AN EFFICIENT NUMERICAL TECHNIQUE
FOR
CALCULATING THERMAL SPREADING RESISTANCE

Final Report

8 February 1973

A Study Prepared for
National Aeronautics and Space Administration
under Contract NAS8-28516

Prepared by
Dr. Earl H. Gale, Jr.
General Electric Company
Aerospace Electronic Systems Department
Utica, New York
ACKNOWLEDGMENTS

The author wishes to express his appreciation to the following General Electric people for their support on this project:

Stephen A. Smith, who reviewed the basic technique and made calculations of the computer work required for this and several other techniques.

Edwin Kelly, who translated my program from BASIC to the FORTRAN Y version listed in the Appendix of this report.

Claude Lindeman, who contributed the section on the use of superposition.
AN EFFICIENT NUMERICAL TECHNIQUE
FOR
CALCULATING THERMAL SPREADING RESISTANCE

Final Report

8 February 1973

A Study Prepared for
National Aeronautics and Space Administration
under Contract NAS8-28516

Prepared by
Dr. Earl H. Gale, Jr.
General Electric Company
Aerospace Electronic Systems Department
Utica, New York
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>1</td>
</tr>
<tr>
<td>I.</td>
<td>1</td>
</tr>
<tr>
<td>II</td>
<td>2</td>
</tr>
<tr>
<td>II.</td>
<td>2</td>
</tr>
<tr>
<td>III</td>
<td>3</td>
</tr>
<tr>
<td>III.</td>
<td>3</td>
</tr>
<tr>
<td>IV</td>
<td>6</td>
</tr>
<tr>
<td>IV.</td>
<td>6</td>
</tr>
<tr>
<td>V</td>
<td>8</td>
</tr>
<tr>
<td>V.</td>
<td>8</td>
</tr>
<tr>
<td>VI</td>
<td>11</td>
</tr>
<tr>
<td>VI.</td>
<td>11</td>
</tr>
<tr>
<td>VII</td>
<td>14</td>
</tr>
<tr>
<td>VII.</td>
<td>14</td>
</tr>
<tr>
<td>VIII</td>
<td>16</td>
</tr>
<tr>
<td>VIII.</td>
<td>16</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>17</td>
</tr>
<tr>
<td>APPENDIX</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>49</td>
</tr>
<tr>
<td></td>
<td>51</td>
</tr>
<tr>
<td>Figure</td>
<td>Title</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>1</td>
<td>Semiconductor in an Integrated Circuit</td>
</tr>
<tr>
<td>2</td>
<td>Microscopic View of the joint of Contacting Pieces of Metal</td>
</tr>
<tr>
<td>3</td>
<td>Temperature Profiles Described by Holm's Equation for Isothermal</td>
</tr>
<tr>
<td></td>
<td>Circular Source on Semi-infinite Slab</td>
</tr>
<tr>
<td>4</td>
<td>Percent of Total Constriction Resistance for a Single Isothermal</td>
</tr>
<tr>
<td></td>
<td>Circular Source on a Semi-infinite Slab as a Function of Distance into</td>
</tr>
<tr>
<td></td>
<td>Body of Contact</td>
</tr>
<tr>
<td>5</td>
<td>Rectangular Field Divided into 25 Finite Elements</td>
</tr>
<tr>
<td>6</td>
<td>Nomenclature for Nodal Interconductances</td>
</tr>
<tr>
<td>7</td>
<td>System of Equations in Matrix Notation</td>
</tr>
<tr>
<td>8</td>
<td>System of Submatrices in Matrix Notation</td>
</tr>
<tr>
<td>9</td>
<td>Cost of Computation vs Size of Coefficient Matrix</td>
</tr>
<tr>
<td>10</td>
<td>Matrix Representation of System of Equations Describing</td>
</tr>
<tr>
<td></td>
<td>Three-Dimensional Field Showing Pattern Established by</td>
</tr>
<tr>
<td></td>
<td>Submatrices of Coefficient Matrix</td>
</tr>
<tr>
<td>11</td>
<td>Mathematical Model for Spreading Resistance</td>
</tr>
<tr>
<td>12</td>
<td>Nodal Pattern for Sample Problem</td>
</tr>
<tr>
<td>13</td>
<td>Convention for Nodal Conductances</td>
</tr>
<tr>
<td>14</td>
<td>Computer Program for Illustrative Example</td>
</tr>
<tr>
<td>15</td>
<td>Identification and Internal Makeup of Submatrices of Coefficient Matrix</td>
</tr>
<tr>
<td>16</td>
<td>Sample Solutions of Illustrative Problem</td>
</tr>
<tr>
<td>17</td>
<td>Maximum Nodal Temperature for Minimum a/b as a Function of Number of Nodes</td>
</tr>
<tr>
<td>18</td>
<td>Memory Board Module</td>
</tr>
<tr>
<td>19</td>
<td>Location of Conductances and Temperatures</td>
</tr>
</tbody>
</table>
SECTION I
INTRODUCTION

This final report has been prepared by the General Electric Company, Aerospace Electronic Systems Department, Utica, New York under contract NAS8-28516. The report documents the results of a thermal spreading resistance data generation technique study. The method developed is discussed in detail, illustrative examples given, and the resulting computer program is included.
SECTION II
BACKGROUND

A. GENERAL

"Thermal spreading resistance" is defined as the conductive thermal resistance between a source region and a sink region in a solid where the geometry is such as to preclude one dimensional heat flow.

Knowledge of thermal spreading resistance is needed in two aerospace engineering areas. These are the thermal design of electronic components or equipments and in the prediction and control of thermal contact resistance.

1. Importance To The Design Of Electronic Components and Equipments

The thermal analysis of a power semiconductor or integrated circuit can be reduced to the problem of determining the appropriate spreading and bonding thermal resistances. As an example, the problem of calculating the junction-to-case thermal resistance of a semiconductor bonded to a substrate which is bonded in a metal case will be considered. Figure 1 illustrates this problem.

![Figure 1. Semiconductor in an Integrated Circuit](image)

Heat is generated in a region of known size, the junction region of the semiconductor. The first, and most significant, spreading resistance of interest occurs between the junction and the opposite face of the silicon chip. The next thermal resistance of interest is that across the bond between chip and substrate. It is of significance that these thermal resistances are not independent although many thermal designers, under the pressures of a design schedule, have treated them as such. The thermal conductance of the bond proper can vary several thousand-fold depending on the use of a metallic or nonmetallic bonding material. The resistance to heat flow between the semiconductor chip bond region and the rear of the substrate represents a second spreading resistance, etc. In a typical integrated circuit package the entire bottom region of the substrate would not be available as a sink for a single semiconductor chip due to the presence of other heat dissipating chips. It is usually possible to estimate the effective sink region on the rear of the substrate from considerations of symmetry or because it exceeds dimensions which appreciably affect the thermal spreading resistance. In those few cases where interactions must be considered, the key analytical tool is superposition; Green's function approach
may also be employed to advantage. For example, see reference 1 and the discussion beginning on page 37 of this report.

The importance of being able to predict thermal spreading resistances in single and multi-layered material in the evaluation of the thermal design of semiconductor or integrated circuits has been shown. Spreading thermal resistances are important in other electrical devices such as phased array antenna elements, Peltier coolers, Seebeck generators and many devices which utilize conductive heat transfer.

B. PREDICTION AND CONTROL OF THERMAL CONTACT RESISTANCE

The resistance to heat flow between two mating (touching, as in a joint) pieces of metal is called thermal contact resistance. When the actual microscopic regions of contact between two mating surfaces are examined, it is found that metal-to-metal contact occurs in small discrete regions where the asperities or microscopic protuberances make contact. References 2 and 3 describe this model of contact in great detail. Figure 2 illustrates this contact model.

![Regions of Asperitic Contact](image)

Figure 2. Microscopic View of the Joint of Contacting Pieces of Metal

The heat flow to and from a region of asperitic contact into the contacting proper is seen to be of the "spreading" type. In fact, the effective thermal contact resistance of any contact may be considered as the sum of the parallel microscopic spreading resistances in the contacts themselves. References 2 and 3 above deal largely with isentropic contacts in which the thermal conductivity within the bodies of both contacts is uniform.

Analysis has shown that the bulk of the spreading resistance occurs close to the region of actual asperitic contact and that the spreading resistance in any region varies inversely with the thermal conductivity of the material. Figures 3 and 4 illustrate the first of these points. Figure 3, drawn to scale, shows the equipotential lines about a circular contact region each drawn to show one-tenth of the total spreading resistance between the circular source region and the body of a very large contact. It is seen that half of this total resistance occurs within one contact radius from the circular contact or source region and 80 percent occurs within three contact radii. Figure 4 illustrates these relationships. Figures 3 and 4 are taken from reference 4.

The thermal conductivity of contact close to the surface is of such importance that even a thin 45 Angstrom thick layer of oxide on an aluminum contact can contribute measurably to the thermal contact resistance of an aluminum contact. This has been shown by Gale, reference 4.

Mikic and Carnasciali, reference 5, have utilized the above principle to enhance thermal contact conductance by plating materials of higher conductivity on the contacting faces of a
Figure 3. Temperature Profiles Described by Holm's Equation for Isothermal Circular Source on a Semi-infinite Slab
metallic joint. They have attempted an analysis of spreading resistance from a circular contact into a contact composed of two layers of materials with different conductivities. An exact boundary value solution of this basic problem has proven too difficult as no mathematical function has been found which will satisfy the boundary conditions between the plating and the body materials.

Professor C. J. Moore, Jr. in his discussion printed at the end of reference 5 felt this two layered spreading resistance problem could best be handled by a "well-conditioned finite difference computer code." Mikic and Carnasciali then question the economic feasibility of such calculations.

Attempts by the author of this study to solve the two layer thermal spreading resistance problem using a finite difference approach utilizing Gauss-Seidel iteration have shown the cost of digital computer calculation to be great.
SECTION III
THEORY

A. GENERAL

The governing differential equation for the thermal spreading resistance problem is Poisson's equation. For those spreading resistance problems that are two-dimensional or may be reduced to two-dimensional problems, the equation is:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = q''''$$

(1)

Consider a rectangular field subdivided into rectangular subregions as illustrated in Figure 5. The heat balance equation describing the heat flow among element \( m, n \) and its four principal neighbors is:

$$(T_m, n - T_{m+1, n}) H_m, n + (T_m, n - T_{m-1, n}) V_m, n + (T_m, n - T_{m, n-1}) H_m, n-1$$

$$+ (T_m, n - T_{m+1, n}) V_{m+1, n} = q_{m, n}$$

(2)

where:

- \( T \) is temperature
- \( q'''' \) heat generated per unit volume
- \( x, y, z \) are spatial coordinates
- \( H, V \) are horizontal and vertical conductances, respectively
- \( q_{m, n} \) heat generated in mode \( m, n \)

The convention for the horizontal and vertical conductances used is shown in Figure 6.

Each of the following observations below will be helpful in understanding the discussion which follows:

1. When any temperature \( t_{m, n} \) is known (e.g., as a boundary condition), it will affect equation \( m, n \) by yielding a term \( q_{m, n} \), which is subtracted from the right hand side of equation (2) where \( q_{m, n} \) is:

$$q_{m, n} = T_{m, n} (H_{m, n} + V_{m, n} + H_{m, n-1} + V_{m+1, n})$$

(3)

1Associate Professor of Mechanical and Aerospace Engineering, North Carolina State University, Raleigh, N.C.

2The method developed is applicable to three-dimensional problems as will be shown later in the report.
Figure 5. Rectangular Field Divided into 25 Finite Elements

Figure 6. Nomenclature for Nodal Interconductances
(2) If the original field is divided into $M$ rows and $N$ columns, and further if $M = N$, then:

(a) There will be $N^2$ linear equations.

(b) There will be not more than $N^2$ unknowns (fewer if some temperatures are initially prescribed).

(c) There can be as many different and distinct nodal conductances as there are interconnections between nodes.

Now, if the system of linear finite difference equations is written in matrix form (taking the nodes of Figure 5 into consideration) from left to right, top row to bottom row, as in reading English, a coefficient matrix results that has a pattern characteristic for field problems described by Poisson's or Laplace's equations. This pattern is illustrated in Figure 7.

It was noted by Karlqvist (Reference 6) that the matrix in Figure 7 may be partitioned as shown. It can be seen that each of the submatrices is $N \times N$ and the coefficient matrix is $N^2 \times N^2$ where the original finite element matrix was $N \times N$ in size.

B. DERIVATION OF AN EFFICIENT TECHNIQUE FOR EXACT SOLUTION OF THIS SYSTEM OF EQUATIONS

Defining the sub-matrices shown in Figure 7 as follows:

\[
\begin{bmatrix}
B_1 & C_1 & 0 & 0 & 0 \\
A_2 & B_2 & C_2 & 0 & 0 \\
0 & A_3 & B_3 & C_3 & 0 \\
0 & 0 & A_4 & B_4 & C_4 \\
0 & 0 & 0 & A_5 & B_5
\end{bmatrix}
\begin{bmatrix}
T_1 \\
T_2 \\
T_3 \\
T_4 \\
T_5
\end{bmatrix}
= 
\begin{bmatrix}
Q_1 \\
Q_2 \\
Q_3 \\
Q_4 \\
Q_5
\end{bmatrix}
\]

Figure 8. System Of Submatrices In Matrix Notation

Expanding the partitioned matrices (Figure 8) into a system of equations, having normalized each equation with respect to the diagonal element:

\[
T_1 - B_1^{-1}C_1T_2 = 0 \\
B_2^{-1}A_2T_1 - B_2^{-1}C_2T_3 = 0 \\
0 - B_3^{-1}A_3T_2 - B_3^{-1}C_3T_4 = 0 \\
0 - B_4^{-1}A_4T_3 - B_4^{-1}C_4T_5 = 0 \\
0 - B_5^{-1}A_5T_4 = 0
\]

0 = $B_1^{-1}Q_1$

0 = $B_2^{-1}Q_2$

0 = $B_3^{-1}Q_3$

$B_4^{-1}C_4T_5 = B_4^{-1}Q_4$

$T_5 = B_5^{-1}Q_5$

Upon redefining constants in the following manner:

$B_2^{-1}A_2 = -B_2$, $B_2^{-1}C_2 = -A_2$, and $B_2^{-1}Q_2 = C_2$, etc.
Figure 7. System of Equations in Matrix Notation
the general equation has the form:

\[-B_i T_{i-1} + T_i - A_i T_{i+1} = C_i \quad (4)\]

The first equation can be solved for \( T_1 \):

\[T_1 = C_1 + A_1 T_2 \quad (5)\]

and the \( i \)th for \( T_i \):

\[T_i = C_i + A_i T_{i+1} + B_i T_{i-1} \quad (6)\]

The goal is to find a recursion relationship built upon successive substitutions, which provides a solution for the \( i \)th unknown in terms of the \((i+1)\)th. That is:

\[T_i = A'_i T_{i+1} + B'_i \quad (7)\]

Examining equation (5) for \( T_1 \) above, it can be seen that:

\[A'_1 = A_1 \text{ and } B'_1 = C_1\]

The equation for \( T_2 \) is:

\[T_2 = C_2 + A_2 T_3 + B_2 T_1 \quad (8)\]

which, when written in terms of the equation for \( T_1 \), becomes:

\[T_2 = \left[I - B_2 A_1 \right]^{-1} A_2 T_3 + \left[I - B_2 A_1' \right]^{-1} \left[C_2 + B_2 B_1' \right] \quad (9)\]

The general coefficients found in this manner become

\[A'_1 = \left[I - B_1 A_{i-1}' \right]^{-1} A_i\]

and

\[B'_1 = \left[I - B_1 A_{i-1}' \right]^{-1} \left[B_i B_{i-1}' + C_i \right]\]

Therefore

\[T_i = A'_i T_{i+1} + B'_i \quad (10)\]

The temperature matrices (columns) are found starting at the \( N \)th row by making

\[T_N = B_N' \quad (11)\]

\[A_N = 0 \text{ as a boundary condition results in modification of } C_N \text{ above } \left[ \text{ see equation } (3) \right].\]

The system of equations has been solved by operating on \( 3\sqrt{N} - 2 \) submatrices, each of which is the square root of the size of the original \( N \times N \) coefficient. \( 3\sqrt{N} \) inversions of these
submatrices are required. The total number of multiplications (an indication of the effort) required during solution is:

\[
\text{No. of Multiplications} = 3N^2 + N^{3/2} - N + N^{1/2}
\]  

(12)

This may be compared against other direct methods (see Ref. 7):

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of Multiplications Required during Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian Elimination</td>
<td>(\frac{1}{3}N^3 + \frac{1}{2}N - \frac{1}{3}N)</td>
</tr>
<tr>
<td>Jordan</td>
<td>(\frac{1}{2}N^3 + \frac{1}{2}N - \frac{1}{2}N)</td>
</tr>
<tr>
<td>Doolittle</td>
<td>(\frac{1}{3}N^3 + \frac{1}{3}N)</td>
</tr>
<tr>
<td>Cholesky</td>
<td>(\frac{1}{6}N^3 + \frac{3}{2}N^2 + \frac{1}{3}N)</td>
</tr>
</tbody>
</table>

Cornock's method (Ref. 8), a triangulation type, also makes use of the characteristic pattern of submatrices which results during a finite difference solution for fields described by Poisson's equation. When the field properties are homogeneous and isentropic, Cornock's method is very powerful since only one of the above submatrices of order \(\sqrt{N}\) need be inverted. However, for the general solution of the nonhomogeneous field, the number of multiplications required is

\[
\frac{13}{2}N^2 - \frac{3}{2}N^{3/2} - 2N - 5
\]  

(13)

A serious drawback to Cornock's method is that it does not lend itself to ready general programming for matrices of variable size as does the method described in this report.

That equation (12) is indicative of the computer effort required for solution has been substantiated in practice. Figure 9 shows the variation in cost realized in the solution of very large matrices using the method developed in this report. Also, a strong feature of this method is that very large systems of equations, e.g., 2500, can be conveniently handled in a direct solution.

The FORTRAN Y program contained in the Appendix was used on a Honeywell 630 computer in generating the dollar costs shown in Figure 9. Out-of-core storage of submatrices was utilized for very large systems.

C. APPLICABILITY OF TECHNIQUE TO THREE-DIMENSIONAL PROBLEMS

The technique discussed above is suited to the solution of field problems having three or more dimensions. Figure 10 illustrates the characteristic pattern of the coefficient matrix for a three dimensional finite element array. It is seen that the submatrices are \(\sqrt{N}\) in size. The same block tridiagonal pattern of submatrices is seen to occur as in the two-dimensional case so the derivation above for the technique of solution for two dimensional matrices is still applicable. Thus, although the BASIC and FORTRAN Y program presented later in this report are written for two-dimensional problems, little revision of these programs is required to handle three dimensional programs. Since the submatrices are \(\sqrt{N}\) rather than the \(\sqrt{N}\) in size, the technique is even more powerful for three dimensional problems. The number of multiplications
required for solution of the three-dimensional problem is a function of $N^{4/3}$ as opposed to $N^2$ for the two-dimensional array where $N$ is the order of the coefficient matrix.
Figure 10. Matrix Representation of System of Equations Describing Three-Dimensional Field Showing Pattern Established by Submatrices of Coefficient Matrix
SECTION IV
ILLUSTRATIVE PROBLEM AND PROGRAM

A sample thermal spreading resistance problem will be solved to illustrate the technique presented in Section II. The computer program used in the problem is written in BASIC language. A general version of the same program written in FORTRAN is included in the Appendix.

A. DESCRIPTION OF SAMPLE PROBLEM

The thermal spreading resistance problem to be considered is depicted in Figure 11. Heat is uniformly generated in a plane circular region of radius \(a\) and flows to a circular sink of radius \(b\) both concentric with, and parallel to, the source region a distance \(H\) away in a conductive medium. The conductive medium is divided into two regions of different conductivity. An exact closed form solution of this problem has not been found.

\[\text{Figure 11. Mathematical Model for Spreading Resistance Nomographs}\]
Figure 12. Nodal Pattern for Sample Problem
Ratios of geometries and conductivities used in the generation of sample data are summarized below. All data is generated and presented in nondimensional parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Number of Values</th>
<th>Parametric Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>a/b</td>
<td>6</td>
<td>0.111, 0.222, 0.166, 0.551, 0.388, 0.5</td>
</tr>
<tr>
<td>H/b</td>
<td>6</td>
<td>0.1, 0.2, 0.5, 1, 2, 5</td>
</tr>
<tr>
<td>C/H</td>
<td>5</td>
<td>0.1, 0.2, 0.3, 0.5,</td>
</tr>
<tr>
<td>K1/K2</td>
<td>5</td>
<td>0.01, 0.1, 0.2, 0.5, 1</td>
</tr>
</tbody>
</table>

The above three-dimensional problem can be viewed as two-dimensional since all heat flow within the cylinder is in the axial and radial directions. Further, there is symmetry about the axis of the cylinder.

Figure 12 shows the arrangement of finite elements used in the illustrative data generation program.

B. CALCULATION OF NODAL CONDUCTANCES

The convention used for the nodal conductances was that the vertical conductance \( V_{(m,n)} \) associated with each node was that in the upward direction and the horizontal conductance \( H_{(m,n)} \) was that connecting with the node on the right. This is illustrated in Figure 13.

![Figure 13. Convention for Nodal Conductances](image)

The calculation of conductances is straightforward for all modes except for those nodes of horizontal conductance lying on the axis of the cylinder, i.e., \( n = 1 \). The horizontal conductance of this inner node was approximated by using the exact solution of Jacob (Ref. 9) for two-dimensional heat flow within a cylinder having uniformly distributed internal heat generation for the difference between the mean temperature of the cylinder and the outside surface with radial flow.

The conductances of all other nodes could be calculated in an exact manner using calculus. General expressions for the \( M, N \)th nodal conductances in terms of \( M, N \) were developed and used in the sample problem to facilitate changing the program to allow the use of different numbers of finite elements.
C. DETAILED DESCRIPTION OF ILLUSTRATIVE PROGRAM

Figure 14 shows the computer program for the illustrative program. The major steps in the program are described below.

<table>
<thead>
<tr>
<th>Line No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-60</td>
<td>Dimensioning symbols will be defined as used in this program. The same symbol may be redefined several times.</td>
</tr>
<tr>
<td>70</td>
<td>( P = \pi )</td>
</tr>
<tr>
<td>80</td>
<td>( C_1 ) is that part of the horizontal thermal resistance of nodes 10, 1 and 1, 1 between the nodal point and the surface of these nodes. (See discussion above concerning Jacob's formula.)</td>
</tr>
<tr>
<td>90</td>
<td>Expressions for entire horizontal resistance between nodes 10, 1 and 10, 2 as well as 1, 1 and 1, 2.</td>
</tr>
<tr>
<td>110</td>
<td>( H ) is the horizontal conductance.</td>
</tr>
<tr>
<td>160</td>
<td>( V ) is the vertical conductance to node above.</td>
</tr>
<tr>
<td>200</td>
<td>Generalized expression for horizontal conductance for most modes, ( m, n ).</td>
</tr>
<tr>
<td>210</td>
<td>Generalized expression for vertical conductance for most modes, ( m, n ).</td>
</tr>
<tr>
<td>290, 320</td>
<td>Entering adiabatic boundary conditions at top and sides.</td>
</tr>
<tr>
<td>380-460</td>
<td>( A_9 ) is the radius of the heat generating region.</td>
</tr>
<tr>
<td>470</td>
<td>Calculation and print of ( a/b ) (see Figure 11).</td>
</tr>
<tr>
<td>480-250</td>
<td>Entering uniform heat input in the circular region described by ( A_9 ). Use is made of the area dependence share by vertical conductance and heat input.</td>
</tr>
<tr>
<td>530-651</td>
<td>Setting of ( H/b ) (see Figure 11).</td>
</tr>
<tr>
<td>680-690</td>
<td>Adjustment of horizontal and vertical conductance for ( H/b ).</td>
</tr>
<tr>
<td>720-801</td>
<td>Setting of ( C/H ) (see Figure 11).</td>
</tr>
<tr>
<td>810-971</td>
<td>Establishment and assignment of values for ( K_1/K_2 ) (see Figure 11).</td>
</tr>
<tr>
<td>990, 1000</td>
<td>Optional printout of ( H(m, n) ) and ( V(m, n) )</td>
</tr>
<tr>
<td>1010-1220</td>
<td>Generation of coefficient matrix.</td>
</tr>
<tr>
<td>1010</td>
<td>( M ) is row number of physical nodal pattern.</td>
</tr>
<tr>
<td>1080</td>
<td>Sets subdiagonal and superdiagonal in the coefficient matrix (line arrays ( W ) immediately either side of the main diagonal) to zero (see Figure 15d).</td>
</tr>
</tbody>
</table>

(Continued on page 25)
2 PRINT
10 DIM H(12,12), V(12,12), T(12,12), Q(10,10)
20 DIM X(10,10), Y(10,10), W(10,10), Z(10,10)
30 DIM A(10,10), B(10,10), C(10,10), D(10,10)
40 DIM P(10,10), G(10,10), I(10,10), J(10,10), K(10,10)
50 DIM L(10,10), M(10,10), N(10,10), O(10,10), P(10,10)
60 DIM R(10,10), S(10,10), U(10,10)
70 P=3.14159265
90 CI=-125/P
90 RI=CI*(LOG(2))/((2+P)
100 R2=R1/2
101 FOR N=4 TO 5
102 FOR N3=1 TO 5
103 FOR N2=1 TO 6
104 FOR N1=1 TO 9
110 H(1,1)=H(10,1)=1/R1
120 FOR M=2 TO 9
130 H(M,1)=2*H(1,1)
140 NEXT M
150 FOR M=1 TO 10
160 V(C,1)=P/2
170 NEXT M
180 FOR N=1 TO 10
190 FOR N2=1 TO 10
200 H(M+N)=1/(1/(4*P)*LOG(N/(N-1)))
210 V(M,N)=P*4*(N-1)
220 IF M>1.1 THEN 240
230 H(1,N)=5*H(1,N)
240 H(10,N)=5*H(10,N)
250 V(M,10)=P*(4*N-5)/2
260 NEXT N
270 NEXT M
280 FOR M=1 TO 10
290 H(M,10)=0
300 NEXT M
310 FOR N=1 TO 10
320 V(1,N)=0
330 NEXT N
340 IF N1=1 THEN 2660
350 IF N1=5 THEN 2660
360 IF N1=7 THEN 26610
370 IF N1=9 THEN 2660
380 A9=1
390 IF N1<2 THEN 470
400 A9=2*N1-1
410 IF N1<10 THEN 470
420 A9=18
430 PRINT USING 471, A9/15
471: ###.##
440 C(1,1)=V(2,1)*2.000000
450 IF N1=1 THEN 530
460 FOR N=2 TO 10
490 C(N,1)=0

Figure 14. Computer Program for Illustrative Example
(Sheet 1 of 6)
493 NEXT N
500 FOR N=1 TO N1
510 C(N,1)=V(2,N)*2.0000
520 NEXT N
530 H1=.1
540 IF N2<2 THEN 650
550 H1=.2
560 IF N2<3 THEN 650
570 H1=.5
580 IF N2<4 THEN 650
590 H1=1
600 IF N2<5 THEN 650
610 H1=2
615 IF N2<6 THEN 650
620 H1=5
630 IF N2<7 THEN 650
640 H1=10
650 PRINT USING 651,H1,
651:##########
660 FOR M=1 TO 10
670 FOR N=1 TO 10
680 V(M,N)=V(M,N)/H1
690 H(M,N)=H(M,N)*H1
700 NEXT N
710 NEXT M
720 C1=1
730 IF N3<2 THEN 800
740 C1=2
750 IF N3<3 THEN 800
760 C1=3
770 IF N3<4 THEN 800
780 C1=5
785 IF N3<5 THEN 800
790 C1=7
800 PRINT USING 801,C1/10,
801:#############
810 K2=1
820 IF N4<2 THEN 910
830 K2=2
840 IF N4< 3 THEN 910
850 K2=5
860 IF N4<4 THEN 910
870 K2=10
880 IF N4<5 THEN 910
890 K2=100
910 FOR M=C1+1 TO 10
920 FOR N=1 TO 10
930 V(M,N)=V(M,N)*K2
940 H(M,N)=H(M,N)*K2
950 NEXT N
960 NEXT M
970 PRINT USING 972.1/K2,
972:#############
980 GOTO 1010

Figure 14. Computer Program for Illustrative Example
(Sheet 2 of 6)
990 MAT PRINT H;
1000 MAT PRINT V;
1010 FOR M=1 TO 10
1020 MAT X=ZER
1030 MAT Y=ZER
1040 MAT W=ZER
1050 FOR N=1 TO 10
1060 IF N=1 THEN 1090
1070 IF N=10 THEN 1090
1080 W(N,N-1)=W(N,N+1)=0
1090 IF N<2 THEN 1110
1100 W(N,N-1)=-W(M,N-1)
1110 IF V>9 THEN 1130
1120 W(N,N+1)=-W(M,N)
1130 X(N,N)=V(M+1,N)
1140 Y(N,N)=-V(M,N)
1150 C3=0
1160 IF N=1 THEN 1180
1170 C3=W(N,N-1)
1180 IF N=10 THEN 1200
1190 C3=C3+W(N,N+1)
1200 W(N,N)=C3+X(N,N)+Y(N,N)
1210 W(N,N)=-W(N,N)
1220 NEXT N
1230 MAT Z=INV(W)
1240 MAT W=Z*X
1250 MAT T=(-1)*W
1260 MAT W=T*(1)
1270 MAT X=Z*Y
1280 MAT T=ZER
1290 MAT T=(-1)*X
1300 MAT X=T*(1)
1310 IF M<>1 THEN 1350
1320 MAT A=Z*C
1330 MAT C=ZER
1340 MAT B=W*(1)
1350 IF M<>2 THEN 1380
1360 MAT C=X*(1)
1370 MAT D=W*(1)
1380 IF M<>3 THEN 1410
1390 MAT E=X*(1)
1400 MAT F=W*(1)
1410 IF M<>4 THEN 1440
1420 MAT G=X*(1)
1430 MAT I=W*(1)
1440 IF M<>5 THEN 1470
1450 MAT J=X*(1)
1460 MAT K=W*(1)
1470 IF M<>6 THEN 1500
1480 MAT L=X*(1)
1490 MAT M=W*(1)
1500 IF M<>7 THEN 1530
1510 MAT N=X*(1)
1520 MAT O=W*(1)

Figure 14. Computer Program for Illustrative Example
(Sheet 3 of 6)
1530 IF M<10 THEN 1560
1540 MAT P=X*(1)
1550 MAT Q=W*(1)
1560 IF M<9 THEN 1590
1570 MAT R=X*(1)
1580 MAT S=W*(1)
1590 IF M<10 THEN 1630
1600 MAT T=ZER
1610 MAT T=X*(1)
1620 MAT U=W*(1)
1630 NEXT M
1640 FOR M=1 TO 10
1650 FOR N=1 TO 10
1660 W(M,N)=X(M,N)*Y(M,N)=Z(M,N)=0
1670 NEXT N
1680 NEXT M
1690 MAT X=C*B
1700 MAT Y=IDN
1710 MAT Z=Y-X
1720 MAT X=INV(Z)
1730 MAT Z=X*D
1740 MAT D=Z*(1)
1750 MAT Z=C*A
1760 MAT C*XZ
1770 MAT X=E*D
1780 MAT Z=Y-X
1790 MAT X=INV(Z)
1800 MAT Z=X*F
1810 MAT F=Z*(1)
1820 MAT Z=E+C
1830 MAT E=X*Z
1840 MAT X=G*F
1850 MAT Z=Y-X
1860 MAT X=INV(Z)
1870 MAT Z=X*I
1880 MAT I=Z*(1)
1890 MAT Z=G*E
1900 MAT G=X*Z
1910 MAT X=J*I
1920 MAT Z=Y-X
1930 MAT X=INV(Z)
1940 MAT Z=X*K
1950 MAT K=Z*(1)
1960 MAT Z=J*G
1970 MAT J=X*Z
1980 MAT X=L*K
1990 MAT Z=Y-X
2000 MAT X=INV(Z)
2010 MAT Z=X*M
2020 MAT M=Z*(1)
2030 MAT Z=L*J
2040 MAT L=X*Z
2050 MAT X=N*M
2060 MAT Z=Y-X

Figure 14. Computer Program for Illustrative Example
(Sheet 4 of 6)
Figure 14. Computer Program for Illustrative Example (Sheet 5 of 6)
2570 MAT Y=X+C
2580 MAT C=Y*(1)
2590 MAT X=B*Y
2600 MAT Y=X+A
2610 MAT A=Y*(1)
2630 R7=A(1,1)/A9
2640 PRINT USING 2650,R7
2650 1

2660 NEXT N1
2665 PRINT
2670 NEXT N2
2675 PRINT
2680 NEXT N3
2685 PRINT
2690 NEXT N4
2695 PRINT
2700 END

Figure 14. Computer Program for Illustrative Example
(Sheet 6 of 6)
Figure 15. Identification and Internal Makeup of Submatrices of Coefficient Matrix
### Line No. | Description
---|---
1100 | Assigns values to first subdiagonal of the coefficient matrix W (see Figure 15d) describing the mth row of the physical nodal pattern.
1120 | Assigns values to first subdiagonal of the coefficient matrix W as in line 1100.
1130 | Assigns values to the X submatrix (see Figure 15c).
1140 | Assigns values to the Y submatrix (see Figure 15b).
1200, 1210 | Enters elements down the main diagonal of the W submatrices.
1230 | First submatrix manipulation statement.
1240 | Normalizes the submatrix to the right of the W matrix diagonal (see Figure 15a).
1270 | Normalizes the submatrix to the left of the W matrix (see Figure 15a).
1280 | Empties T matrix.
1320-1630 | Assigns each of the submatrices, subdiagonal and superdiagonal, mapped by Figure 15a after normalization. Note that matrices X and W are functions of M, the row of the physical nodal pattern.
1640-1680 | Clears matrices W, X, Y, Z, so they may be redefined below by entering only nonzero elements.
1690-2250 | These steps calculate the recurrence coefficients $A_i'$ and $B_i'$ according to equation (10) of Section II. As these are calculated, the superdiagonal and subdiagonal matrices of Figure 15a are sequentially redefined to be these recurrence coefficients.
2320 | This statement enters the boundary condition that the temperatures along the bottom row of the physical matrix are zero.
2330-2610 | Using the recurrence relationship developed in 1690-2250, equation (10) is used to calculate the temperatures, one row (of the physical nodal model) at a time. These (column) matrices of temperatures are calculated in the following order (see Figure 15a) and with the following nomenclature: T, R, P, N, L - J, G, E, C, A.
2630 | $R_7$ is the temperature of node 1, 1 divided by the radius of the heat source $A_9$, defined in statements 420-460.

### D. SAMPLE SOLUTIONS

Figure 16 presents the sample solutions of the illustrative problem. $t_{MAX}$ is the average temperature nodal point 1, 1, see Figure 12. It is obvious that, for these solutions, increasing the number of nodes in the physical model would have the result of increasing the temperature in this hottest element. This was done and the results are discussed below.
Figure 16. Sample Solution of Illustrative Problem
(Sheet 1 of 10)
Figure 16. Sample Solution of Illustrative Problem
(Sheet 2 of 10)
Figure 16. Sample Solution of Illustrative Problem
(Sheet 3 of 10)
Figure 16. Sample Solution of Illustrative Problem
(Sheet 4 of 10)
Figure 16. Sample Solution of Illustrative Problem
(Sheet 6 of 10)
Figure 16. Sample Solution of Illustrative Problem
(Sheet 7 of 10)
Figure 16. Sample Solution of Illustrative Problem
(Sheet 8 of 10)
Figure 16. Sample Solution of Illustrative Problem
(Sheet 9 of 10)
Figure 16. Sample Solution of Illustrative Problem  
(Sheet 10 of 10)
Effect of Increasing the Number of Finite Elements on the Solution of the Illustrative Problem

An exact closed form solution exists for the cylindrical spreading resistance problem in a medium of uniform conductivity. Kennedy (Ref. 10) shows that, as \( \frac{a}{b} \to 0 \),

\[
\frac{(t_{\text{MAX}} - t_s) K_1}{q'' a} \to 1
\]

Figure 17 shows this trend for the finite difference model. With 1600 nodes, \( \frac{(t_{\text{MAX}} - t_s) K_1}{q''} \) was calculated to be 0.9788. With 2500 nodes, the nondimensional resistance dropped to 0.9766; this reduction is attributed to the inexact treatment of the horizontal resistance of the nodes in the inner column using the equation of Jacob (Ref. 9) as described earlier.

\[
\frac{K_1}{K_2} = 1 \quad \frac{H}{b} = 1 \quad a = 1
\]

Figure 17. Maximum Nodal Temperature for Minimum \( \frac{a}{b} \) as a Function of Number of Nodes

It would appear that a nodal model of 900 nodes would be a near optimum number for generating solutions to this particular problem using the finite element approach.
SECTION V
THEORY OF SUPERPOSITION OF SOLUTIONS OF SPREADING THERMAL RESISTANCE PROBLEMS

A. GENERAL

The general steady state heat-conduction equation in Cartesian coordinates, known as the Poisson equation, is given by

$$\frac{\partial^2 t}{\partial x^2} + \frac{\partial^2 t}{\partial y^2} + \frac{\partial^2 t}{\partial z^2} + \frac{q'''(x, y, z)}{K} = 0$$  \hspace{1cm} (14)

where

- \( t = t(x, y, z) \) = temperature (° F)
- \( K \) = thermal conductivity (taken to be independent of temperature and position) (Btu/hr-ft-° F)
- \( q''' \) = internal volumetric heat source (Btu/hr-ft³)
- \( x, y, z \) = Cartesian coordinates

The generalized boundary conditions vary. For example, specified temperature:

\[ t(x, y, z) \bigg|_{x_b, y_b, z_b} = f(x_b, y_b, z_b) \]  \hspace{1cm} (15a)

where:

- \( f \) = specified function
- \( x_b, y_b, z_b \) = values of \( x, y, z \) on the boundary (b)

or, convection to a fluid:

\[ \frac{\partial t}{\partial n} \bigg|_{x_b, y_b, z_b} = h \frac{k}{2} \left[ t(x, y, z) \bigg|_{on \ the \ boundary} - t_f \right] \]  \hspace{1cm} (15b)

where

- \( n \) = outward directed vector normal to the boundary
- \( h \) = Newtonian convective film coefficient (Btu/hr-ft²-° F)
- \( t_f \) = bulk temperature of convecting fluid (° F)
It is convenient to rewrite equations (14), (15a), and (15b) using a temperature difference for the dependent variable that contains a reference temperature. This reference temperature is typically taken to be a specified boundary temperature, as in equation (15a), or the fluid temperature as in equation (15b).

Hence, we define this temperature difference by

$$u \triangleq t - t_{\text{reference}}$$

(16)

Equations (14), (15a), and (15b) can then be rewritten

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} + G = 0$$

(17)

$$u(x, y, z) \bigg|_{x_b', y_b', z_b'} = g(x_b', y_b', z_b')$$

(18a)

$$\frac{\partial u}{\partial n} \bigg|_{x_b', y_b', z_b'} = \frac{h}{K} u(x_b', y_b', z_b')$$

(18b)

where

$$G = q'''/K$$

These equations are linear as can be seen by observing that they contain no products of the dependent variable (u) or its derivatives. Since they are linear, any linearly independent combination of solutions will satisfy these equations due to the distributive property of linear operators, i.e.,

$$L(x_1 + x_2 + \ldots) = L(x_1) + L(x_2) + \ldots$$

where L is a generalized linear operator.

This property may be applied to equation (17) as follows: Take $u_1$ and $u_2$ to be independent solutions to equation (17). Next, define

$$u_3 = a_1 u_1 + a_2 u_2$$

where $a_1 u_1 + a_2 u_2 = 0$ if, and only if, $a_1 = a_2 = 0$, i.e., $u_1$ and $u_2$ are linearly independent.

Now substitute $a_1 u_1$ into equation (17), then $a_2 u_2$ into equation (17), and add the two expressions (using (a) the shorthand operator

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$

and (b) $G_1$ corresponding to the $a_1 u_1$ solution and $G_2$ corresponding to the $a_2 u_2$ solution):

$$\nabla^2 a_1 u_1 + \nabla^2 a_2 u_2 + G_1 + G_2 = 0$$

38
Since $\nabla^2$ is a linear operator, and defining $G = G_1 + G_2$, we can write:
\[
\nabla^2(a_1 u_1 + a_2 u_2) + G = 0
\]
but
\[
a_1 u_1 + a_2 u_2 = u_3
\]
therefore,
\[
\nabla^2 u_3 + G = 0
\]
Thus, $u_3$ is also a solution to equation (17). Applying the same procedure to, say, equation (18b)
\[
\frac{\partial u_1}{\partial n} = \frac{h}{K} u_1
\]
\[
\frac{\partial u_2}{\partial n} = \frac{h}{K} u_2
\]
Adding
\[
\frac{\partial u_1}{\partial n} + \frac{\partial u_2}{\partial n} = \frac{h}{K} (u_1 + u_2)
\]
Since $\partial/\partial n$ is a linear operator, this can be written
\[
\frac{\partial}{\partial n} (u_1 + u_2) = \frac{h}{K} (u_1 + u_2)
\]
But $u_1 + u_2 = u_3$, then
\[
\frac{\partial u_3}{\partial n} = \frac{h}{K} u_3
\]
Thus, $u_3$ also satisfies the boundary condition.

B. ADDITIVE SOLUTIONS

A useful ramification of the superposition principle lies in the fact that the solution to a complicated system may be formed by linear combinations of known solutions.

Consider a rectangular plate with internal heat generation. The governing differential equation and boundary conditions are illustrated in the following sketch:
Explicitly, we have the system
\[ \nabla^2 u(x, y) + G(x, y) = 0 \]
\[ u(x, 0) = f_0(x) \]
\[ u(x, a) = f_\alpha(x) \]
\[ u(0, y) = g_o(y) \]
\[ \nabla u(1, y) = (h/K) u(1, y) \]

The solution may now be written
\[ u(x, y) = u_1(x, y) + u_2(x, y) + u_3(x, y) + u_4(x, y) \]

(20)

The number of ancillary problems is taken equal to the number of nonhomogeneities in the system. Since the governing equation and the first three boundary conditions are not homogeneous, the number of ancillary problems is four.

Substituting equation (20) into equation (19), we obtain the complete system:
\[ \nabla^2 u_1 + \nabla^2 u_2 + \nabla^2 u_3 + \nabla^2 u_4 + G = 0 \]
\[ u_1 + u_2 + u_3 + u_4 = f_0 \]
\[ u_1 + u_2 + u_3 + u_4 = f_\alpha \]
\[ u_1 + u_2 + u_3 + u_4 = g_o \]
\[ \nabla u_1 + \nabla u_2 + \nabla u_3 + \nabla u_4 = \left( \frac{h}{K} \right) \left[ u_1 + u_2 + u_3 + u_4 \right] \text{ at } x = 1, y = y \]

The set of ancillary problems corresponding to this system can now be written as:

<table>
<thead>
<tr>
<th>Problem 1</th>
<th>Problem 2</th>
<th>Problem 3</th>
<th>Problem 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \nabla^2 u_1 + G = 0 )</td>
<td>( \nabla^2 u_2 = 0 )</td>
<td>( \nabla^2 u_3 = 0 )</td>
<td>( \nabla^2 u_4 = 0 )</td>
</tr>
<tr>
<td>( u_1 = 0 )</td>
<td>( u_1 = 0 )</td>
<td>( u_1 = 0 )</td>
<td>( u_1(x, 0) = f_0(x) )</td>
</tr>
<tr>
<td>( u_2 = 0 )</td>
<td>( u_2(x, \alpha) = f_\alpha(x) )</td>
<td>( u_2 = 0 )</td>
<td>( u_2 = 0 )</td>
</tr>
<tr>
<td>( u_3 = 0 )</td>
<td>( u_3 = 0 )</td>
<td>( u_3(0, y) = g_o(y) )</td>
<td>( u_3 = 0 )</td>
</tr>
<tr>
<td>( \nabla u_1(1, y) = \frac{h}{K} u_1(1, y) )</td>
<td>( \nabla u_2(1, y) = \frac{h}{K} u_2(1, y) )</td>
<td>( \nabla u_3(1, y) = \frac{h}{K} u_3(1, y) )</td>
<td>( \nabla u_4(1, y) = \frac{h}{K} u_4(1, y) )</td>
</tr>
</tbody>
</table>

Note that each of the ancillary problems now contains only one nonhomogeneity, a much simpler form. Note also that their sum is equal to equations (21).

Hopefully, we can solve each of the ancillary problems or find the solutions in the literature. Once we have these, we merely add them to obtain the total solution to equations (19).

It was mentioned that once a system has been degenerated to a set of ancillary problems, the final solution is the sum of the individual solutions. This holds, provided the ancillary
problems are properly defined. Care must be exercised in specifying boundary conditions so that the sum of the individual solutions equals the total solution.

Consider, for example, the following three problems (taken from reference 10):

![Diagram of three cases](image)

Heat, which is generated uniformly over a circular disk $S$, spreads by conduction through a cylinder of height $H$ and diameter $D$ to a constant temperature heat sink.

At first glance, the analyst might be inclined to assume that case III is the sum of cases I and II; he would be wrong. To understand why, we must correctly define the boundary conditions on each ancillary problem.

The governing differential equation in each case will be the same

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \quad (22)$$

The boundary conditions will be written in four parts corresponding to the regions $b_1$, $b_2$, $b_3$ and $S$.

**Case I:**

$u_I(b_1) = 0$

$$\frac{\partial u_I}{\partial n}(b_2) = 0 \quad \text{(adiabatic surface)} \quad (23a)$$

$$\frac{\partial u_I}{\partial n}(b_3) = 0$$

$$\frac{\partial u_I}{\partial n}(S) = G_I \quad \text{(constant flux)}$$

where $n$ is an outward directed unit vector normal to the surface.
Case II:
\[ \frac{\partial u_{II}(b1)}{\partial n} = 0 \]
\[ u_{II}(b2) = 0 \]
\[ \frac{\partial u_{II}(b3)}{\partial n} = 0 \]
\[ \frac{\partial u_{II}(S)}{\partial n} = G_{II} \]

Case III:
\[ u_{III}(b1) = 0 \]
\[ u_{III}(b2) = 0 \]
\[ \frac{\partial u_{III}(b3)}{\partial n} = 0 \]
\[ \frac{\partial u_{III}(S)}{\partial n} = G_{III} \]

Now, add the governing differential equations for cases I and II
\[ \frac{\partial^2 u_I}{\partial x^2} + \frac{\partial^2 u_I}{\partial y^2} + \frac{\partial^2 u_{II}}{\partial x^2} + \frac{\partial^2 u_{II}}{\partial y^2} = 0 \]

Regrouping
\[ \frac{\partial^2}{\partial x^2} (u_I + u_{II}) + \frac{\partial^2}{\partial y^2} (u_I + u_{II}) = 0 \]

By our initial assumption \( u_{III} = u_I + u_{II} \). Thus,
\[ \frac{\partial^2 u_{III}}{\partial x^2} + \frac{\partial^2 u_{III}}{\partial y^2} = 0 \]

The differential equation is satisfied.

Next, add boundary conditions, beginning in region b1:
\[ u_I(b1) + \frac{\partial u_{II}(b1)}{\partial n} = 0 \]

Note that we have mixed conditions which are inconsistent. This situation also exists in regions b2 and S (except that in region S we could define \( G_I + G_{II} \triangle G_{III} \) to make the boundary conditions additive. Thus, case III is not the sum of cases I and II; we have, in fact, three nonanalogous systems.
C. FURTHER EXAMPLES

An excellent set of examples of the application of superposition principles in the calculation of thermal spreading resistances may be found in Reference 1 in which the author utilizes Green's function in the calculation of thermal spreading resistances. For a discussion of Green's function see Reference 11.
The two handiest "rules of thumb" relationships that can be used to calculate or estimate thermal spreading resistances were developed by Holm (Ref. 12) and Raillard (Ref. 13). Holm derives the equation for the thermal resistance of a circular isothermal source on the face of a semi-infinite slab as:

\[ R = \frac{1}{4 \pi a k} \]  

where

- \( a \) = radius of circular source
- \( k \) = thermal conductivity of medium

Figure 3 (Section II) shows the temperature profiles described by Holm's equation. Figure 3 shows that 80 percent of the total resistance in a semi-infinite slab occurs within three radii of the source. It can be seen that, when the size of the source is small compared to the thickness of a slab of finite extent, Holm's equation can be used to make a conservative (high) estimate of thermal resistance. Such geometries occur often in microelectronic components.

Raillard presents similar exact solutions for circular and rectangular sources having uniform generation located on the faces of semi-infinite slabs. The equation for the thermal spreading resistance of the circular source bears a close resemblance to Holm's equation:

\[ R = \frac{1}{\pi a k} \]  

In Appendix B of his report, Raillard presents an exact closed-form solution for the rectangular source of uniform generation on a semi-infinite slab. He also derives the solution for uniform generation in a strip of infinite length and finite width on a semi-infinite slab.

Two references treat the case of the rectangular source of finite width and infinite length on a slab of finite depth. Wilcox (Ref. 14) treats the uniform heat generation source, while Gale (Ref. 15) presents thermal spreading resistances for the uniform temperature source. The results of these two studies can also be found in Reference 16.

Muller (Ref. 17) and Kennedy (Ref. 10) present exact solutions to the problem of a circular source at one end of a right cylinder with conduction to the side of the cylinder, the other end of the cylinder, or to both places. Kennedy's source is uniform while the intensity of Muller's circular heat source varies exponentially within the source region.

Hein (Ref. 1) examines steady-state heat transfer in a rectangular substrate or slab having multiple heat sources. He has integrated circuits in mind. He considers convective heat transfer and lead conduction for various heat-sinking conditions and his solutions are mathematically exact.

44
Finally, it might be well to point out a principle which is somewhat analogous to Saint-Venant's Principle in the theory of elasticity. Simply stated, $T$, the temperature in any thermal spreading resistance problem, varies with $1/L$ where $L$ is the distance from the source, when $L$ is large compared with the characteristic dimension of the source. That is,

$$T \propto \frac{1}{L} \quad \text{when } L >> a \quad \text{where } a \text{ is characteristic source dimension}$$

This is indicated by the form of the solution of Fourier's equation for spherical flow, i.e.,

$$q = k \frac{A \, dt}{dt} = \frac{k_m \left(t_1 - t_2\right)}{L_2} \int_{L_1}^{L_2} \frac{r^2 \, dr}{4\pi r^2} = \frac{4\pi k_m \left(t_1 - t_2\right)}{\left(\frac{1}{L_2} - \frac{1}{L_1}\right)}$$
SECTION VII

PLANNED APPLICATION OF COMPUTATIONAL TECHNIQUE

General Electric's Aerospace Electronic Systems Department has developed an extremely compact and thermally efficient packaging configuration for computer circuitry. Integrated circuits are bonded to multilayered printed wiring boards which are in turn bonded to compact forced-air-cooled heat exchangers. This configuration is illustrated in Figures 18 and 19.

A heat transfer analysis program that calculates the temperature of each flatpack using Gauss-Seidel iteration is currently employed. This technique has proven costly: 200+ iterations are required. Computer costs of $25 to $50 per analysis have been experienced.

The exact computational technique described in this report will be implemented in the near future for this type of analysis. Costs are expected to be an order of magnitude lower than those with the iterative technique. (See Figure 9 of Section III.)
Figure 18. Memory Board Module

Figure 19. Location of Conductances and Temperatures
SECTION VIII
RECOMMENDATIONS FOR FUTURE WORK

1. The program should be rewritten for three-dimensional fields. Special attention to such techniques as using the method developed in this report for the main coefficient matrix on the submatrices themselves should be examined. The fact that very large matrices can be efficiently handled by this technique should be capitalized upon.

2. The program as it now stands should be used in a thermal spreading resistance parametric study similar to but larger in scope than the illustrative problem of this study. Nine hundred to sixteen hundred finite elements should be used in such data generation.
REFERENCES


16. References 15 and 16 may be found in General Electric Heat Transfer Design Data Book, Section G502-4, p. 15, January 1968 (this book is both in print and in open literature).

SUBROUTINE MAIN
PARAMETER MAXCOR=1000
COMMON /MEMORY/CORE(MAXCOR)
COMMON /FILES/INFILE,IOFILE,IFILE1,IFILE2
COMMON /TIMES/ITIME(2),IDATE(2),DELTIM
COMMON /DEBUG/IDDEBUG
PARAMETER MAXMAT=69
PARAMETER MAXOFF=4
DIMENSION IOFF(MAXOFF)
DIMENSION KARD(14)
PARAMETER MAXTIT=12
DIMENSION ITITLE(MAXTIT)
PARAMETER MAXTIM=2
DATA KDRUG/SHDEBUG/
DATA IDDEBUG/1/
DATA INFILE/05/
DATA IOFILE/36/
DATA IFFILE/17/
DATA IFFILE/SHUO307/
DATA IFFILE/SH/
DATA ISIZE/HSIZE/
DATA CORE/MAXCOR/
DATA KTITLE/HTITITLE /
CALL FXOPT(67,1,1,0)
CALL FXOPT(68,0,0,0)
CALL FXOPT(69,1,1,0)
CALL FXOPT(70,1,1,0)
CALL FXOPT(71,0,0,0)
READ(INFILE,77)KARD
IF(KARD(1).NE.KTITLE) GO TO 94
CALL SUPERT(MAXTIT,KARD(2))
24 CONTINUE
CALL NEWLIN
WRITE(IOFILE,70)KARD
CALL TIMDAT(ITIME,IDATE)
CALL ELTIME(DELTIM)
CONTINUE READ(INFILE,77)KARD
CALL NEWLIN WRITE(IOFILE,78)KARD FORMAT(2H*,13A6,A2,1H*)
IF(KARD(1).NE.KDEBUG) GO TO 557
IDEBUG=0
GO TO 558

CONTINUE DECODE(KARD,1)KEY,N FORMAT(A6,4X,I5)
IF(KEY.EQ.ISIZE) GO TO 2
CALL NEWLIN WRITE(IOFILE,55)

FORMAT(40H THE ABOVE CARD SHOULD BE A 'SIZE' CARD.)
STOP

IF(N*LT.MAXMAT) GO TO 222
CALL NEWLIN WRITE(IOFILE,223)MAXMAT FORMAT(18H SIZE GREATER THAN,110)
STOP

IF(N.GT.0) GO TO 224
CALL NEWLIN WRITE(IOFILE,225)

FORMAT(17H SIZE LESS THAN 1)
STOP

NSQ=N*N CALL SETNUM(2*N*3)

C RECORDS ARE IN SYSTEM STANDARD RANDOM FORMAT
C WHICH MEANS THAT IF A RECORD IS GREATER THAN 318 WORDS
C THEN THE RECORD WILL BEGIN IN A NEW BLOCK
C AND END A BLOCK EVERY TIME
C
C I TRIED USING PURE DATA RANDOM FILES(11-13-72), BUT
FOR SOME REASON THEY DID NOT SEEM TO WORK PROPERLY.
NBL=NSQ/318+1)*(2*N*3)
CALL NEWLIN WRITE(IOFILE,87)NBL

FORMAT(116,43H BLOCKS OF RANDOM DISC STORAGE ARE REQUIRED)
NBL=NSQ/12+2
CALL NEWLIN WRITE(IOFILE,80)NBLINKS
66 FORMAT(110,42H LINKS OF RAND DISC STORAGE ARE REQUIRED)
   NTIMES=0
61 CALL GETMOR(1, IERR, NLINKS, JFILE1)
   IF(IERR.EQ.0) GO TO 88
   REQUEST WAS REFUSED
   CALL NEWLIN
   WRITE(IOFILE,82)
62 FORMAT(29H REQUEST FOR DISC WAS REFUSED)
   NTIMES=NTIMES+1
   IF(NTIMES.LT.MAXTIM) GO TO 81
   CALL NEWLIN
   WRITE(IOFILE,83), NTIMES
63 FORMAT(29H REQUEST FOR DISC REFUSED. 18,16H TIMES, GIVE UP.)
   STOP
88 CALL SETSIZ(NSQ).
   C SEE IF WE HAVE ENOUGH CORE
   MCORE=5*NSQ + N
   CALL NEWLIN
   WRITE(IOFILE,101), ICORE
101 FORMAT(110,38H WORDS OF CORE ARE CURRENTLY AVAILABLE)
   CALL NEWLIN
   WRITE(IOFILE,102), MCORE
102 FORMAT(110,39H WORDS OF CORE ARE REQUIRED FOR THE JOB)
   IF(MCORE.LE.ICORE) GO TO 3
   GET MORE CORE
   C ICORE IS AUTOMATICALLY UPDATED TO REFLECT THE ACTUAL NUMBER
   OF WORDS THERE ARE UPON RETURN
   C THE ROUTINE DOES NOT FAIL
   C IT keeps TRYING UNTIL IT GETS THE CORE IT WANTS
   CALL NEWLIN
   WRITE(IOFILE,103)
103 FORMAT(33H GET THE ADDITIONAL CORE REQUIRED)
   CALL GIMMF(MCORE, ICORE, CORE)
   C COMPUTE LINEAR OFFSET FOR EACH MATRIX
3   IOFF(1)=1
   DO 4 I=2, MAXOFF
   IOFF(I)=IOFF(I-1)+NSQ
   NUM=N
4   J=2*NUM
   DO 91 I=1, J
   CALL NEWNUM(K)
   CALL FIRST(NUM, CORE(IOFF(1)), NUM, NUM, CORE(IOFF(2)), NUM, NUM,
91
CALL FINISH
STOP
END

SUBROUTINE FIRST(NUM,Q1,NR1,NC1,Q2,NR2,NC2,Q3,NR3,NC3,Q4,NR4,NC4,
Q5,NR5,NC5,IVEC,NVEC)
COMMON /FILES/INFILE,IOFILE,IFILE1,IFILE2
COMMON /DEBUG/DEBUG
DIMENSION IVEC,NVEC)
REAL Q1(NR1,NC1)
REAL Q2(NR2,NC2)
REAL Q3(NR3,NC3)
REAL Q4(NR4,NC4)
REAL Q5(NR5,NC5)
DIMENSION I01(4)
DIMENSION I02(4)
DIMENSION I03(4)
DIMENSION I04(4)
DIMENSION I05(4)
I01(1)=0
I02(1)=0
I03(1)=0
I04(1)=0
I05(1)=0
I01(2)=0
I02(2)=0
I03(2)=0
I04(2)=0
I05(2)=0
I01(3)=NR1
I02(3)=NR2
I03(3)=NR3
I04(3)=NR4
I05(3)=NR5
I01(4)=NC1
I02(4)=NC2
I03(4)=NC3
I04(4)=NC4
I05(4)=NC5
CALL SETUPA(Q1,NR1,NC1,Q2,NR2,NC2,Q3,NR3,NC3,Q4,NR4,NC4,
Q5,NR5,NC5,IVEC,NVEC,NUM)
GENERATE RECURSION COEFFICIENTS A1 AND B1
CALL RSTR(Q1, I01, 1)
CALL RSTR(Q2, I02, 2)
CALL RSTR(Q3, I03, 3)
DO 10 NAME = 3, (NUM = 2 - 1), 2
CALL MMPY(Q3, I03, Q2, I02, Q1, I01, IERR)
CALL MATHZER(QO, I02, NUM)
CALL MSUB(O1, O2, Q1, I01, Q2, I02, IERR)
CALL MINY(QO, NUM, NUM, IVEC, DET)
CALL RSTR(QO, I05, NAME = 1)
CALL MMPY(O1, I03, Q2, I05, Q2, I02, IERR)
CALL MMPY(Q1, IO1, Q3, I02, Q1, I01, IERR)
CALL MMPY(Q2, I02, Q3, I05, Q2, I01, IERR)
CALL MMPY(Q3, I03, Q3, I05, Q1, I01, IERR)
CALL MMPY(Q1, I01, Q1, IO5, Q1, I01, IERR)
10 CONTINUE
IF (DEBUG .NE. 0) GO TO 666
CALL NEWLIN
WRITE ( IOFILE, 566)
566 FORMAT (14H DEBUG POINT C)
CALL MATMAT
666 CONTINUE

USE A1 AND B1 TO SOLVE FOR UNKNOWNS
CALL MATZER(Q1, IO1)
CALL RSTR(Q2, I02, NUM = 2 - 1)
CALL MATZER(Q2, I02)
CALL SAVE(Q2, I02, NUM = 2 - 1)

NAME = NUM - 2
CALL RSTR(Q3, IO3, NAME)

C = S, Q, 0, ... 
CALL MMPY(Q3, IO3, Q2, I02, Q1, I01, IERR)
IF (IERR .NE. 0) CALL QUIT(IERR)

C = 01 = X
CALL RSTR(Q2, I02, NAME = 1)

C = R, P, Q, ... 
CALL MADD(Q1, IO1, Q2, I02, Q2, I02, IERR)

56
IF(IERR .NE. 0) CALL QUIT(IERR)
CALL SAVE(02,102,NAME-1)
CALL R, P PEND
IF(NAME .GE. 4) GO TO 20
IF(IDEBUG .NE. 0) GO TO 667
CALL NEWLIN
WRITE(IOFILE,567)
567 FORMAT(14H DEBUG POINT: D)
CALL MATMAT
667 CONTINUE

C
C PRINT OUT THE RESULTS
CALL NEWPAG
CALL NEWLIN
WRITE(IOFILE,8801)
8801 FORMAT(9H A MATRIX)
CALL RSTR(Q1, IQ11)
CALL PMAT(Q1, IQ11, IQ12, IQ13, IQ14)
CALL NEWPAG
CALL NEWLIN
WRITE(IOFILE,8802)
8802 FORMAT(9H C MATRIX)
CALL RSTR(Q1, IQ13)
CALL PMAT(Q1, IQ11, IQ12, IQ13, IQ14)
CALL NEWPAG
CALL NEWLIN
WRITE(IOFILE,8803)
8803 FORMAT(9H P MATRIX)
CALL RSTR(Q1, IQ1, (NUM-2)*2-1)
CALL PMAT(Q1, IQ11, IQ12, IQ13, IQ14)
CALL NEWPAG
CALL NEWLIN
WRITE(IOFILE,8804)
8804 FORMAT(9H R MATRIX)
CALL RSTR(Q1, IQ1, (NUM-1)*2-1)
CALL PMAT(Q1, IQ11, IQ12, IQ13, IQ14)
CALL NEWPAG
CALL NEWLIN
WRITE(IOFILE,8805)
8805 FORMAT(9H T MATRIX)
CALL RSTR(Q1, IQ1, (NUM-1)*2-1)
CALL PMAT(Q1, IQ11, IQ12, IQ13, IQ14)
SETUP

SUBROUTINE SETUPA(NRH, NCH, NRV, NCV, NRC, NCC, X, NRX, NCX, Y, NRY, NCY, IVECN, IVVEC, NUM)

COMMON /FILES/INFILE, IofilE, FILEI, FILE2
COMMON /DEBUG/DEBUG

DIMENSION IVECN(IVECN)
DIMENSION IVF(NRV, NCV)
DIMENSION C(NRC, NCC)
DIMENSION X(NRX, NCX)
DIMENSION Y(NRY, NCY)
DIMENSION IX(4)
DIMENSION IV(4)
DIMENSION IC(4)
DIMENSION IN(4)
DIMENSION KARD(14)
DATA IHEAT/4HHEAT/

IH(1)=NUM
IH(2)=NUM
IH(3)=NRH
IH(4)=NCH
IV(1)=NUM
IV(2)=NUM
IV(3)=NRV
IV(4)=NCV
IC(1)=NUM
IC(2)=I
IC(3)=NRC
IC(4)=NCC
IX(1)=NUM
IX(2)=NUM
IX(3)=NRX
IX(4)=NCX
IY(1)=NUM
IY(2)=NUM
IY(3)=NRY
IY(4)=NCY
CALL NEWMNUM(INUECN)
CALL NEWMNUM(INUECX)
CALL NEWNUM(INDEXV)
CALL MATZER(H,1H)
CALL MATZER(V,IV)
CALL MATZER(C,IC)
PI=3.14159265
C1=5/PI
R1=PI*(ALOG(2.0))/(2.0*PI)
R2=R1/2.0
H(1,1)=1.0/R1
DO 190 M=2,NUM
   H(M,1)=2.0*H(1,1)
   DO 210 N=1,NUM
      V(N,1)=PI/2.0
   DO 210 M=1,NUM
      V(N,M)=PI*(4.0*FLOAT(N)/FLOAT(N)-1.0))
      IF(N.I.GT.1) GO TO 280
      H(1,N)=5*H(1,1)
      DO 300 N=2,NUM
         H(N,N)=1.0/(1.0/(4.0*PI)*ALOG(FLOAT(N)/FLOAT(N)-1.0))
      V(N,N)=PI*4.0*(FLOAT(N)-1.0)
   280 H(N,N)=5*H(N,N)
   V(M,1)=PI*(4.0*FLOAT(N)-5.0)/2.0
   300 CONTINUE
   DO 310 M=1,NUM
      H(M,NUM)=0.0
   DO 370 N=1,NUM
      V(1,N)=0.0
      C1=1.0,C2=1.0
      DO 460 M=1,NUM
         V(M,1)=C1*V(M,1)
         DO 450 N=1,NUM
            V(N,N)=C3*V(N,N)
            H(N,N)=C2*H(N,N)
         450 CONTINUE
      460 CONTINUE
      READ(INFILE,461,END=488)KARD
      461 FORMAT(13A6,A2)
      CALL NEWLIN
      WRITE(0FILE,464)KARD
      464 FORMAT(2H*13A6,A2,1H*)
      IF(KARD(1).EQ.1HEAT) GO TO 462
      CALL NEWLIN


WRITE(IOFILE,463)
463 FORMAT(35H ABOVE CARD SHOULD BE A 'HEAT' CARD)
STOP

462 DECODE(KARD,463) I1, I2
465 FORMAT(IOX,2:5)
   IF(I1.LE.0.) GO TO 472
   IF(I1.GT. NUM) GO TO 473
   GO TO 479
472 CALL NEWLIN
   WRITE(IOFILE,474)
474 FORMAT(20H 11 IS OUT OF BOUNDS)
   STOP

473 CALL NEWLIN
   WRITE(IOFILE,475)
475 FORMAT(20H 12 IS OUT OF BOUNDS)
   STOP

479 CONTINUE
   DO 466 N=1, NUM
466 C(N,1)=0., 0.
   DO 467 N=1, 12
467 C(N,1)=V(N,1)=2., 0.
   CALL NEWPAG
   CALL NEWLIN
   WRITE(IOFILE,469)
469 FORMAT(26H INITIAL HEAT INPUT MATRIX)
   CALL PMAT(C, IC(1), IC(2), IC(3), IC(4))
   GO TO 471

470 CALL NEWLIN
470 FORMAT(49H UNEXPECTED END OF FILE, EXPECTING A 'HEAT' CARD)
   WRITE(IOFILE,470)
   STOP

471 CONTINUE
   CALL NEWPAG
   CALL NEWLIN
   WRITE(IOFILE,701)
701 FORMAT(9H H MATRIX)
   CALL PMAT(H, IH(1), IH(2), IH(3), IH(4))
   CALL NEWPAG
   CALL NEWLIN
   WRITE(IOFILE,702)
702 FORMAT(9H V MATRIX)
   CALL PMAT(V, IV(1), IV(2), IV(3), IV(4))
   CALL SAVE(C, IC, INDEXC)
CALL SAVE(H,H_INDEXH)
CALL SAVE(V,V_INDEXV)
IF (IDEBUG.NE.0) GO TO 664
CALL NEWLIN
WRITE (IOFILE, 564)
FORMAT (14H DEBUG POINT A)
664 CONTINUE

C
C THE MATRICES ON THE DIAGONAL (AND OF COURSE THEIR INVERSES)
C ARE DIFFERENT ONLY FOR THE FIRST, SECOND, AND LAST TIMES.

DO 1150 M = 1, NUM
   100 IX(1) = NUM
   101 IX(2) = NUM
   CALL MATZER(X, IX)
   102 IY(1) = NUM
   103 IY(2) = NUM
   CALL MATZER(Y, IY)
   IF (IG0.EQ.1 OR M.EQ.2 OR M.EQ.NUM) 100 = 0
   IF (M.EQ.1 OR M.EQ.2 OR M.EQ.NUM) 100 = 0
   IF (IGO.EQ.0) CALL MATZER(C, IC)
   CALL RSTR(H, IH_INDEXH)
   DO 720 N = 1, NUM
      590 IF (N.EQ.1 OR N.EQ.NUM) GO TO 590
      IF (IGO.EQ.0) C(N, N+1) = 0.0
      IF (IGO.EQ.0) C(N, N-1) = 0.0
      590 IF (N.LE.2) GO TO 610
      IF (IGO.EQ.0) C(N, N-1) = H(M, N-1)
      IF (M.EQ.NUM) GO TO 630
      630 IF (IGO.EQ.0) C(N, N+1) = H(M, N)
      IF (M.LE.NUM) GO TO 640
      X(N, N) = V(N+1, N)
      640 Y(N, N) = V(N, N)
      C33 = 0.0
      IF (N.EQ.1) GO TO 680
      C33 = C(N, N+1)
      680 IF (N.EQ.NUM) GO TO 700
      C33 = C33* C(N, N+1)
      IF (IGO.EQ.0) C(N, N) = (C33* X(N, N) + Y(N, N))
      700 CONTINUE
      IF (IGO.EQ.0) CALL MINV(C, NUM, NUM, IVEC, DET)
      CONTINUE
CALL MMPY(C,IC,X,IX,M,1H,1ERR)
IF(IERR.EQ.O) CALL QUIT(IERR)
CALL MNEG(H,1H)

C
SAVE H (B,F,*)
CALL SAVE(H,1H,M*2)
IF(M.GT.1) GO TO 888
CALL RSTR(H,1H,INDEX)
CALL MMPY(C,IC,H,1M,X,IX,1ERR)
IF(IERR.GT.0) CALL QUIT(IERR)
C
SAVE (C,E,G,*)
CALL SAVE(C,1E,1G,M*2-1)
1150 CONTINUE
IF(IDBUG.EQ.O) GO TO 665
CALL NEWLIN
WRITE(IOFILE,565)
565 FORMAT(14H DEBUG POINT *)
CALL MATHAT
CONTINUE
RETURN
C
END SETUP

CSAVE
SAVE
SUBROUTINE SAVE(A,IA,INDEX)
COMMON /FILES/INFLEDOFLILEIFILEIFILE2
DIMENSION A(1)
DIMENSION IA(4)
DIMENSION IR(3)
DATA NRW/O/
DATA KOUNT/0/
IF(INDEX.LT.1 OR INDEX.GT.KOUNT) GO TO 801
IF(IA(1).LE.0) CALL QUIT(1)
IF(IA(2).LE.0) CALL QUIT(1)
C
CHECK TO SEE IF RANDOM RECORD SIZE IS LARGE ENOUGH
IF((IA(1)*IA(2)).GT.MAXWRD) CALL QUIT(1)
IB(1)=IA(1)
IB(2)=IA(2)
IB(3)=1
WRITE(IFILE2*INDEX)IB
NR=IA(1)
NC=IA(2)
NCOLS=IA(4)
I2=NR-1
I3=NCOLS*(NR-1)+1
WRITE(IFILE1*INDEX)((A(I) I=11 I1=I2, I1=1, I3, NCOLS)
NRW=NRW+1
RETURN

ENTRY RSTR(A, IA, INDEX)
IF(INDEX.LT.1.OR.INDEX.GT.KOUNT) GO TO 801
READ(IFILE2*INDEX)IB
IF(IB(3).EQ.0) CALL QUIT(INDEX)
CHECK TO SEE IF THE DIMENSIONS OF THE MATRIX ARE LARGE ENOUGH
IF(IB(1).GT.IA(3)) CALL QUIT(1)
IF(IB(2).GT.IA(4)) CALL QUIT(1)
IA(1)=IB(1)
IA(2)=IB(2)
NR=IA(1)
NC=IA(2)
NCOLS=IA(4)
I2=NR-1
I3=NCOLS*(NR-1)+1
READ(IFILE1*INDEX)((A(I) I=11 I1=I2, I1=1, I3, NCOLS)
NRW=NRW+1
RETURN

ENTRY FINISH
CALL NEWPAG
CALL NEWLIN
WRITE(IOFILE,1111)NRW
1111 FORMAT(110,21H READ-WRITES EXECUTED)
CALL NEWLIN
WRITE(IOFILE,6)
6 FORMAT(12H NORMAL HALT)
STOP

ENTRY SETSIZ(NUMBR)
MAXWRD=NUMBR
CALL RANSIZ(IFILE1, MAXWRD, 0)
CALL RANSIZ(IFILE2, 3)
RETURN

63
ENTRY SETNUM(NUMBR)
MAXKNT=NUMBR
CALL SETDUM(NUMBR)
RETURN

ENTRY NEWNUM(NUMBR)
KOUNT=KOUNT+1
IF(KOUNT.LE.MAXKNT) GO TO 501
WRITE(IOFILE,502)MAXKNT
502 FORMAT(10H MORE THAN,110, 9H MATRICES)
CALL QUIT(1)
501 CONTINUE
NUMBR=KOUNT
IA(1)=0
IB(2)=0
IB(3)=0
WRITE(IOFILE,502)KOUNT
RETURN

ENTRY MATZERO(A,IA)
NSQ=IA(3)*IA(4)
DO 1 I=1,NSQ
1 A(I)=0.0
RETURN

ENTRY MNEG(A,IA)
NSQ=IA(3)*IA(4)
DO 2 I=1,NSQ
2 A(I)=A(I)
RETURN

ENTRY MATIDN(A,IA,NUM)
IF(IA(3).LT.NUM.OR.IA(4).LT NUM) CALL QUIT(1)
IA(1)=NUM
IA(2)=NUM
NSQ=IA(3)*IA(4)
DO 3 I=1,NSQ
3 A(I)=A(3)*I+1
3 A(NSQ)=1.0
RETURN

CALL NEWLIN

WRITE(IOFILE,802)INDEX

FORMAT(20H BAD MATRIX INDEX OF,I10)

CALL QUIT(1)

STOP

END SAVE

END

NEWLIN

SUBROUTINE NEWLIN

COMMON /DATTIM/ITIME(2),IDATE(2)

PARAMETER MAX=20

DIMENSION IHEAD(MAX)

DIMENSION JHEAD(MAX)

DIMENSION IVEC(NWORDS)

DATA JHEAD/MAX*6H

DATA IOFILE/6/

DATA MAXLIN/55/

DATA INAME/1H/

DATA NLINES/3/

DATA IGO1/1/

DATA IPAGE/0/

DATA N2/3/

DATA ILANK/1H/

1002=1

GO TO (1,32),I001

32 IF(NLINES.GT.0) GO TO 5

IPAGE=IPAGE+1

WRITE(IOFILE,18)

18 FORMAT(1M1,/) WRITE(IOFILE,6) ITIME, IDATE,IPAGE

6 FORMAT(9H TIME IS ,2A6 ,4H ON , ,2A6,8X,

15)

WRITE(IOFILE,87)JHEAD

87 FORMAT(10H TITLE.,20A6)

WRITE(IOFILE,7)IHEAD

7 FORMAT(10H SUBTITLE.,20A6)

DO 8 I=1,N2

8 WRITE(IOFILE,9)

9 FORMAT(1:)

NLINES=NLINES+1

5 NLINES=NLINES+1
IF(NLINES.GT.MAXLIN)NLINES=0
RETURN

ENTRY NEWPAG
NLINES=0
RETURN

ENTRY SETHEd(NWORDS, IVEC)
IGN0=2
GO TO (1,2),IGO1
1 CA LL TIMDAT(ITIME, IDATE)
IGO1=2
2 NLINES=0
IPAGE=0
DO 3 1=1,MAX
3 IHEAd(1)=IBlank
GO TO (32,22),IGO2
22 DO 4 I=1,NWORDS
4 IHEAd(1)=IVEC(I)
RETURN

ENTRY SUPERT(NWORDS, IVEC)
DO 80 I=1,NWORDS
80 J=HAd(I)=IVEC(I)
RETURN

C NEWLmN
C END

C QUIT
SUBROUTINE QUIT(IERR)
COMMON /FILES/INFILE,IOFILE,IFILE1,IFILE2
DIMENSION IB(3)
NSO=0
NSQ=NSQ**NSQ
WRITE(IOFILE,21)IERR
21 FORMAT(22H QUIT BECAUSE OF IERR=,110)
IGO=1
GO TO 777

ENTRY MATMAT
IGO=0
777 CONTINUE
WRITE(IOFILE,12)
12 FORMAT(19H MATRIX INFORMATION)
WRITE(10, FORMAT(80H, MATRIX NUMBER, NO. ROWS IN USE, NO. COLS IN USE)
1 USE EVER USED)
DO 10 I=1, MAXKNT
READ(IFILE2, 11), IB
10 WRITE(IOFILE, 11) I, IB
11 FORMAT(4120)
IF(IOO.EQ.0) RETURN
CALL PDUMP
STOP
C
ENTRY SETDUM(ERR)
MAXKNT = ERR
RETURN
C
END QUIT

* INVS
SUBROUTINE TO OBTAIN INVERSE OF MATRIX, CALL DECOM FIRST
CD600D4,007 DATE 05/04/65

SUBROUTINE INVS(A, INTR, MSIZE, NN)
DIMENSION A(MSIZE, MSIZE), INTR(MSIZE)
SUBROUTINE DECOM MUST BE CALLED FIRST
N=NN
IF(INTR(N)) 18, 17, 18
1 DO 13 K=1, N
KM*K-1
IF(KK) 2, 7, 2
2 IF(K=N) 3, 7, 3
COMPLETE REDUCTION BELOW DIAGONAL
3 KP*K=1
DO 6 I=KP, N
X=A(I, K)
IF(X) 4, 6, 4
6 CONTINUE
DIVIDE THRU OF BY PIVOT ELEMENT
7 X=1.0/A(K, K)
A(K, K)=1.0
DO 8 J=1, N
4 DO 5 J=1, KM
5 A(I, J)=A(K, J)*X*A(I, J)
6 CONTINUE
REDUCE TERMS ABOVE DIAGONAL
8 A(K, J)=A(K, J)*X
IF(KM) 9, 13, 9
9 CONTINUE
REPLACE K WITH N
9 CONTINUE

*
9 DO 12 I=1,KM
  X=A(I,K)
  IF(X).LT.1.0,12,10
10   A(I,K)=0.0
  DO 11 J=1,N
11   A(I,J)=A(K,J)*X+A(I,J)
12 CONTINUE
13 CONTINUE

* INTERCHANGE COLUMNS

* KM=N-1 FROM PREVIOUS LOOP
  DO 16 J=1,KM
  KM=J
  IF(KM).LT.1,14,16
14   DO 15 I=1,N
  X=A(I,KM)
  A(I,KM)=A(I,K)
15   A(I,K)=X
16 CONTINUE
17 RETURN
18 KM=N-1
  DO 21 I=2,KM
  KM=INTR(I)
  IF(KM).LT.1,19,21
19   DO 20 J=1,KM
  X=A(I,J)
  A(I,J)=A(KM,J)
20   A(KM,J)=X
21 CONTINUE

GO TO 1

**MINV**

**MOTH**
SUBROUTINE MMPY(A,IDA,B,IBD,C,IDC,IND)
DIMENSION A(1),B(1),C(1),IDA(4),IBD(4),IDC(4)
IA=IDA(1)
JA=IDA(2)
KA=IDA(3)
IB=IBD(1)
JB=IBD(2)
KB=IBD(3)
IC=IDC(1)
JC=IDC(2)
KC=IDC(3)
NC=IDC(4)

IND=0
IF(JA.GE.JB)IND=2
IF((KC.LT.KA).OR.(NC.LT.JB))IND=1
IF(IND.GE.0)RETURN

IND=IND+1

DO 1 I=IA,JA

DO 1 K=IB,KB

C(LC)=0.D

DO 1 J=1,JB

CALL KJ

CLA=MA.CJ-1.

LA=MA.(K-1)

LB=K-1

1 C(LC)=C(LC)+A(LA)*B(LB)

RETURN

END

SUBROUTINE MADD(A,IDA,B,IBD,C,IDC,IND)
DIMENSION A(1),B(1),C(1),IDA(4),IBD(4),IDC(4)
IA=IDA(1)
JA=IDA(2)
KA=IDA(3)
IB=IBD(1)
JB=IBD(2)
KB=IBD(3)
IC=IDC(1)
JC=IDC(2)
KC=IDC(3)
NC=IDC(4)
IF((IA .NE. 18) .OR. (JA .NE. JB)) IND = 2
IF((MC .LT. IA) .OR. (NC .LT. JA)) IND = 1
IF(IND .NE. 0) RETURN
IDC(1) = IDA(1)
IDC(2) = IDA(2)
DO 1 J = 1, JA
JA = JA + 1
LB = MB + JM
LC = MC + JM
DO 1 I = 1, IA
LA = IA + 1
LB = LB + 1
LC = LC + 1
IC(LC) = A(LA) - B(LB)
R E T U R N
1 C(LC) = A(LA) - B(LB)
M A T R I X  S U B S T R A C T  R O U T I N E  05/18/66
D I M E N S I O N A(1), B(1), C(1), IDA(4), IDB(4), IDC(4)
IA = IDA(1)
JA = IDA(2)
IB = IDB(1)
JB = IDB(2)
IC = IDC(1)
JA = IDC(2)
MB = IDA(3)
MB = IDB(3)
MC = IDC(3)
NC = IDC(4)
IND = 0
IF((IA .NE. 1B) .OR. (JA .NE. JB)) IND = 2
IF((MC .LT. IA) .OR. (NC .LT. JA)) IND = 1
IF(IND .NE. 0) RETURN
IDC(1) = IDA(1)
IDC(2) = IDA(2)
DO 1 J = 1, JA
JA = JA + 1
LA = MA + JM
LB = MB + JM
LC = MC + JM
R E T U R N
70
*DECOM* SUBROUTINE TO DECOMPOSE MATRIX FOR SIMULTANEOUS EQUATIONS

SUBROUTINE DECOM(A, INTR, MSIZE, BN)
DIMENSION A(MSIZE, MSIZE), INTR(MSIZE)
MATRIX DECOMPOSITION USED WITH SOLV SUBROUTINE FOR SOLUTION
* IF MATRIX A IS SINGULAR INTR(N) WILL BE SET TO ZERO

N=NN
INTR=1
IN=0

DO 2 I=1, NM
AMAX=ABS(A(J, J))
JP=J+1

IF(A(AMAX)4, 3, 4)

1 AMAX=AT
IN=1
CONTINUE
IF(A(AMAX))4, 3, 4
3 INTR(J)=J
GO TO 11
5 INTR=IN
DO 6 I=J, N
AT=A(J, I)
A(J, I)=A(IN, I)
6 A(IN, I)=AT

7 INTR(J)=IN
AMAX=1.0/A(J, J)
DO 8 I=JP, N
IF(A(I, J))8, 10, 8

8 AT=A(I, J)*AMAX
A(I, J)=AT
DO 9 K=JP, N

END

CD60004.007 DATE 05/04/65
DECOM020
DECOM030
DECOM040
DECOM050
DECOM060
DECOM070
DECOM080
DECOM090
DECOM100
DECOM110
DECOM120
DECOM130
DECOM140
DECOM150
DECOM160
DECOM170
DECOM180
DECOM190
DECOM200
DECOM210
DECOM220
DECOM230
DECOM240
DECOM250
DECOM260
DECOM270
DECOM280
DECOM290
DECOM300
DECOM310
DECOM320
DECOM330
DECOM340
DECOM350
DECOM360
DECOM370
*DECOM*

DECOM380

10 CONTINUE

DECOM390

11 NTR=6

DECOM400

12 INTR(N)=NTR

DECOM410

RETURN

DECOM420

END

DECOM430

*DECOM*

DECOM440

SUBROUTINE DTMN(A,INTR,MSIZE,NN,DET)

DECOM450

DIMENSION A(MSIZE,MSIZE),INTR(MSIZE)

DECOM460

COMPUTES DET, THE DETERMINANT OF THE DECOMPOSED MATRIX

DECOM470

SUBROUTINE DECOM MUST BE CALLED FIRST

DECOM480

INTR(N) WILL CONTAIN INTEGER POWER OF TEN OF MULTIPLYING FACTOR

DECOM490

EPM=1.E38

DECOM500

NE=38

DECOM510

N=NN

DECOM520

NTR=INTR(N)

DECOM530

IF(NTR)2,1,3

DECOM540

1 DTT=0.0

DECOM550

GO TO 10

DECOM560

2 DTT=-10.

DECOM570

GO TO 4

DECOM580

3 DTT=0.1

DECOM590

GO 9 I=1,N

DECOM600

DT=APS(DTT)

DECOM610

GO TO 9

DECOM620

4 IF(DT-1.)699

DECOM630

6 DT=DT*EP

DECOM640

NTR=NTR-NE

DECOM650

GO TO 9

DECOM660

5 IF(DT-1.0)9,9,7

DECOM670

7 IF(DT-1.0)9,9,8

DECOM680

8 DTT=DT/NTR

DECOM690

GO TO 9

DECOM700

9 DTT=DT*A(I,1)

DECOM710

INTR(N)=NTR

DECOM720

GO TO 9

DECOM730

10 DET=DTT

DECOM740

RETURN

DECOM750

END

DECOM760

*DECOM*

DECOM770

SUBROUTINE TO PRINT MATRIX

DECOM780

SUBROUTINE PMAT(A,NR,NC,MM,NN)

DECOM790

DIMENSION A(MM,NN),P(6)

DECOM800

NRW=NR

DECOM810

PMAT0020

PMAT0030

PMAT0040

PMAT0050

PMAT0060
NCOL = NC
1=1
J=1
1
IP=1
JP=J
DO 2 K=1,6
KK=K
P(K)=A(I,J)
J=J+1
IF(J.GT.NCOL) GO TO 2
2 CONTINUE
CALL NEWLIN
WRITE(6,4) I, J, (P()K, K=1,KK)
GO TO 1
3 CALL NEWLIN
WRITE(6,4) I, J, (P(K), K=1, KK)
1=1+1
J=1
IF(1.LE.NROW) GO TO 1
RETURN
END
$GMAP DECK
$INCODE IBMF
LBL GETMOR, GETMOR
TTL GETMOR
SY=M GETMOR
REM TO OBTAIN MORE CORE OR DISC
REM CALL GETMOR(TYPE, RESULT, NUM, FC)
REM TYPE=0 FOR CORE
REM TYPE=1 FOR RANDOM DISC
REM TYPE=2 FOR LINKED DISC
REM RESULT=0 IF SUCCESSFUL
REM RESULT=1 IF UNSUCCESSFUL
REM NUM IS NUMBER OF LINKS DESIRED OR
REM NUMBER OF K (1024 WORDS) DESIRED
REM FC IS THE FILE CODE IN THE FORM 6H0000FC
REM (USED ONLY WHEN GETTING MORE DISC)
GETMOR LDA 4,1*
LDC 2,1*
SBO =6H000001
TMI STORA CURE REQUEST
TZE RANDOM RANDOM DISC REQUEST
<table>
<thead>
<tr>
<th>LDO</th>
<th>5,1*</th>
<th>LINKED DISC REQUEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRA</td>
<td>ORTOA</td>
<td></td>
</tr>
<tr>
<td>RANDOM LDO</td>
<td>5,1*</td>
<td></td>
</tr>
<tr>
<td>ORTOA ORA</td>
<td>=1,DU</td>
<td></td>
</tr>
<tr>
<td>STORA STA</td>
<td>ZERO</td>
<td></td>
</tr>
<tr>
<td>ZERO BSS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>YES LDO</td>
<td>=6H000000</td>
<td></td>
</tr>
<tr>
<td>TRA CONT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NO LDO</td>
<td>=6H000001</td>
<td></td>
</tr>
<tr>
<td>CONT STG</td>
<td>3,1*</td>
<td></td>
</tr>
<tr>
<td>TRA 0,1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>END</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Subroutine to Adjust Size of Specified Array to Specified**

- **New size**, **MME GEMORE** is used to add more words of memory.
- **MME GEMREL** is used to release surplus words of memory.

**Calling Sequence**
- **Call GIMME (NEW,OLD, ARRAY NAME)**

**The Array Specified**
- Must be loaded at the very top of memory.
- Using either A & S use control card or a block data subprogram.

**Warning:** This subroutine changes the second argument to reflect the new size of the array.

**Set Done**
- **GIME** SETIME, set time of day

**Set Done**
- **GIME** SETIME, set time of day

**Save**
- **GIME** SETIME, set time of day

**End**
- **GIME** SETIME, set time of day
MME GELAPS GET PROCESSOR TIME USED TILL NOW IMME0026
STO TEMP2 IMME0027

* CHECK THAT THE REFERENCED ARRAY IS LOADED AT THE TOP OF CORE IMME0028
* SBAR **1 GET THE ADDRESS OF THE TOP OF CORE IMME0029
LDQ **,DL IMME0030
AND #0777,DL IMME0031
GLS 9 IMME0032

EA XO 4,1* GET ADDRESS OF ARRAY IMME0033
STXO ARADR SAVE IMME0034
SBG **,DL SUBTRACT ADDRESS OF ARRAY IMME0035

LDA LOWLOAD CHECK LOWLOAD IMME0036
CA HA #1018,DL INDICATOR IMME0037
THZ LOW ***LOWLOADED IMME0038

* HIGHLOAD PREPROCESS AND CHECKS

LDA =-1,DU SET FLAG IMME0039
STCA RLSFL0,70 TO ALLOW MEMORY RELEASE IMME0040
STO TEMP SAVE ARRAY SIZE IMME0041
SBQ 3,1* SUBTRACT CLAIMED ARRAY SIZE IMME0042
TZE **3 ***EQUAL IMME0043
C MP G 1,DU MAYBE ARRAY WAS LOADED ON ODD LOCATION IMME0044
THZ ERROR *** SORRY, SOMETHING ELSE AT TOP OF CORE IMME0045
LJO TEMP RESTORE ARRAY SIZE IMME0046
TRA GIM ***CONTINUE IMME0047

* LOWLOAD PREPROCESS AND CHECKS

LDA #-1,DU ADDRESS OF ARRAY IMME0048
STCA RLSFL0,70 COMPARISON WITH LOWEST UNUSED IMME0049
STO TEMP SAVE ARRAY SIZE IMME0050
SBQ 3,1* SUBTRACT CLAIMED ARRAY SIZE IMME0051
TZE **3 ***EQUAL IMME0052
CM PG 1,DU MAYBE ARRAY WAS LOADED ON ODD LOCATION IMME0053
LDA LOWLOAD CHECK LOWLOAD IMME0054
LDO TEMP RESTORE ARRAY SIZE IMME0055
TRA GIM ***CONTINUE IMME0056

LDO ARADR ADDRESS OF ARRAY IMME0057
CM PXO LIMITS COMPARISON WITH LOWEST UNUSED IMME0058
TMI ERROR *** ARRAY NOT LOADED ABOVE PROGRAM IMME0059
LX L0 LIMITS ADDRESS OF HIGH UNUSED LIMIT IMME0060
CM PXO ARADR COMPARISON WITH BASE OF ARRAY IMME0061
TMI **4 ***OK LIMIT IS BELOW BASE IMME0062
LDXO ARADR ADJUST LIMIT IMME0063
SBLXO #1,DU TO BE IMME0064
SX LD LIMITS BELOW ARRAY IMME0065
LDA 2,1* COMPUTE FLAG IMME0066
SBA 3,1* INDICATING THAT IMME0067
STCA RLSFL0,70 RELEASE MAY OCCUR (IF NEGATIVE) IMME0068
GIM STC 3,1* MAKE SURE OLD ARRAY SIZE IS CORRECT IMME0069
**COMPUTE THE NUMBER OF 1024 WORD BLOCKS THAT ARE REQUIRED**

**LDC** 2,1*  
**LDS** 3,1*  
**ADD** =1023, DL  
**DRS** 10  
**STO** TEMP  
**TZE** DONE  

*IF GIME IS NOT TO RELEASE CORE, INSERT THE FOLLOWING INSTRUCTION HERE*

**THI** DONE  
**LDA** TEMP  
**ALS** 10  
**ASA** 3,1*  

*IF GIME IS NOT TO RELEASE SURPLUS CORE, REMOVE THE FOLLOWING TWO INSTRUCTIONS.*

**CAPA** 0, DL  
**THI** GIVEUP  

**LOOP TO FETCH REQUIRED CORE IN 2K INCREMENTS**

**LOOP**  
**STO** TEMP3  
**CMPO** 1, DL  
**TZE** LAST  
**TZE** GEMORE  
**ZERO** 0, 2  
**TSX1** CNT  
**LDD** TEMP3  
**SBD** 2, DL  
**TZE** DONE  
**TRA** LOOP  

**LAST**  
**TZE** GEMORE  
**ZERO** 0, 1  
**TSX1** CNT  
**TRA** DONE  

**CNT**  
**AOS** COUNT  
**LDC** 4+1000+064, DL  
**MME** GENAKE  
**TRA** -3, 1  

**SEQUENCE TO RELEASE SURPLUS CORE**
<table>
<thead>
<tr>
<th>IMME0084</th>
<th>GIVEUP SIZ, **, DU</th>
<th>CHECK RELEASE FLAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMME0085</td>
<td>LCQ TEMP **, **, DU</td>
<td>CHECK RELEASE FLAG</td>
</tr>
<tr>
<td>IMME0086</td>
<td>QLS **, 10</td>
<td>RELEASE QUEUED</td>
</tr>
<tr>
<td>IMME0087</td>
<td>EAA DONE</td>
<td>SET RETURN ADDRESS FOR GERMEL</td>
</tr>
<tr>
<td>IMME0088</td>
<td>**, MME GERMEL **</td>
<td>RELEASE CORE</td>
</tr>
<tr>
<td>IMME0089</td>
<td>PRINT STATISTICS FOR THIS CALL TO GIMME</td>
<td></td>
</tr>
</tbody>
</table>

### Completed Tasks

- **DONE**
  - **IMME0090**
    - **IMME0091**
      - **IMME0092**
        - **IMME0093**
          - **IMME0094**
            - **IMME0095**
              - **IMME0096**
                - **IMME0097**
                  - **IMME0098**
                    - **IMME0099**
                      - **IMME0100**
                        - **IMME0101**
                          - **IMME0102**
                            - **IMME0103**
                              - **IMME0104**
                                - **IMME0105**
                                  - **IMME0106**
                                    - **IMME0107**
                                      - **IMME0108**
                                        - **IMME0109**
                                          - **IMME0110**
                                            - **IMME0111**
                                              - **IMME0112**
                                                - **IMME0113**

### Instructions

- **IMME0084**
  - **IMME0085**
    - **IMME0086**
      - **IMME0087**
        - **IMME0088**
          - **IMME0089**
            - **IMME0100**
              - **IMME0101**
                - **IMME0102**
                  - **IMME0103**
                    - **IMME0104**
                      - **IMME0105**
                        - **IMME0106**
                          - **IMME0107**
                            - **IMME0108**
                              - **IMME0109**
                                - **IMME0110**
                                  - **IMME0111**
                                    - **IMME0112**
                                      - **IMME0113**

---

**ERROR CALL**

- **FXEH(#61, MESS, #5)**
RLSFLG EQU GIVEUP
LIMITS BOOL 37
LOWLOD BOOL 24
TEMP RSS 1
TEMP2 RSS 2
TEMP3 RSS 1
COUNT ZERO
MESS GCI 5, ARRAY NOT LOADED ABOVE PROGRAM
FORM GCI 9, TIME, DATE, AND ELAPSED TIME SUBPROGRAM
FILE DEC 06
END $ GMAP DECK, COMDK 77739 011671TIME
$ INCODE IRMF
LBL TIME, TIME AND DATE SUBPROGRAM
TTL DRILL CONVERSION PROGRAM
TTL TIME, DATE, AND ELAPSED TIME SUBPROGRAM
* CALL TIMDAT (TIME, DATE)
* WHERE TIME = 2 CONSECUTIVE WORDS WHERE THE CURRENT TIME WILL BE
* PLACED AS HH:MM:SS
* DATE = 2 CONSECUTIVE WORDS WHERE THE CURRENT DATE WILL BE
* PLACED AS MM/DD/YY
* TIMDAT SAVE
  MME GETIME SET DATE IN A AS MM/DD/YY AND TIME IN Q IN
  REM 64 THS MSEC SINCE MIDDNIWHT.
  STQ TIME SAVE TIME
  LRL 36 PLACE DATE IN Q
  LDA =6H PLACE SPACES IN A
  LLR 12 MOVE MM INTO A
  ALS 6 INSERT SLASH
  ORA =3H68/,,DL
  LLR 12 MOVE DD INTO A
  ALS 6 INSERT SLASH
  ORA =3H68/,,DL
  LDX 3,1 LOAD ADDRESS OF DATE
  STA 0,0 STORE FIRST WORD
  STQ 1,0 AND SECOND WORD
  LDO TIME PUT TIME IN Q
  ORS 6 CONVERT TO MSEC

IMME0115
IMME0116
IMME0117
IMME0010
IMME0020
IMME0030
IMME0040
IMME0050
IMME0060
IMME0070
IMME0080
IMME0090
IMME0100
IMME0110
IMME0120
IMME0130
IMME0140
IMME0150
IMME0160
IMME0170
IMME0180
IMME0190
IMME0200
IMME0210
IMME0220
IMME0230
IMME0240
IMME0250
IMME0260
IMME0270
IMME0280

78
DIV 1000,DL  CONVERT TO SEC  TIME0290
DIV 10,DL  STORE UNIT SECONDS  TIME0300
DIV 6,DL  STORE TENS OF SECONDS  TIME0310
DIV 10,0L  UNIT HRS TO A, TENS OF HRS IN Q  TIME0320
STA TIME  TIME0330
STA TIME+1  STORE UNIT MINUTES  TIME0340
STA TIME+2  STORE TENS OF MINUTES  TIME0350
DIV 15,0L  UNIT HRS TO A, TENS OF HRS IN Q  TIME0370
ALS 30  TIME0380
LLR 6  ASSEMBLE NH IN RIGHT OF Q  TIME0390
QLS 6  TIME0410
ORQ TIME+3  INSERT COLON  TIME0420
QLS 6  TIME0430
ORQ TIME+2  INSERT NH  TIME0440
QLS 6  TIME0460
ORQ =015,0L  INSERT COLON  TIME0470
LDA =6H  PLACE SPACES IN A  TIME0490
LLS 6  TIME0500
ORQ TIME+1  TIME0510
LLS 6  TIME0520
ORG TIME  INSERT SS  TIME0530
LLR 24  LOAD ADDRESS OF TIME  TIME0540
STA 0,0  STORE FIRST AND  TIME0550
STO 1,0  SECOND WORD  TIME0560
RETURN TIMDAT  TIME0570
EJECT  TIME0580

CALL ELTIME(TIMEL)  TIME0590
* WHERE TIMEL = LOCATION WHERE ELAPSED TIME IN MSEC. IS PLACED.

ELTIME MME GELAPS  GET ELAPSED TIME IN Q IN 64'THS MSEC.  TIME0610
ORS 6  CONVERT TO MSEC  TIME0620
STO 2,1  STORE IN TIMEL  TIME0630
RETURN ELTIME  TIME0640
TIME BSS 4  TIME0650
END  TIME0660

EXECUTE  TIME0670
FILE 07  TIME0680
FILE 08,1R  TIME0690
LIMITS 200,17K,,20K  TIME0700
TITLE THERMAL TEST CASE
SIZE 50  TIME0710
HEAT 1 5  TIME0720
END JOB  TIME0730