics
*
* Modify here for multiple cases
*
* RS/6000 version - double precision
*
* Program structure similar to that detailed in
* "Computation of Conduction and Duct Flow Heat Transfer",
*  S.V. Patankar, Maple Grove, MN:  Innovative Research, Inc.
*
************************************************************************
*
      INCLUDE 'common3d.f'

* set up loops for multiple cases (need to use input file
*    in layer*.f)  # of cases = icases

      i = 0
      icases = 1
      do 13 i = 1, icases
      kstop = 0


      CALL DEFLT
      CALL GRID
      CALL READY
      CALL BEGIN

   10 CONTINUE

* start iteration of outer loop
       CALL OUTPUT
* check to see if convergence has occured
       IF(KSTOP.EQ.0) then

              CALL HEART
                  GO TO 10

      ENDIF

   13  continue


      STOP
      END
```

```
* include other subroutines in invariant part
      INCLUDE 'defrd3d.f'
      INCLUDE 'heart3d.f'
      INCLUDE 'solve3d.f'
      INCLUDE 'tools3d.f'
```

Appendix C - LAYER3D.F

```
SUBROUTINE ADAPT
c
c  MCM example - see chapter 5
c
C-----------------------------------------------------------------------
C------- STEADY CONDUCTION in a 3-D layered structure
C        uniform material with contact resistance between layers
C-----------------------------------------------------------------------
      INCLUDE 'common3d.f'
C***********************************************************************

c set number of layers here with NL
      parameter(  NL=3    ,NSC=50,NSF=50,
     +   NED=5,NDV=50,NPT=3000)

      DIMENSION                T(NI,NJ,NK),
     1 KNSRC(NSC),XSRC(NSC),YSRC(NSC),DXSRC(NSC),DYSRC(NSC),QSRC(NSC),
     3 ISRC(NPT),JSRC(NPT),KSRC(NPT),NSRC(NPT),
     4 KNSURF(NSC),XSURF(NSC),YSURF(NSC),
     5 DXSURF(NSC),DYSURF(NSC),RSURF(NSC),TSURF(NSC),
     6 ISURF(NPT),JSURF(NPT),KSURF(NPT),NSURF(NPT),
     7 KNEDGE(NED),NWEDGE(NED),PEDGE(NED),DPEDGE(NED),REDGE(NED),
     8 TEDGE(NED),IEDGE(NPT),JEDGE(NPT),KEDGE(NPT),NEDGE(NPT),
     9 K1VIA(NDV),X1VIA(NDV),Y1VIA(NDV),K2VIA(NDV),X2VIA(NDV),
     9 Y2VIA(NDV),RVIA(NDV),I1VIA(NDV),J1VIA(NDV),
     9 I2VIA(NDV),J2VIA(NDV),
     9 intemp(4), cond(NL), Tguess(NL)

      EQUIVALENCE (F(1,1,1,1),T(1,1,1))

      INTEGER CVX, CVY, CVZ
      REAL HTEDGE,HTSURF

C*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
      ENTRY GRID

      OPEN (1, FILE = 'input')
      OPEN (2, FILE = 'prtmax')
      OPEN (3, FILE = 'pts')
* files used for graphing with MATLAB
* needs to be a file for each layer to be graph
      OPEN (11, FILE = 'mat1.dat')
      OPEN (12, FILE = 'mat2.dat')
      OPEN (13, FILE = 'mat3.dat')
      OPEN (14, FILE = 'mat4.dat')
      OPEN (15, FILE = 'mat5.dat')

      CVX = 80
      CVY = 80
      CVZ = NL

      epsi = 1.E-5
      Tguess(1)=60.
      Tguess(2)=70.
      Tguess(3)=80.

c text to indentify output
```

```
      HEADER ='STEADY CONDUCTION in a true 3-D Layer Ortho'
c output file
      PRINTF ='print'

c x, y, and z dimensions of model
      XL     = 0.025d0
      YL     = 0.025d0

* define "tiny" to be used to make sure some of the logic loops work
      tiny = XL / 1000.

c  zoned grid method
c  set up x and y zones to match sources and different size layers
c    or holes in a layer
       NZX = 1
       XZONE(1) = .025d0
        NCVX(1) = cvx

       NZY = 1
       YZONE(1) = .0250d0
        NCVY(1) = cvy

c set up z grid to match the layers
c ZZONE is the thickness of each layer
c NCVZ is the control volume width of each layer
        NZZ = 3
        ZZONE(1) = .01d0
        NCVZ(1)  = 1
      ZZONE(2) = .001
      NCVZ(2)  = 1
      ZZONE(3) = .0005d0
        NCVZ(3)  = 1

      CALL ZGRID

      RETURN
C*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
      ENTRY BEGIN

c set maximum number of outer interations
      LAST  = 60

c define top and bottom convective cooling coeffecients and temperatures
      Htop = 15.0d0
      Hbot = 1500.0d0
      Ttop  = 0.0d0
      Tbot  = 0.0d0

c define conductivity for each layer (remember layer1 = bottom layer)
        cond(1) = 29.
        cond(2) = 1.0
        cond(3) = 153.

c  set outer limits for top layer
      XLAY  = 0.005
      YLAY  = 0.005
      DXLAY = 0.015
      DYLAY = 0.015

* set contact resistances
```

```
      do 95 nf = 1, 20
            rc(nf) = 0.0
   95 continue

c name for NF = 1 (temperature) variable
      TITLE(1)='   Temperature    '
      KSOLVE(1)=1
      KPRINT(1)=1
      KPLOT(1) =0

c  Set initial temperature level - estimates

      DO 100 K=1,NL
      DO 100 J=1,M1
      DO 100 I=1,L1
         T(I,J,K)=Tguess(K)
  100 CONTINUE

****************** added

* source variables
* remember that QSRC is heat input, not flux
      KNSRC(1) = 5
      XSRC(1)      = 0.005
      YSRC(1)      = 0.005
      DXSRC(1) = 0.015
      DYSRC(1) = 0.015
      QSRC(1)      = 14.0625

* vias
c resistance for connection from layer 2 to layer 3 (K=3 to K=4)
      R43 = 67.93
c resistance for connection from layer 1 to layer 2 (K=2 to K=3)
      R32 = 1816.2


c define via connections
c ntemp,xtemp, and ytemp are used to get the correct spacing
      ntemp = 1
      ytemp = 0.005d0
      do 121 j = 1, 4
      xtemp = 0.005d0
      ytemp = ytemp + 0.003d0
      do 121 i = 1, 4
      xtemp = xtemp + 0.003d0
        K1VIA(ntemp) = 4
        X1VIA(ntemp) = xtemp
        Y1VIA(ntemp) = ytemp
        K2VIA(ntemp) = 3
        X2VIA(ntemp) = xtemp
        Y2VIA(ntemp) = ytemp
        RVIA(ntemp)  = R43
      ntemp = ntemp + 1
 121  continue


        ytemp = 0.005d0
        do 122 j = 1, 4
        xtemp = 0.005d0
        ytemp = ytemp + 0.003d0
```

```
        do 122 i = 1, 4
        xtemp = xtemp + 0.003d0
        K1VIA(ntemp) = 3
        X1VIA(ntemp) = xtemp
        Y1VIA(ntemp) = ytemp
        K2VIA(ntemp) = 2
        X2VIA(ntemp) = xtemp
        Y2VIA(ntemp) = ytemp
        RVIA(ntemp)  = R32
        ntemp = ntemp + 1
 122    continue


* set resistance for wire bonds
      Rwb = 6780.
c puts in the vias for the wire bonds on the left side
        ytemp = 0.005d0
        do 113 j = 1, 4
        ytemp = ytemp + 0.003d0

        K1VIA(ntemp) = 3
        X1VIA(ntemp) = 0.0025
        Y1VIA(ntemp) = ytemp
        K2VIA(ntemp) = 4
        X2VIA(ntemp) = 0.0070
        Y2VIA(ntemp) = ytemp
        RVIA(ntemp)  = Rwb

      ntemp = ntemp + 1
 113    continue
c puts in the vias for the wire bonds on the right side
        ytemp = 0.005d0
        do 114 j = 1, 4
        ytemp = ytemp + 0.003d0
        K1VIA(ntemp) = 3
        X1VIA(ntemp) = 0.0225
        Y1VIA(ntemp) = ytemp
        K2VIA(ntemp) = 4
        X2VIA(ntemp) = 0.0180
        Y2VIA(ntemp) = ytemp
        RVIA(ntemp)  = Rwb

      ntemp = ntemp + 1
 114    continue
c puts in the vias for the wire bonds on the bottom side
        xtemp = 0.005d0
        do 115 i = 1, 4
        xtemp = xtemp + 0.003d0

        K1VIA(ntemp) = 3
        X1VIA(ntemp) = xtemp
        Y1VIA(ntemp) = 0.0025
        K2VIA(ntemp) = 4
        X2VIA(ntemp) = xtemp
        Y2VIA(ntemp) = 0.0070
        RVIA(ntemp)  = Rwb

      ntemp = ntemp + 1
 115    continue
c puts in the vias for the wire bonds on the top side
```

```
        xtemp = 0.005d0
        do 116 i = 1, 4
        xtemp = xtemp + 0.003d0

        K1VIA(ntemp) = 3
        X1VIA(ntemp) = xtemp
        Y1VIA(ntemp) = 0.0225
        K2VIA(ntemp) = 4
        X2VIA(ntemp) = xtemp
        Y2VIA(ntemp) = 0.0180
        RVIA(ntemp)  = Rwb

      ntemp = ntemp + 1
 116    continue


* surface variables
c set resistance for the connection
            resist = 62.5
c           resist = 1000000000000.
      xtemp = 0.02055
      ytemp = 0.02305
      do 125 i = 1, 10
      KNSURF(i) = 1
      XSURF(i)  = xtemp
      YSURF(i)  = ytemp - real(i-1)*0.0025
      DXSURF(i) = 0.0014
      DYSURF(i) = 0.0014
      RSURF(i)  = resist
      TSURF(i)  = 0.0
 125  continue

        xtemp = 0.02305
        ytemp = 0.02305
        do 126 j = 1, 10
      i = j + 10
        KNSURF(i) = 1
        XSURF(i)  = xtemp
        YSURF(i)  = ytemp - real(j-1)*0.0025
        DXSURF(i) = 0.0014
        DYSURF(i) = 0.0014
        RSURF(i)  = resist
        TSURF(i)  = 0.0
 126    continue



        xtemp = 0.01805
        ytemp = 0.02305
        do 127 j = 1, 8
      i = j + 20
        KNSURF(i) = 1
        XSURF(i)  = xtemp - real(j-1)*0.0025
        YSURF(i)  = ytemp
        DXSURF(i) = 0.0014
        DYSURF(i) = 0.0014
        RSURF(i)  = resist
        TSURF(i)  = 0.0
 127    continue
```

```
        xtemp = 0.01805
        ytemp = 0.02055
        do 128 j = 1, 8
      i = j + 28
        KNSURF(i) = 1
        XSURF(i)  = xtemp - real(j-1)*0.0025
        YSURF(i)  = ytemp
        DXSURF(i) = 0.0014
        DYSURF(i) = 0.0014
        RSURF(i)  = resist
        TSURF(i)  = 0.0
 128    continue


c129                    continue

* edge variables
      KNEDGE(1) = 3
      NWEDGE(1) = 2
      PEDGE(1)  = 0.005
      DPEDGE(1) = 0.003
      REDGE(1)  = resist
      TEDGE(1)  = 0.

        KNEDGE(2) = 3
        NWEDGE(2) = 2
        PEDGE(2)  = 0.010
        DPEDGE(2) = 0.005
        REDGE(2)  = resist
        TEDGE(2)  = 0.


* DETERMINE POINTS CORRESPONDING TO EACH CONNECTION
            nscpt = 1
            nsfpt = 1
            nept  = 1
* loop over all i,j points
      do 130 i = 1, L1
      do 131 j = 1, M1

* loop vias
        do  148 n = 1, ndv
              if ( X(I)-XCV(I)/2. .LT. X1VIA(N) .AND.
     1              X(I)+XCV(I)/2. .GT. X1VIA(N) ) then

                    I1VIA(N) = I

              endif


            if (Y(J)-YCV(J)/2. .LT. Y1VIA(N) .AND.
     3             Y(J)+YCV(J)/2. .GT. Y1VIA(N)  )    then
                   J1VIA(N) = J
            endif


            if (X(1)-tiny .LT. X1VIA(N) .AND.
     1              X(1)+tiny .GT. X1VIA(N) ) then
                  I1VIA(N) = 1
            endif
```

```
            if (X(L1)-tiny .LT. X1VIA(N) .AND.
     1              X(L1)+tiny .GT. X1VIA(N) )then
                  I1VIA(N) = L1
            endif

            if (Y(1)-tiny .LT. Y1VIA(N) .AND.
     1              Y(1)+tiny .GT. Y1VIA(N) )then
                  J1VIA(N) = 1
            endif

                if (Y(M1)-tiny .LT. Y1VIA(N) .AND.
     1              Y(M1)+tiny .GT. Y1VIA(N) ) then
                       J1VIA(N) = M1
                endif




                if ( X(I)-XCV(I)/2. .LT. X2VIA(N) .AND.
     1               X(I)+XCV(I)/2. .GT. X2VIA(N) ) then

                       I2VIA(N) = I

                endif


                if (Y(J)-YCV(J)/2. .LT. Y2VIA(N) .AND.
     3             Y(J)+YCV(J)/2. .GT. Y2VIA(N)  )     then
                       J2VIA(N) = J
                endif

                tiny = .001

                if (X(1)-tiny .LT. X2VIA(N) .AND.
     1              X(1)+tiny .GT. X2VIA(N) ) then
                       I2VIA(N) = 1
                endif

                if (X(L1)-tiny .LT. X2VIA(N) .AND.
     1              X(L1)+tiny .GT. X2VIA(N) ) then
                       I2VIA(N) = L1
                endif

                if (Y(1)-tiny .LT. Y2VIA(N) .AND.
     1              Y(1)+tiny .GT. Y2VIA(N) ) then
                       J2VIA(N) = 1
                endif

                if (Y(M1)-tiny .LT. Y2VIA(N) .AND.
     1              Y(M1)+tiny .GT. Y2VIA(N) ) then
                       J2VIA(N) = M1
                endif

 148   continue


* loop sources
        do 150 n = 1, nsc
                if ( X(I) .GE. XSRC(N) .AND.
```

```
     1                X(I) .LE. XSRC(N) + DXSRC(N) .AND.
     2                Y(J) .GE. YSRC(N) .AND.
     3                Y(J) .LE. YSRC(N) + DYSRC(N)    ) then
                        ISRC(NSCPT) = I
                        JSRC(NSCPT) = J
                        KSRC(NSCPT) = KNSRC(N)
                        NSRC(NSCPT) = N
                        NSCPT = NSCPT + 1
                endif

 150    continue

* loop surfaces
        do 160 n = 1, nsf

                if ( X(I) .GE. XSURF(N) .AND.
     1               X(I) .LE. XSURF(N) + DXSURF(N) .AND.
     2               Y(J) .GE. YSURF(N) .AND.
     3               Y(J) .LE. YSURF(N) + DYSURF(N)    ) then

                  ISURF(NSFPT) = I
                        JSURF(NSFPT) = J
                        KSURF(NSFPT) = KNSURF(N)
                        NSURF(NSFPT) = N
                        NSFPT = NSFPT + 1

                endif

  160   continue

* loop edges
        do 170 n = 1, ned

                if (NWEDGE(N) .EQ. 1 .AND. I .EQ. 1) then
                if (Y(J) .GT. PEDGE(N) .AND.
     1              Y(J) .LT. PEDGE(N) + DPEDGE(N) ) then
                        IEDGE(NEPT) = 1
                        JEDGE(NEPT) = J
                        KEDGE(NEPT) = KNEDGE(N)
                        NEDGE(NEPT) = N
                        NEPT = NEPT + 1

                endif
                endif

                if (NWEDGE(N) .EQ. 2 .AND. I .EQ. L1) then
                if (Y(J) .GT. PEDGE(N) .AND.
     1              Y(J) .LT. PEDGE(N) + DPEDGE(N) ) then
                        IEDGE(NEPT) = L1
                        JEDGE(NEPT) = J
                        KEDGE(NEPT) = KNEDGE(N)
                        NEDGE(NEPT) = N
                        NEPT = NEPT + 1
                endif
                endif

                if (NWEDGE(N) .EQ. 3 .AND. J .EQ. 1) then
                if (X(I) .GT. PEDGE(N) .AND.
     1              X(I) .LT. PEDGE(N) + DPEDGE(N) ) then
                        IEDGE(NEPT) = I
```

```
                        JEDGE(NEPT) = 1
                        KEDGE(NEPT) = KNEDGE(N)
                        NEDGE(NEPT) = N
                        NEPT = NEPT + 1
                endif
                endif

                if (NWEDGE(N) .EQ. 4 .AND. J .EQ. M1) then
                if (X(I) .GT. PEDGE(N) .AND.
     1              X(I) .LT. PEDGE(N) + DPEDGE(N) ) then
                        IEDGE(NEPT) = I
                        JEDGE(NEPT) = M1
                        KEDGE(NEPT) = KNEDGE(N)
                        NEDGE(NEPT) = N
                        NEPT = NEPT + 1
                endif
                endif

 170  continue


 131  continue
 130  continue


      RETURN
C*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
      ENTRY OUTPUT



      DO 200 IUNIT=IU1,IU2
         IF(ITER.EQ.0) THEN
c output to PRINTF file
         write(iunit,202)'Htop =',Htop,'Hbot =',Hbot
         write(iunit,202)'Temp top =',Ttop,'Temp bottom =',Tbot
         write(iunit,203)'NCVLX =',CVX,'NCVLY =',CVY,'NCVLZ =',NZZ
         write(iunit,204)'Convergence =',epsi,'Max Iterations =',LAST

  202   format(2x,a16,f7.2,5x,a16,f9.2)
  203   format(/,2x,a16,i4,3x,a16,i4,3x,a16,i3)
  204   format(2x,a16,1e8.1,4x,a16,i4)

        do 206 nf = nl, 1, -1
           write(iunit,'(/,2x,a5,i2)')'Layer',nf
         write(iunit,202)'Conductivity =',cond(nf),
     +       'Thickness =', zcv(nf+1)
  206   continue

         ENDIF

  200 CONTINUE




c  calculate Qbase, Qtop to use in energy balance
         Qbase = 0.0d0
         Qtop  = 0.0d0
```

```
         do 230 j = 2,m2
         do 230 i = 2,12
            Qbase = Qbase -
     1              xcv(i)*ycvr(j)*fluxk1(i,j,1)
            Qtop  = Qtop  +
     1              xcv(i)*ycvr(j)*fluxn1(i,j,1)

  230    continue

* calculate edge heat transfer to use in energy balance
* J1 and M1 edges
       Qedge = 0.0d0
       do 232 k = 2, N2
       do 232 i = 2, L2
             if (K .NE. 4 .OR. YBLAY-tiny .LE. 0.) then
          Qedge = Qedge + rv(j)*xcv(i)*zcv(k)*FLUXJ1(i,k,1)
             endif
             if (K .NE. 4 .OR. YLAY+DYLAY+tiny .GE. Y(M1)) then
          Qedge = Qedge + rv(j)*xcv(i)*zcv(k)*FLUXM1(i,k,1)
             endif

  232 continue
* I1 and L1 edges
       do 233 k = 2, N2
       do 233 j = 2, M2
             if (K .NE. 4 .OR. XLAY-tiny .LE. 0.) then
           Qedge = Qedge + arx(j)*zcv(k)*FLUXI1(j,k,1)
             endif
             if (K .NE. 4 .OR. XLAY+DXLAY+tiny .GE. X(L1)) then
           Qedge = Qedge + arx(j)*zcv(k)*FLUXL1(j,k,1)
             endif
  233 continue


*  calculate energy balance error

         Ebal = (Qbase - Qtop - Qedge - Qgen)/Qtop

            write(*,*)'iteration,ebal'
            write(*,*)iter,ebal

* if convergence criterion has been reached then allow one more
iteration
       if (iflag .eq. 1) KSTOP = 1
         if (abs(Ebal) .LE. epsi .AND. ITER .GT. 1) iflag = 1

       IF(ITER.EQ.LAST .OR. KSTOP .EQ. 1) THEN


* reset convergence flag so execution of multiple cases works
       iflag = 0
* write out energy balance values
       write(iu1,*)
       write(iu2,*)
          write (iu1,'(5x,a,1p3g12.5)')
     1           'Qbase, Qtop, Qedge = ', Qbase, Qtop, Qedge
          write (iu2,'(5x,a,1p3g12.5)')
     1           'Qbase, Qtop, Qedge = ', Qbase, Qtop, Qedge
          write (iu1,'(5x,a,2pf12.6,/)') 'Ebal,% = ', Ebal
          write (iu2,'(5x,a,2pf12.6,/)') 'Ebal,% = ', Ebal
```

```
* write out iterations and linear counter value
      write(iu2,'(5x,a,i4,5x,a,i4)')'Iterations =',ITER, 'NTC =',NTC(1)
      write(iu2,*)' '

c  calculate avg temp for top layer of cvs
       Tavg = 0.0d0

         do 250 j = 2,m2
         do 250 i = 2,12
            Tavg = Tavg + xcv(i)*ycvr(j)*T(i,j,N1)
  250    continue
         Tavg = Tavg/(XL*YL)

* CHECK FOR MAX TEMP
* iterate over all T's on the TOP SURFACE ONLY
* and take the highest one (2nd if)
* (omitting those that are outside of the chip boundaries (1st if) )
* this does not interpolate between grid points
* K loop needed to check all temperatures in the domain *!!!!!!*
       Tmax = 0.0d0
      do 260 j = 2, M2
      do 261 i = 2, L2

        IF ( X(I).GT.XLAY.AND.X(I).LT.XLAY+DXLAY  .AND.
     +       Y(J).GT.YLAY.AND.Y(J).LT.YLAY+DYLAY ) THEN
           if ( T(i,j,N1) .GT. Tmax ) then
                 Tmax = T(i,j,N1)
                 imax = i
                 jmax = j
           endif
      ENDIF
 261  continue
 260  continue
* set kmax to N1 since only checking the top surface
* this would be determined in a k loop if the entire domain was checked
                 kmax = n1



         write (iu1,'(5x,a,1pg12.6,/)') 'Tavg (top layer) = ', Tavg
         write (iu2,'(5x,a,1pg12.6,/)') 'Tavg (top layer) = ', Tavg

         write (iu1,'(5x,a,1pg15.7,/)') 'Tmax    = ', Tmax
         write (iu2,'(5x,a,1pg15.7,/)') 'Tmax    = ', Tmax
       write (iu1,'(1x,a,i4,i4,i4)') 'Location =',imax,jmax,kmax
         write (iu2,'(1x,a,i4,i4,i4)') 'Location =',imax,jmax,kmax


* correct spacing for printer
      if (nl .eq. 3) then
            write(iu2,'(////////////////////////////////)')
      endif

      if (nl .eq. 5) then
            write(iu2,'(////////////////////////)')
      endif
```

```
* create MATLAB data files; don't include edge values
          DO 991 J=2,NCVY(1)+1
               WRITE(11,990) (T(I,J,1),I=2,NCVX(1)+1)
               WRITE(12,990) (T(I,J,2),I=2,NCVX(1)+1)
               WRITE(13,990) (T(I,J,3),I=2,NCVX(1)+1)
  990          FORMAT(1X,1P80E9.2)
  991       CONTINUE
* for K=4 and K=5 layers, limit the data to the area within the layer
boundary
            DO 993 J=18,65
               WRITE(14,990) (T(I,J,4),I=18,65)
               WRITE(15,990) (T(I,J,5),I=18,65)
  992          FORMAT(1X,1P48E9.2)
  993       CONTINUE

* output raw data to file if desired (includes all temperatures, very
large)
c         CALL PRINT


      ENDIF
      RETURN
C*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
      ENTRY PHI


* implement source, via, surface, and edge connections as source and
flux
* terms

* loop over SOURCE points
       do 400 n = 1, nscpt

* if on a boundary express as a flux
       if (KSRC(N) .EQ. 1) then
               FLXCK1(ISRC(N),JSRC(N)) = FLXCK1(ISRC(N),JSRC(N)) +
     1         QSRC(NSRC(N))  /
     2         (  DXSRC(NSRC(N))*DYSRC(NSRC(N))  )
       elseif (KSRC(N) .EQ. N1) then
               FLXCN1(ISRC(N),JSRC(N)) = FLXCN1(ISRC(N),JSRC(N)) +
     1         QSRC(NSRC(N))  /
     2         (  DXSRC(NSRC(N))*DYSRC(NSRC(N))  )
       else
* express interior locations as source terms
            SC(ISRC(N),JSRC(N),KSRC(N)) = SC(ISRC(N),JSRC(N),KSRC(N)) +
     1         QSRC(NSRC(N))  /
     2         (  DXSRC(NSRC(N))*DYSRC(NSRC(N)) * ZCV(KSRC(N))  )
       endif

 400  continue


* loop over SURFACE points
       do 410 n = 1, nsfpt
* if on a boundary express as a flux
       if (KSURF(N) .EQ. 1) then
               FLXCK1(ISURF(N),JSURF(N)) = FLXCK1(ISURF(N),JSURF(N)) +
     1         TSURF(NSURF(N)) /
     2         ( DXSURF(NSURF(N))*DYSURF(NSURF(N))*RSURF(NSURF(N)) )
               FLXPK1(ISURF(N),JSURF(N)) = FLXPK1(ISURF(N),JSURF(N)) +
```

```
     1                    (- 1.)   /
     2             ( DXSURF(NSURF(N))*DYSURF(NSURF(N))*RSURF(NSURF(N)) )

        elseif (KSURF(N) .EQ. N1) then
                FLXCN1(ISURF(N),JSURF(N)) = FLXCN1(ISURF(N),JSURF(N)) +
     1          TSURF(NSURF(N)) /
     2          ( DXSURF(NSURF(N))*DYSURF(NSURF(N))*RSURF(NSURF(N)) )
                FLXPN1(ISURF(N),JSURF(N)) = FLXPN1(ISURF(N),JSURF(N)) +
     1                  (- 1.)   /
     2          ( DXSURF(NSURF(N))*DYSURF(NSURF(N))*RSURF(NSURF(N)) )
        else
* express others as source terms
       SC(ISURF(N),JSURF(N),KSURF(N)) = SC(ISURF(N),JSURF(N),KSURF(N)) +
     1          TSURF(NSURF(N)) /
     2 (ZCV(KSURF(N))*DXSURF(NSURF(N))*DYSURF(NSURF(N))*RSURF(NSURF(N)))
       SP(ISURF(N),JSURF(N),KSURF(N)) = SP(ISURF(N),JSURF(N),KSURF(N)) +
     1                  (- 1.)    /
     2 (ZCV(KSURF(N))*DXSURF(NSURF(N))*DYSURF(NSURF(N))*RSURF(NSURF(N)))

        endif

 410    continue


* loop over EDGE points
* all connections are boundary connections, expressed as fluxes

        do 420 n = 1, nept

        if (NWEDGE(NEDGE(N)) .EQ. 1) then
                FLXCI1(JEDGE(N),KEDGE(N)) = FLXCI1(JEDGE(N),KEDGE(N)) +
     1          1. / REDGE(NEDGE(N)) * TEDGE(NEDGE(N)) /
     2          ( ZCV(KEDGE(N)) * DPEDGE(NEDGE(N)) )
                FLXPI1(JEDGE(N),KEDGE(N)) = FLXPI1(JEDGE(N),KEDGE(N)) +
     1          (- 1.) / REDGE(NEDGE(N))   /
     2          ( ZCV(KEDGE(N)) * DPEDGE(NEDGE(N)) )

        elseif (NWEDGE(NEDGE(N)) .EQ. 2) then
                FLXCL1(JEDGE(N),KEDGE(N)) = FLXCL1(JEDGE(N),KEDGE(N)) +
     1          1. / REDGE(NEDGE(N)) * TEDGE(NEDGE(N)) /
     2          ( ZCV(KEDGE(N)) * DPEDGE(NEDGE(N)) )
                FLXPL1(JEDGE(N),KEDGE(N)) = FLXPL1(JEDGE(N),KEDGE(N)) +
     1          (- 1.) / REDGE(NEDGE(N))   /
     2          ( ZCV(KEDGE(N)) * DPEDGE(NEDGE(N)) )


        elseif (NWEDGE(NEDGE(N)) .EQ. 3) then
                FLXCJ1(IEDGE(N),KEDGE(N)) = FLXCJ1(IEDGE(N),KEDGE(N)) +
     1          1. / REDGE(NEDGE(N)) * TEDGE(NEDGE(N)) /
     2          ( ZCV(KEDGE(N)) * DPEDGE(NEDGE(N)) )
                FLXPJ1(IEDGE(N),KEDGE(N)) = FLXPJ1(IEDGE(N),KEDGE(N)) +
     1          (- 1.) / REDGE(NEDGE(N))   /
     2          ( ZCV(KEDGE(N)) * DPEDGE(NEDGE(N)) )

        elseif (NWEDGE(NEDGE(N)) .EQ. 4) then
                FLXCM1(IEDGE(N),KEDGE(N)) = FLXCM1(IEDGE(N),KEDGE(N)) +
     1          1. / REDGE(NEDGE(N)) * TEDGE(NEDGE(N)) /
     2          ( ZCV(KEDGE(N)) * DPEDGE(NEDGE(N)) )
                FLXPM1(IEDGE(N),KEDGE(N)) = FLXPM1(IEDGE(N),KEDGE(N)) +
     1          (- 1.) / REDGE(NEDGE(N))   /
```

```
     2          ( ZCV(KEDGE(N)) * DPEDGE(NEDGE(N)) )

c234567890123456789012345678901234567890123456789012345678901234567890123456789012

        endif

 420  continue


* loop VIA points

        do 430 n = 1, ndv

* express via connections as fluxes if they are on a boundary
* I1 boundary
                if (I1VIA(N) .EQ. 1) then
                FLXCI1(J1VIA(N),K1VIA(N)) = FLXCI1(J1VIA(N),K1VIA(N)) +
     1          1. / RVIA(N) * T(I2VIA(N),J2VIA(N),K2VIA(N)) /
     2          (ARX(J1VIA(N))*ZCV(K1VIA(N)))
                FLXPI1(J1VIA(N),K1VIA(N)) = FLXPI1(J1VIA(N),K1VIA(N)) +
     1          (- 1.) / RVIA(N)  /
     2          (ARX(J1VIA(N))*ZCV(K1VIA(N)))
* L1 boundary
                elseif (I1VIA(N) .EQ. L1) then
                FLXCL1(J1VIA(N),K1VIA(N)) = FLXCL1(J1VIA(N),K1VIA(N)) +
     1          1. / RVIA(N) * T(I2VIA(N),J2VIA(N),K2VIA(N)) /
     2          (ARX(J1VIA(N))*ZCV(K1VIA(N)))
                FLXPL1(J1VIA(N),K1VIA(N)) = FLXPL1(J1VIA(N),K1VIA(N)) +
     1          (- 1.) / RVIA(N)  /
     2          (ARX(J1VIA(N))*ZCV(K1VIA(N)))
* J1 boundary
                elseif (J1VIA(N) .EQ. 1) then
                FLXCJ1(I1VIA(N),K1VIA(N)) = FLXCJ1(I1VIA(N),K1VIA(N)) +
     1          1. / RVIA(N) * T(I2VIA(N),J2VIA(N),K2VIA(N)) /
     2          (RV(1)*XCV(I1VIA(N))*ZCV(K1VIA(N)))
                FLXPJ1(I1VIA(N),K1VIA(N)) = FLXPJ1(J1VIA(N),K1VIA(N)) +
     1          (- 1.) / RVIA(N)  /
     2          (RV(1)*XCV(I1VIA(N))*ZCV(K1VIA(N)))
* M1 boundary
                elseif (J1VIA(N) .EQ. M1) then
                FLXCM1(I1VIA(N),K1VIA(N)) = FLXCM1(I1VIA(N),K1VIA(N)) +
     1          1. / RVIA(N) * T(I2VIA(N),J2VIA(N),K2VIA(N)) /
     2          (RV(M1)*XCV(I1VIA(N))*ZCV(K1VIA(N)))
                FLXPM1(I1VIA(N),K1VIA(N)) = FLXPM1(J1VIA(N),K1VIA(N)) +
     1          (- 1.) / RVIA(N)  /
     2          (RV(M1)*XCV(I1VIA(N))*ZCV(K1VIA(N)))
* K1 boundary
                elseif (K1VIA(N) .EQ. 1) then
                FLXCK1(I1VIA(N),J1VIA(N)) = FLXCK1(I1VIA(N),J1VIA(N)) +
     1          1. / RVIA(N) * T(I2VIA(N),J2VIA(N),K2VIA(N)) /
     2          (XCV(I1VIA(N))*YCVR(J1VIA(N)))
                FLXPK1(I1VIA(N),J1VIA(N)) = FLXPK1(J1VIA(N),K1VIA(N)) +
     1          (- 1.) / RVIA(N)  /
     2          (XCV(I1VIA(N))*YCVR(J1VIA(N)))
* N1 boundary
                elseif (K1VIA(N) .EQ. N1) then
                FLXCN1(I1VIA(N),J1VIA(N)) = FLXCN1(I1VIA(N),J1VIA(N)) +
     1          1. / RVIA(N) * T(I2VIA(N),J2VIA(N),K2VIA(N)) /
     2          (XCV(I1VIA(N))*YCVR(J1VIA(N)))
                FLXPN1(I1VIA(N),J1VIA(N)) = FLXPN1(J1VIA(N),K1VIA(N)) +
```

```
     1          (- 1.) / RVIA(N)   /
     2          (XCV(I1VIA(N))*YCVR(J1VIA(N)))
*
* now model interior points as source terms
                else
       SC(I1VIA(N),J1VIA(N),K1VIA(N)) = SC(I1VIA(N),J1VIA(N),K1VIA(N)) +
     1          1. / RVIA(N) * T(I2VIA(N),J2VIA(N),K2VIA(N)) /
     2          (XCV(I1VIA(N))*YCVR(J1VIA(N))*ZCV(K1VIA(N)))
       SP(I1VIA(N),J1VIA(N),K1VIA(N)) = SP(I1VIA(N),J1VIA(N),K1VIA(N)) +
     1          (- 1.) / RVIA(N)   /
     2          (XCV(I1VIA(N))*YCVR(J1VIA(N))*ZCV(K1VIA(N)))


                endif

* now for I2,J2,K2 wrt I1,J1,K1
* I1 boundary
                if (I2VIA(N) .EQ. 1) then
                FLXCI1(J2VIA(N),K2VIA(N)) =  FLXCI1(J2VIA(N),K2VIA(N)) +
     1          1. / RVIA(N) * T(I1VIA(N),J1VIA(N),K1VIA(N)) /
     2          (ARX(J2VIA(N))*ZCV(K2VIA(N)))
                FLXPI1(J2VIA(N),K2VIA(N)) = FLXPI1(J2VIA(N),K2VIA(N)) +
     1          (- 1.) / RVIA(N)   /
     2          (ARX(J2VIA(N))*ZCV(K2VIA(N)))
* L1 boundary
                elseif (I2VIA(N) .EQ. L1) then
                FLXCL1(J2VIA(N),K2VIA(N)) = FLXCL1(J2VIA(N),K2VIA(N)) +
     1          1. / RVIA(N) * T(I1VIA(N),J1VIA(N),K1VIA(N)) /
     2          (ARX(J2VIA(N))*ZCV(K2VIA(N)))
                FLXPL1(J2VIA(N),K2VIA(N)) = FLXPL1(J2VIA(N),K2VIA(N)) +
     1          (- 1.) / RVIA(N)   /
     2          (ARX(J2VIA(N))*ZCV(K2VIA(N)))
* J1 boundary
                elseif (J2VIA(N) .EQ. 1) then
                FLXCJ1(I2VIA(N),K2VIA(N)) = FLXCJ1(I2VIA(N),K2VIA(N)) +
     1          1. / RVIA(N) * T(I1VIA(N),J1VIA(N),K1VIA(N)) /
     2          (RV(1)*XCV(I2VIA(N))*ZCV(K2VIA(N)))
                FLXPJ1(I2VIA(N),K2VIA(N)) = FLXPJ1(J2VIA(N),K2VIA(N)) +
     1          (- 1.) / RVIA(N)   /
     2          (RV(1)*XCV(I2VIA(N))*ZCV(K2VIA(N)))
* M1 boundary
                elseif (J2VIA(N) .EQ. M1) then
                FLXCM1(I2VIA(N),K2VIA(N)) = FLXCM1(I2VIA(N),K2VIA(N)) +
     1          1. / RVIA(N) * T(I1VIA(N),J1VIA(N),K1VIA(N)) /
     2          (RV(M1)*XCV(I2VIA(N))*ZCV(K2VIA(N)))
                FLXPM1(I2VIA(N),K2VIA(N)) = FLXPM1(J2VIA(N),K2VIA(N)) +
     1          (- 1.) / RVIA(N)   /
     2          (RV(M1)*XCV(I2VIA(N))*ZCV(K2VIA(N)))
* K1 boundary
                elseif (K2VIA(N) .EQ. 1) then
                FLXCK1(I2VIA(N),J2VIA(N)) = FLXCK1(I2VIA(N),J2VIA(N)) +
     1          1. / RVIA(N) * T(I1VIA(N),J1VIA(N),K1VIA(N)) /
     2          (XCV(I2VIA(N))*YCVR(J2VIA(N)))
                FLXPK1(I2VIA(N),J2VIA(N)) = FLXPK1(J2VIA(N),K2VIA(N)) +
     1          (- 1.) / RVIA(N)   /
     2          (XCV(I2VIA(N))*YCVR(J2VIA(N)))
* N1 boundary
                elseif (K2VIA(N) .EQ. N1) then
                FLXCN1(I2VIA(N),J2VIA(N)) = FLXCN1(I2VIA(N),J2VIA(N)) +
     1          1. / RVIA(N) * T(I1VIA(N),J1VIA(N),K1VIA(N)) /
```

```
     2          (XCV(I2VIA(N))*YCVR(J2VIA(N)))
                FLXPN1(I2VIA(N),J2VIA(N)) = FLXPN1(J2VIA(N),K2VIA(N)) +
     1          (- 1.) / RVIA(N)   /
     2          (XCV(I2VIA(N))*YCVR(J2VIA(N)))
* now model interior points as source terms
                else
       SC(I2VIA(N),J2VIA(N),K2VIA(N)) = SC(I2VIA(N),J2VIA(N),K2VIA(N)) +
     1          1. / RVIA(N) * T(I1VIA(N),J1VIA(N),K1VIA(N)) /
     2          (XCV(I2VIA(N))*YCVR(J2VIA(N))*ZCV(K2VIA(N)))
       SP(I2VIA(N),J2VIA(N),K2VIA(N)) = SP(I2VIA(N),J2VIA(N),K2VIA(N)) +
     1          (- 1.) / RVIA(N)   /
     2          (XCV(I2VIA(N))*YCVR(J2VIA(N))*ZCV(K2VIA(N)))

                endif

 430    continue
******** end of connections section


c  Set conductivity for each layer
      DO 300 J=2,M2
      DO 300 I=2,L2

         GAMX(I,J,2) = cond(1)
         GAMY(I,J,2) = cond(1)
         GAMZ(I,J,2) = cond(1)


         GAMX(I,J,3) = cond(2)
         GAMY(I,J,3) = cond(2)
         GAMZ(I,J,3) = cond(2)

* make the top layer k = 0 where it does not exist
        IF ( X(I).GT.XLAY.AND.X(I).LT.XLAY+DXLAY  .AND.
     +       Y(J).GT.YLAY.AND.Y(J).LT.YLAY+DYLAY ) THEN
                GAMX(I,J,4) = cond(3)
                GAMY(I,J,4) = cond(3)
                GAMZ(I,J,4) = cond(3)
        else
                GAMX(I,J,4) = 0.0
                GAMY(I,J,4) = 0.0
                GAMZ(I,J,4) = 0.0
        endif


  300 CONTINUE

* Set edge bc in x and y - convection with Tinf = 0, insulated on one
side
* (insulated boundaries are commented out)
* if Ttop = 0 then flxc terms have no effect

      do 310 k = 2,n2
      do 310 j = 2,m2
          kbci1(j,k) = 2
          kbcl1(j,k) = 2
       flxci1(j,k) = flxci1(j,k) + Ttop
       flxcl1(j,k) = flxcl1(j,k) + Ttop
c        flxpi1(j,k) = flxpi1(j,k) - Htop
         flxpl1(j,k) = flxpl1(j,k) - Htop
```

```
  310 continue

      do 320 k = 2,n2
      do 320 i = 2,l2
          kbcj1(i,k) = 2
          kbcm1(i,k) = 2
       flxcj1(i,k) = flxcj1(i,k) + Ttop
       flxcm1(i,k) = flxcm1(i,k) + Ttop
c        flxpj1(i,k) = flxpj1(i,k) - Htop
         flxpm1(i,k) = flxpm1(i,k) - Htop
  320 continue

* Set convection from bottom (Tinf = 0), and top (Tinf = 0)

        do 330 j = 2,m2
        do 330 i = 2,l2
            kbck1(i,j) = 2
            kbcn1(i,j) = 2

* convection from top and bottom only from appropriate areas
* model convection from top of second layer as a source term
        IF ( X(I).GT.XLAY.AND.X(I).LT.XLAY+DXLAY  .AND.
     +       Y(J).GT.YLAY.AND.Y(J).LT.YLAY+DYLAY ) THEN
           flxcn1(i,j) = flxcn1(i,j) + Ttop
         flxpn1(i,j) = flxpn1(i,j) - Htop
        ELSE
           SP(i,j,3) = SP(i,j,3) - Htop / ZCV(2)
      ENDIF

* for heat sink on bottom
      IF (X(I) .LT. XLAY+DXLAY   .AND.
     +          Y(J) .LT. YLAY+DYLAY  )  THEN
      flxck1(i,j) = flxck1(i,j) + Tbot
      flxpk1(i,j) = flxpk1(i,j) - Hbot
      ENDIF

  330   CONTINUE

c     Calculate energy generated by source terms within the model
c     located here because some values change within the invariant part
c     Qgen is used in OUTPUT to check convergence
      Qgen = 0.0
        do 236 i = 2,l2
        do 235 j = 2,m2
        do 234 k = 2,n2
            Qgen = Qgen + ( sc(i,j,k) + sp(i,j,k)*T(i,j,k) )
     1          * xcv(i) * ycvr(j) * zcv(k)
  234   continue
  235   continue
  236   continue

      RETURN
      END
```

Appendix D - LAYERMOD.F

```
SUBROUTINE ADAPT
c
c  layer approximation version - see chapter 6
c
C-----------------------------------------------------------------------
C------- STEADY CONDUCTION in a 3-D layered structure
C        uniform material with no contact resistance between layers
C-----------------------------------------------------------------------
      INCLUDE 'common3d.f'
C***********************************************************************

      parameter (NL = 2)

DIMENSION                T(NI,NJ,NK),
     1 KNSRC(NSC),XSRC(NSC),YSRC(NSC),DXSRC(NSC),DYSRC(NSC),QSRC(NSC),
     3 ISRC(NPT),JSRC(NPT),KSRC(NPT),NSRC(NPT),
     4 KNSURF(NSC),XSURF(NSC),YSURF(NSC),
     5 DXSURF(NSC),DYSURF(NSC),RSURF(NSC),TSURF(NSC),
     6 ISURF(NPT),JSURF(NPT),KSURF(NPT),NSURF(NPT),
     7 KNEDGE(NED),NWEDGE(NED),PEDGE(NED),DPEDGE(NED),REDGE(NED),
     8 TEDGE(NED),IEDGE(NPT),JEDGE(NPT),KEDGE(NPT),NEDGE(NPT),
     9 K1VIA(NDV),X1VIA(NDV),Y1VIA(NDV),K2VIA(NDV),X2VIA(NDV),
     9 Y2VIA(NDV),RVIA(NDV),I1VIA(NDV),J1VIA(NDV),
     9 I2VIA(NDV),J2VIA(NDV),
     9 intemp(4), cond(NL), Tguess(NL)

      EQUIVALENCE (F(1,1,1,1),T(1,1,1))

      INTEGER CVX, CVY, CVZ
      REAL HTEDGE,HTSURF


C*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
      ENTRY GRID

      OPEN (1, FILE = 'input')
      OPEN (2, FILE = 'prtmax')


* set grid resolution
      CVX = 80
      CVY = 40
      CVZ = NL

* initial guesses
      Tguess(1) = 0.1
      Tguess(2) = 0.05
      xkneg = 16.0
      tzneg = 0.1



      HEADER ='STEADY CONDUCTION in a true 3-D Layer Ortho'
      PRINTF ='layer.out'
      XL     = 20.0d0
      YL     = 10.0d0
      ZL     =  2.0d0 + tzneg
```

```
c  zoned grid method to do the same as above:
c  set up x and y zones to match sources and different size layers
c    or holes in a layer
       NZX = 3
       XZONE(1) = 47.5d0
        NCVX(1) = ( cvx - 0.25*cvx ) / 2
       XZONE(2) = 5.0d0
        NCVX(2) = cvx - ncvx(1) - ncvx(1) + 1
       XZONE(3) = 47.5d0
        NCVX(3) = ncvx(1)

       NZY = 2
       YZONE(1) = 2.5d0
        NCVY(1) = 0.25 * cvy
       YZONE(2) = 47.5d0
        NCVY(2) = 0.75 * cvy


c  set up z grid to match material changes for each layer
       NZZ = 1
       ZZONE(1) = 1.0d0
       NCVZ(1) = NL


      CALL ZGRID

      RETURN
C*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
      ENTRY BEGIN

* set maximum number of outer iterations and convergence criterion
      LAST  = 100
      epsi  = 0.00001d0


      Qpp   = 1.0d0
      Htop = 1.0d0
      Hbot = 1.0d0
      Ttop  = 0.0d0
      Tbot  = 0.0d0
      xk    = 1.0d0
      xkz   = 1.0d0
      tz    = 2.0d0 / NL


c  set outer limit variables for source terms
c  xl = left boundary  xr = right b.  yb = bottom b.  yt = top b.
c  s1 denotes the first source
      xls1 = 47.5
      xrs1 = 52.5
      ybs1 = 0.0
      yts1 = 2.5
      XLAY  = 47.5
      YLAY  = 0.0
      DXLAY = 5.0
      DYLAY = 2.5

        KNSRC(1) = 5
        XSRC(1)  = 47.5
```

```
        YSRC(1)  = 0.0
        DXSRC(1) = 5.0
        DYSRC(1) = 2.5
        QSRC(1)  = 12.5
* (1 W/m^2 * area = 12.5)


* set contact resistances
      do 95 nf = 1, 20
            rc(nf) = 0.0
   95 continue
* for removed layer=bottom case
c     rc(2) = tzneg / xkneg
* for removed layer=middle case
c     rc(3) = tzneg / xkneg
* for removed layer on top case
      rc(nl+2) = tzneg / xkneg

* calculate modified conductivity to be used for layers
*     adjacent to "neg" layer
* to modify two layer (middle case)
c      xkmod = ( ( 1 / xk / tz + 2 / xkneg / tzneg ) /
c     1 ( 2 * tz / (xk*tz*xkneg*tzneg) ) - xk ) / 1.5  + xk
* adjust to modify one layer ( take out 2 in previous)
* (bottom or top case)
      xkmod = ( ( 1 / xk / tz + 1 / xkneg / tzneg ) /
     1 ( tz / (xk*tz*xkneg*tzneg) ) - xk ) / 1.5  + xk

* enable solution for NF=1 (temperature)
      TITLE(1)='   Temperature   '
      KSOLVE(1)=1
      KPRINT(1)=1
      KPLOT(1) =0

c  Set initial temperature level - just estimated for now
      DO 100 K=1,NL
      DO 100 J=1,M1
      DO 100 I=1,L1
         T(I,J,K)=Tguess(K)
  100 CONTINUE

* DETERMINE POINTS CORRESPONDING TO EACH CONNECTION
               nscpt = 1
               nsfpt = 1
               nept  = 1
* loop over all i,j points
       do 130 i = 1, L1
       do 131 j = 1, M1

* loop vias
       do  148 n = 1, ndv
               if ( X(I)-XCV(I)/2. .LT. X1VIA(N) .AND.
     1              X(I)+XCV(I)/2. .GT. X1VIA(N) ) then

                     I1VIA(N) = I

               endif


               if (Y(J)-YCV(J)/2. .LT. Y1VIA(N) .AND.
```

```
3              Y(J)+YCV(J)/2. .GT. Y1VIA(N)  )    then
                   J1VIA(N) = J
               endif


               if (X(1)-tiny .LT. X1VIA(N) .AND.
1                  X(1)+tiny .GT. X1VIA(N) ) then
                   I1VIA(N) = 1
               endif

               if (X(L1)-tiny .LT. X1VIA(N) .AND.
1                  X(L1)+tiny .GT. X1VIA(N) )then
                   I1VIA(N) = L1
               endif

               if (Y(1)-tiny .LT. Y1VIA(N) .AND.
1                  Y(1)+tiny .GT. Y1VIA(N) )then
                   J1VIA(N) = 1
               endif

               if (Y(M1)-tiny .LT. Y1VIA(N) .AND.
1                  Y(M1)+tiny .GT. Y1VIA(N) ) then
                   J1VIA(N) = M1
               endif




               if ( X(I)-XCV(I)/2. .LT. X2VIA(N) .AND.
1                   X(I)+XCV(I)/2. .GT. X2VIA(N) ) then

                   I2VIA(N) = I

               endif


               if (Y(J)-YCV(J)/2. .LT. Y2VIA(N) .AND.
3              Y(J)+YCV(J)/2. .GT. Y2VIA(N)  )    then
                   J2VIA(N) = J
               endif

               tiny = .001

               if (X(1)-tiny .LT. X2VIA(N) .AND.
1                  X(1)+tiny .GT. X2VIA(N) ) then
                   I2VIA(N) = 1
               endif

               if (X(L1)-tiny .LT. X2VIA(N) .AND.
1                  X(L1)+tiny .GT. X2VIA(N) ) then
                   I2VIA(N) = L1
               endif

               if (Y(1)-tiny .LT. Y2VIA(N) .AND.
1                  Y(1)+tiny .GT. Y2VIA(N) ) then
                   J2VIA(N) = 1
               endif

               if (Y(M1)-tiny .LT. Y2VIA(N) .AND.
```

```
     1              Y(M1)+tiny .GT. Y2VIA(N) ) then
                       J2VIA(N) = M1
                endif


 148    continue

* loop sources
        do 150 n = 1, nsc
                if ( X(I) .GE. XSRC(N) .AND.
     1               X(I) .LE. XSRC(N) + DXSRC(N) .AND.
     2               Y(J) .GE. YSRC(N) .AND.
     3               Y(J) .LE. YSRC(N) + DYSRC(N)    ) then
                       ISRC(NSCPT) = I
                       JSRC(NSCPT) = J
                       KSRC(NSCPT) = KNSRC(N)
                       NSRC(NSCPT) = N
                       NSCPT = NSCPT + 1
                endif

 150    continue

* loop surfaces
        do 160 n = 1, nsf

                if ( X(I) .GE. XSURF(N) .AND.
     1               X(I) .LE. XSURF(N) + DXSURF(N) .AND.
     2               Y(J) .GE. YSURF(N) .AND.
     3               Y(J) .LE. YSURF(N) + DYSURF(N)    ) then

                       ISURF(NSFPT) = I
                       JSURF(NSFPT) = J
                       KSURF(NSFPT) = KNSURF(N)
                       NSURF(NSFPT) = N
                       NSFPT = NSFPT + 1

                endif

  160   continue

* loop edges
        do 170 n = 1, ned

                if (NWEDGE(N) .EQ. 1 .AND. I .EQ. 1) then
                if (Y(J) .GT. PEDGE(N) .AND.
     1              Y(J) .LT. PEDGE(N) + DPEDGE(N) ) then
                       IEDGE(NEPT) = 1
                       JEDGE(NEPT) = J
                       KEDGE(NEPT) = KNEDGE(N)
                       NEDGE(NEPT) = N
                       NEPT = NEPT + 1

                endif
                endif

                if (NWEDGE(N) .EQ. 2 .AND. I .EQ. L1) then
                if (Y(J) .GT. PEDGE(N) .AND.
     1              Y(J) .LT. PEDGE(N) + DPEDGE(N) ) then
                       IEDGE(NEPT) = L1
                       JEDGE(NEPT) = J
```

```
                        KEDGE(NEPT) = KNEDGE(N)
                        NEDGE(NEPT) = N
                        NEPT = NEPT + 1
                endif
                endif

                if (NWEDGE(N) .EQ. 3 .AND. J .EQ. 1) then
                if (X(I) .GT. PEDGE(N) .AND.
     1              X(I) .LT. PEDGE(N) + DPEDGE(N) ) then
                        IEDGE(NEPT) = I
                        JEDGE(NEPT) = 1
                        KEDGE(NEPT) = KNEDGE(N)
                        NEDGE(NEPT) = N
                        NEPT = NEPT + 1
                endif
                endif

                if (NWEDGE(N) .EQ. 4 .AND. J .EQ. M1) then
                if (X(I) .GT. PEDGE(N) .AND.
     1              X(I) .LT. PEDGE(N) + DPEDGE(N) ) then
                        IEDGE(NEPT) = I
                        JEDGE(NEPT) = M1
                        KEDGE(NEPT) = KNEDGE(N)
                        NEDGE(NEPT) = N
                        NEPT = NEPT + 1
                endif
                endif

 170    continue


 131    continue
 130    continue



      RETURN

C*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
      ENTRY OUTPUT

      DO 200 IUNIT=IU1,IU2
         IF(ITER.EQ.0) THEN

         write(iunit,202)'CONDUCTIVITY =',xkneg,'THICKNESS =',tzneg
         write(iunit,*)' '
         write(iunit,202)'Htop =',Htop,'Hbot =',Hbot
         write(iunit,202)'Temp top =',Ttop,'Temp bottom =',Tbot
         write(iunit,203)'NCVLX =',CVX,'NCVLY =',CVY,'NCVLZ =',NL
         write(iunit,204)'Convergence =',epsi,'Max Iterations =',LAST

  202   format(2x,a16,f7.2,5x,a16,f7.2)
  203   format(/,2x,a16,i4,3x,a16,i4,3x,a16,i3)
  204   format(2x,a16,1e8.1,4x,a16,i4)

        do 206 nf = nl, 1, -1
           write(iunit,'(/,2x,a5,i2)')'Layer',nf

        if (nf.eq.nl) then
         write(iunit,202)'Conductivity =',xkmod,'Thickness =', tz
```

```
      else
         write(iunit,202)'Conductivity =',xk,'Thickness =', tz
      endif

           write(iunit,'(2x,a6,i2,a3,e12.6)')'Tavg',nf,' =',T(2,2,nf)
  206   continue

        ENDIF

  200 CONTINUE



c  calculate Qbase, Qtop ,includes heat flux input and convection out

        Qbase = 0.0d0
        Qtop  = 0.0d0
        do 230 j = 2,m2
        do 230 i = 2,12
           Qbase = Qbase -
     1             xcv(i)*ycvr(j)*fluxk1(i,j,1)
           Qtop  = Qtop  +
     1             xcv(i)*ycvr(j)*fluxn1(i,j,1)
  230   continue

* calculate edge heat transfer
      Qedge = 0.0d0
      do 232 k = 2, N2
      do 232 i = 2, L2
         Qedge = Qedge + rv(j)*xcv(i)*zcv(k)*FLUXJ1(i,k,1)
         Qedge = Qedge + rv(j)*xcv(i)*zcv(k)*FLUXM1(i,k,1)
  232 continue

      do 233 k = 2, N2
      do 233 j = 2, M2
           Qedge = Qedge + arx(j)*zcv(k)*FLUXI1(j,k,1)
           Qedge = Qedge + arx(j)*zcv(k)*FLUXL1(j,k,1)
  233 continue



*  calculate energy balance error
        Ebal = (Qbase - Qtop - Qedge)/Qtop

* test for convergence
       if (iflag .eq. 1) KSTOP = 1
       if (abs(Ebal) .LE. epsi .AND. ITER .GT. 1) iflag = 1



     IF(ITER.EQ.LAST .OR. KSTOP .EQ. 1) THEN
* reset convergence flag so execution of multiple cases works
      iflag = 0

         write (iu1,'(5x,a,1p3g12.5)')
     1          'Qbase, Qtop, Qedge = ', Qbase, Qtop, Qedge
         write (iu2,'(5x,a,1p3g12.5)')
     1          'Qbase, Qtop, Qedge = ', Qbase, Qtop, Qedge
```

```
         write (iu1,'(5x,a,2pf12.6,/)') 'Ebal,% = ', Ebal
         write (iu2,'(5x,a,2pf12.6,/)') 'Ebal,% = ', Ebal


* write out iterations and linear counter value
      write(iu2,'(5x,a,i4,5x,a,i4)')'Iterations =',ITER, 'NTC =',NTC(1)
      write(iu2,*)' '

c  fix corner temps (approx) - need edges too in 3D

c  Extapolate max temp on edge at center of source - grid specific
c   first - in j direction


      ihot = (cvx + 4) / 2

      dif2 = Y(2) - Y(1)
      dif3 = Y(3) - Y(2)
      alpha = dif3 / dif2
      delx = dif2
      f1 = T(ihot,2,N1)
      f2 = T(ihot,3,N1)
      qdp = FLUXJ1(ihot,N2,1)

* forward difference in j direction
      Tmaxj      = - qdp * delx * alpha * (1.0d0 + alpha) / xk / alpha
     1         / (alpha + 2.0d0)   +   (1.0d0 + alpha)**2 * f1 / alpha
     2         / (alpha + 2.0d0)   - f2 / alpha / (alpha + 2.0d0)


c         write (iu1,'(5x,a,1pg15.7,/)') 'Tmax,j  = ', Tmaxj
c         write (iu2,'(5x,a,1pg15.7,/)') 'Tmax,j  = ', Tmaxj

c   Next, extrapolate in z direction

* define values used in k direction

c      dif2 = Z(N2) - Z(N1)
c      dif3 = Z(N3) - Z(N2)
c      alpha = dif3 / dif2
c      delx = abs(dif2)
c      f1 = T(22,1,N2)
c      f2 = T(22,1,N3)
c      qdp = FLUXN1(22,2,1)

* backward difference in k direction
c      Tmaxk    = qdp * delx * alpha * (1.0d0 + alpha) / xkz / alpha
c     1      / (alpha + 2.0d0)   +   (1.0d0 + alpha)**2 * f1 / alpha
c     2      / (alpha + 2.0d0)   - f2 / alpha / (alpha + 2.0d0)


c         write (iu1,'(5x,a,1pg15.7,/)') 'Tmax,k  = ', Tmaxk
c         write (iu2,'(5x,a,1pg15.7,/)') 'Tmax,k  = ', Tmaxk

* average the two for the max edge temperature
c      T(22,1,N1) = (Tmaxj + Tmaxk) / 2.0d0
      T(ihot,1,N1) = Tmaxj

* interpolate Tmax accounting for Rc of missing layer
      T(ihot,1,N1) = T(ihot,1,N1) + QSRC(1)/DXSRC/DYSRC * rc(nl+2)
```

```
*  calculate avg temp for top layer of cvs (not surface)
       Tavg = 0.0d0

         do 250 j = 2,m2
         do 250 i = 2,l2
            Tavg = Tavg + xcv(i)*ycv(j)*T(i,j,N2)
  250    continue

         Tavg = Tavg/(XL*YL)
         write (iu1,'(5x,a,1pg12.6,/)') 'Tavg (top layer) = ', Tavg
         write (iu2,'(5x,a,1pg12.6,/)') 'Tavg (top layer) = ', Tavg

         write (iu1,'(5x,a,1pg15.7,/)') 'Tmax    = ', T(ihot,1,N1)
         write (iu2,'(5x,a,1pg15.7,/)') 'Tmax    = ', T(ihot,1,N1)
       write (2,'(1pg15.6)') T(ihot,1,N1)

* correct spacing for printer
      if (nl .eq. 2) then
            write(iu2,'(////////////////////////////////)')
      endif

      if (nl .eq. 4) then
            write(iu2,'(////////////////////////////)')
      endif

* calls print subroutine (prints all temps - very large output file)
c         CALL PRINT


      ENDIF
      RETURN
C*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
      ENTRY PHI

* implement source, via, surface, and edge connections as source and
flux
* terms

* loop over SOURCE points
       do 400 n = 1, nscpt

* if on a boundary express as a flux
       if (KSRC(N) .EQ. 1) then
               FLXCK1(ISRC(N),JSRC(N)) = FLXCK1(ISRC(N),JSRC(N)) +
     1         QSRC(NSRC(N))   /
     2         (  DXSRC(NSRC(N))*DYSRC(NSRC(N))   )
          elseif (KSRC(N) .EQ. N1) then
               FLXCN1(ISRC(N),JSRC(N)) = FLXCN1(ISRC(N),JSRC(N)) +
     1         QSRC(NSRC(N))   /
     2         (  DXSRC(NSRC(N))*DYSRC(NSRC(N))   )
          else
* express interior locations as source terms
            SC(ISRC(N),JSRC(N),KSRC(N)) = SC(ISRC(N),JSRC(N),KSRC(N)) +
     1         QSRC(NSRC(N))   /
     2         (  DXSRC(NSRC(N))*DYSRC(NSRC(N)) * ZCV(KSRC(N))   )
          endif

 400    continue
```

```
* loop over SURFACE points
       do 410 n = 1, nsfpt
* if on a boundary express as a flux
       if (KSURF(N) .EQ. 1) then
              FLXCK1(ISURF(N),JSURF(N)) = FLXCK1(ISURF(N),JSURF(N)) +
     1        TSURF(NSURF(N)) /
     2        ( DXSURF(NSURF(N))*DYSURF(NSURF(N))*RSURF(NSURF(N)) )
              FLXPK1(ISURF(N),JSURF(N)) = FLXPK1(ISURF(N),JSURF(N)) +
     1                (- 1.)  /
     2        ( DXSURF(NSURF(N))*DYSURF(NSURF(N))*RSURF(NSURF(N)) )

       elseif (KSURF(N) .EQ. N1) then
              FLXCN1(ISURF(N),JSURF(N)) = FLXCN1(ISURF(N),JSURF(N)) +
     1        TSURF(NSURF(N)) /
     2        ( DXSURF(NSURF(N))*DYSURF(NSURF(N))*RSURF(NSURF(N)) )
              FLXPN1(ISURF(N),JSURF(N)) = FLXPN1(ISURF(N),JSURF(N)) +
     1                (- 1.)  /
     2        ( DXSURF(NSURF(N))*DYSURF(NSURF(N))*RSURF(NSURF(N)) )
       else
* express others as source terms
      SC(ISURF(N),JSURF(N),KSURF(N)) = SC(ISURF(N),JSURF(N),KSURF(N)) +
     1        TSURF(NSURF(N)) /
     2 (ZCV(KSURF(N))*DXSURF(NSURF(N))*DYSURF(NSURF(N))*RSURF(NSURF(N)))
      SP(ISURF(N),JSURF(N),KSURF(N)) = SP(ISURF(N),JSURF(N),KSURF(N)) +
     1                (- 1.)   /
     2 (ZCV(KSURF(N))*DXSURF(NSURF(N))*DYSURF(NSURF(N))*RSURF(NSURF(N)))

       endif

 410   continue


* loop over EDGE points
* all connections are boundary connections, expressed as fluxes

       do 420 n = 1, nept

       if (NWEDGE(NEDGE(N)) .EQ. 1) then
              FLXCI1(JEDGE(N),KEDGE(N)) = FLXCI1(JEDGE(N),KEDGE(N)) +
     1        1. / REDGE(NEDGE(N)) * TEDGE(NEDGE(N)) /
     2        ( ZCV(KEDGE(N)) * DPEDGE(NEDGE(N)) )
              FLXPI1(JEDGE(N),KEDGE(N)) = FLXPI1(JEDGE(N),KEDGE(N)) +
     1        (- 1.) / REDGE(NEDGE(N))  /
     2        ( ZCV(KEDGE(N)) * DPEDGE(NEDGE(N)) )

       elseif (NWEDGE(NEDGE(N)) .EQ. 2) then
              FLXCL1(JEDGE(N),KEDGE(N)) = FLXCL1(JEDGE(N),KEDGE(N)) +
     1        1. / REDGE(NEDGE(N)) * TEDGE(NEDGE(N)) /
     2        ( ZCV(KEDGE(N)) * DPEDGE(NEDGE(N)) )
              FLXPL1(JEDGE(N),KEDGE(N)) = FLXPL1(JEDGE(N),KEDGE(N)) +
     1        (- 1.) / REDGE(NEDGE(N))  /
     2        ( ZCV(KEDGE(N)) * DPEDGE(NEDGE(N)) )


       elseif (NWEDGE(NEDGE(N)) .EQ. 3) then
              FLXCJ1(IEDGE(N),KEDGE(N)) = FLXCJ1(IEDGE(N),KEDGE(N)) +
     1        1. / REDGE(NEDGE(N)) * TEDGE(NEDGE(N)) /
     2        ( ZCV(KEDGE(N)) * DPEDGE(NEDGE(N)) )
              FLXPJ1(IEDGE(N),KEDGE(N)) = FLXPJ1(IEDGE(N),KEDGE(N)) +
     1        (- 1.) / REDGE(NEDGE(N))  /
```

```
     2           ( ZCV(KEDGE(N)) * DPEDGE(NEDGE(N)) )

        elseif (NWEDGE(NEDGE(N)) .EQ. 4) then
                FLXCM1(IEDGE(N),KEDGE(N)) = FLXCM1(IEDGE(N),KEDGE(N)) +
     1          1. / REDGE(NEDGE(N)) * TEDGE(NEDGE(N)) /
     2          ( ZCV(KEDGE(N)) * DPEDGE(NEDGE(N)) )
                FLXPM1(IEDGE(N),KEDGE(N)) = FLXPM1(IEDGE(N),KEDGE(N)) +
     1          (- 1.) / REDGE(NEDGE(N))  /
     2          ( ZCV(KEDGE(N)) * DPEDGE(NEDGE(N)) )

c234567890123456789012345678901234567890123456789012345678901234567890123456789012

        endif

 420    continue

* loop VIA points

        do 430 n = 1, ndv

* express via connections as fluxes if they are on a boundary
* I1 boundary
                if (I1VIA(N) .EQ. 1) then
                FLXCI1(J1VIA(N),K1VIA(N)) = FLXCI1(J1VIA(N),K1VIA(N)) +
     1          1. / RVIA(N) * T(I2VIA(N),J2VIA(N),K2VIA(N)) /
     2          (ARX(J1VIA(N))*ZCV(K1VIA(N)))
                FLXPI1(J1VIA(N),K1VIA(N)) = FLXPI1(J1VIA(N),K1VIA(N)) +
     1          (- 1.) / RVIA(N)  /
     2          (ARX(J1VIA(N))*ZCV(K1VIA(N)))
* L1 boundary
                elseif (I1VIA(N) .EQ. L1) then
                FLXCL1(J1VIA(N),K1VIA(N)) = FLXCL1(J1VIA(N),K1VIA(N)) +
     1          1. / RVIA(N) * T(I2VIA(N),J2VIA(N),K2VIA(N)) /
     2          (ARX(J1VIA(N))*ZCV(K1VIA(N)))
                FLXPL1(J1VIA(N),K1VIA(N)) = FLXPL1(J1VIA(N),K1VIA(N)) +
     1          (- 1.) / RVIA(N)  /
     2          (ARX(J1VIA(N))*ZCV(K1VIA(N)))
* J1 boundary
                elseif (J1VIA(N) .EQ. 1) then
                FLXCJ1(I1VIA(N),K1VIA(N)) = FLXCJ1(I1VIA(N),K1VIA(N)) +
     1          1. / RVIA(N) * T(I2VIA(N),J2VIA(N),K2VIA(N)) /
     2          (RV(1)*XCV(I1VIA(N))*ZCV(K1VIA(N)))
                FLXPJ1(I1VIA(N),K1VIA(N)) = FLXPJ1(J1VIA(N),K1VIA(N)) +
     1          (- 1.) / RVIA(N)  /
     2          (RV(1)*XCV(I1VIA(N))*ZCV(K1VIA(N)))
* M1 boundary
                elseif (J1VIA(N) .EQ. M1) then
                FLXCM1(I1VIA(N),K1VIA(N)) = FLXCM1(I1VIA(N),K1VIA(N)) +
     1          1. / RVIA(N) * T(I2VIA(N),J2VIA(N),K2VIA(N)) /
     2          (RV(M1)*XCV(I1VIA(N))*ZCV(K1VIA(N)))
                FLXPM1(I1VIA(N),K1VIA(N)) = FLXPM1(J1VIA(N),K1VIA(N)) +
     1          (- 1.) / RVIA(N)  /
     2          (RV(M1)*XCV(I1VIA(N))*ZCV(K1VIA(N)))
* K1 boundary
                elseif (K1VIA(N) .EQ. 1) then
                FLXCK1(I1VIA(N),J1VIA(N)) = FLXCK1(I1VIA(N),J1VIA(N)) +
     1          1. / RVIA(N) * T(I2VIA(N),J2VIA(N),K2VIA(N)) /
     2          (XCV(I1VIA(N))*YCVR(J1VIA(N)))
                FLXPK1(I1VIA(N),J1VIA(N)) = FLXPK1(J1VIA(N),K1VIA(N)) +
     1          (- 1.) / RVIA(N)  /
```

```
     2           (XCV(I1VIA(N))*YCVR(J1VIA(N)))
* N1 boundary
                 elseif (K1VIA(N) .EQ. N1) then
                 FLXCN1(I1VIA(N),J1VIA(N)) = FLXCN1(I1VIA(N),J1VIA(N)) +
     1           1. / RVIA(N) * T(I2VIA(N),J2VIA(N),K2VIA(N)) /
     2           (XCV(I1VIA(N))*YCVR(J1VIA(N)))
                 FLXPN1(I1VIA(N),J1VIA(N)) = FLXPN1(J1VIA(N),K1VIA(N)) +
     1           (- 1.) / RVIA(N)  /
     2           (XCV(I1VIA(N))*YCVR(J1VIA(N)))
*
* now model interior points as source terms
                 else
       SC(I1VIA(N),J1VIA(N),K1VIA(N)) = SC(I1VIA(N),J1VIA(N),K1VIA(N)) +
     1           1. / RVIA(N) * T(I2VIA(N),J2VIA(N),K2VIA(N)) /
     2           (XCV(I1VIA(N))*YCVR(J1VIA(N))*ZCV(K1VIA(N)))
       SP(I1VIA(N),J1VIA(N),K1VIA(N)) = SP(I1VIA(N),J1VIA(N),K1VIA(N)) +
     1           (- 1.) / RVIA(N)  /
     2           (XCV(I1VIA(N))*YCVR(J1VIA(N))*ZCV(K1VIA(N)))


                 endif

* now for I2,J2,K2 wrt I1,J1,K1
* I1 boundary
                 if (I2VIA(N) .EQ. 1) then
                 FLXCI1(J2VIA(N),K2VIA(N)) =  FLXCI1(J2VIA(N),K2VIA(N)) +
     1           1. / RVIA(N) * T(I1VIA(N),J1VIA(N),K1VIA(N)) /
     2           (ARX(J2VIA(N))*ZCV(K2VIA(N)))
                 FLXPI1(J2VIA(N),K2VIA(N)) = FLXPI1(J2VIA(N),K2VIA(N)) +
     1           (- 1.) / RVIA(N)  /
     2           (ARX(J2VIA(N))*ZCV(K2VIA(N)))
* L1 boundary
                 elseif (I2VIA(N) .EQ. L1) then
                 FLXCL1(J2VIA(N),K2VIA(N)) = FLXCL1(J2VIA(N),K2VIA(N)) +
     1           1. / RVIA(N) * T(I1VIA(N),J1VIA(N),K1VIA(N)) /
     2           (ARX(J2VIA(N))*ZCV(K2VIA(N)))
                 FLXPL1(J2VIA(N),K2VIA(N)) = FLXPL1(J2VIA(N),K2VIA(N)) +
     1           (- 1.) / RVIA(N)  /
     2           (ARX(J2VIA(N))*ZCV(K2VIA(N)))
* J1 boundary
                 elseif (J2VIA(N) .EQ. 1) then
                 FLXCJ1(I2VIA(N),K2VIA(N)) = FLXCJ1(I2VIA(N),K2VIA(N)) +
     1           1. / RVIA(N) * T(I1VIA(N),J1VIA(N),K1VIA(N)) /
     2           (RV(1)*XCV(I2VIA(N))*ZCV(K2VIA(N)))
                 FLXPJ1(I2VIA(N),K2VIA(N)) = FLXPJ1(J2VIA(N),K2VIA(N)) +
     1           (- 1.) / RVIA(N)  /
     2           (RV(1)*XCV(I2VIA(N))*ZCV(K2VIA(N)))
* M1 boundary
                 elseif (J2VIA(N) .EQ. M1) then
                 FLXCM1(I2VIA(N),K2VIA(N)) = FLXCM1(I2VIA(N),K2VIA(N)) +
     1           1. / RVIA(N) * T(I1VIA(N),J1VIA(N),K1VIA(N)) /
     2           (RV(M1)*XCV(I2VIA(N))*ZCV(K2VIA(N)))
                 FLXPM1(I2VIA(N),K2VIA(N)) = FLXPM1(J2VIA(N),K2VIA(N)) +
     1           (- 1.) / RVIA(N)  /
     2           (RV(M1)*XCV(I2VIA(N))*ZCV(K2VIA(N)))
* K1 boundary
                 elseif (K2VIA(N) .EQ. 1) then
                 FLXCK1(I2VIA(N),J2VIA(N)) = FLXCK1(I2VIA(N),J2VIA(N)) +
     1           1. / RVIA(N) * T(I1VIA(N),J1VIA(N),K1VIA(N)) /
     2           (XCV(I2VIA(N))*YCVR(J2VIA(N)))
```

```
                FLXPK1(I2VIA(N),J2VIA(N)) = FLXPK1(J2VIA(N),K2VIA(N)) +
     1          (- 1.) / RVIA(N)  /
     2          (XCV(I2VIA(N))*YCVR(J2VIA(N)))
* N1 boundary
                elseif (K2VIA(N) .EQ. N1) then
                FLXCN1(I2VIA(N),J2VIA(N)) = FLXCN1(I2VIA(N),J2VIA(N)) +
     1          1. / RVIA(N) * T(I1VIA(N),J1VIA(N),K1VIA(N)) /
     2          (XCV(I2VIA(N))*YCVR(J2VIA(N)))
                FLXPN1(I2VIA(N),J2VIA(N)) = FLXPN1(J2VIA(N),K2VIA(N)) +
     1          (- 1.) / RVIA(N)  /
     2          (XCV(I2VIA(N))*YCVR(J2VIA(N)))
* now model interior points as source terms
                else
       SC(I2VIA(N),J2VIA(N),K2VIA(N)) = SC(I2VIA(N),J2VIA(N),K2VIA(N)) +
     1          1. / RVIA(N) * T(I1VIA(N),J1VIA(N),K1VIA(N)) /
     2          (XCV(I2VIA(N))*YCVR(J2VIA(N))*ZCV(K2VIA(N)))
       SP(I2VIA(N),J2VIA(N),K2VIA(N)) = SP(I2VIA(N),J2VIA(N),K2VIA(N)) +
     1          (- 1.) / RVIA(N)  /
     2          (XCV(I2VIA(N))*YCVR(J2VIA(N))*ZCV(K2VIA(N)))

                endif

 430    continue
******** end of connections section


*  Set internal Gams , source terms = 0.0 (default) - Iso Gamma so far
*                      need to modify GamZ for contact R between layers

      DO 300 K=2,N2
      DO 300 J=2,M2
      DO 300 I=2,L2
         GAMX(I,J,K) = xk
         GAMY(I,J,K) = xk

* modify conductivities - IF STATEMENT SPECIFIC FOR MIDDLE CASE!
        if (k .eq. 2 .or. k .eq. 3) then
            GAMX(I,J,K) = xkmod
            GAMY(I,J,K) = xkmod
      endif

         GAMZ(I,J,K) = xkz
  300 CONTINUE

*  Set edge bc in x and y - convection with Tinf = 0, insulated on one
side
*  (insulated boundaries are commented out)
*  if Ttop = 0, then flxc terms have no effect
      do 310 k = 2,n2
      do 310 j = 2,m2
          kbci1(j,k) = 2
          kbcl1(j,k) = 2
         flxci1(j,k) = flxci1(j,k) + Ttop
         flxcl1(j,k) = flxcl1(j,k) + Ttop
c        flxpi1(j,k) = flxpi1(j,k) - Htop
         flxpl1(j,k) = flxpl1(j,k) - Htop
  310 continue

      do 320 k = 2,n2
```

```
      do 320 i = 2,12
          kbcj1(i,k) = 2
          kbcm1(i,k) = 2
         flxcj1(i,k) = flxcj1(i,k) + Ttop
         flxcm1(i,k) = flxcm1(i,k) + Ttop
c        flxpj1(i,k) = flxpj1(i,k) - Htop
         flxpm1(i,k) = flxpm1(i,k) - Htop
  320 continue

c  Set convection from bottom (Tinf = 0), and top (Tinf = 0)

        do 330 j = 2,m2
        do 330 i = 2,12
            kbck1(i,j) = 2
            kbcn1(i,j) = 2
           flxpk1(i,j) = flxpk1(i,j) - Hbot
           flxpn1(i,j) = flxpn1(i,j) - Htop
  330   CONTINUE

c  Add heat flux ( = 1) from devices on top layer
c    in center 11x5 cvs in 41x20 grid,
c    turn off convection where device is attached

        do 340 j = 2, m2
        do 340 i = 2, 12

        IF ( X(I).GT.XLAY.AND.X(I).LT.XLAY+DXLAY  .AND.
     +       Y(J).GT.YLAY.AND.Y(J).LT.YLAY+DYLAY ) THEN
           flxcn1(i,j) = Qpp
           flxpn1(i,j) = 0.0d0
      ENDIF

  340   CONTINUE

* adjust top and bottom heat fluxes for rc added on top or bottom
* for Tinf = 0
        do 350 j = 2,m2
        do 350 i = 2,12
            kbck1(i,j) = 2
            kbcn1(i,j) = 2
           flxpk1(i,j) = flxpk1(i,j) - Hbot
           flxpn1(i,j) = flxpn1(i,j) - 1 / ( 1/Htop + rc(nl + 2) )

        IF ( X(I).GT.XLAY.AND.X(I).LT.XLAY+DXLAY  .AND.
     +       Y(J).GT.YLAY.AND.Y(J).LT.YLAY+DYLAY ) THEN

           flxcn1(i,j) = flxcn1(i,j) + 0.0d0
           flxpn1(i,j) = flxpn1(i,j) + 0.0d0

        ENDIF


  350   CONTINUE


      RETURN
      END
```

Appendix E - DEFRD3D.F

```
SUBROUTINE DEFRD
c  3D version - DP version - Orthotropic Gamma
C*********************************************************************
      INCLUDE 'common3d.f'
C*********************************************************************
C
      ENTRY DEFLT
C
COME HERE TO SET THE DEFAULT VALUES
C
      HEADER='USE THE CHARACTER VARIABLE HEADER TO SPECIFY A PROBLEM TIT
     1LE'
      PRINTF='PRINT.OUT'
      PLOTF ='PLOT.OUT'
      KSTOP=0
      LAST =5
      ITER =0
      KORD =2
      MODE =1
      KPGR =1
      KOUT =3
      SMALL=1.d-32
      BIG  =1.d+32
      TIME =0.0d0
      DT   =BIG
      R(1) =0.0d0
      POWERX=1.0d0
      POWERY=1.0d0
      POWERZ=1.0d0

      DO 10 NZ=1,NZMAX
         POWRX(NZ)=1.0d0
         POWRY(NZ)=1.0d0
   10    POWRZ(NZ)=1.0d0

      DO 20 NF=1,NFMAX
         CRIT(NF)  =1.d-5
         KSOLVE(NF)=0
         NTIMES(NF)=10
         KBLOC(NF) =1
         RELAX(NF) =1.0d0
         TITLE(NF) ='                    '
         KPRINT(NF)=0
         KPLOT(NF) =0

         DO 30 K=2,NK
         DO 30 I=2,NI
            FLUXJ1(I,K,NF)=0.0d0
   30       FLUXM1(I,K,NF)=0.0d0

         DO 40 K=2,NK
         DO 40 J=2,NJ
            FLUXI1(J,K,NF)=0.0d0
   40       FLUXL1(J,K,NF)=0.0d0

         DO 42 J=2,NJ
         DO 42 I=2,NI
```

```
            FLUXK1(I,J,NF)=0.0d0
   42       FLUXN1(I,J,NF)=0.0d0

   20 CONTINUE

      DO 50 K=1,NK
      DO 50 J=1,NJ
      DO 50 I=1,NI
         CON(I,J,K)    =0.0d0
          AP(I,J,K)    =0.0d0
        ALAM(I,J,K)    =1.0d0

c  Orthotropic Gamma 3D

        GAMX(I,J,K)    =1.0d0
        GAMY(I,J,K)    =1.0d0
        GAMZ(I,J,K)    =1.0d0

C     IBLOCK(I,J,K)    =0

         DO 60 NF=1,NFMAX
   60       F(I,J,K,NF) =0.0d0

   50 CONTINUE

      DO 70 K=2,NK
      DO 70 I=2,NI
          KBCJ1(I,K) =1
          KBCM1(I,K) =1
         FLXCJ1(I,K)=0.0d0
         FLXCM1(I,K)=0.0d0
         FLXPJ1(I,K)=0.0d0
         FLXPM1(I,K)=0.0d0
   70 CONTINUE

      DO 80 K=2,NK
      DO 80 J=2,NJ
          KBCI1(J,K) =1
          KBCL1(J,K) =1
         FLXCI1(J,K)=0.0d0
         FLXCL1(J,K)=0.0d0
         FLXPI1(J,K)=0.0d0
         FLXPL1(J,K)=0.0d0
   80 CONTINUE

      DO 90 J=2,NJ
      DO 90 I=2,NI
          KBCK1(I,J) =1
          KBCN1(I,J) =1
         FLXCK1(I,J)=0.0d0
         FLXCN1(I,J)=0.0d0
         FLXPK1(I,J)=0.0d0
         FLXPN1(I,J)=0.0d0
   90 CONTINUE
C
      RETURN
C*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
      ENTRY READY
C
      IF(KOUT.NE.1) OPEN(UNIT=7,FILE=PRINTF)
```

```
      IU1=6
      IF(KOUT.EQ.2) IU1=7
      IU2=7
      IF(KOUT.EQ.1) IU2=6
CREATE INITIAL OUTPUT
      DO 100 IUNIT=IU1,IU2
C
c      IF(MODE.EQ.1) WRITE(IUNIT,1)
    1 FORMAT(1X,'RESULTS OF Ortho-3D FOR CARTESIAN COORD SYSTEM'
     1/1X,50(1H*)//)
c Note that MODE=2 is not valid for this 3D version - use MODE=3
      IF(MODE.EQ.2) WRITE(IUNIT,2)
    2 FORMAT(1X,'Do not use Mode=2 in CONDUCT-3D, use Mode=3 instead'
     1/1X,53(1H*)//)
      IF(MODE.EQ.3) WRITE(IUNIT,3)
    3 FORMAT(1X,'RESULTS OF Ortho-3D FOR POLAR-Z COORDINATE SYSTEM'
     1/1X,46(1H*)//)
      WRITE(IUNIT,5) HEADER
    5 FORMAT(1X,64('-')/1X,A64/1X,64('-')//)
      IF  (L1.GT.NI .OR. M1.GT.NJ .OR. N1.GT.NK
     $ .OR.L1.LT.4  .OR. M1.LT.4  .OR. N1.LT.4) THEN
        WRITE(IUNIT,6)
    6 FORMAT(1X,'EXECUTION TERMINATED DUE TO ONE(OR MORE) OF THE FOLLOWI
     1NG REASON(S)',/2X,'1) L1,M1,N1 GREATER THAN NI,NJ,NK',/2X,
     2                  '2) L1,M1,N1 LESS THAN 4',/)
        KSTOP=1
      ELSE IF (MODE .EQ. 2) THEN
        KSTOP=1
      ENDIF

  100 CONTINUE

      IF(KSTOP.NE.0) STOP ' Error in READY'
CALCULATE GEOMETRICAL QUANTITIES
      L2=L1-1
      L3=L2-1
      M2=M1-1
      M3=M2-1
      N2=N1-1
      N3=N2-1
      X(1)=XU(2)

      DO 110 I=2,L2
  110    X(I)=0.5d0*(XU(I+1)+XU(I))

      X(L1)=XU(L1)
      Y(1) =YV(2)

      DO 120 J=2,M2
  120    Y(J)=0.5d0*(YV(J+1)+YV(J))

      Y(M1)=YV(M1)
      Z(1) =ZW(2)

      DO 122 K=2,N2
  122    Z(K)=0.5d0*(ZW(K+1)+ZW(K))

      Z(N1)=ZW(N1)

      DO 130 I=2,L2
```

```
  130    XCV(I)=XU(I+1)-XU(I)
      DO 140 J=2,M2
  140    YCV(J)=YV(J+1)-YV(J)
      DO 142 K=2,N2
  142    ZCV(K)=ZW(K+1)-ZW(K)

      IF(MODE.EQ.1) THEN
        DO 150 J=1,M1
           RV(J)=1.0d0
  150      R(J) =1.0d0
      ELSE
        RY1=R(1)-Y(1)
        DO 160 J=2,M1
  160      R(J)=Y(J)+RY1
        RV(2)=R(1)
        DO 170 J=3,M2
  170      RV(J)=RV(J-1)+YCV(J-1)
        RV(M1)=R(M1)
      ENDIF

      IF(MODE.EQ.3) THEN
        DO 180 J=1,M1
  180      SX(J)=R(J)
      ELSE
        DO 190 J=1,M1
  190      SX(J)=1.0d0
      ENDIF

      DO 200 J=2,M2
         YCVR(J)=R(J)*YCV(J)
         IF(MODE.EQ.3) THEN
           ARX(J)=YCV(J)
         ELSE
           ARX(J)=YCVR(J)
         ENDIF
  200 CONTINUE
C
      RETURN
      END
```

Appendix F - HEART3D.F

```
SUBROUTINE HEART
c  3D version - HP f77 - dp version, Orthotropic Gamma
C*********************************************************************
      INCLUDE 'common3d.f'
C*********************************************************************
CONSTRUCT LOOP FOR ALL EQUATIONS
      DO 999 NF=1,NFMAX
c     NF=N
      IF(KSOLVE(NF).EQ.0) GO TO 999
C
      CALL PHI
C
CALCULATE COEFFICIENTS IN THE DISCRETIZATION EQUATION
C
      BETA=4.d0/3.d0
      IF(KORD.EQ.1) BETA=1.0
      RLX=(1.0d0-RELAX(NF))/RELAX(NF)
CONSIDER VOLUMETRIC TERMS
      DO 10 K=2,N2
      DO 10 J=2,M2
      DO 10 I=2,L2
         VOL=YCVR(J)*XCV(I)*ZCV(K)
         APT=ALAM(I,J,K)/DT
         CON(I,J,K)=(CON(I,J,K)+APT*F(I,J,K,NF))*VOL
          AP(I,J,K)=(APT-AP(I,J,K))*VOL
   10 CONTINUE

COEFFICIENTS FOR X-DIRECTION DIFFUSION
      DO 20 K=2,N2
      DO 20 J=2,M2
      DO 20 I=2,L3
         DIFF=ARX(J)*ZCV(K)*2.d0*GAMX(I,J,K)*GAMX(I+1,J,K)/
     1        ((XCV(I)*GAMX(I+1,J,K)+XCV(I+1)*GAMX(I,J,K)+SMALL)*SX(J))
         AIP(I,J,K)=DIFF+SMALL
         AIM(I+1,J,K)=AIP(I,J,K)
   20 CONTINUE

      DO 30 K=2,N2
      DO 30 J=2,M2
CONSIDER I=1 BOUNDARY
         DIFF=GAMX(2,J,K)/(0.5d0*XCV(2)*SX(J))+SMALL
         AIM(2,J,K)=BETA*DIFF
         AIP(1,J,K)=AIM(2,J,K)
         AIM(2,J,K)=AIM(2,J,K)*ARX(J)*ZCV(K)
         AIM(1,J,K)=(BETA-1.d0)*AIP(2,J,K)/(ARX(J)*ZCV(K))
         AIP(2,J,K)=AIP(2,J,K)+AIM(1,J,K)*ARX(J)*ZCV(K)
         IF(KBCI1(J,K).EQ.1) THEN
           CON(2,J,K)=CON(2,J,K)+AIM(2,J,K)*F(1,J,K,NF)
         ELSE
            AP(1,J,K)=AIP(1,J,K)-FLXPI1(J,K)
           CON(1,J,K)=FLXCI1(J,K)
           TEMP=AIM(2,J,K)/AP(1,J,K)
            AP(2,J,K) =AP(2,J,K)-AIP(1,J,K)*TEMP
           AIP(2,J,K)=AIP(2,J,K)-AIM(1,J,K)*TEMP
           CON(2,J,K)=CON(2,J,K)+CON(1,J,K)*TEMP
         ENDIF
          AP(2,J,K)=AP(2,J,K)+AIM(2,J,K)
```

```
          AIM(2,J,K)=0.0d0
CONSIDER I=L1 BOUNDARY
          DIFF=GAMX(L2,J,K)/(0.5d0*XCV(L2)*SX(J))+SMALL
          AIP(L2,J,K)=BETA*DIFF
          AIM(L1,J,K)=AIP(L2,J,K)
          AIP(L2,J,K)=AIP(L2,J,K)*ARX(J)*ZCV(K)
          AIP(L1,J,K)=(BETA-1.d0)*AIM(L2,J,K)/(ARX(J)*ZCV(K))
          AIM(L2,J,K)=AIM(L2,J,K)+AIP(L1,J,K)*ARX(J)*ZCV(K)
          IF(KBCL1(J,K).EQ.1) THEN
            CON(L2,J,K)=CON(L2,J,K)+AIP(L2,J,K)*F(L1,J,K,NF)
          ELSE
             AP(L1,J,K)=AIM(L1,J,K)-FLXPL1(J,K)
            CON(L1,J,K)=FLXCL1(J,K)
            TEMP=AIP(L2,J,K)/AP(L1,J,K)
             AP(L2,J,K) =AP(L2,J,K)-AIM(L1,J,K)*TEMP
            AIM(L2,J,K)=AIM(L2,J,K)-AIP(L1,J,K)*TEMP
            CON(L2,J,K)=CON(L2,J,K)+CON(L1,J,K)*TEMP
          ENDIF
           AP(L2,J,K)=AP(L2,J,K)+AIP(L2,J,K)
          AIP(L2,J,K)=0.0d0
   30 CONTINUE

COEFFICIENTS FOR Y-DIRECTION DIFFUSION
      DO 40 K=2,N2
      DO 40 J=2,M3
      DO 40 I=2,L2
          AREA=RV(J+1)*XCV(I)*ZCV(K)
          DIFF=AREA*2.d0*GAMY(I,J,K)*GAMY(I,J+1,K)/(YCV(J)*GAMY(I,J+1,K)+
     1         YCV(J+1)*GAMY(I,J,K)+SMALL)
          AJP(I,J,K)=DIFF+SMALL
          AJM(I,J+1,K)=AJP(I,J,K)
   40 CONTINUE

      DO 50 K=2,N2
      DO 50 I=2,L2
CONSIDER J=1 BOUNDARY
          AREA=RV(2)*XCV(I)*ZCV(K)
          DIFF=GAMY(I,2,K)/(0.5d0*YCV(2))+SMALL
          AJM(I,2,K)=BETA*DIFF
          AJP(I,1,K)=AJM(I,2,K)
          AJM(I,2,K)=AJM(I,2,K)*AREA
          AJM(I,1,K)=(BETA-1.d0)*AJP(I,2,K)/(RV(3)*XCV(I)*ZCV(K))
          AJP(I,2,K)=AJP(I,2,K)+AJM(I,1,K)*AREA
          IF(KBCJ1(I,K).EQ.1) THEN
            CON(I,2,K)=CON(I,2,K)+AJM(I,2,K)*F(I,1,K,NF)
          ELSE
             AP(I,1,K)=AJP(I,1,K)-FLXPJ1(I,K)
            CON(I,1,K)=FLXCJ1(I,K)
            TEMP=AJM(I,2,K)/AP(I,1,K)
             AP(I,2,K)= AP(I,2,K)-AJP(I,1,K)*TEMP
            AJP(I,2,K)=AJP(I,2,K)-AJM(I,1,K)*TEMP
            CON(I,2,K)=CON(I,2,K)+CON(I,1,K)*TEMP
          ENDIF
           AP(I,2,K)=AP(I,2,K)+AJM(I,2,K)
          AJM(I,2,K)=0.0d0
CONSIDER J=M1 BOUNDARY
          AREA=RV(M1)*XCV(I)*ZCV(K)
          DIFF=GAMY(I,M2,K)/(0.5d0*YCV(M2))+SMALL
          AJP(I,M2,K)=BETA*DIFF
          AJM(I,M1,K)=AJP(I,M2,K)
```

```
          AJP(I,M2,K)=AJP(I,M2,K)*AREA
          AJP(I,M1,K)=(BETA-1.d0)*AJM(I,M2,K)/(RV(M2)*XCV(I)*ZCV(K))
          AJM(I,M2,K)=AJM(I,M2,K)+AJP(I,M1,K)*AREA
          IF(KBCM1(I,K).EQ.1) THEN
            CON(I,M2,K)=CON(I,M2,K)+AJP(I,M2,K)*F(I,M1,K,NF)
          ELSE
             AP(I,M1,K)=AJM(I,M1,K)-FLXPM1(I,K)
            CON(I,M1,K)=FLXCM1(I,K)
            TEMP=AJP(I,M2,K)/AP(I,M1,K)
             AP(I,M2,K)= AP(I,M2,K)-AJM(I,M1,K)*TEMP
            AJM(I,M2,K)=AJM(I,M2,K)-AJP(I,M1,K)*TEMP
            CON(I,M2,K)=CON(I,M2,K)+CON(I,M1,K)*TEMP
          ENDIF
           AP(I,M2,K)=AP(I,M2,K)+AJP(I,M2,K)
          AJP(I,M2,K)=0.0d0
   50 CONTINUE

*************
* contact resistance added here (RC)
COEFFICIENTS FOR Z-DIRECTION DIFFUSION
      DO 45 K=2,N3
      DO 45 J=2,M2
      DO 45 I=2,L2
          AREA=YCVR(J)*XCV(I)
          DIFF=AREA*2.d0*GAMZ(I,J,K)*GAMZ(I,J,K+1)/(ZCV(K)*GAMZ(I,J,K+1)+
     1       ZCV(K+1)*GAMZ(I,J,K)
     2       + 2.d0*RC(K+1)*GAMZ(I,J,K)*GAMZ(I,J,K+1)+SMALL)
          AKP(I,J,K)=DIFF+SMALL
          AKM(I,J,K+1)=AKP(I,J,K)



   45 CONTINUE


      DO 56 J=2,M2
      DO 56 I=2,L2
CONSIDER K=1 BOUNDARY
          AREA=YCVR(J)*XCV(I)
          DIFF=GAMZ(I,J,2)/(0.5d0*ZCV(2))+SMALL
          AKM(I,J,2)=BETA*DIFF
          AKP(I,J,1)=AKM(I,J,2)
          AKM(I,J,2)=AKM(I,J,2)*AREA
          AKM(I,J,1)=(BETA-1.d0)*AKP(I,J,2)/AREA
          AKP(I,J,2)=AKP(I,J,2)+AKM(I,J,1)*AREA
          IF(KBCK1(I,J).EQ.1) THEN
            CON(I,J,2)=CON(I,J,2)+AKM(I,J,2)*F(I,J,1,NF)
          ELSE
             AP(I,J,1)=AKP(I,J,1)-FLXPK1(I,J)
            CON(I,J,1)=FLXCK1(I,J)
            TEMP=AKM(I,J,2)/AP(I,J,1)
             AP(I,J,2)= AP(I,J,2)-AKP(I,J,1)*TEMP
            AKP(I,J,2)=AKP(I,J,2)-AKM(I,J,1)*TEMP
            CON(I,J,2)=CON(I,J,2)+CON(I,J,1)*TEMP
          ENDIF
           AP(I,J,2)=AP(I,J,2)+AKM(I,J,2)
          AKM(I,J,2)=0.0d0
```

```
CONSIDER K=N1 BOUNDARY
          AREA=YCVR(J)*XCV(I)
          DIFF=GAMZ(I,J,N2)/(0.5d0*ZCV(N2))+SMALL
          AKP(I,J,N2)=BETA*DIFF
          AKM(I,J,N1)=AKP(I,J,N2)
          AKP(I,J,N2)=AKP(I,J,N2)*AREA
          AKP(I,J,N1)=(BETA-1.d0)*AKM(I,J,N2)/AREA
          AKM(I,J,N2)=AKM(I,J,N2)+AKP(I,J,N1)*AREA
          IF(KBCN1(I,J).EQ.1) THEN
            CON(I,J,N2)=CON(I,J,N2)+AKP(I,J,N2)*F(I,J,N1,NF)
          ELSE
             AP(I,J,N1)=AKM(I,J,N1)-FLXPN1(I,J)
            CON(I,J,N1)=FLXCN1(I,J)
            TEMP=AKP(I,J,N2)/AP(I,J,N1)
             AP(I,J,N2)= AP(I,J,N2)-AKM(I,J,N1)*TEMP
            AKM(I,J,N2)=AKM(I,J,N2)-AKP(I,J,N1)*TEMP
            CON(I,J,N2)=CON(I,J,N2)+CON(I,J,N1)*TEMP
          ENDIF
           AP(I,J,N2)=AP(I,J,N2)+AKP(I,J,N2)
          AKP(I,J,N2)=0.0d0
   56 CONTINUE



COME HERE TO INTRODUCE UNDERRELAXATION
CONSTRUCT AP(I,J,K) AND CON(I,J,K) IN THEIR FINAL FORM
      DO 60 K=2,N2
      DO 60 J=2,M2
      DO 60 I=2,L2
         ANB=AIP(I,J,K)+AIM(I,J,K)+AJP(I,J,K)+AJM(I,J,K)
     1                             +AKP(I,J,K)+AKM(I,J,K)
         AINR=ANB*RLX
          AP(I,J,K)= AP(I,J,K)+ANB+AINR
         CON(I,J,K)=CON(I,J,K)+AINR*F(I,J,K,NF)
   60 CONTINUE
C
CALL THE SOLVE ROUTINE TO OBTAIN THE SOLUTION OF THE DISCRETIZATION
C  EQUATIONS
C
      CALL SOLVE
  999 CONTINUE
C
      TIME=TIME+DT
      ITER=ITER+1
      IF(ITER.GE.LAST) KSTOP=1

      RETURN
      END
```

Appendix G - SOLVE3D.F

```
SUBROUTINE SOLVE
c  3D version - HP f77 - dp version
C*********************************************************************
      INCLUDE 'common3d.f'
      DIMENSION RT(8)
C*********************************************************************
      BIG1  =1.d+10
      SMALL1=1.0d-5
      LL2   =2*L2
      LL    =LL2-2
      MM2   =2*M2
      MM    =MM2-2
      NN2   =2*N2
      NN    =NN2-2
c     N     =NF
      NTM   =NTIMES(NF)

      DO 999 NT=1,NTM
      NTT =NT
      ICON=1
COME HERE TO PERFORM THE I-DIRECTION BLOCK CORRECTION
C---------------------------------------------------------------------
      PTX(1)=0.0d0
      QTX(1)=0.0d0

      DO 10 I=2,L2
         BL=SMALL
         BLP=0.0d0
         BLM=0.0d0
         BLC=0.0d0

         DO  20 K=2,N2
         DO  20 J=2,M2
             IF(AP(I,J,K).LT.BIG1) THEN
               BL=BL+AP(I,J,K)
               IF(AP(I,J+1,K).LT.BIG1) BL =BL -AJP(I,J,K)
               IF(AP(I,J-1,K).LT.BIG1) BL =BL -AJM(I,J,K)
               IF(AP(I,J,K+1).LT.BIG1) BL =BL -AKP(I,J,K)
               IF(AP(I,J,K-1).LT.BIG1) BL =BL -AKM(I,J,K)
               IF(AP(I+1,J,K).LT.BIG1) BLP=BLP+AIP(I,J,K)
               IF(AP(I-1,J,K).LT.BIG1) BLM=BLM+AIM(I,J,K)
CONVERGENCE CRITERION FOR THE SOLUTION ROUTINE
               RT(1)=AIP(I,J,K)*F(I+1,J,K,NF)
               RT(2)=AIM(I,J,K)*F(I-1,J,K,NF)
               RT(3)=AJP(I,J,K)*F(I,J+1,K,NF)
               RT(4)=AJM(I,J,K)*F(I,J-1,K,NF)
               RT(5)=AKP(I,J,K)*F(I,J,K+1,NF)
               RT(6)=AKM(I,J,K)*F(I,J,K-1,NF)
               RT(7)=-AP(I,J,K)*F(I,J,K,NF)
               RT(8)=CON(I,J,K)
               RES=0.0d0
               TERM=1.0d-8

               DO 30 IRT=1,8
                  RES=RES+RT(IRT)
   30             TERM=MAX(TERM,ABS(RT(IRT)))
```

```
              IF(ABS(RES/TERM) .GT. CRIT(NF)) ICON=0
              BLC=BLC+RES

           ENDIF
   20    CONTINUE

         DENOM=BL-PTX(I-1)*BLM
         IF(ABS(DENOM/BL).LT.SMALL1) DENOM=BIG
         PTX(I)=BLP/DENOM
         QTX(I)=(BLC+BLM*QTX(I-1))/DENOM
   10 CONTINUE

      IF(NTT.NE.1.AND.ICON.EQ.1) GO TO 990
      IF(KBLOC(NF).EQ.0) GO TO 80
      BL=0.0d0

      DO 40 I=L2,2,-1
         BL=BL*PTX(I)+QTX(I)

         DO 40 K=2,N2
         DO 40 J=2,M2
            IF(AP(I,J,K).LT.BIG1) F(I,J,K,NF)=F(I,J,K,NF)+BL
   40 CONTINUE
COME HERE TO PERFORM THE J-DIRECTION BLOCK CORRECTION
C-----------------------------------------------------------------------
      PTY(1)=0.0d0
      QTY(1)=0.0d0

      DO 50 J=2,M2
         BL=SMALL
         BLP=0.0d0
         BLM=0.0d0
         BLC=0.0d0

         DO 60  K=2,N2
         DO 60  I=2,L2
            IF(AP(I,J,K).LT.BIG1) THEN
            BL=BL+AP(I,J,K)
            IF(AP(I+1,J,K).LT.BIG1) BL =BL -AIP(I,J,K)
            IF(AP(I-1,J,K).LT.BIG1) BL =BL -AIM(I,J,K)
            IF(AP(I,J,K+1).LT.BIG1) BL =BL -AKP(I,J,K)
            IF(AP(I,J,K-1).LT.BIG1) BL =BL -AKM(I,J,K)
            IF(AP(I,J+1,K).LT.BIG1) BLP=BLP+AJP(I,J,K)
            IF(AP(I,J-1,K).LT.BIG1) BLM=BLM+AJM(I,J,K)
            BLC=BLC + CON(I,J,K)                - AP(I,J,K)*F(I, J, K,NF)
     1              + AIP(I,J,K)*F(I+1,J,K,NF) +AIM(I,J,K)*F(I-1,J,K,NF)
     2              + AJP(I,J,K)*F(I,J+1,K,NF) +AJM(I,J,K)*F(I,J-1,K,NF)
     3              + AKP(I,J,K)*F(I,J,K+1,NF) +AKM(I,J,K)*F(I,J,K-1,NF)

            ENDIF
   60    CONTINUE

         DENOM=BL-PTY(J-1)*BLM
         IF(ABS(DENOM/BL).LT.SMALL1) DENOM=BIG
         PTY(J)=BLP/DENOM
         QTY(J)=(BLC+BLM*QTY(J-1))/DENOM
   50 CONTINUE

      IF(KBLOC(NF).EQ.0) GO TO 82
```

```
      BL=0.0d0

      DO 70 J=M2,2,-1
         BL=BL*PTY(J)+QTY(J)
         DO 70 K=2,N2
         DO 70 I=2,L2
            IF(AP(I,J,K).LT.BIG1) F(I,J,K,NF)=F(I,J,K,NF)+BL
   70 CONTINUE

   80 CONTINUE
COME HERE TO PERFORM THE K-DIRECTION BLOCK CORRECTION
C---------------------------------------------------------------------
      PTZ(1)=0.0d0
      QTZ(1)=0.0d0

      DO 52 K=2,N2
         BL=SMALL
         BLP=0.0d0
         BLM=0.0d0
         BLC=0.0d0

         DO 62  J=2,M2
         DO 62  I=2,L2
            IF(AP(I,J,K).LT.BIG1) THEN
            BL=BL+AP(I,J,K)
            IF(AP(I+1,J,K).LT.BIG1) BL =BL -AIP(I,J,K)
            IF(AP(I-1,J,K).LT.BIG1) BL =BL -AIM(I,J,K)
            IF(AP(I,J+1,K).LT.BIG1) BL =BL -AJP(I,J,K)
            IF(AP(I,J-1,K).LT.BIG1) BL =BL -AJM(I,J,K)
            IF(AP(I,J,K+1).LT.BIG1) BLP=BLP+AKP(I,J,K)
            IF(AP(I,J,K-1).LT.BIG1) BLM=BLM+AKM(I,J,K)
            BLC=BLC + CON(I,J,K)                - AP(I,J,K)*F(I, J, K,NF)
     1              + AIP(I,J,K)*F(I+1,J,K,NF) +AIM(I,J,K)*F(I-1,J,K,NF)
     2              + AJP(I,J,K)*F(I,J+1,K,NF) +AJM(I,J,K)*F(I,J-1,K,NF)
     3              + AKP(I,J,K)*F(I,J,K+1,NF) +AKM(I,J,K)*F(I,J,K-1,NF)

            ENDIF
   62    CONTINUE

         DENOM=BL-PTZ(K-1)*BLM
         IF(ABS(DENOM/BL).LT.SMALL1) DENOM=BIG
         PTZ(K)=BLP/DENOM
         QTZ(K)=(BLC+BLM*QTZ(K-1))/DENOM
   52 CONTINUE

      BL=0.0d0

      DO 72 K=N2,2,-1
         BL=BL*PTZ(K)+QTZ(K)
         DO 72 J=2,M2
         DO 72 I=2,L2
            IF(AP(I,J,K).LT.BIG1) F(I,J,K,NF)=F(I,J,K,NF)+BL
   72 CONTINUE

   82 CONTINUE

CARRY OUT THE I-DIRECTION TDMA
C---------------------------------------------------------------------
      DO 90 KK=2,NN
       K=MIN(KK,NN2-KK)
```

```
       DO 90 JJ=2,MM
         J=MIN(JJ,MM2-JJ)
         PTX(1)=0.0d0
         QTX(1)=0.0d0

         DO 100 I=2,L2
            DENOM=AP(I,J,K)-PTX(I-1)*AIM(I,J,K)
            PTX(I)=AIP(I,J,K)/DENOM
            TEMP=CON(I,J,K)
     1         + AJP(I,J,K)*F(I,J+1,K,NF) + AJM(I,J,K)*F(I,J-1,K,NF)
     2         + AKP(I,J,K)*F(I,J,K+1,NF) + AKM(I,J,K)*F(I,J,K-1,NF)
            QTX(I)=(TEMP+AIM(I,J,K)*QTX(I-1))/DENOM
  100    CONTINUE

         DO 110 I=L2,2,-1
  110       F(I,J,K,NF)=F(I+1,J,K,NF)*PTX(I)+QTX(I)

   90 CONTINUE

CARRY OUT THE J-DIRECTION TDMA
C-------------------------------------------------------------------------
      DO 120 KK=2,NN
       K=MIN(KK,NN2-KK)
       DO 120 II=2,LL
         I=MIN(II,LL2-II)
         PTY(1)=0.0d0
         QTY(1)=0.0d0

         DO 130 J=2,M2
            DENOM=AP(I,J,K)-PTY(J-1)*AJM(I,J,K)
            PTY(J)=AJP(I,J,K)/DENOM
            TEMP=CON(I,J,K)
     1         + AIP(I,J,K)*F(I+1,J,K,NF) + AIM(I,J,K)*F(I-1,J,K,NF)
     2         + AKP(I,J,K)*F(I,J,K+1,NF) + AKM(I,J,K)*F(I,J,K-1,NF)
            QTY(J)=(TEMP+AJM(I,J,K)*QTY(J-1))/DENOM
  130    CONTINUE

         DO 140 J=M2,2,-1
  140       F(I,J,K,NF)=F(I,J+1,K,NF)*PTY(J)+QTY(J)

  120 CONTINUE
C
CARRY OUT THE K-DIRECTION TDMA
C-------------------------------------------------------------------------
      DO 122 JJ=2,MM
       J=MIN(JJ,MM2-JJ)
       DO 122 II=2,LL
         I=MIN(II,LL2-II)
         PTZ(1)=0.0d0
         QTZ(1)=0.0d0

         DO 132 K=2,N2
            DENOM=AP(I,J,K)-PTZ(K-1)*AKM(I,J,K)
            PTZ(K)=AKP(I,J,K)/DENOM
            TEMP=CON(I,J,K)
     1         + AIP(I,J,K)*F(I+1,J,K,NF) + AIM(I,J,K)*F(I-1,J,K,NF)
     2         + AJP(I,J,K)*F(I,J+1,K,NF) + AJM(I,J,K)*F(I,J-1,K,NF)
            QTZ(K)=(TEMP+AKM(I,J,K)*QTZ(K-1))/DENOM
  132    CONTINUE
```

```
        DO 142 K=N2,2,-1
  142      F(I,J,K,NF)=F(I,J,K+1,NF)*PTZ(K)+QTZ(K)

  122 CONTINUE
C-------------------------------------------------------------------------
  999 CONTINUE

      NTC(NF)=NTT
      GO TO 991
  990 NTC(NF)=NTT-1
  991 CONTINUE

CALCULATE THE UNKNOWN BOUNDARY VALUES AND FLUXES
C-------------------------------------------------------------------------
      DO 160 K=2,N2
      DO 160 I=2,L2
         TEMP=AJM(I,1,K)*(F(I,3,K,NF)-F(I,2,K,NF))
         IF(KBCJ1(I,K).EQ.2)
     1          F(I,1,K,NF)=(AJP(I,1,K)*F(I,2,K,NF)-
     2                    TEMP+CON(I,1,K))/AP(I,1,K)
         FLUXJ1(I,K,NF)=AJP(I,1,K)*(F(I,1,K,NF)-F(I,2,K,NF))+TEMP
         TEMP=AJP(I,M1,K)*(F(I,M3,K,NF)-F(I,M2,K,NF))
         IF(KBCM1(I,K).EQ.2)
     1          F(I,M1,K,NF)=(AJM(I,M1,K)*F(I,M2,K,NF)-
     2                     TEMP+CON(I,M1,K))/AP(I,M1,K)
  160    FLUXM1(I,K,NF)=AJM(I,M1,K)*(F(I,M1,K,NF)-F(I,M2,K,NF))+TEMP

      DO 170 K=2,N2
      DO 170 J=2,M2
         TEMP=AIM(1,J,K)*(F(3,J,K,NF)-F(2,J,K,NF))
         IF(KBCI1(J,K).EQ.2)
     1       F(1,J,K,NF)=(AIP(1,J,K)*F(2,J,K,NF)-
     2                 TEMP+CON(1,J,K))/AP(1,J,K)
         FLUXI1(J,K,NF)=AIP(1,J,K)*(F(1,J,K,NF)-F(2,J,K,NF))+TEMP
         TEMP=AIP(L1,J,K)*(F(L3,J,K,NF)-F(L2,J,K,NF))
         IF(KBCL1(J,K).EQ.2)
     1      F(L1,J,K,NF)=(AIM(L1,J,K)*F(L2,J,K,NF)-
     2                 TEMP+CON(L1,J,K))/AP(L1,J,K)
  170    FLUXL1(J,K,NF)=AIM(L1,J,K)*(F(L1,J,K,NF)-F(L2,J,K,NF))+TEMP

      DO 172 J=2,M2
      DO 172 I=2,L2
         TEMP=AKM(I,J,1)*(F(I,J,3,NF)-F(I,J,2,NF))
         IF(KBCK1(I,J).EQ.2)
     1          F(I,J,1,NF)=(AKP(I,J,1)*F(I,J,2,NF)-
     2                    TEMP+CON(I,J,1))/AP(I,J,1)
         FLUXK1(I,J,NF)=AKP(I,J,1)*(F(I,J,1,NF)-F(I,J,2,NF))+TEMP
         TEMP=AKP(I,J,N1)*(F(I,J,N3,NF)-F(I,J,N2,NF))
         IF(KBCN1(I,J).EQ.2)
     1          F(I,J,N1,NF)=(AKM(I,J,N1)*F(I,J,N2,NF)-
     2                     TEMP+CON(I,J,N1))/AP(I,J,N1)
  172    FLUXN1(I,J,NF)=AKM(I,J,N1)*(F(I,J,N1,NF)-F(I,J,N2,NF))+TEMP

C
COME HERE TO RESET CON,AP,KBC,FLXC, AND FLXP -  3d is complete
C-------------------------------------------------------------------------
      DO 180 K=2,N2
      DO 180 J=2,M2
          KBCI1(J,K) =1
          KBCL1(J,K) =1
```

```
      FLXCI1(J,K)=0.0d0
      FLXCL1(J,K)=0.0d0
      FLXPI1(J,K)=0.0d0
      FLXPL1(J,K)=0.0d0

      DO 180 I=2,L2
         CON(I,J,K) =0.0d0
          AP(I,J,K) =0.0d0
180 CONTINUE

    DO 190 K=2,N2
    DO 190 I=2,L2
        KBCJ1(I,K) =1
        KBCM1(I,K) =1
       FLXCJ1(I,K)=0.0d0
       FLXCM1(I,K)=0.0d0
       FLXPJ1(I,K)=0.0d0
       FLXPM1(I,K)=0.0d0
190 CONTINUE

    DO 192 J=2,M2
    DO 192 I=2,L2
        KBCK1(I,J) =1
        KBCN1(I,J) =1
       FLXCK1(I,J)=0.0d0
       FLXCN1(I,J)=0.0d0
       FLXPK1(I,J)=0.0d0
       FLXPN1(I,J)=0.0d0
192 CONTINUE

    RETURN
    END
```

Appendix H - TOOLS3D.F

```
SUBROUTINE TOOLS
c 3D version - HP f77 - dp version
C*********************************************************************
      INCLUDE 'common3d.f'
C*********************************************************************
      ENTRY EZGRID
C
CONSTRUCT THE X-DIRECTION GRID
      L1    =NCVLX+2
      XU(2) =0.0d0
      XU(L1)=XL
      L2    =L1-1
      FCVLX =FLOAT(NCVLX)

      DO 20 I=3,L2
         DD=FLOAT(I-2)/FCVLX
         IF(POWERX.GT.0.d0) THEN
           XU(I)=XL*DD**POWERX
         ELSE
           XU(I)=XL*(1.d0-(1.d0-DD)**(-POWERX))
         ENDIF
   20 CONTINUE

CONSTRUCT THE Y-DIRECTION GRID
      M1    =NCVLY+2
      YV(2) =0.0d0
      YV(M1)=YL
      M2    =M1-1
      FCVLY =FLOAT(NCVLY)

      DO 30 J=3,M2
         DD=FLOAT(J-2)/FCVLY
         IF(POWERY.GT.0.d0) THEN
           YV(J)=YL*DD**POWERY
         ELSE
           YV(J)=YL*(1.d0-(1.d0-DD)**(-POWERY))
         ENDIF
   30 CONTINUE

CONSTRUCT THE Z-DIRECTION GRID
      N1    =NCVLZ+2
      ZW(2) =0.0d0
      ZW(N1)=ZL
      N2    =N1-1
      FCVLZ =FLOAT(NCVLZ)

      DO 40 K=3,N2
         DD=FLOAT(K-2)/FCVLZ
         IF(POWERZ.GT.0.d0) THEN
           ZW(K)=ZL*DD**POWERZ
         ELSE
           ZW(K)=ZL*(1.d0-(1.d0-DD)**(-POWERZ))
         ENDIF
   40 CONTINUE

      RETURN
C*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
```

```
      ENTRY ZGRID
CONSTRUCT THE GRID ZONE-BY-ZONE
C
CONSIDER THE X DIRECTION
      XU(2)=0.0d0
      I2   =2

      DO 100 NZ=1,NZX
         FCVLX=FLOAT(NCVX(NZ))
         ILAST=I2
         I1   =ILAST+1
         I2   =ILAST+NCVX(NZ)

         DO 100 I=I1,I2
            DD=FLOAT(I-ILAST)/FCVLX
            IF(POWRX(NZ).GT.0.d0) THEN
              XU(I)=XU(ILAST)+XZONE(NZ)*DD**POWRX(NZ)
            ELSE
              XU(I)=XU(ILAST)+XZONE(NZ)*(1.d0-(1.d0-DD)**(-POWRX(NZ)))
            ENDIF
  100 CONTINUE

      L1=I2
C
CONSIDER THE Y DIRECTION
      YV(2)=0.0d0
      J2   =2

      DO 110 NZ=1,NZY
         FCVLY=FLOAT(NCVY(NZ))
         JLAST=J2
         J1   =JLAST+1
         J2   =JLAST+NCVY(NZ)

         DO 110 J=J1,J2
            DD=FLOAT(J-JLAST)/FCVLY
            IF(POWRY(NZ).GT.0.d0) THEN
              YV(J)=YV(JLAST)+YZONE(NZ)*DD**POWRY(NZ)
            ELSE
              YV(J)=YV(JLAST)+YZONE(NZ)*(1.d0-(1.d0-DD)**(-POWRY(NZ)))
            ENDIF
  110 CONTINUE

      M1=J2
C
CONSIDER THE Z DIRECTION
      ZW(2)=0.0d0
      K2   =2

      DO 120 NZ=1,NZZ
         FCVLZ=FLOAT(NCVZ(NZ))
         KLAST=K2
         K1   =KLAST+1
         K2   =KLAST+NCVZ(NZ)

         DO 120 K=K1,K2
            DD=FLOAT(K-KLAST)/FCVLZ
            IF(POWRZ(NZ).GT.0.d0) THEN
              ZW(K)=ZW(KLAST)+ZZONE(NZ)*DD**POWRZ(NZ)
            ELSE
```

```
              ZW(K)=ZW(KLAST)+ZZONE(NZ)*(1.d0-(1.d0-DD)**(-POWRZ(NZ)))
            ENDIF
  120 CONTINUE

      N1=K2

      RETURN
C*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
      ENTRY PRINT
C
      DO 999 IUNIT=IU1,IU2
C
COME HERE TO ARRANGE THE PRINTOUT OF Three-DIMENSIONAL FIELDS

      IF(KPGR.NE.0) THEN
C
CREATE PRINTOUT FOR GRID
C
        WRITE(IUNIT,1)
    1   FORMAT(' ')
        IBEG=1
        IEND=L1
        IREP=(IEND-IBEG+7)/7

        DO 200 IK=1,IREP
           INCR   =MIN(6,IEND-IBEG)
           ISTOP  =IBEG+INCR
           istopu = istop + 1
           if (istop .eq. l1) istopu = l1
           WRITE(IUNIT,2) (I,I=IBEG,ISTOP)
    2      FORMAT(/2X,'I =',2X,7(I4,5X))
           IF(MODE.EQ.3) THEN
             WRITE(IUNIT,3) (X(I),I=IBEG,ISTOP)
    3        FORMAT(1X,'TH =',1P7E9.2)
             WRITE(IUNIT,33) (XU(I),I=IBEG+1,ISTOPU)
   33        FORMAT(4X,'THU = ',1P7E9.2)
           ELSE
             WRITE(IUNIT,4) (X(I),I=IBEG,ISTOP)
    4        FORMAT(2X,'X =',1P7E9.2)
             WRITE(IUNIT,44) (XU(I),I=IBEG+1,ISTOPU)
   44        FORMAT(5X,'XU = ',1P7E9.2)
           ENDIF
           IBEG=ISTOP+1
  200   CONTINUE
C
        WRITE(IUNIT,1)
        JBEG=1
        JEND=M1
        JREP=(JEND-JBEG+7)/7

        DO 210 JK=1,JREP
           INCR   =MIN(6,JEND-JBEG)
           JSTOP  =JBEG+INCR
           jstopV = jstop + 1
           if (jstop .eq. m1) jstopV = m1
           WRITE(IUNIT,5) (J,J=JBEG,JSTOP)
    5      FORMAT(/2X,'J =',2X,7(I4,5X))
           WRITE(IUNIT,6) (Y(J),J=JBEG,JSTOP)
    6      FORMAT(2X,'Y =',1P7E9.2)
           WRITE(IUNIT,66) (YV(J),J=JBEG+1,JSTOPV)
```

```
   66       FORMAT(5X,'YV = ',1P7E9.2)
            JBEG=JSTOP+1
  210    CONTINUE
C
         WRITE(IUNIT,1)
         KBEG=1
         KEND=N1
         KREP=(KEND-KBEG+7)/7

         DO 215 KK=1,KREP
            INCR   =MIN(6,KEND-KBEG)
            KSTOP  =KBEG+INCR
            KstopW = Kstop + 1
            if (Kstop .eq. n1) KstopW = n1
            WRITE(IUNIT,75) (K,K=KBEG,KSTOP)
   75       FORMAT(/2X,'K =',2X,7(I4,5X))
            WRITE(IUNIT,76) (Z(K),K=KBEG,KSTOP)
   76       FORMAT(2X,'Z =',1P7E9.2)
            WRITE(IUNIT,77) (ZW(K),K=KBEG+1,KSTOPW)
   77       FORMAT(5X,'ZW = ',1P7E9.2)
            KBEG=KSTOP+1
  215    CONTINUE

       ENDIF

CREATE PRINTOUT FOR THE VALUES OF DEPENDENT VARIABLES

       DO 220 NF=1,NFMAX
          IF(KPRINT(NF).NE.0) THEN
            WRITE(IUNIT,7) TITLE(NF)
    7       FORMAT(//1X,6(1H*),3X,A18,3X,6(1H*)/9X,20(1H-))
            KBEG=1
            KEND=N1

            DO 225 K = KEND, KBEG, -1

              if (K .EQ. 1) then
                WRITE(IUNIT, 97) K , Z(K)
   97           FORMAT(///,9x,'K =',I4,'  Z(K) =',1pg12.5,//)

              else

                WRITE(IUNIT, 98) K , Z(K), ZW(K)
   98           FORMAT(///,9x,'K =',I4,'  Z(K) =',1pg12.5,
     1                                '  ZW(K) =',1pg12.5,//)
              end if

            IBEG=1
            JBEG=1
            IEND=L1
            JEND=M1
            IREP=(IEND-IBEG+7)/7

            DO 230 KK=1,IREP
               INCR =MIN(6,IEND-IBEG)
               ISTOP=IBEG+INCR
               WRITE(IUNIT,8) (I,I=IBEG,ISTOP)
    8          FORMAT(/'  I =',I6,6I9)
               WRITE(IUNIT, 9)
    9          FORMAT('  J')
```

```
              DO 240 J=JEND,JBEG,-1
                 WRITE(IUNIT,10) J,(F(I,J,K,NF),I=IBEG,ISTOP)
   10            FORMAT(1X,I2,3X,1P7E9.2)
  240           CONTINUE

             IBEG=ISTOP+1

  230        CONTINUE
  225        CONTINUE

           ENDIF

  220 CONTINUE

  999 CONTINUE
      RETURN
C*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
c Note - dont use PLOT for 3D without first modifying the plot prog.
c  note that the IBLOCK array was removed from common.
      ENTRY PLOT
      OPEN(UNIT=8,FILE=PLOTF)
COME HERE TO CREATE DATA FOR PLOTTING
C
      KFLOW=2
      WRITE(8,300) HEADER
  300 FORMAT(A64)
      WRITE(8,310) KFLOW,L1,M1,N1,NFMAX,MODE,(KPLOT(I),I=1,NFMAX)
  310 FORMAT(18I5)
c     IBLOK=0
c
c     DO 320 K=2,N2
c     DO 320 J=2,M2
c     DO 320 I=2,L2
c        IF(IBLOCK(I,J,K).EQ.1) THEN
c          IBLOK=1
c          GO TO 330
c        ENDIF
c 320 CONTINUE
c 330 CONTINUE

c     WRITE(8,310) IBLOK
      WRITE(8,340) (TITLE(N),N=1,NFMAX)
  340 FORMAT(4A18)
      WRITE(8,350) (X(I),I=1,L1),   (Y(J),J=1,M1),   (Z(K),K=1,N1),
     1            (XU(I),I=2,L1), (YV(J),J=2,M1), (ZW(K),K=2,N1),
     2                             (R(J),J=1,M1)
  350 FORMAT(5E12.6)

      DO 360 N=1,NFMAX
         IF(KPLOT(N).NE.0)
     1   WRITE(8,350) (((F(I,J,K,N),I=1,L1),J=1,M1),K=1,N1)
  360 CONTINUE

c     IF(IBLOK.EQ.1) THEN
c       WRITE(8,310) (((IBLOCK(I,J,K),I=1,L1),J=1,M1),K=1,N1)
c     ENDIF
      CLOSE(8)
      RETURN
      END
```

Appendix I - COMMON3D.F

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
c Common include file for 3D version of CONDUCT - Orthotropic Gamma case
c    adds  K,N  for Z direction equiv to  I,L  in X and  J,M  in Y
c
c    double precision version

      implicit real*8 (a-h,o-z)
      implicit integer*4 (i-n)

      PARAMETER (NI=105, NJ=105, NK=20, NFMAX=1, NZMAX=10)
      CHARACTER*18 TITLE
      CHARACTER*64 HEADER, PRINTF, PLOTF

      COMMON  F(NI,NJ,NK,NFMAX),     ALAM(NI,NJ,NK),
     1 GAMX(NI,NJ,NK), GAMY(NI,NJ,NK), GAMZ(NI,NJ,NK),
     2  CON(NI,NJ,NK),   AP(NI,NJ,NK),  AIP(NI,NJ,NK), AIM(NI,NJ,NK),
     3  AJP(NI,NJ,NK),  AJM(NI,NJ,NK),  AKP(NI,NJ,NK), AKM(NI,NJ,NK),
     4             FLUXI1(NJ,NK,NFMAX),          FLUXL1(NJ,NK,NFMAX),
     5             FLUXJ1(NI,NK,NFMAX),          FLUXM1(NI,NK,NFMAX),
     6             FLUXK1(NI,NJ,NFMAX),          FLUXN1(NI,NJ,NFMAX),
     7    X(NI),  XU(NI), XCV(NI), ARX(NJ),   Y(NJ),  YV(NJ), YCV(NJ),
     8 YCVR(NJ),   R(NJ),  RV(NJ),  SX(NJ),   Z(NK),  ZW(NK), ZCV(NK),
     9  PTX(NI), QTX(NI), PTY(NJ), QTY(NJ), PTZ(NK), QTZ(NK), RC(NK)
c
      COMMON /BCarray/
     1 FLXCI1(NJ,NK),       FLXCL1(NJ,NK),
     2 FLXCJ1(NI,NK),       FLXCM1(NI,NK),
     3 FLXCK1(NI,NJ),       FLXCN1(NI,NJ),
     4 FLXPI1(NJ,NK),       FLXPL1(NJ,NK),
     5 FLXPJ1(NI,NK),       FLXPM1(NI,NK),
     6 FLXPK1(NI,NJ),       FLXPN1(NI,NJ),
     7  KBCI1(NJ,NK),        KBCL1(NJ,NK),
     8  KBCJ1(NI,NK),        KBCM1(NI,NK),
     9  KBCK1(NI,NJ),        KBCN1(NI,NJ)
c
      COMMON /GENL/ NF, L1,L2,L3, M1,M2,M3, N1,N2,N3, ITER,SMALL,BIG,
     1             LAST,TIME,DT, MODE,KORD,KOUT, IU1,IU2, KPGR,KSTOP
      COMMON /NFF/   RELAX(NFMAX), KPRINT(NFMAX), KSOLVE(NFMAX),
     1 KBLOC(NFMAX), KPLOT(NFMAX), NTIMES(NFMAX), NTC(NFMAX),CRIT(NFMAX)
      COMMON /TTL/ TITLE(NFMAX), HEADER, PRINTF, PLOTF
      COMMON /EZG/ NCVLX,NCVLY,NCVLZ, XL,YL,ZL, POWERX,POWERY,POWERZ
      COMMON /ZG/  NZX,NZY,NZZ, NCVX(NZMAX),  NCVY(NZMAX),  NCVZ(NZMAX),
     1                          XZONE(NZMAX), YZONE(NZMAX), ZZONE(NZMAX),
     2                          POWRX(NZMAX), POWRY(NZMAX), POWRZ(NZMAX)
c
c
      COMMON Tmaxj, Qgen
c
      DIMENSION        SC(NI,NJ,NK), SP(NI,NJ,NK)
      EQUIVALENCE (CON,SC),       (AP,SP)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

Appendix J Plotting With MATLAB

Three-dimensional plots can easily be obtained using MATLAB. The temperature distribution must be written to a data file in rows and columns format. For a model of any appreciable size, the lines in the data file will be extremely wide. Although viewing a file like this is a visual mess, MATLAB can successfully interpret the data. The FORTRAN commands used to write the data files for the example problem are given.

Placed at beginning of adaptation subroutine

```
      OPEN (11, FILE = 'mat1.dat')
      OPEN (12, FILE = 'mat2.dat')
      OPEN (13, FILE = 'mat3.dat')
      OPEN (14, FILE = 'mat4.dat')
      OPEN (15, FILE = 'mat5.dat')
```

Placed near the end of the OUTPUT section; should only be executed after the convergence criteria is satisfied

```
            icvs = 81
            DO 991 J=2, icvs
             WRITE(11,990) (T(I,J,1),I=2,icvs)
             WRITE(12,990) (T(I,J,2),I=2,icvs)
             WRITE(13,990) (T(I,J,3),I=2,icvs)
991         CONTINUE

990       FORMAT(1X,1P80E9.2)

            DO 993 J=18,65
             WRITE(14,992) (T(I,J,4),I=18,65)
             WRITE(15,992) (T(I,J,5),I=18,65)
993         CONTINUE

992         FORMAT(1X,1P48E9.2)
```

Note that the limits on the do loops exclude the boundary values, which have not been solved for. For layers K=4 and K=5, the data outside the boundary of the layer is excluded, since it is not meaningful.

Within MATLAB, a data file is loaded by typing:
load mat1.dat
This creates a matrix called mat1.
The data can then be plotted using any of several commands.

mesh(mat1)
creates a three dimensional mesh plot
surf(mat1)
creates a three-dimensional surface plot
contour(mat1)
creates a two-dimensional contour plot
pcolor(mat1)

creates a two-dimensional contour plot with shaded regions

These commands plot the temperature within the model. The location of the temperatures is determined by the values location in the matrix. On a surface plot, the z-axis represents temperature and the x and y axes represent the matrix row and column number. Thus, this procedure is only useful when the data is obtained from a uniform grid.

The surface plots in chapter 5 were created using a sequence of commands resembling:

load mat5.dat	load data file
surf(mat5)	create surface plot
colormap(gray)	use gray color scheme for b&w printer
brighten(gray,0.7)	lighten color scheme (use value between -1 and 1, 0 is default)
view(135,30)	change view (for plots K=5 through K=2; for K=2 use default view of (-45,30))
print -dps surf5	create surf5.ps b&w PostScript file

The contour plots were created using commands resembling:

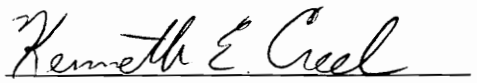
load mat5.dat	load data file (if not already in memory)
pcolor(mat5)	create filled color contour plot
colormap(gray)	use gray color scheme for b&w printer
brighten(gray,0.5)	lighten color scheme (use value between -1 and 1, 0 is default)
hold	hold current plot and superimpose later commands
contour(mat5,10,'w')	plot 10 white contour lines (will print in black)
print -dps con5	create con5.ps b&w PostScript file
hold	releases figure window for next plot

The plots can be modified and combined in a variety of ways. Further information is provided in the MATLAB user's manuals.

Vita

Kenneth Creel was born in South Carolina on May 5, 1970. After spending his childhood in Clinton, he attended Virginia Tech, obtaining his B.S. in Aerospace Engineering in 1992. He then entered graduate school to study Mechanical Engineering, graduating with an M.S. in 1994.

His path in the future is undetermined, although ideally it will contain novelty, curiosity, learning, and enlightenment, in essence, a second childhood.



Kenneth E. Creel