# NASA Contractor Report 172240



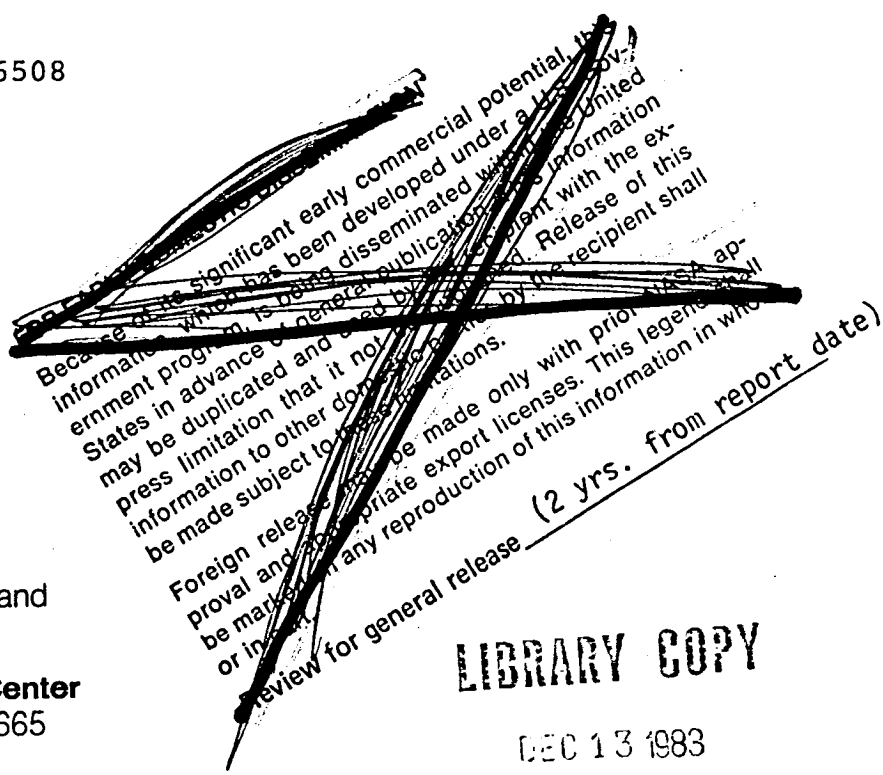
NASA-CR-172240
19860003275

# Transient Thermal Modeling of the Nonscanning ERBE Detector

J. R. Mahan

VIRGINIA POLYTECHNIC INSTITUTE & STATE UNIVERSITY
Blacksburg, Virginia 24061

Contract NAS1-16508
November 1983




National Aeronautics and
Space Administration

**Langley Research Center**
Hampton, Virginia 23665

ENTER:
4B

DISPLAY 26/6/1
86N12743*# ISSUE 3 PAGE 453 CATEGORY 43 RPT#: NASA-CR-172240 NAS 1.26:172240 CNT#: NAS1-16508 83/11/00 107 PAGES UNCLASSIFIED DOCUMENT

UTTL: Transient thermal modeling of the nonscanning ERBE detector TLSP: Fina Report, 13 Nov. 1980 - 30 Sep. 1982

AUTH: A/MAHAN, J. R.

CORP: Virginia Polytechnic Inst. and State Univ., Blacksburg. AVAIL.NTIS

SAP: HC A06/MF A01

CIO: UNITED STATES

MAJS: /*HEAT TRANSFER/*MONTE CARLO METHOD/*THERMODYNAMIC PROPERTIES/*TRANSIENT RESPONSE

MINS: / EARTH RADIATION BUDGET EXPERIMENT/ MATHEMATICAL MODELS/ RADIOMETERS

ABA: E.A.K.

TABLE OF CONTENTS

# Transient Thermal Modeling of the Nonscanning ERBE Detector

## 1. Introduction

This report summarizes the activity at VPI under Contract NAS1-16508 between 13 November 1980 and 30 September 1982. During this period a numerical model of the ERBE wide-field-of-view total radiometer channel was developed, and the model was applied to the TRW calibration procedure. Other activities included consulting with NASA and contractor scientists and engineers on the design and calibration of radiometric instruments and on the interpretation of data from such instruments.

Three major documents have resulted from this work: a mid-term report entitled, Application of the Monte Carlo Method to the Transient Thermal Modeling of a Diffuse-Specular Radiometer Cavity (Leo Eskin's M.S. thesis)[1], a Users' Manual for the cavity transient thermal model program (included as an appendix to reference 1, and the TRW Calibration Simulation Users' Manual , the final revised version which appears in the Appendix of this final report. In addition, a paper was presented at the Fourth AMS Conference on Atmospheric Radiation in Toronto in June of 1981[2].

The final numerical model of the ERBE instrument, the "TRW Calibration Simulation Model", consists of two sub-models which interact with each other within the structure of the overall model. One of these sub-models computes the transient thermal response of the cone

cavity, and the other computes the transient thermal response of the instrument body itself. The details of the cavity sub-model are described in reference 1 and thus are not reiterated here. In the present report we document the instrument body sub-model and its interaction with the cavity sub-model and its surroundings. We indicate in some detail how this model can be modified to take into account the "two-temperature" aspect of the TRW calibration source with its mounting collar, or baffle. We also outline how the "TRW Calibration Model" can be modified and extended to serve as a flight simulation model, including operation as a visible channel. Finally, we suggest a procedure for using the model to establish a real-time interpretation strategy.

## 2. The TRW Calibration Simulation Model

The Users' Manual for the "TRW Calibration Simulation Model" appears in the Appendix of this report. This is the final revised version originally transmitted to NASA Langley by Dave Parekh in September, 1982. At that time Mr. Parekh also sent NASA a magnetic tape containing this model. In this section we describe the parts of this model not already documented in Reference 1. This is also where we suggest the modifications referred to in the Introduction.

### 2.1. TRW FRONT

"TRW FRONT" is a subprogram of the TRW Calibration Simulation Model in which Monte Carlo techniques are used to compute the

distribution factors for radiation among the surfaces of the "front end" of the instrument, between these surfaces and the blackbody calibration source, and between these surfaces and the cavity. By "front end", we mean the field-of-view (FOV) limiter and the surface in the plane of the precision aperture to which the FOV limiter is attached. The general principles of the Monte Carlo technique are described in great detail in Reference 1. Because the logic of "TRW FRONT" is very similar to that of the cavity distribution factor program documented in Reference 1, we will not reiterate the details here. However, the program listing, with notes, appears on pages 47-76 of the Appendix.

The case documented in the Appendix is for the wide field-of-view total channel. It can be easily modified for any other geometry by changing values of the data read by the program, "READ(5,97)", page 49. These data appear after the subroutines which serve "TRW FRONT", on page 75 of the Appendix. Each element of this data set is carefully labeled on page 76, described on page 47, and referred to in Fig. 1 of the February, 1982, Progress Report.

### 2.1.1. Modification to Account for a Two-Temperature Calibration Source

The present version of "TRW FRONT" assumes that the instrument faces a uniform temperature blackbody source. The blackbody calibration source is thus modeled as a perfectly absorbing sector in the plane of the FOV limiter. This is exactly equivalent to any real

geometry black cavity into which the instrument might face as long as the cavity is isothermal. If, however, as in the case of the actual TRW calibration source, the calibration cavity is composed of two surfaces having different temperatures, "TRW FRONT" must be modified to take into account its actual shape. The page-by-page procedure for this modification is outlined below:

1. Page 49 : change NN3 = NN2 + 3

   to NN3 = NN2 + 4

   change MM3 = M3 + 1

   to MM3 = M3 - NTH

2. Page 50 : change

```
    GOTO(101, 102,..., 110),J
101 CONTINUE
       ...
102 CONTINUE
```

   to

```
    GOTO(100, 101, 102,..., 110),J
100 CONTINUE
```

      statements similar to those presently between "101 CONTINUE" and "102 CONTINUE" which establish the x, y and z coordinants of a randomly selected source location on the controlled surface of the calibration cavity

```
101 CONTINUE
```

      statements which establish the x, y and z coordinants of a randomly selected source location on the concentric collar, or baffle, which couples the calibration source to the ERBE instrument

```
102 CONTINUE
```

3. Page 55: change IF(TZ1.EQ.H2) GOTO 201

   to IF(TZ1.GT.H2) GOTO 201

4. Page 56 : change three statements between "201 CONTINUE" and "GOTO 23" to the logic necessary to compute values of UNX, UNY and UNZ, depending on whether the source point is on the blackbody surface itself or on the baffle. This logic depends on the geometry of the calibration source and its interface with the instrument, but should be easy to implement.

5. Page 61 : After "11 CONTINUE" the logic must be changed depending on the geometry of the calibration source/radiometer interface. First, determine if the source point is on the calibration blackbody surface or on the baffle by testing the value of TZI in a manner similar to that on page 60. Then compute TXJ and TYJ in a manner similar to that in the present version of the program. If the source point is on the blackbody surface, begin the search to see if the energy packet has entered the instrument field of view by comparing the value of

DSQRT(TXJ*TXJ + TYJ*TYJ)

with the radius of the FOV limiter at TZJ = H2. If this value exceeds the radius of the FOV limiter, the ray is reabsorbed somewhere within the calibration source, in which case go to "RETURN". However, if this value is

less than the radius of the FOV limiter at TZJ = H2, the energy packet has entered the instrument field of view, in which case the search continues as presently organized under "11 CONTINUE". The logic is similar if the source point is on the baffle rather than the blackbody surface itself.

### 2.1.2. Adaptation to a "Real" Source Field

If the "TRW Calibration Simulation Model" is to be modified as a flight simulator, the calibration blackbody source must be replaced with a "real" source field. This real field must, of course, be time varying. It is impractical to work in terms of distribution factors from the earth scene to the surfaces of the instrument, since these factors would all be essentially zero (the fraction of radiation emitted diffusely from Kansas which is absorbed by a 5-mm square surface in earth orbit is too small to compute). We also have demonstrated that an artificial "grid" placed over the FOV inlet plane to represent the earth scene is unsatisfactory because of its "focusing" effect. The most efficient way to adapt the existing model to a "real" field, such as GENDAT, is outlined below:

1. Use GENDAT (or some equivalent source field model) to compute the flux incident to the disk defined by the FOV limiter, as well as to the disk defined by the precision aperture. These calculations are both within the present capabilities of GENDAT.

2. The difference between these two values is the flux incident to the FOV limiter and the "floor" surrounding the precision aperture. Assume that none of this energy is specularly reflected into the cone cavity, in keeping with the design principle of the FOV limiter. The fraction of this energy which is diffusely reflected into the precision aperture is

$$\sum_{i=1}^{n} D_{i-A} \rho_i^d ,$$

where $D_{i-A}$ is the distribution factor from the ith front end surface element to the disk defined by the precision aperture, computed in "TRW FRONT", and $\rho_i^d$ is the diffuse component of the reflectivity of this element. Note that this calculation assumes that the energy from the scene incident to the front end optics is uniformly distributed across the FOV limiter and floor. This assumption is of minor consequence in view of the fact that it is approximately true for any realistic earth scene, and that the result in any case is a small contribution to the total energy input to the cavity.

3. The fraction of the flux entering the cavity directly from the earth scene which is absorbed by cavity surface element j is

$$D_{A-j} = \epsilon_j \frac{A_j}{A_A} D_{j-A} ,$$

where $A_j$ is the surface area of cavity element j, $\varepsilon_j$ is its diffuse emissivity, $A_A$ is the precision aperture opening area, and $D_{j-A}$ is the distribution factor from cavity element j to the precision aperture opening computed in "TRW NODE". This calculation assumes that the flux from the earth scene incident to the precision aperture opening is diffuse. Although this assumption is not strictly valid, the error associated with it will be negligible in view of the cavity properties.

### 2.1.3. Inclusion of a Filter Dome

The extension of the "TRW Calibration Simulation Model" to include the visible channel would be a major modification. It would involve altering the paths of the energy packets which intercepted the filter dome as follows:

1. Upon the intersection of a ray with the filter dome, a random number would be generated to determine if it was reflected. If not reflected, another random number would be generated to see if it was absorbed or transmitted.
2. If the energy packet is absorbed, the dome element would be treated like any other surface element; that is, the counter for the distribution factor from the source element to the dome element would be incremented.
3. If the energy packet is transmitted, diffraction can be included according to Snell's law.
4. If the energy packet is reflected, we have to generate

another random number to determine if the reflection is specular or diffuse.

a) If specular, treat the dome element like any other surface.

b) If diffuse, the possibility of forward scattering should be included; that is, uniformly distributed scattering in $4\pi$ space rather than $2\pi$ space.

After reflection, or scattering, the energy packet is traced to its next surface just as in the case without a dome.

## 2.2. TRW TEMP

This subprogram computes the transient temperature distributions in the active cavity and body of the instrument based on the distribution factors and conductances computed in the previous subprograms (or user generated in the case of the body elements). As presently configured, this calculation is based on an assumed isothermal blackbody source. Arbitrary heat sink and mounting beam temperatures can be introduced to correspond to actual calibration conditions. A detailed annotated listing of this subprogram appears in the Appendix, pages 77-98. The data listings appear on pages 99-104.

The thermal mass of a typical instrument body node is much larger than that of a typical cavity node. For this reason the temperatures of the body nodes change much more slowly than those of the cavity nodes. Thus, we can realize a certain efficiency by using a "two-timing" technique to compute the transient temperature distributions in the instrument body and on the active cavity surfaces. The body

node temperature calculations are performed only after every tenth cavity node temperature calculation. The radiation heat transfer from the FOV limiter to the active cavity used in each cavity temperature calculation is then based on the quasi-steady temperatures of the FOV limiter nodes. The conduction heat transfer from the cavity nodes to the body nodes is also calculated based on the quasi-steady body node temperatures and stored in a buffer until needed for the next body node temperature calculation, at which time this accumulated energy is distributed to the appropriate body nodes.

### 2.2.1. Detailed Description of the Body Node Temperature Calculation

The active cavity node temperature calculation is well documented in Reference 1. In this section the body node temperature calculation and its interaction with the cavity node temperature calculation is described.

The values of certain constants, defined on page 77 , are established on page 80 . In particular, the user must change "HST", "TCLAMP", "HEAT" and "TSOURC" to conform to the actual calibration conditions. The statements on pages 80-86 establish the values of the constants, coefficients, conductances, distribution factors, and initial temperatures for the calculations. The actual calculations begin on page 86 with the "280" DO-loop, which extends to the end of the subprogram.

The first step in this sequence is to check to see if it is time to do a body temperature calculation ("IF(KOUNT.EQ.10) GOTO 678", page 87 ). The actual body temperature calculations lie between

"678 CONTINUE", page 87 , and "679 CONTINUE", page 88 . Between "219 CONTINUE", page 87 , and "GOTO 219", page 88 , an estimate of the new node temperatures is computed based on the old node temperatures, the input from the cavity, the input from the field (calibration source), and the inputs from the heat sink and mounting beam. This calculation loop is repeated until convergence is obtained to a cavity temperature distribution whose rms value no longer changes from one loop to the next. In this loop, "TGBODY" represents the "guess" for the temperature distribution, and "TUBODY" represents the updated temperature distribution. At the end of each cycle through this loop, the guess is updated and, if there is a significant difference between the update and the preceding guess, the loop is repeated. In the present version of the subprogram a change of one part in a million is tolerated in the rms difference between the last two iterations. This tolerance can be changed by the user in the statement "IF(TEST.LE.0.000001) GOTO 241", page 88.

The updated temperature distribution is actually computed in the "230" DO-loop, pages 87-88. The logic in this loop is "hard wired" in that, if the body shape or node configuration is changed, the logic in this loop must be changed accordingly. These changes would be made in the statements between "220 CONTINUE" and "637 CONTINUE", page 87. Between "637 CONTINUE", page 87, and "230 CONTINUE", page 88, the actual temperature calculation is made based on Newton's method. From the statement "230 CONTINUE", page 88, down to the statement "IF(TEST.LE.0.000001) GOTO 241", page 88, the relative rms change in the temperature distribution is calculated and tested.

If the temperature distribution has converged, the body node temperatures "T1BODY" are changed to their new values, "TUBODY", in the "260" DO-loop, page 88. The clock is updated and the new body temperature distributions are written out in the statements which immediately follow. Finally, the power input to the cavity by radiation from the FOV limiter, "QBOD", is computed between "246 CONTINUE" and "679 CONTINUE", page 88.

The statements after "679 CONTINUE" in which the cavity node temperatures are computed, are for the most part identical to those documented in Reference 1. The main differences are on page 91, where the heat input from the front end, "QBOD", is added ("IF (I.LE.4) Q2 = Q2 + QBOD"), and on page 93, where the heat conduction "QTOT" is buffered from the cavity to the body nodes ("IF(IJK.LE.4) QTOT(LL) = QTOT(LL) + ...").

### 2.2.2. Modifications for the Addition of a Filter Dome

If a filter dome is installed, additional body nodes must be provided. The conductances and distribution factors associated with these new nodes must be calculated and included in the data files. The calculation of the distribution factors would be carried out in "TRW FRONT" as discussed in Section 2.1.3. However, the conductances must be provided by the user. What follows is a page-by-page outline of the modifications required for the inclusion of the filter dome.

1. Page 80: Increase all dimensions "104" to a number which includes the filter dome nodes, and change all dimensions "106" to this new number plus 2 (the number "106" presently refers to the cavity aperture).

Increase the dimension of "SOURCE" to 41 + n, where n is the number of filter dome nodes.

Increase the value of "NBODY" to include the additional filter dome nodes.

2. Page 82: Change the upper limit of the "601" DO-loop to include the filter dome nodes (that is, increase 41 to 41 + n, where n is the total number of filter dome nodes).

3. Page 83: Change the "105" in "IF(J.LT.5) JJ = 105" to the total number of body plus filter dome nodes + 1.

Change "IF(J.EQ.41) JJ = 106" to "IF(J.EQ. number1) JJ = number2", where number1 = 41 + n (n being the number of filter dome nodes) and number2 = total number of body plus filter dome nodes + 2.

After "IF(J.GE.33.AND.J.LT.41) etc.", add a similar statement which converts the dome node position indices (established in the data set) to node numbers.

Change the upper limit in the two "401" DO-loops to the total number of body plus filter dome nodes.

4. Page 87: Between "220 CONTINUE" and "636 CONTINUE", add statements similar to those for the existing body nodes, for the filter dome nodes.

5. Page 88: Change the upper limit of the "246" and "661" DO-loops to the total number of body plus filter dome nodes.
   In the line above "661 CONTINUE", change "106" to the total number of body plus filter dome nodes + 2.
6. Pages 100,101 & 104: Add thermal physical properties and conductance data (user generated) for filter dome nodes.

## 3. Use of the Model to Establish a Real-Time Flight Data Interpretation Strategy

The "TRW Calibration Simulation Model" could be used to establish a real-time flight data interpretation strategy. The first step would be to adjust the model (with the filter dome modifications for the visible channel) until it is in good agreement with the actual instrument in the calibration environment. The next step would be to implement the modifications outlined above to convert the model into a flight simulator. Finally, the "calibrated" model would be used to establish an empirical relation between the actual source energy flux at the instrument, computed independently of the instrument response using, for example, GENDAT, the active cavity heater power signal, and one or more (the more the better) key instrument temperatures. It would be hoped that these "key instrument temperatures" would play only secondary roles in this relation. A logical and yet relatively simple form, consistent with Figs. 1 and 2 of the July, 1982, Progress Report, is

suggested by the superposition theorem[3]:

$$Q_{heater}(t) = A \int_o^t \frac{\partial}{\partial t} Q_{source}(t-\tau)d\tau + B \int_o^t \frac{\partial}{\partial t} T_{HS}(t-\tau)d\tau$$

$$+ C \int_o^t \frac{\partial}{\partial t} T_c(t-\tau)d\tau \ .$$

In this relation, $Q_{heater}(t)$ is the instantaneous heater power (or at least the "count" which represents this power), $Q_{source}(t-\tau)$ is the flux incident to the precision aperture from the source a time delay $\tau$ earlier, $T_{HS}(t-\tau)$ is the heat sink temperature a time delay $\tau$ earlier, and $T_c(t-\tau)$ is the temperature of the mounting beam a time delay $\tau$ earlier. The sensitivity coefficients A, B and C would be obtained by exercising the calibrated model. This relation is inconvenient because $Q_{source}$ appears under an integral. However, it should be possible to write an algorithm capable of inverting this integral in real time.

It is possible that a more complicated model involving more key temperatures will be required. In any case, the numerical model, modified and calibrated as described in this report, should prove very useful in defining this relation.

Once flight data are obtained and interpreted by some data interpretation strategy, the result can be introduced into the complete model to verify that the correct count is indeed obtained.

## 4. References

1. Eskin, L.D., Application of the Monte Carlo Method to the Transient Thermal Modeling of a Diffuse-Specular Radiometer Cavity, Master's Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, August 1981.

2. Mahan, J. R., and L. D. Eskin, "Application of Monte Carlo Techniques to Transient Thermal Modeling of Cavity Radiometers Having Diffuse-Specular Surfaces," Fourth American Meteorological Society Conference on Atmospheric Radiation, Toronto, Ontario, Canada, June 16-18, 1981.

3. Eckert, E. R. G., and R. M. Drake, Jr., Analysis of Heat and Mass Transfer (McGraw-Hill, New York, 1972), p. 170.

# 5. APPENDIX

## Transient Thermal Modeling of the Nonscanning ERBE Detector

## TRW Calibration Simulation

## USERS MANUAL

Final Revision: March 1983

## 1. TRW SHAPE

This program subdivides a TRW-type cavity into NN2*NTH elements, where NN2 is the number of divisions along the cavity axis and NTH is the number of circumferential divisions. When the cavity has been subdivided into elements, the node surface areas and thermal capacities and the internode thermal conductances are computed and stored in online disk files.

This program requires two previously generated data file areas on system disk. The first file name should be named A51961.DATA10 and be of sufficient size to hold M2 + 1 card images. This file will be written to as unit 10. The second file should be named A51961.DATA15 and should be of sufficient size to hold M2*M2 + NN2 + 2 card images. This file will be written to as unit 15.

### 1.1 Nomenclature

| | |
|---|---|
| ZTB | Matrix of the z-coordinate for each node level top boundary. Must be dimensioned NN2. |
| DZ | Matrix of the z-coordinate width of each node level. Must be dimensioned NN2. |
| AA | Matrix of the node surface areas (m**2). Must be dimensioned M2. |
| CC | Matrix of the node thermal capacities (W-s/K). Must be dimensioned M2. |
| VX(I,I) | X-position of the Ith node in Cartesian coordinates. |
| VY(I,I) | Y-position of the Ith node in Cartesian coordinates. |
| VZ(I,I) | Z-position of the Ith node in Cartesian coordinates. |
| VX(I,J) | "Half-conductance" between nodes I and J where I > J. |
| VY(J,I) | "Half-conductance" between nodes I and J where J > I. |

(Note: VX, VY, and VZ must all be dimensioned M2 by M2.)

| | |
|---|---|
| A | Distance from plane of precision aperture to junction of barrel and cone (mm). |
| B | Distance from plane of precision aperture to apex of cone (mm). |
| C | Radius of barrel (mm). |

(Note: For the purpose of stating these dimensions, it is assumed that there is no "space ring" gap between the precision aperture and the cavity opening.)

| | |
|---|---|
| ALPHA | Slope of cone (-). |
| PI | Ratio of circumference to diameter of a circle (-). |
| COND | Thermal conductivity (W/m-K). |
| DENS | Mass density (kg/m**3). |
| SPEC | Specific heat (W-s/kg-K). |
| THCK | Thickness (mm). |
| NARSZ | Number of columns (or rows) in arrays VX, VY, and VZ. This must be the dimensioned value. |
| NTH | The number of circumferential divisions desired. |
| FNTH | Floating point version of "NTH". |
| N1 | Number of axial divisions of barrel. |
| N2 | Number of axial divisions of cavity. |
| FN1, FN2 | Floating point versions of N1 and N2. |
| NN1 | Total number of axial divisions through barrel. |
| NN2 | Total number of axial divisions of cavity. |
| M1 | Total number of cylinder nodes. |
| M2 | Total number of cavity nodes. |
| I | Node number. |
| J | Index of node level. |
| FJ | Real value of J. |
| TZ | Temporary value of z-coordinate of node level J. |
| TZ1 | Temporary value of z-coordinate of node level J-1. |
| TZ2 | Temporary value of z-coordinate of node level J+1. |
| GK1 | "Half-conductance" to node to the "left" of a node on level J. |
| GK2 | "Half-conductance" to node "above" a node on level J, i.e., on level J-1. |
| GK3 | "Half-conductance" to node to the "right" of a node on level J; note that GK3 = GK1. |
| GK4 | "Half-conductance" to node "below" a node on level J, i.e., on level J+1. |
| TZB1 | Temporary z-coordinate of the boundary between nodes on level J and nodes on level J-1. |
| TZB2 | Temporary z-coordinate of the boundary between nodes on level J and nodes on level J+1. |
| R | Radial location of nodes at node level J. |
| RB1 | Radial position of boundary between nodes on level J and nodes on level J-1. |
| RB2 | Radial position of boundary between nodes on level J and nodes on level J+1. |
| AREA | Area occupied by a node at node level J (m**2). |
| J | Auxillary counter defined in "10" DO-loop used to determine which segment of the cavity is being considered. |
| FJ | Floating point version of J. |
| TZ | Temporary z-coordinate of node; later converted to a permanent value, VZ(I,I). |
| R | Radial coordinate of node; later resolved into x- and y-coordinates, VX(I,I) and VY(I,I). |
| THETA | Circumferential angular location of node. |

## 1.2 Description of Program

It is necessary to specify double precision when this program is compiled on IBM equipment. This step may not be necessary with other processors.

```
    IMPLICIT REAL*8 (A-H,O-Z)
```

Dimension subscripted variables. The dimensions shown permit the cavity to be divided into 100 nodes, with 10 divisions in each direction. Of course, these can be increased by the user if desired.

```
    DIMENSION AA(100), CC(100)
    DIMENSION VX(100,100), VY(100,100), VZ(100,100)
    DIMENSION ZTB(10), DZ(10)
```

Establish COMMON for subroutines called.

```
    COMMON /NODE1/ FN1,FN2,N1,N2,M1,M2
    COMMON /NODE2/ FNTH,PI,NTH,NN1,NN2,MM2
```

Read in and write out the cavity dimensions and other constants.

```
    READ(5,97) A,B,C,N1,N2,NTH,NSHOTN,ABS,REFR,DT,ELIMIT
    WRITE(6,96) A,B,C,N1,N2,NTH,NSHOTN,ABS,REFR,DT,ELIMIT
```

Establish the values of the physical constants.

```
    COND = 406.64D0
    ALPHA = C/(B-A)
    SPEC = 234.04D0
    THCK = 2.0D-1
    PI = 3.1415926D0
    DENS = 10524.15D0
```

The cavity is subdivided into NTH*NN2 elements, or nodes, in this portion of the program.

```
    NARSZ = 100
    FNTH = NTH
    FN1 = N1
    FN2 = N2
    NN1 = N1
    NN2 = NN1 + N2
    M1 = NTH*NN1
    M2 = NTH*NN2
    MM2 = M2 + 1
    I = 0
```

In the "10" DO-loop we compute temporary z-coordinates and the r-coordinate of each node. We use computed GO TO statements to direct the calculation to the proper relations depending on which segment of the cavity is being considered (as determined by the value of "J").

```
    DO 10 J = 1, NN2
```

Calculation of the node coordinates, area, thermal capacity and conductances to surrounding nodes is different depending on the surface on which the node is located. Thus we must branch to the appropriate segment of the program.

If the node lies on the barrel, branch to the barrel calculation segment.

```
    IF(J.LE.NN1) GOTO 11
```

If the node lies on the cone, branch to the "cone" calculation segment.

```
    GOTO 12
```

The node is on the barrel; thus we compute the node coordinates, area, and half-conductances for the barrel.

```
 11 CONTINUE
    FJ = J
    TZ = A*(2.0D0*FJ - 1.0D0)/(2.0D0*FN1)
    IF(J.GT.NN2) GOTO 900
    DZ(J) = A/FN1
    ZTB(J) = TZ - DZ(J)/2.0D0
900 CONTINUE
    R = C
    AREA = 2.0D-6*PI*C*A/(FN1*FNTH)
    GK1 = A*COND*THCK*1.0D-3*FNTH/(PI*R*FN1)
    GK2 = COND*PI*R*THCK*4.0D-3/(FNTH*A)
    GK3 = GK1
    GK4 = GK2
    IF(J.EQ.1) GK2 = 0.0D0
    GOTO 17
```

The node is on the cone; thus we compute the coordinates, area, and half-conductances for the cone.

```
 12 CONTINUE
    FJ = J
    TZ = 2*(FN1 + FN2 - FJ) + 1.0D0
    TZ = (A-B)*DSQRT(TZ)/DSQRT(2.0D0*FN2) + B
    R = ALPHA*(B-TZ)
    AREA = PI*C*1.0D-6*DSQRT(C*C + (B-A)*(B-A))
    AREA = AREA/(FNTH*FN2)
    TZ1 = 2*(FN1 + FN2 - FJ)
    TZ1 = (A - B)*DSQRT(TZ1)/DSQRT(2.0D0*FN2)
    TZ1 = TZ1 + B
    IF(J.EQ.NN2) GOTO 800
    TZ2 = 2*(FN1 + FN2 - FJ) - 2.0D0
    TZ2 = (A - B)*DSQRT(TZ2)/DSQRT(2.0D0*FN2)
    TZ2 = TZ2 + B
800 TZB2 = (A - B)*DSQRT((FN1+FN2-FJ)/FN2) + B
    TZB1 = (A - B)*DSQRT((FN1+FN2-FJ+1.0D0)/FN2)
    TZB1 = TZB1 + B
    IF(J.GT.NN2) GOTO 901
    ZTB(J) = TZB1
    DZ(J) = TZB2 - TZB1
901 CONTINUE
802 GK1 =(TZB2-TZB1)*(TZB2-TZB1)
    GK1 = GK1 + ALPHA*ALPHA*(TZB2-TZB1)*(TZB2-TZB1)
    GK1 = COND*THCK*FNTH*DSQRT(GK1)*1.0D-3/(PI*R)
    GK2 = (TZ-TZB1)*(TZ-TZB1)
    GK2 = GK2 + ALPHA*ALPHA*(TZ-TZB1)*(TZ-TZB1)
    GK2 = COND*PI*THCK*2.0D-3*ALPHA*(B-TZB1)
    GK2 = GK2/(DSQRT(GK2)*FNTH)
    GK3 = GK1
    IF(J.NE.NN2) GOTO 803
    GK4 = 0.0D0
    GOTO 17
803 GK4 = (TZB2-TZ)*(TZB2-TZ)
    GK4 = GK4 + ALPHA*ALPHA*(TZB2-TZ)*(TZB2-TZ)
    GK4 = COND*PI*THCK*2.0D-3*ALPHA*(TZB2-TZ)
    GK4 = GK4/(DSQRT(GK4)*FNTH)
    GOTO 17
```

This segment of the program is common to both parts of the cavity, barrel and cone. In this section, the coordinates, half-conductances, and areas are stored, and the thermal capacities are computed and stored.

```
 17 CONTINUE
    DO 10 K = 1, NTH
    FK = K
    I = I + 1
    VZ(I,I) = TZ
    THETA = 6.283185306D0*FK/FNTH
    VX(I,I) = R*DCOS(THETA)
    VY(I,I) = R*DSIN(THETA)
    IF(I.GT.M2) GOTO 936
    AA(I) = AREA
    CC(I) = AREA*THCK*DENS*SPEC*1.0D-3
936 CONTINUE
    IF(J.GT.NN2) GOTO 10
    ICK = I - 1
    ICK = ICK/NTH
    ICK = I - ICK*NTH
    IF(ICK.EQ.1) GOTO 362
    VX(I,I-1) = GK1
    GOTO 363
362 CONTINUE
    VX(I+NTH-1,I) = GK1
363 CONTINUE
    IF(I.LE.NTH) GOTO 364
    VX(I,I-NTH) = GK2
364 CONTINUE
    IF(ICK.EQ.NTH) GOTO 365
    VY(I+1,I) = GK3
    GOTO 366
365 CONTINUE
    VY(I,I-NTH+1) = GK3
366 CONTINUE
    IF(I.GT.(M2-NTH)) GOTO 10
    VY(I+NTH,I) = GK4
 10 CONTINUE
```

(Note: The conductance between node I and adjacent node J is given by VX(I,J)*VY(I,J)/(VX(I,J) + VY(I,J)), where I > J; that is, series conductances add like parallel resistances.)

Write out results.

```
    DO 201 I = 1, M2
    WRITE(10,2) I, AA(I), CC(I)
201 CONTINUE
```

Create a trip card image.

```
      ISTAR = 0
      AASTAR = 0.0D0
      CCSTAR = 0.0D0
      WRITE(10,2) ISTAR, AASTAR, CCSTAR
```

Continue writing.

```
      DO 202 I = 1, M2
      DO 202 J = 1, M2
      WRITE(15,1) I, J, VX(I,J), VY(I,J), VZ(I,J)
202   CONTINUE
```

Create another trip card image.

```
      I = 0
      J = 0
      VDUMMY = 0.0D0
      WRITE(15,1) I, J, VDUMMY, VDUMMY, VDUMMY
```

And write some more.

```
      DO 203 J = 1, NN2
      WRITE(15,2) J, ZTB(J), DZ(J)
203   CONTINUE
```

And one last trip card image.

```
      J = 0.0D0
      ZDUMMY = 0.0D0
      WRITE(15,2) J, ZDUMMY, ZDUMMY
```

Subroutine POSMTX is a subroutine which writes out the node position matrix on the line printer so that the values can be verified by the user. Thus, it is optional and does not have to be called.

```
      CALL POSMTX(VX,VY,VZ,NARSZ)
```

Format statements:

```
 97 FORMAT(F5.2,F5.2,F5.2,I2,I2,I2,I6,F3.1,F3.1,F4.1,
   &       E8.2,4I1)
 96 FORMAT(5X,'THE SYSTEM VARIABLES ARE AS ',
   &       'FOLLOWS:' ,//10X,'A = ',F5.2 ,' MM',
   &       /,10X,'B = ',F5.2,' MM',/,10X,'C = ',
   &       F5.2,' MM',/,10X,'N1 = ',I2,/,10X,'N2 '
   &       ,'= ',I2,/,10X,'NTH =',I2,/,10X,'NSHOTN = ',
   &       I6,/,10X,'ABS = ',F3.1,/,10X,'REFR = '
   &       ,F3.1,/,10X,'DT = ',F4.1,/,10X,'ELIMIT = '
   &       ,E8.2///)
  1 FORMAT(5X, 2I5, 3D14.5)
  2 FORMAT(5X, I5, 2D14.5)
```

End of Program TRW SHAPE.

```
    STOP
    END
```

## 1.3. Subroutine 'POSMTX'

This subroutine rearranges the node coordinate results and writes them out with the appropriate labels so that they can be checked by the user. Use of this subroutine is optional.

```
      SUBROUTINE POSMTX(VX,VY,VZ,NARSZ)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION VX(NARSZ,NARSZ),VY(NARSZ,NARSZ)
      DIMENSION VZ(NARSZ,NARSZ)
      COMMON /NODE1/ FN1,FN2,N1,N2,M1,M2
      COMMON /NODE2/ FNTH,PI,NTH,NN1,NN2,MM2
      WRITE(6,1)
      DO 100 I = 1, M2
      R = DSQRT(VX(I,I)*VX(I,I) + VY(I,I)*VY(I,I))
      PHI = DARCOS(VX(I,I)/R)
      IF(VY(I,I).LT.0.0D0) PHI = 2.0D0*PI - PHI
      WRITE(6,2) I, VX(I,I), VY(I,I), VZ(I,I), R , PHI
  100 CONTINUE
      RETURN
    1 FORMAT('1',5X'NODE',T20,'X-COORDINATE',T40,
     &       'Y-COORDINATE',T60,'Z-COORDINATE',T83,
     &       'RADIUS',T103,'ANGLE'//)
    2 FORMAT(6X,I3,T18,D14.5,T38,D14.5,T58,D14.5,
     &       T80,D14.5,T100,D14.5)
```

End of Subroutine 'POSMTX'.

```
      END
```

## 2. TRW NODES

This program uses Monte Carlo techniques to compute the distribution factors for the radiation emitted from the cavity nodes for a TRW-type cavity radiometer. The "photon" specular reflections are computed "exactly". Program 'TRW SHAPE' must be executed before this program is submitted.

This program requires one previously generated data file area on system disk. The file should be named A51961.DATA12 and should be of sufficient size to hold (M2+1)*NN2 + 1 card images. This file will be written to as unit 12.

### 2.1. Nomenclature

| | |
|---|---|
| DF | Matrix of distribution factors. Must be dimensioned M2 + 1. DF is written out for each source node. |
| XPOS | Matrix of node x-coordinates. Must be dimensioned M2. |
| YPOS | Matrix of node y-coordinates. Must be dimensioned M2. |
| ZPOS | Matrix of node z-coordinates. Must be dimensioned M2. |
| ZTB | Matrix of node level upper boundary z-coordinates. Must be dimensioned NN2. |
| DZ | Matrix of node level z-coordinate widths. Must be dimensioned NN2. |
| Z | Matrix of random numbers generated by 'GGUBS', a subroutine called by Subroutine 'RAND'. |
| A | Distance from plane of precision aperture to junction of barrel and cone (mm). |
| B | Distance from plane of precision aperture to apex of cone (mm). |
| C | Radius of barrel (mm). |

(Note: For the purpose of stating these dimensions it is assumed that there is no "space ring" gap between the precision aperture and the cavity opening.)

| | |
|---|---|
| ALPHA | Slope of cone. |
| NSHOTS | Number of packets into which energy is divided for the Monte Carlo procedure. |
| RSHOTS | Real value of nshots. |
| M2 | Total number of cavity nodes. |
| NTH | Number of circumferential divisions. |
| PI | Ratio of circumference of circle to its diameter. |
| MM2 | Index of aperture. |
| SEED | Seed number needed for random number generator. |
| ABS | Absorptivity. |
| REFR | Ratio of specular component of reflectivity to diffuse + specular (= total) reflectivity. |
| TXI | X-coordinate of the source position. |
| TYI | Y-coordinate of the source position. |
| TZI | Z-coordinate of the source position. |
| TXIOLD | X-coordinate of the previous source position. |
| TYIOLD | Y-coordinate of the previous source position. |
| TZIOLD | Z-coordinate of the previous source position. |
| TXJ | X-coordinate of the receiving position. |
| TYJ | Y-coordinate of the receiving position. |
| TZJ | Z-coordinate of the receiving position. |

(Note: All of the above coordinates (TXI, TYI,..., TZJ) advance with each reflection. Initially TXI, TYI, and TZI are used in a test to determine the first trip through position (TXI,TYI,TZI) by a given "photon".)

| | |
|---|---|
| NSD | Test variable whose value is set to "1" when a specular reflection occurs and to "2" when a diffuse reflection occurs. |

## 2.2. Description of Program

IBM computers require double precision to execute this program. Other compilers may not need this first step:

```
IMPLICIT REAL*8 (A-H, O-Z)
REAL*4 RN
```

Dimension subscripted variables. The subscript range shown below permits the cavity to be divided into up to 100 nodes, with up to 10 divisions in each direction.

```
DIMENSION DF(101), ZTB(10), DZ(10)
DIMENSION XPOS(100),YPOS(100),ZPOS(100)
```

Establish variables common to subroutines called.

```
      COMMON /NODE1/ FN1,FN2,N1,N2,NN1
      COMMON /NODE2/ FNTH,NTH,NN2,M1,M2,MM2
      COMMON /GEOM/ A,B,C,ALPHA,PI
```

Read in and write out cavity dimensions and property values.

```
      READ(5,97) A,B,C,N1,N2,NTH,NSHOTS,ABS,REFR,DT,ELIMIT
      WRITE(6,96) A,B,C,N1,N2,NTH,NSHOTS,ABS,REFR,DT,ELIMIT
```

Establish values of constants.

```
      ALPHA = C/(B-A)
      PI = 3.141592D0
      FNTH = NTH
      FN1 = N1
      FN2 = N2
      NN1 = N1
      NN2 = NN1 + N2
      M1 = NN1*NTH
      M2 = NN2*NTH
      MM2 = M2 + 1
      RSHOTS = NSHOTS
      SEED = 1234567.D0
      RN = -1.0D0
```

Read in values of x-, y- and z-positions of nodes; store as XPOS, YPOS and ZPOS, respectively.

```
   11 CONTINUE
      READ(15,1) I, J, XX, YY, ZZ
      IF(I.NE.J) GOTO 11
      IF(I.EQ.0) GOTO 12
      XPOS(J) = XX
      YPOS(J) = YY
      ZPOS(J) = ZZ
      GOTO 11
   12 CONTINUE
```

Read in values of ZTB and DZ.

```
899 READ(15,4) J, TZBDUM, DZDUM
    IF(J.EQ.0) GOTO 900
    ZTB(J) = TZBDUM
    DZ(J) = DZDUM
    GOTO 899
900 CONTINUE
```

In the "20" DO-loop we loop through the first nodes of each ring of cavity nodes, treating each as a diffuse source. The distribution factors not computed directly in this loop can be determined when needed from symmetry.

```
    I = MM2
    DO 20 J = 1, M2, NTH
```

Zero the 'DF' vector.

```
    DO 9 I1 = 1, MM2
    DF(I1) = 0.0D0
  9 CONTINUE
```

In the "10" DO-loop we allow node J to emit NSHOTS volleys of energy packets and follow them through the cavity until they are either absorbed or leave the cavity through the aperture.

```
    DO 10 ISHOT = 1, NSHOTS
    CALL RAND (SEED, IN, RN)
    JL = J - 1
    JL = JL/NTH + 1
    TZI = ZTB(JL) + RN*DZ(JL)
    TZIOLD = TZI
    TZJ = TZI
    CALL RAND(SEED, IN, RN)
    PHII = (2.0D0*RN + 1.0D0)*PI/FNTH
    IF(J.GT.M1) GOTO 902
    R = C
    GOTO 903
902 R = ALPHA*(B - TZI)
903 CONTINUE
    TXI = R*DCOS(PHII)
    TXIOLD = TXI
    TXJ = TXI
    TYI = R*DSIN(PHII)
    TYIOLD = TYI
    TYJ = TYI
```

We return to this junction ("13 CONTINUE") each time a reflection occurs.

```
 13 CONTINUE
```

If this is the first time this volley has passed through J, the volley striking node J is "emitted" diffusely.

```
    IF((TXI.EQ.TXIOLD).AND.(TYI.EQ.TYIOLD)
   &       .AND.(TZI.EQ.TZIOLD)) GOTO 137
```

Test to see if the energy packet is absorbed or reflected. If the random number RN is less than or equal to ABS, the absorptivity, the packet is absorbed; otherwise it is reflected.

```
    CALL RAND(SEED, IN, RN)
    IF(RN.LE.ABS) GOTO 16
```

Test to see if reflected energy packet is specularly reflected or diffusely reflected. If the random number RN is less than or equal to REFR, the reflectivity ratio, the packet is specularly reflected; otherwise it is diffusely reflected.

```
    CALL RAND(SEED, IN, RN)
    NSD = 1
    IF(RN.LE.REFR) GOTO 17
```

Randomly select the direction of the diffuse reflection.

```
137 CONTINUE
    CALL RAND(SEED, IN, RN)
    IF(RN.EQ.1.0D0) GOTO 14
    THETA = ARSIN(SQRT(RN))
    GOTO 15
 14 CONTINUE
    THETA = 0.0D0
 15 CONTINUE
    CALL RAND(SEED, IN, RN)
    PHI = 2.0D0*PI*RN
    NSD = 2
    GOTO 17
115 CONTINUE
```

Compute the x-, y-, and z-components of the unit vector in the direction (THETA,PHI) with respect to the central coordinate system.

```
      CALL VECTOR(UNX,UNY,UNZ,PHI,THETA,VOX,VOY,VOZ)
```

Call the search subroutine which identifies the receiving point of the diffusely reflected photon.

```
      CALL SEARCH(TXI,TYI,TZI,TXJ,TYJ,TZJ,VOX,VOY,VOZ)
```

If the z-coordinate of the receiving position is negative, the photon has escaped.

```
      IF(TZJ.LT.0.0D0) GOTO 24
```

The photon is reflected to the next position whose coordinates are (TXJ, TYJ, TZJ).

```
      GOTO 25
```

The photon is absorbed. Call Subroutine NODE which determines the absorbing node.

```
   16 CONTINUE
      CALL NODE(ZTB,TXJ,TYJ,TZJ,J1,JJ1,JJ2)
      DF(J1) = DF(J1) + 1.0D0
      GOTO 10
   17 CONTINUE
```

Determine the inward-directed unit normal vector of the source position. This depends on the shape of the source surface.

```
      IF(TZI.LE.A) GOTO 19
```

The reflecting surface is the cone.

```
 18 CONTINUE
    UN = ALPHA*ALPHA*ALPHA*ALPHA
    UN = UN*(B - TZI)*(B - TZI)
    UN = UN + TXI*TXI + TYI*TYI
    UN = DSQRT(UN)
    UNX = - TXI/UN
    UNY = - TYI/UN
    UNZ = - ALPHA*ALPHA*(B - TZI)/UN
    GOTO 23
```

The reflecting surface is the barrel.

```
 19 CONTINUE
    UNX = - TXI/C
    UNY = - TYI/C
    UNZ = 0.0D0
 23 CONTINUE
```

If reflection is diffuse return to the diffuse sequence; otherwise continue in the specular sequence.

```
    IF(NSD.EQ.2) GOTO 115
```

Compute the "ideal" vector of the reflected packet.

```
    VDOT = (TXI-TXIOLD)*UNX
    VDOT = VDOT + (TYI-TYIOLD)*UNY
    VDOT = VDOT + (TZI-TZIOLD)*UNZ
    VOX = TXI - TXIOLD - 2.0D0*VDOT*UNX
    VOY = TYI - TYIOLD - 2.0D0*VDOT*UNY
    VOZ = TZI - TZIOLD - 2.0D0*VDOT*UNZ
    VMAG = DSQRT(VOX*VOX + VOY*VOY + VOZ*VOZ)
    VOX = VOX/VMAG
    VOY = VOY/VMAG
    VOZ = VOZ/VMAG
```

Call the search subroutine which identifies the receiving point of the "exactly" reflected photon.

```
    CALL SEARCH(TXI,TYI,TZI,TXJ,TYJ,TZJ,VOX,VOY,VOZ)
```

If the z-coordinate of the receiving position is negative, the photon has escaped.

```
      IF(TZJ.LT.0.0D0) GOTO 24
```

The photon is reflected to the next position whose coordinates are (TXJ, TYJ, TZJ).

```
      GOTO 25
```

The packet has left the cavity through the aperture. Record and fire another volley.

```
 24 CONTINUE
      DF(MM2) = DF(MM2) + 1.0D0
      GOTO 10
```

A reflection to another surface has occurred. Thus, increment TXI, TYI, TZI, TXIOLD, TYIOLD, TZIOLD, TXJ, TYJ, and TZJ.

```
 25 CONTINUE
      TXIOLD = TXI
      TYIOLD = TYI
      TZIOLD = TZI
      TXI = TXJ
      TYI = TYJ
      TZI = TZJ
```

Loop back to test sequence to determine disposition of photon when it arrives at this new surface.

```
      GOTO 13
 10 CONTINUE
```

Write vector of distribution factors corresponding to node J.

```
      JJ = (J + NTH - 1)/NTH
      DO 30 JK = 1, MM2
      DF(JK) = DF(JK)/RSHOTS
      WRITE(12,3) JJ, JK, DF(JK)
   30 CONTINUE
   20 CONTINUE
```

Generate a "trip" card image at the end of the file.

```
      JJ = 0
      JK = 0
      DFDUM = 0.0D0
      WRITE(12,3) JJ, JK, DFDUM
```

Format statements:

```
    1 FORMAT(5X, 2I5, 3D14.5)
    2 FORMAT(8X, I3)
    3 FORMAT(5X, 2I5, D14.5)
    4 FORMAT(5X, I5, 2D14.5)
   96 FORMAT(5X,'THE SYSTEM VARIABLES ARE AS ',
     &       'FOLLOWS:' ,//10X,'A = ',F5.2 ,' MM',
     &       /,10X,'B = ',F5.2,' MM',/,10X,'C = ',
     &       F5.2,' MM',/,10X,'N1 = ',I2,/,10X,'N2 '
     &       ,'= ',I2,/,10X,'NTH =',I2,/,10X,'NSHOTN = '
     &       ,I6,/,10X,'ABS = ',F3.1,/,10X,'REFR = '
     &       ,F3.1,/,10X,'DT = ',F4.1,/,10X,'ELIMIT = '
     &       ,E8.2///)
   97 FORMAT(F5.2,F5.2,F5.2,I2,I2,I2,I6,F3.1,F3.1,F4.1,
     &       E8.2,4I1)
```

End of Program 'TRW NODES'.

```
      STOP
      END
```

## 2.3. Subroutine SEARCH

This subroutine calculates the magnitude of the vector from the source position to the receiving position of the reflected photon. Since the direction of this vector is known (VOX, VOY, VOZ), the magnitude is all that is needed to calculate the receiving position.

```
      SUBROUTINE SEARCH(TXI,TYI,TZI,TXJ,TYJ,TZJ,VOX,VOY,VOZ)
```

When this subroutine is compiled on IBM equipment, double precision must be specified. This step may not be necessary on other processors.

```
      IMPLICIT REAL*8 (A-H,O-Z)
```

Establish COMMON variables.

```
      COMMON /GEOM/ A,B,C,ALPHA,PI
```

If the source position is on the cone and VOZ is positive, search for the receiving position on the cone. If the z-component of the direction unit vector is zero or positive, the escape check may be skipped. Otherwise, begin the search on the aperture, proceeding to the cylinder and cone as necessary.

```
      IF((TZI.GE.A).AND.(VOZ.GE.0.0D0)) GOTO 10
```

Check to see if the photon has escaped.

```
      IF(VOZ.GE.0.0D0) GOTO 5
      VMAG1 = - TZI/VOZ
      TXJ = TXI + VMAG1*VOX
      TYJ = TYI + VMAG1*VOY
      RADIUS = DSQRT(TXJ*TXJ + TYJ*TYJ)
      IF(RADIUS.GE.C) GOTO 5
      TZJ = - 1.0D0
      RETURN
```

Begin the search for the receiving position on the cylinder.

```
 5 A1 = VOX*VOX + VOY*VOY
   B1 = 2.0D0*(TXI*VOX + TYI*VOY)
   C1 = TXI*TXI + TYI*TYI - C*C
   VMAG2 = (-B1 + DSQRT(B1*B1 - 4.0D0*A1*C1))
   VMAG2 =VMAG2/(2.0D0*A1)
   VMAG3 = (-B1 - DSQRT(B1*B1 - 4.0D0*A1*C1))
   VMAG3 =VMAG3/(2.0D0*A1)
   VMAGB = VMAG2
   IF(VMAGB.LE.0.0D0) VMAGB = VMAG3
   TXJ = TXI + VMAGB*VOX
   TYJ = TYI + VMAGB*VOY
   TZJ = TZI + VMAGB*VOZ
```

Check to see if the receiving position calculated is on the barrel. If not, continue to search on the cone.

```
   IF(TZJ.LE.A) RETURN
```

The receiving position is on the cone.

```
10 A2 = VOX*VOX+VOY*VOY-ALPHA*ALPHA*VOZ*VOZ
   B2 = ALPHA*ALPHA*VOZ*(B-TZI)
   B2 = B2 + TXI*VOX + TYI*VOY
   B2 = B2*2.0D0
   C2 = 2.0D0*B*TZI - TZI*TZI - B*B
   C2 = C2*ALPHA*ALPHA
   C2 = C2 + TXI*TXI + TYI*TYI
   VMAG4 = - B2 + DSQRT(B2*B2 - 4.0D0*A2*C2)
   VMAG4 = VMAG4/(2.0D0*A2)
   VMAG5 = - B2 - DSQRT(B2*B2 - 4.0D0*A2*C2)
   VMAG5 = VMAG5/(2.0D0*A2)
   VMAGC = VMAG4
   IF(VMAGC.LE.0.0D0) VMAGC = VMAG5
```

The equation for the magnitude is quadratic, giving two solutions. The zero or negative root is ignored.

Calculate the receiving position on the cone.

```
   TXJ = TXI + VMAGC*VOX
   TYJ = TYI + VMAGC*VOY
   TZJ = TZI + VMAGC*VOZ
   RETURN
   END
```

## 2.4. Subroutine VECTOR

This subroutine converts the randomly selected diffuse angle with respect to the source position into the central coordinate system.

### 2.4.1. Nomenclature

| | |
|---|---|
| UNX | X-component of inward-directed unit normal vector of source position. |
| UNY | Y-component of inward-directed unit normal vector of source position. |
| UNZ | Z-component of inward-directed unit normal vector of source position. |
| UTX | X-component of unit tangent vector on source position. |
| UTY | Y-component of unit tangent vector on source position. |
| UTZ | Z-component of unit tangent vector on source position. |
| USX | X-component of the mutually orthogonal vector. |
| USY | Y-component of the mutually orthogonal vector. |
| USZ | Z-component of the mutually orthogonal vector. |
| THETA | Angle of diffuse vector with respect to normal vector. |
| PHI | Angle of diffuse vector with respect to tangent vector. |
| VOX | X-component of diffuse vector with respect to central coordinate system. |
| VOY | Y-component of diffuse vector with respect to central coordinate system. |
| VOZ | Z-component of diffuse vector with respect to central coordinate system. |

### 2.4.2. Subroutine Description

```
SUBROUTINE VECTOR(UNX,UNY,UNZ,PHI,THETA,VOX,VOY,VOZ)
```

When this subroutine is compiled on IBM equipment, it is necessary to specify double precision. This step may not be necessary with other compilers.

```
IMPLICIT REAL*8 (A-H, O-Z)
```

Compute the x-, y- and z-components of the unit tangent vector, UT, and the mutually orthogonal vector, US, on the source position.

If the surface is cylindrical, go to 11.

```
      IF(UNZ.EQ.0.0D0) GOTO 11
```

Calculations for the general surface.

```
      UTX = UNX*DSQRT(UNZ*UNZ/(1.0D0 - UNZ*UNZ))
      UTY = UNY*DSQRT(UNZ*UNZ/(1.0D0 - UNZ*UNZ))
      UTZ = DSQRT(1.0D0 - UNZ*UNZ)
      GOTO 13
```

Calculations for the cylindrical surfaces.

```
   11 CONTINUE
      UTX = 0.0D0
      UTY = 0.0D0
      UTZ = 1.0D0
   13 CONTINUE
```

Compute the x-, y- and z-components of the unit vector on the source position which is mutually orthogonal to UN and UT.

```
      USX = UNY*UTZ - UNZ*UTY
      USY = UNZ*UTX - UNX*UTZ
      USZ = 0.0D0
```

Compute components of the vector at angle (THETA, PHI) with respect to the UN,UT,US-coordinate system and express the result in the I,J,K-coordinate system.

```
      SINTH = DSIN(THETA)
      COSTH = DCOS(THETA)
      SINPH = DSIN(PHI)
      COSPH = DCOS(PHI)
      VOX = SINTH*COSPH*UTX + SINTH*SINPH*USX
      VOX = VOX + COSTH*UNX
      VOY = SINTH*COSPH*UTY + SINTH*SINPH*USY
      VOY = VOY + COSTH*UNY
      VOZ = SINTH*COSPH*UTZ + SINTH*SINPH*USZ
      VOZ = VOZ + COSTH*UNZ
```

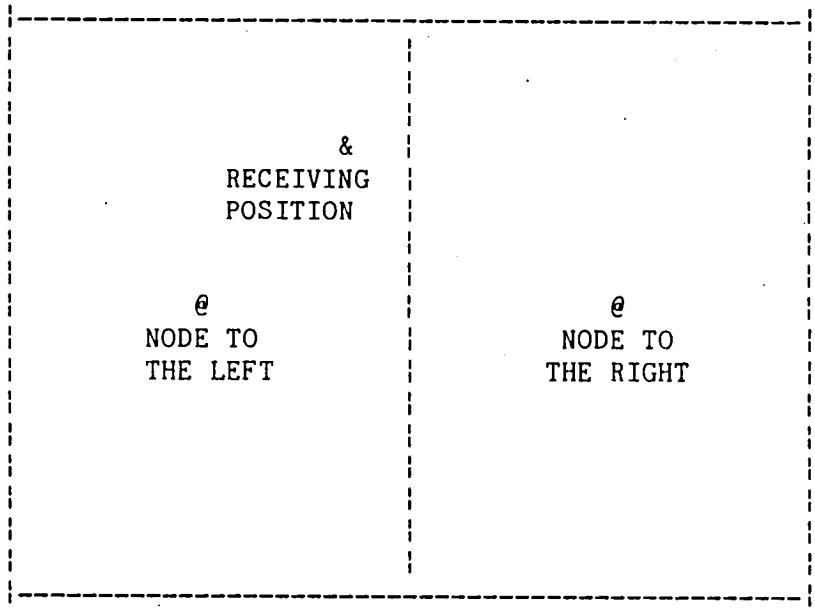
End of Subroutine 'VECTOR'.

```
RETURN
END
```

## 2.5. Subroutine NODE

This subroutine identifies the node which absorbs a given photon.

### 2.5.1. Nomenclature




| | |
|---|---|
| PHI | Circumferential angle of the receiving position. |
| PHILE | Circumferential angle of the left edge of the node to the right. |
| NRA | Number of the node ring "above" the receiving position. |
| NRB | Number of the node ring "below" the receiving position. |
| NLP | Position of the node to the left. |
| NRP | Position of the node to the right. |

### 2.5.2. Subroutine Description

```
SUBROUTINE NODE(ZTB,TXJ,TYJ,TZJ,J1,JJ1,JJ2)
```

When this subroutine is compiled on IBM equipment, it is necessary to specify double precision. This step may not be necessary on other compilers.

```
      IMPLICIT REAL*8 (A-H, O-Z)
```

Dimension the subscripted variables.

```
      DIMENSION ZTB(NN2)
```

Establish COMMON variables.

```
      COMMON /NODE1/ FN1,FN2,FN3,N1,N2,N3,NN1
      COMMON /NODE2/ FNTH,NTH,NN2,NN3,M1,M2,M3,MM2
      COMMON /GEOM/ A,B,C,ALPHA,PI
```

Determine which surface the receiving position lies on. If the receiving position lies on the cone, jump to the cone calculation segment.

```
      IF(TZJ.GT.A) GOTO 11
```

The receiving position is on the cylinder.

```
      PHI = DAR OS(TXJ/C)
      IF(TYJ.LT.0.0D0) PHI = 2.0D0*PI - PHI
      NRA = (1.0D0 + 2.0D0*FN1*TZJ/A)/2.0D0
      NRB = NRA + 1
      NRP = FNTH*PHI/(2.0D0*PI)
      IF(NRP.EQ.0) NRP = NTH
      NLP = NRP + 1
      IF(NLP.EQ.(NTH+1)) NLP = 1
      PHILE = (2.0D0*NRP + 1.0D0)*PI/FNTH
      IF(NRP.EQ.NTH) PHILE = PHILE - 2.0D0*PI
      JJ2 = NRP
      IF(PHI.GT.PHILE) JJ2 = NLP
      JJ1 = NRA
      IF(NRA.NE.0) GOTO 998
      JJ1 = NRB
      GOTO 999
  998 IF(TZJ.GT.ZTB(NRB)) JJ1 = NRB
  999 J1 = (JJ1 - 1.0D0)*NTH + JJ2
      RETURN
```

The receiving position is on the cone.

```
 11 R = DSQRT(TXJ*TXJ + TYJ*TYJ)
    PHI = DAR OS(TXJ/R)
    IF(TYJ.LT.0.0D0) PHI = 2.0D0*PI - PHI
    FNRA = (TZJ-B)*(DSQRT(2.0D0*FN2))/(A-B)
    FNRA = FNRA*((TZJ-B)*(DSQRT(2.0D0*FN2))/(A-B))
    FNRA = FN1 + FN2 - (FNRA-1.0D0)/2.0D0
    NRA = FNRA
    NRB = NRA + 1
    NRP = FNTH*PHI/(2.0D0*PI)
    IF(NRP.EQ.0)NRP = NTH
    NLP = NRP + 1
    IF(NLP.EQ.(NTH+1)) NLP = 1
    PHILE = (2.0D0*NRP + 1.0D0)*PI/FNTH
    IF(NRP.EQ.NTH) PHILE = PHILE - 2.0D0*PI
    JJ2 = NRP
    IF(PHI.GT.PHILE) JJ2 = NLP
    JJ1 = NRA
    IF(NRA.EQ.NN2) GOTO 997
    IF(TZJ.GT.ZTB(NRB)) JJ1 = NRB
997 J1 = (JJ1 - 1.0D0)*NTH + JJ2
```

End of Subroutine 'NODE'.

```
    RETURN
    END
```

## 2.6. Subroutine RAND

This subroutine uses library Subroutine GGUBS to obtain random numbers in an efficient way. A sequence of 100 random numbers is generated each time the 100th number from the previous generation is used.

### 2.6.1. Nomenclature

| | |
|---|---|
| SEED | A seed number required by GGUBS. It must be specified in the main program as a double precision integer value. |
| IN | An integer between 1 and 100 inclusive which represents which of the 100 random numbers of a given generation is obtained. |
| RN | The random number passed back to MAIN. |

### 2.6.2. Subroutine Description

```
SUBROUTINE RAND(SEED, IN, RN)
```

When this subroutine is compiled on IBM equipment, it is necessary to specify double precision. This step may not be necessary with other processors.

```
IMPLICIT REAL*8 (A-H,O-Z)
REAL*4 RN, Z(100)
```

If this is the first call to RAND, RN is equal to the negative value set in MAIN. In this case we must call GGUBS to generate the first generation of 100 random numbers.

```
IF(RN.LT.0.0D0) GOTO 11
```

If this is not the first call to RAND, then increment to the next value in this generation of 100 random numbers.

```
IN = IN + 1
```

If we have not yet used up our allotment of random numbers from this generation, we simply take the next one in the sequence back to MAIN.

```
    IF(IN.LE.100) GOTO 13
```

However, if we have used all the members of this generation, we must create a new generation of 100 random numbers.

```
 11 CONTINUE
    IN = 1
    DSEED = SEED
    NUMBER = 100
    CALL GGUBS(DSEED, NUMBER, Z)
    SEED = DSEED
 13 CONTINUE
    RN = Z(IN)
```

End of Subroutine 'RAND'.

```
    RETURN
    END
```

## 3. TRW FRONT

This program uses Monte Carlo techniques to compute the distribution factors for the radiation emitted from the 'front end' nodes for a TRW-type cavity radiometer. The specular reflections are computed "exactly".

This program requires one previously generated data file area on system disk. The file should be named A51961.DATA17 and should be of sufficient size to hold (M2+1)*NN2 + 1 card images. This file will be written to as unit 17, and will contain the distribution factor matrix after the program has been executed.

### 3.1. Nomenclature

| | |
|---|---|
| DF | Matrix of distribution factors. Must be dimensioned M2 + 1. DF is written out for each source node. |
| H1 | Distance to floor, or face, of instrument (cm). |
| H2 | Distance to face of F-O-V limiter (cm). |
| R1 | Radius which describes the inside face of the F-O-V limiter; also outside radius of the face of the instrument (cm). |
| R2 | Inside radius of the face of the instrument (cm). |
| R3 | Radius of precision aperture (cm). |
| F1 | Radial distance to the inner edge of the first floor node (cm). |
| F2 | Radial distance to the inner edge of the second floor node (cm). |
| F3 | Radial distance to the inner edge of the third floor node (cm). |
| FOV1 | Axial distance from the floor level (Z = H1) to the lower edge of the top F-O-V limiter node (cm). |
| FOV2 | Axial distance from the floor level (Z = H1) to the lower edge of the central F-O-V limiter node (cm). |
| EFOV | Emissivity (= absorptivity = 1 - reflectivity) of the face of the F-O-V limiter (-). |
| RFOV | Ratio of specular to specular + diffuse reflectivity for the face of the F-O-V limiter (-). |
| EF | Emissivity (= absorptivity = 1 - reflectivity) of the face of the instrument (-). |
| RF | Ratio of specular to specular + diffuse reflectivity for the face of the instrument (-). |
| NSHOTS | Number of "photons" emitted in Monte Carlo method. |
| RSHOTS | Real value of "NSHOTS". |

(Note: The distances H1, H2, R1, R2, R3, and R4 are all measured with respect to the origin of coordinates (0,0,0) located on the axis in the plane of the precision aperture. See Fig. 1 of the February Progress Report.)

| | |
|---|---|
| PI | Ratio of circumference to diameter of a circle (-). |
| NTH | Number of circumferential divisions (set at 4). |
| FNTH | Floating point version of NTH. |
| N1 | Number of axial divisions on F-O-V limiter (-). (set at 3). |
| FN1 | Floating point version of N1. |
| N2 | Number of radial divisions on floor (set at 4). |
| FN2 | Floating point version of N2. |
| NN2 | Number of axial divisions on F-O-V limiter plus number of radial divisions on floor (-). |
| NN3 | Total number of "front end" nodes divided by NTH. |
| M1 | Total number of nodes on F-O-V limiter (-). |
| M2 | Total number of nodes on the face of the instrument including F-O-V limiter and floor (-). |
| M3 | Total number of nodes (= M1 + M2 + 3*NTH), including the edge of the floor, the top surface of the precision aperture, and the imaginary pie shaped segment in the plane of the F-O-V limiter. |
| SEED | Seed number needed for the random number generator. |
| RN | Uniformly distributed random number between 0 and 1 generated by Subroutine RAND. |
| MM3 | Total number of possible receiving nodes, including cavity opening. |
| TXI | X-coordinate of the source position. |
| TYI | Y-coordinate of the source position. |
| TZI | Z-coordinate of the source position. |
| TXIOLD | X-coordinate of the previous source position. |
| TYIOLD | Y-coordinate of the previous source position. |
| TZIOLD | Z-coordinate of the previous source position. |
| TXJ | X-coordinate of the receiving position. |
| TYJ | Y-coordinate of the receiving position. |
| TZJ | Z-coordinate of the receiving position. |

## 3.2. Program Description

When this program is compiled on IBM equipment, double precision must be specified. This step may not be necessary on other processors.

```
IMPLICIT REAL*8 (A-H, O-Z)
REAL*4 RN
```

Dimension the subscripted variable.

```
DIMENSION DF(41)
```

Identify the COMMON variables for the subroutines called.

```
      COMMON H1,H2,R1,R2,R3,FOV1,FOV2,F1,F2,F3,PI
```

Read in and write out values of front end dimensions and optical properties.

```
      READ(5,97) NSHOTS,H1,H2,R1,R2,R3,F1,F2,F3,FOV1,FOV2,
     &       EFOV,RFOV,EF,RF
      WRITE(6,96) H1,H2,R1,R2,R3,F1,F2,F3,FOV1,FOV2,
     &       EFOV,RFOV,EF,RF,NSHOTS
```

Set values of constants.

```
      PI = 3.141592D0
      NTH = 4
      FNTH = NTH
      N1 = 3
      FN1 = N1
      N2 = 4
      FN2 = N2
      NN2 = N1 + N2
      NN3 = NN2 + 3
      M1 = N1*NTH
      M2 = NN2*NTH
      M3 = NN3*NTH
      MM3 = M3 + 1
      RSHOTS = NSHOTS
      SEED = 1234567.D0
      RN = -1.0D0
```

In the "20" DO-loop we loop through the first nodes of each ring of NTH nodes, treating each as a diffuse source. The distribution factors not computed directly in this loop can be determined when needed from symmetry.

```
      DO 20 J = 1, NN3
      JJ = (J-1)*NTH + 1
```

Zero the "DF" vector.

```
      DO 9 I1 = 1, MM3
      DF(I1) = 0.0D0
    9 CONTINUE
```

In the "10" DO-loop we allow node JJ to emit NSHOTS volleys of energy packets and follow them until they are either absorbed or leave the instrument.

```
    DO 10 ISHOT = 1, NSHOTS
```

Locate emission point on node JJ.

```
    CALL RAND (SEED, IN, RN)
    RN1 = RN
    CALL RAND(SEED,IN,RN)
    RN2 = RN
    GOTO(101,102,103,104,105,106,107,108,109,110), J
101 CONTINUE
```

The source point is on the blackbody, which we model as a black sector in the plane of the F-O-V limiter.Note that this limits the analysis to a reasonably uniform calibration source.

```
    ANGLE = RN1*PI/2.0D0
    RADIUS = DSQRT(R1*R1 - H2*H2)
    RADIUS = RADIUS*DSQRT(RN2)
    TXI = RADIUS*DCOS(ANGLE)
    TYI = RADIUS*DSIN(ANGLE)
    TZI = H2
    GOTO 111
```

The source point is on the upper ring of the F-O-V limiter face.

```
102 CONTINUE
    ANGLE1 = RN1*PI/2.0D0
    RAD1 = DSQRT(R1*R1 - H2*H2)
    A = DARSIN(RAD1/R1)
    RAD2 = DSQRT(R1*R1 - (H1+FOV1)*(H1+FOV1))
    B = DARSIN(RAD2/R1)
    ANGLE2 = DARCOS(DCOS(A) - RN2*(DCOS(A)-DCOS(B)))
    TXI = R1*DSIN(ANGLE2)*DCOS(ANGLE1)
    TYI = R1*DSIN(ANGLE2)*DSIN(ANGLE1)
    TZI = R1*DCOS(ANGLE2)
    GOTO 111
```

The source point is on the middle ring of the F-O-V limiter face.

```
103 CONTINUE
    ANGLE1 = RN1*PI/2.0D0
    RAD1 = DSQRT(R1*R1 - (H1+FOV1)*(H1+FOV1))
    A = DARSIN(RAD1/R1)
    RAD2 = DSQRT(R1*R1 - (H1+FOV2)*(H1+FOV2))
    B = DARSIN(RAD2/R1)
    ANGLE2 = DARCOS(DCOS(A) - RN2*(DCOS(A)-DCOS(B)))
    TXI = R1*DSIN(ANGLE2)*DCOS(ANGLE1)
    TYI = R1*DSIN(ANGLE2)*DSIN(ANGLE1)
    TZI = R1*DCOS(ANGLE2)
    GOTO 111
```

The source point is on the bottom ring of the F-O-V limiter face.

```
104 CONTINUE
    ANGLE1 = RN1*PI/2.0D0
    RAD1 = DSQRT(R1*R1 - (H1+FOV2)*(H1+FOV2))
    A = DARSIN(RAD1/R1)
    RAD2 = DSQRT(R1*R1 - H1*H1)
    B = DARSIN(RAD2/R1)
    ANGLE2 = DARCOS(DCOS(A) - RN2*(DCOS(A)-DCOS(B)))
    TXI = R1*DSIN(ANGLE2)*DCOS(ANGLE1)
    TYI = R1*DSIN(ANGLE2)*DSIN(ANGLE1)
    TZI = R1*DCOS(ANGLE2)
    GOTO 111
```

The source point is on the first (outer) ring of the floor.

```
105 CONTINUE
    ANGLE = RN1*PI/2.0D0
    RAD2 = DSQRT(R1*R1 - H1*H1)
    RADIUS = DSQRT(F1*F1 + RN2*(RAD2*RAD2 - F1*F1))
    TXI = RADIUS*DCOS(ANGLE)
    TYI = RADIUS*DSIN(ANGLE)
    TZI = H1
    GOTO 111
```

The source point is on the second ring of the floor.

```
106 CONTINUE
    ANGLE = RN1*PI/2.0D0
    RADIUS = DSQRT(F2*F2 + RN2*(F1*F1-F2*F2))
    TXI = RADIUS*DCOS(ANGLE)
    TYI = RADIUS*DSIN(ANGLE)
    TZI = H1
    GOTO 111
```

The source point is on the third ring of the floor.

```
107 CONTINUE
    ANGLE = RN1*PI/2.0D0
    RADIUS = DSQRT(F3*F3 + RN2*(F2*F2-F3*F3))
    TXI = RADIUS*DCOS(ANGLE)
    TYI = RADIUS*DSIN(ANGLE)
    TZI = H1
    GOTO 111
```

The source point is on the fourth (inner) ring of the floor.

```
108 CONTINUE
    ANGLE = RN1*PI/2.0D0
    RADIUS = DSQRT(R2*R2 + RN2*(F3*F3-R2*R2))
    TXI = RADIUS*DCOS(ANGLE)
    TYI = RADIUS*DSIN(ANGLE)
    TZI = H1
    GOTO 111
```

The source point is on the inner edge of the floor.

```
109 CONTINUE
    ANGLE = RN1*PI/2.0D0
    ZLEVEL = RN2*H1
    TXI = R2*DCOS(ANGLE)
    TYI = R2*DSIN(ANGLE)
    TZI = ZLEVEL
    GOTO 111
```

The source point is on the precision aperture.

```
110 CONTINUE
    ANGLE = RN1*PI/2.0D0
    RADIUS = R3*DSQRT(RN2)
    TXI = RADIUS*DCOS(ANGLE)
    TYI = RADIUS*DSIN(ANGLE)
    TZI = 0.0D0
```

A reflection to another surface has occurred; thus, increment TXIOLD, TYIOLD, TZIOLD, TXJ, TYJ, and TZJ.

```
111 CONTINUE
    TXIOLD = TXI
    TYIOLD = TYI
    TZIOLD = TZI
    TXJ = TXI
    TYJ = TYI
    TZJ = TZI
```

(Note: All of the above coordinates advance with each reflection. Initially TXI, TYI, and TZI are used in a test to determine the first trip through position (TXI,TYI,TZI) by a given photon.)

We return to this junction ("13 CONTINUE") each time a reflection occurs.

```
 13 CONTINUE
```

If this is the first time this volley has passed through J, the volley striking node J is "emitted" diffusely.

```
    IF((TXI.EQ.TXIOLD).AND.(TYI.EQ.TYIOLD)
   &       .AND.(TZI.EQ.TZIOLD)) GOTO 137
```

Test to see if the energy packet is absorbed or reflected. if the random number RN is less than or equal to ABS, the absorptivity, the packet is absorbed; otherwise it is reflected.

```
    ABS = EF
    REFR = RF
    IF(TZJ.GT.H1) ABS = EFOV
    IF(TZJ.GT.H1) REFR = RFOV
    CALL RAND(SEED, IN, RN)
    IF(RN.LE.ABS) GOTO 16
```

Test to see if the reflected energy packet is specularly reflected or diffusely reflected. If the random number RN is less than or equal to REFR, the reflectivity ratio, the packet is specularly reflected; otherwise it is diffusely reflected.

```
    CALL RAND(SEED, IN, RN)
    NSD = 1
    IF(RN.LE.REFR) GOTO 17
```

Randomly select the direction of the diffuse reflection.

```
137 CONTINUE
    CALL RAND(SEED, IN, RN)
    IF(RN.EQ.1.0D0) GOTO 14
    THETA = ARSIN(SQRT(RN))
    GOTO 15
 14 CONTINUE
    THETA = 0.0D0
 15 CONTINUE
    CALL RAND(SEED, IN, RN)
    PHI = 2.0D0*PI*RN
    NSD = 2
    GOTO 17
115 CONTINUE
```

Call the vector subroutine which computes the x-, y-, and z-components of the unit vector in direction (THETA,PHI) with respect to the central coordinate system.

```
    CALL VECTOR(UNX,UNY,UNZ,PHI,THETA,VOX,VOY,VOZ)
```

Call the SEARCH subroutine which identifies the receiving point of the diffusely reflected photon.

```
    CALL SEARCH(TXI,TYI,TZI,TXJ,TYJ,TZJ,VOX,VOY,VOZ)
```

If the z-coordinate of the receiving position is H2, the photon has escaped.

```
    IF(TZJ.EQ.H2) GOTO 24
```

If the z-coordinate of the receiving position is 0 and the r-coordinate is less than R3, the photon has entered the cavity.

```
    RAD = DSQRT(TXJ*TXJ + TYJ*TYJ)
    IF(TZJ.EQ.0.0D0.AND.RAD.LT.R3) GOTO 244
```

The photon is reflected to the next position whose coordinates are (TXJ, TYJ, TZJ).

```
    GOTO 25
```

The photon is absorbed. Call the NODE subroutine which determines the absorbing node.

```
 16 CONTINUE
    CALL NODE(TXJ,TYJ,TZJ,J1)
    DF(J1) = DF(J1) + 1.0D0
    GOTO 10
```

Determine the inward-directed unit normal vector of the source position. This depends on the shape of the source surface.

```
 17 CONTINUE
    IF(TZI.EQ.H2) GOTO 201
    IF(TZI.GT.H1) GOTO 202
    IF(TZI.EQ.H1) GOTO 203
    IF(TZI.GT.0.0D0) GOTO 204
```

The source point is on the precision aperture.

```
    UNX = 0.0D0
    UNY = 0.0D0
    UNZ = 1.0D0
    GOTO 23
```

The source is on the imaginary surface in the plane of F-O-V limiter.

```
201 CONTINUE
    UNX = 0.0D0
    UNY = 0.0D0
    UNZ = -1.0D0
    GOTO 23
```

The source point is on the F-O-V limiter.

```
202 CONTINUE
    UNX = -TXI/R1
    UNY = -TYI/R1
    UNZ = -TZI/R1
    GOTO 23
```

The source point is on the floor.

```
203 CONTINUE
    UNX = 0.0D0
    UNY = 0.0D0
    UNZ = 1.0D0
    GOTO 23
```

The source point is on the inner edge of the floor.

```
204 CONTINUE
    UNX = -TXI/R2
    UNY = -TYI/R2
    UNZ = 0.0D0
```

If the reflection is diffuse, return to the diffuse sequence; otherwise continue in the specular sequence.

```
 23 CONTINUE
    IF(NSD.EQ.2) GOTO 115
```

Compute the "ideal" vector of the reflected packet.

```
      VDOT = (TXI-TXIOLD)*UNX
      VDOT = VDOT + (TYI-TYIOLD)*UNY
      VDOT = VDOT + (TZI-TZIOLD)*UNZ
      VOX = TXI - TXIOLD - 2.0D0*VDOT*UNX
      VOY = TYI - TYIOLD - 2.0D0*VDOT*UNY
      VOZ = TZI - TZIOLD - 2.0D0*VDOT*UNZ
      VMAG = DSQRT(VOX*VOX + VOY*VOY + VOZ*VOZ)
      VOX = VOX/VMAG
      VOY = VOY/VMAG
      VOZ = VOZ/VMAG
```

Call the SEARCH subroutine which identifies the receiving point of the "exactly" reflected photon.

```
      CALL SEARCH(TXI,TYI,TZI,TXJ,TYJ,TZJ,VOX,VOY,VOZ)
```

If the z-coordinate of the receiving position is H2, the photon has escaped.

```
      IF(TZJ.EQ.H2) GOTO 24
```

If the z-coordinate of the receiving position is 0 and the r-coordinate is less than R3, the photon has entered the cavity.

```
      RAD = DSQRT(TXJ*TXJ + TYJ*TYJ)
      IF(TZJ.EQ.0.0D0.AND.RAD.LT.R3) GOTO 244
```

The photon is reflected to the next position whose coordinates are (TXJ, TYJ, TZJ).

```
      GOTO 25
```

The photon has escaped through the F-O-V limiter; thus, find the pie segment through which it has escaped and increment the appropriate DF counter.

```
 24 CONTINUE
    IF(TXJ.GE.0.0D0.AND.TYJ.GE.0.0D0) I = 1
    IF(TXJ.LT.0.0D0.AND.TYJ.GE.0.0D0) I = 2
    IF(TXJ.LT.0.0D0.AND.TYJ.LT.0.0D0) I = 3
    IF(TXJ.GE.0.0D0.AND.TYJ.LT.0.0D0) I = 4
    DF(I) = DF(I) + 1.0D0
    GOTO 10
```

The photon has entered the cavity; thus, increment the cavity DF counter.

```
244 CONTINUE
    DF(MM3) = DF(MM3) + 1.0D0
    GOTO 10
```

Reflection to another surface has occurred. Thus, increment TXIOLD, TYIOLD, TZIOLD, TXJ, TYJ, and TZJ.

```
 25 CONTINUE
    TXIOLD = TXI
    TYIOLD = TYI
    TZIOLD = TZI
    TXI = TXJ
    TYI = TYJ
    TZI = TZJ
```

Loop back to test sequence to determine disposition of the photon when it arrives at this new surface.

```
    GOTO 13
```

Write the vector of distribution factors corresponding to node J.

```
 10 CONTINUE
    DO 30 JK = 1, MM3
    DF(JK) = DF(JK)/RSHOTS
 30 CONTINUE
    DO 63 I = 2,MM3,4
    I2 = I + 2
    DF(I) = DF(I) + DF(I2)
    DF(I) = DF(I)/2.0D0
    DF(I2) = DF(I)
 63 CONTINUE
    DO 64 JK = 1,MM3
    WRITE(17,3) JJ, JK, DF(JK)
 64 CONTINUE
```

Generate "trip" card at end of file.

```
 20 CONTINUE
    JJ = 0
    JK = 0
    DFDUM = 0.0D0
    WRITE(17,3) JJ, JK, DFDUM
```

Format statements:

```
  1 FORMAT(5X, 2I5, 3D14.5)
  2 FORMAT(8X, I3)
  3 FORMAT(5X, 2I5, D14.5)
  4 FORMAT(5X, I5, 2D14.5)
 96 FORMAT(5X,'THE SYSTEM VARIABLES ARE AS ',
    &       'FOLLOWS:' ,//10X,'H1 = ',F5.2 ,' CM',
    &       /,10X,'H2 = ',F5.2,' CM',/,10X,'R1 = ',
    &       F5.2,' CM',/,10X,'R2 = ',F5.2,/,10X,'R3 '
    &       ,'= ',F5.2,/,10X,'F1 = ',F5.2,/,10X,'F2 = ',F5.2,
    &       /,10X,'F3 = ',F5.2,/,10X,'FOV1 = ',F5.2,/,10X,
    &       'FOV2 = ',F5.2,/,10X,'EFOV = ',F5.2,/,10X,
    &       'RFOV = ',F5.2,/,10X,'EF = ',F5.2,/,10X,'RF = ',
    &       F5.2,/,10X,'NSHOT = ',I6///)
 97 FORMAT(I6,14F5.2)
```

End of "TRW FRONT".

```
    STOP
    END
```

## 3.3. Subroutine SEARCH

This subroutine calculates the magnitude of the vector from the source position to the receiving position of the reflected photon. Since the direction of this vector is known (VOX, VOY, VOZ), the magnitude is all that is needed to calculate the receiving position.

### 3.3.1. Subroutine Description

```
      SUBROUTINE SEARCH(TXI,TYI,TZI,TXJ,TYJ,TZJ,VOX,VOY,VOZ)
```

When this subroutine is compiled on IBM equipment, double precision must be specified. This step may not be necessary on other processors.

```
      IMPLICIT REAL*8 (A-H,O-Z)
```

Establish COMMON variables.

```
      COMMON H1,H2,R1,R2,R3,FOV1,FOV2,F1,F2,F3,PI
```

If the source point is at the origin of coordinates, RETURN.

```
      IF(TZI.EQ.0.0D0.AND.TXI.EQ.0.0D0) GOTO 27
```

If the source point is on the imaginary F-O-V exit plane surface, go to 11.

```
      IF(TZI.EQ.H2) GOTO 11
```

If the source point is on the F-O-V limiter, go to 12.

```
      IF(TZI.GT.H1) GOTO 12
```

If the source point is on the floor, go to 13.

```
    IF(TZI.EQ.H1) GOTO 13
```

If the source point is on the inner edge of the floor, go to 14.

```
    IF(TZI.GT.0.0D0) GOTO 14
```

If the source point is on the precision aperture, go to 15.

```
    GOTO 15
```

The source point is on the imaginary F-O-V exit plane surface; thus begin search on the F-O-V limiter.

```
 11 CONTINUE
    VMAG = - (H2-H1)/VOZ
    TXJ = TXI + VMAG*VOX
    TYJ = TYI + VMAG*VOY
    TZJ = H1
    RADIUS = DSQRT(TXJ*TXJ + TYJ*TYJ)
```

If 'RADIUS' is less than 'R1', the packet has missed the F-O-V limiter, in which case we go to 16.

```
    IF(RADIUS.LT.R1) GOTO 16
```

The packet has intercepted the F-O-V limiter.

```
 20 CONTINUE
    A = VOX*TXI + VOY*TYI + VOZ*TZI
    B = TXI*TXI + TYI*TYI + TZI*TZI - R1*R1
    VMAG = - A + DSQRT(A*A-B)
    TXJ = TXI + VMAG*VOX
    TYJ = TYI + VMAG*VOY
    TZJ = TZI + VMAG*VOZ
    RETURN
```

The packet has missed the F-O-V limiter. Check to see if it has hit the floor, in which case RETURN.

```
 16 CONTINUE
    IF(RADIUS.GT.R2) RETURN
```

The packet has missed the floor; check to see if it has hit the edge of the floor.

```
    VMAG = - H2/VOZ
 21 CONTINUE
    TXJ = TXI + VMAG*VOX
    TYJ = TYI + VMAG*VOY
    TZJ = 0.0D0
    RADIUS = DSQRT(TXJ*TXJ + TYJ*TYJ)
```

If 'RADIUS' is less than 'R2', the packet has missed the edge of the floor, in which case go to 17.

```
    IF(RADIUS.LT.R2) GOTO 17
```

The packet has hit the edge of the floor.

```
 25 CONTINUE
    A = VOX*VOX + VOY*VOY
    B = TXI*VOX + TYI*VOY
    C = TXI*TXI + TYI*TYI - R2*R2
    ARG = B*B - A*C
    VMAG = (-B + DSQRT(ARG))/A
    TXJ = TXI + VMAG*VOX
    TYJ = TYI + VMAG*VOY
    TZJ = TZI + VMAG*VOZ
    RETURN
```

The packet has missed the edge of the floor. Check to see if it has hit the precision aperture, in which case RETURN.

```
 17 CONTINUE
    IF(RADIUS.GE.R3) RETURN
```

The packet has entered the cavity.

```
      TXJ = 0.0D0
      TYJ = 0.0D0
      TXJ = 0.0D0
      RETURN
```

The source point is on the F-O-V limiter; thus begin on the F-O-V limiter exit plane.

```
   12 CONTINUE
      IF(VOZ.GT.0.0D0) GOTO 18
```

The source point is not on the F-O-V exit plane; thus begin search on the F-O-V limiter.

```
      VMAG = -(TZI-H1)/VOZ
      TXJ = TXI + VMAG*VOX
      TYJ = TYI + VMAG*VOY
      TZJ = H1
      RADIUS = DSQRT(TXJ*TXJ + TYJ*TYJ)
```

If 'RADIUS' is less than 'R1', the packet has missed the F-O-V limiter, in which case we continue the search on the floor.

```
      IF(RADIUS.LT.R1) GOTO 19
```

The packet has hit the F-O-V limiter. Solve for coordinates.

```
      GOTO 20
```

The packet has missed the F-O-V limiter, thus we continue our search on the floor. If 'RADIUS' is greater than 'R2', the packet has hit the floor, in which case we RETURN.

```
   19 CONTINUE
      IF(RADIUS.GT.R2) RETURN
```

The packet has missed the floor, in which case we continue our search on the edge of the floor.

```
    VMAG = - TZI/VOZ
    GOTO 21
```

Search on the F-O-V limiter exit plane.

```
 18 CONTINUE
    VMAG = (H2-TZI)/VOZ
 22 CONTINUE
    TXJ = TXI + VMAG*VOX
    TYJ = TYI + VMAG*VOY
    TZJ = H2
    RADIUS = DSQRT(TXJ*TXJ + TYJ*TYJ)
    REXIT = DSQRT(R1*R1 - H2*H2)
```

If 'RADIUS' is less than 'REXIT', the packet has escaped, in which case we RETURN.

```
    IF(RADIUS.LT.REXIT) RETURN
```

The packet has hit the F-O-V limiter.

```
    GOTO 20
```

The source point is on the floor; thus begin search on the imaginary F-O-V exit plane surface.

```
 13 CONTINUE
    VMAG = (H2-H1)/VOZ
    GOTO 22
```

The source point is on the inner edge of the floor; thus check to see if packet is emitted upward or downward.

```
 14 CONTINUE
    IF(VOZ.LT.0.0D0) GOTO 23
```

The packet has been emitted upward; thus resume the search on the inner edge of the floor.

```
    VMAG = (H1-TZI)/VOZ
 26 CONTINUE
    TXJ = TXI + VMAG*VOX
    TYJ = TYI + VMAG*VOY
    RADIUS = DSQRT(TXJ*TXJ + TYJ*TYJ)
```

If 'RADIUS' is less than 'R2', the packet has missed the inner edge of the floor; thus resume search on the F-O-V limiter.

```
    IF(RADIUS.LT.R2) GOTO 24
```

The packet has intercepted the inner edge of the floor; thus go to 25.

```
    GOTO 25
```

The packet has missed the inner edge of the floor; thus resume search on the F-O-V limiter.

```
 24 CONTINUE
    VMAG = (H2-TZI)/VOZ
    GOTO 22
```

The packet is emitted downward; thus begin search on the inner edge of the floor.

```
 23 CONTINUE
    VMAG = -TZI/VOZ
    TXJ = TXI + VMAG*VOX
    TYJ = TYI + VMAG*VOY
    TZJ = 0.0D0
    RADIUS = DSQRT(TXJ*TXJ + TYJ*TYJ)
```

If 'RADIUS' is greater than 'R2' , the packet has hit the inner edge of the floor, in which case go to 25; otherwise, go to 17.

```
    IF(RADIUS.GT.R2) GOTO 25
    GOTO 17
```

The source point is on the precision aperture; thus begin search on the inner edge of the floor.

```
 15 CONTINUE
    VMAG = H1/VOZ
    GOTO 26
 27 CONTINUE
```

End of Subroutine 'SEARCH'.

```
    RETURN
    END
```

## 3.4. Subroutine VECTOR

This subroutine expresses the randomly selected diffuse angle with respect to the source position in terms of the central coordinate system.

### 3.4.1. Nomenclature

| | |
|---|---|
| UNX | X-component of inward-directed unit normal vector of source position. |
| UNY | Y-component of inward-directed unit normal vector of source position. |
| UNZ | Z-component of inward-directed unit normal vector of source position. |
| UTX | X-component of unit tangent vector of source position. |
| UTY | Y-component of unit tangent vector of source position. |
| UTZ | Z-component of unit tangent vector of source position. |
| USX | X-component of the mutually orthogonal vector. |
| USY | Y-component of the mutually orthogonal vector. |
| USZ | Z-component of the mutually orthogonal vector. |
| THETA | Angle of diffuse vector with respect to normal vector. |
| PHI | Angle of diffuse vector with respect to tangent vector. |
| VOX | X-component of diffuse vector with respect to central coordinate system. |
| VOY | Y-component of diffuse vector with respect to central coordinate system. |
| VOZ | Z-component of diffuse vector with respect to central coordinate system. |

### 3.4.2. Subroutine Description

```
    SUBROUTINE VECTOR(UNX,UNY,UNZ,PHI,THETA,VOX,VOY,VOZ)
```

When this subroutine is compiled on IBM equipment, it is necessary to specify double precision. This step may not be necessary on other processors.

```
    IMPLICIT REAL*8 (A-H, O-Z)
```

Compute the x-, y-, and z-components of the unit tangent vector, UT, and the mutually orthogonal vector, US, on the source position.

If the surface is cylindrical, go to 11.

```
    IF(UNZ.EQ.0.0D0) GOTO 11
```

If the surface is plane, go to 12.

```
    IF(UNZ.EQ.1.0D0.OR.UNZ.EQ.-1.0D0) GOTO 12
```

Calculations for the spherical surface:

```
    ARG = UNZ*UNZ/(1.0D0-UNZ*UNZ)
    UTX = UNX*DSQRT(ARG)
    UTY = UNY*DSQRT(ARG)
    UTZ = DSQRT(1.0D0 - UNZ*UNZ)
    GOTO 13
```

Calculations for the cylindrical surface:

```
11  CONTINUE
    UTX = 0.0D0
    UTY = 0.0D0
    UTZ = 1.0D0
    GOTO 13
```

Calculations for plane surfaces:

```
 12 CONTINUE
    UTX = 1.0D0
    UTY = 0.0D0
    UTZ = 0.0D0
 13 CONTINUE
```

Compute the x-, y-, and z-components of the unit vector on the source position which is mutually orthogonal to UN and UT.

```
    USX = UNY*UTZ - UNZ*UTY
    USY = UNZ*UTX - UNX*UTZ
    USZ = 0.0D0
```

Compute components of the vector at angle (THETA, PHI) with respect to the UN,UT,US-coordinate system and express the result in the I,J,K-coordinate system.

```
    SINTH = DSIN(THETA)
    COSTH = DCOS(THETA)
    SINPH = DSIN(PHI)
    COSPH = DCOS(PHI)
    VOX = SINTH*COSPH*UTX + SINTH*SINPH*USX
    VOX = VOX + COSTH*UNX
    VOY = SINTH*COSPH*UTY + SINTH*SINPH*USY
    VOY = VOY + COSTH*UNY
    VOZ = SINTH*COSPH*UTZ + SINTH*SINPH*USZ
    VOZ = VOZ + COSTH*UNZ
```

End of Subroutine 'VECTOR'.

```
    RETURN
    END
```

## 3.5. Subroutine NODE

This subroutine identifes the node which absorbs a given photon.

### 3.5.1. Subroutine Description

```
      SUBROUTINE NODE(TXJ,TYJ,TZJ,J1)
```

When this subroutine is compiled on IBM equipment, it is necessary to specify double precision. This step may not be necessary on other processors.

```
      IMPLICIT REAL*8 (A-H, O-Z)
```

Identify the COMMON variables.

```
      COMMON H1,H2,R1,R2,R3,FOV1,FOV2,F1,F2,F3,PI
```

Check to see if absorption occurred on the F-O-V limiter, in which case go to 11.

```
      IF(TZJ.GT.H1) GO TO 11
      RADIUS = DSQRT(TXJ*TXJ + TYJ*TYJ)
```

Check to see if absorption occurred on the floor, in which case go to 12.

```
      IF(RADIUS.LT.R1.AND.RADIUS.GE.R2) GO TO 12
```

Check to see if absorption occurred on the edge of the floor, in which case go to 13.

```
      IF(TZJ.LT.H1.AND.TZJ.GT.0.0D0) GOTO 13
```

Absorption occurred on the precision aperture; thus determine which node.

```
    IF(TXJ.GE.0.0D0.AND.TYJ.GE.0.0D0) J1 = 37
    IF(TXJ.LT.0.0D0.AND.TYJ.GE.0.0D0) J1 = 38
    IF(TXJ.LT.0.0D0.AND.TYJ.LT.0.0D0) J1 = 39
    IF(TXJ.GE.0.0D0.AND.TYJ.LT.0.0D0) J1 = 40
    RETURN
```

Absorption occurred on the F-O-V limiter; thus check to see in which ring of nodes the absorption occurred.

```
 11 CONTINUE
    IF(TZJ.GT.FOV1) GOTO 14
    IF(TZJ.GT.FOV2) GOTO 15
```

Absorption occurred in the bottom ring of nodes of the F-O-V limiter; thus identify the node.

```
    IF(TXJ.GE.0.0D0.AND.TYJ.GE.0.0D0) J1 = 13
    IF(TXJ.LT.0.0D0.AND.TYJ.GE.0.0D0) J1 = 14
    IF(TXJ.LT.0.0D0.AND.TYJ.LT.0.0D0) J1 = 15
    IF(TXJ.GE.0.0D0.AND.TYJ.LT.0.0D0) J1 = 16
    RETURN
```

Absorption occurred in the top ring of nodes of the F-O-V limiter; thus identify the node.

```
 14 CONTINUE
    IF(TXJ.GE.0.0D0.AND.TYJ.GE.0.0D0) J1 = 5
    IF(TXJ.LT.0.0D0.AND.TYJ.GE.0.0D0) J1 = 6
    IF(TXJ.LT.0.0D0.AND.TYJ.LT.0.0D0) J1 = 7
    IF(TXJ.GE.0.0D0.AND.TYJ.LT.0.0D0) J1 = 8
    RETURN
```

Absorption occurred in the middle ring of nodes of the F-O-V limiter; thus identify the node.

```
 15 CONTINUE
    IF(TXJ.GE.0.0D0.AND.TYJ.GE.0.0D0) J1 = 9
    IF(TXJ.LT.0.0D0.AND.TYJ.GE.0.0D0) J1 = 10
    IF(TXJ.LT.0.0D0.AND.TYJ.LT.0.0D0) J1 = 11
    IF(TXJ.GE.0.0D0.AND.TYJ.LT.0.0D0) J1 = 12
    RETURN
```

Absorption occurred on the floor; thus check to see in which ring of nodes the absorption occurred.

```
 12 CONTINUE
    IF(RADIUS.GE.F1) GOTO 16
    IF(RADIUS.GT.F2) GOTO 17
    IF(RADIUS.GT.F3) GOTO 18
```

Absorption occurred in the innermost ring of nodes on the floor; thus identify the node.

```
    IF(TXJ.GE.0.0D0.AND.TYJ.GE.0.0D0) J1 = 29
    IF(TXJ.LT.0.0D0.AND.TYJ.GE.0.0D0) J1 = 30
    IF(TXJ.LT.0.0D0.AND.TYJ.LT.0.0D0) J1 = 31
    IF(TXJ.GE.0.0D0.AND.TYJ.LT.0.0D0) J1 = 32
    RETURN
```

Absorption occurred on the outermost ring of nodes on the floor; thus identify the node.

```
 16 CONTINUE
    IF(TXJ.GE.0.0D0.AND.TYJ.GE.0.0D0) J1 = 17
    IF(TXJ.LT.0.0D0.AND.TYJ.GE.0.0D0) J1 = 18
    IF(TXJ.LT.0.0D0.AND.TYJ.LT.0.0D0) J1 = 19
    IF(TXJ.GE.0.0D0.AND.TYJ.LT.0.0D0) J1 = 20
    RETURN
```

Absorption occurred on the second ring of nodes from the outer edge of the floor; thus identify the node.

```
 17 CONTINUE
    IF(TXJ.GE.0.0D0.AND.TYJ.GE.0.0D0) J1 = 21
    IF(TXJ.LT.0.0D0.AND.TYJ.GE.0.0D0) J1 = 22
    IF(TXJ.LT.0.0D0.AND.TYJ.LT.0.0D0) J1 = 23
    IF(TXJ.GE.0.0D0.AND.TYJ.LT.0.0D0) J1 = 24
    RETURN
```

Absorption occurred on the third ring of nodes from the outer edge of the floor; thus identify the node.

```
 18 CONTINUE
    IF(TXJ.GE.0.0D0.AND.TYJ.GE.0.0D0) J1 = 25
    IF(TXJ.LT.0.0D0.AND.TYJ.GE.0.0D0) J1 = 26
    IF(TXJ.LT.0.0D0.AND.TYJ.LT.0.0D0) J1 = 27
    IF(TXJ.GE.0.0D0.AND.TYJ.LT.0.0D0) J1 = 28
    RETURN
```

Absorption occurred on the edge of the floor; thus identify the node.

```
 13 CONTINUE
    IF(TXJ.GE.0.0D0.AND.TYJ.GE.0.0D0) J1 = 33
    IF(TXJ.LT.0.0D0.AND.TYJ.GE.0.0D0) J1 = 34
    IF(TXJ.LT.0.0D0.AND.TYJ.LT.0.0D0) J1 = 35
    IF(TXJ.GE.0.0D0.AND.TYJ.LT.0.0D0) J1 = 36
```

End of Subroutine 'NODE'.

```
    RETURN
    END
```

## 3.6. Subroutine RAND

This subroutine uses library Subroutine GGUBS to obtain random numbers in an efficient way. A sequence of 100 random numbers is generated each time the 100th number from the previous generation is used.

### 3.6.1. Nomenclature

| | |
|---|---|
| SEED | A seed number required by GGUBS. It must be specified in the main program as a double precision integer value. |
| IN | An integer between 1 and 100 inclusive which represents which of the 100 random numbers of a given generation is obtained. |
| RN | The random number passed back to MAIN. |

### 3.6.2. Subroutine Description

```
SUBROUTINE RAND(SEED, IN, RN)
```

When this subroutine is compiled on IBM equipment, it is necessary to specify double precision. This step may not be necessary with other processors.

```
IMPLICIT REAL*8 (A-H,O-Z)
REAL*4 RN, Z(100)
```

If this is the first call to RAND, RN is equal to the negative value set in MAIN. In this case we must call GGUBS to generate the first generation of 100 random numbers.

```
IF(RN.LT.0.0D0) GOTO 11
```

If this is not the first call to RAND, then increment to the next value in this generation of 100 random numbers.

```
IN = IN + 1
```

If we have not yet used up our allotment of random numbers from this generation, we simply take the next one in the sequence back to MAIN.

```
      IF(IN.LE.100) GOTO 13
```

However, if we have used all the members of this generation, we must create a new generation of 100 random numbers.

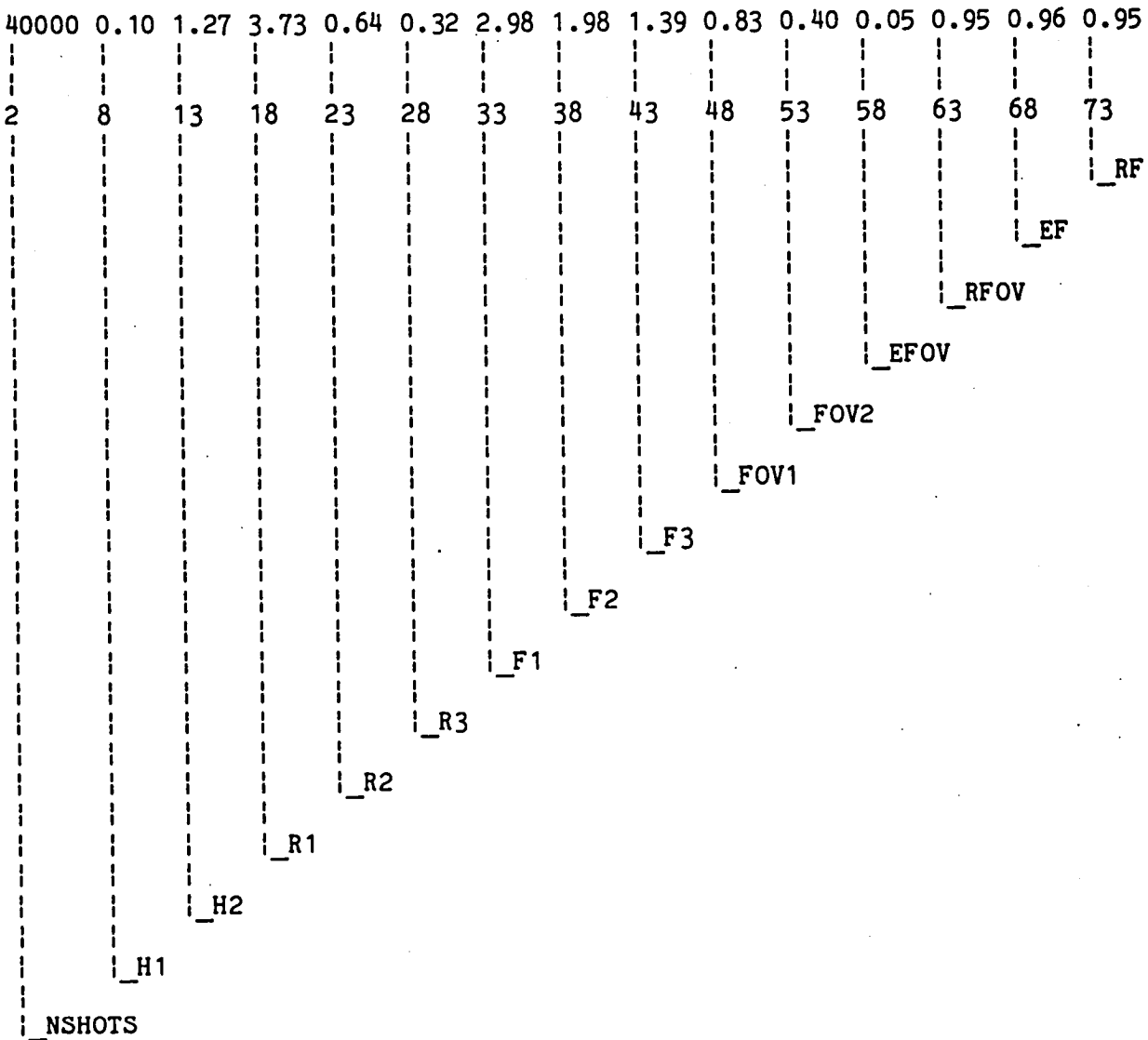
```
   11 CONTINUE
      IN = 1
      DSEED = SEED
      NUMBER = 100
      CALL GGUBS(DSEED, NUMBER, Z)
      SEED = DSEED
   13 CONTINUE
      RN = Z(IN)
```

End of Subroutine 'RAND'.

```
      RETURN
      END
```

## 3.7. 'FRONT' DATA

These data are used only by Program 'TRW FRONT'. This is where all dimensions and properties of the body, other than the internode conductances, must be changed if another front end design is to be analyzed.



| Variable | Type | Field | Description |
|---|---|---|---|
| NSHOT | I6 | 1-6 | Number of volleys fired |
| H1 | F5.2 | 7-11 | Distance to floor of instrument |
| H2 | F5.2 | 12-16 | Distance to face of F-O-V limiter |
| R1 | F5.2 | 17-21 | Outside radius of face |
| R2 | F5.2 | 22-26 | Inside radius of face |
| R3 | F5.2 | 27-31 | Radius of precision aperture |
| F1 | F5.2 | 32-36 | Radius to first floor node |
| F2 | F5.2 | 37-41 | Radius to second floor node |
| F3 | F5.2 | 42-46 | Radius to third floor node |
| FOV1 | F5.2 | 47-51 | Distance to top F-O-V limiter node |
| FOV2 | F5.2 | 52-56 | Distance to center limiter node |
| EFOV | F5.2 | 57-61 | Emissivity of F-O-V limiter face |
| RFOV | F5.2 | 62-66 | Reflectivity ratio of F-O-V face |
| EF | F5.2 | 67-71 | Emissivity of floor |
| RF | F5.2 | 72-76 | Reflectivity ratio of face |

Note: Please see the 'NOMENCLATURE' of this section for a more complete description of these variables. Also, see Fig. 1 of the February Progress Report.

## 4. TRW TEMP

This program computes the transient temperature distribution in the active cavity and body of a TRW-type wide-field-of-view total spectrum radiometer subject to a uniform blackbody irradiance input and arbitrary heat sink and mounting beam temperature variations. It requires as inputs conductance matrices and radiation distribution factor matrices generated by the previous programs.

Programs TRW SHAPE, TRW NODES and TRW FRONT must be executed before this program is submitted. Also, an additional on-line disk file must be created and named A51961.DATA14 which contains the body internode conductances and the body node surface areas, emissivities, thermal capacities, volumes and heat sources. The current version of this file, which is user generated, is given in section 6. of this USER'S MANUAL.

### 4.1. Nomenclature

| | |
|---|---|
| TO | Matrix of "old" cavity node temperatures (K); i.e., temperatures at beginning of calculation time interval DT. Must be dimensioned M2. |
| TN | Matrix of "new" cavity node temperatures (K); i.e., temperatures at the end of calculation time interval DT. Must be dimensioned M2. |
| AA | Matrix of cavity node surface areas (m**2). Must be dimensioned M2. |
| CC | Matrix of cavity node thermal capacities (W-s/K). Must be dimensioned M2. |
| DFC(J,I) | Matrix of cavity distribution factors for radiation emitted diffusely from the first node position on node ring J which is absorbed by node I (-). Must be dimensioned NN2 by (M2 + 1). |
| G(I,J) | Matrix of conductances for conduction path between cavity nodes I and J (W/K). Must be dimensioned M2 by M2. |
| NBODY | Number of radiometer "body" nodes (= 104). |
| NBODY2 | NBODY + 2 (= 106). |
| RNBODY | Real version of NBODY. |
| HST | Heat sink (node "20") temperature (K). |
| TCLAMP | Mounting beam temperature (K). |
| HEAT | Proportional to cavity heater power, P (P = 0.32*HEAT mW). |
| TSOURC | Source equivalent blackbody temperature (K). |
| GLINK | The conductance of the link between the cavity and the thermal impedance (W/K). |
| SIG | Stefan-Boltzmann constant (5.6693E-08 W/m**2-K**4). |
| MBODY | Body node number. |

AABODY Surface area of body node 'MBODY' for purposes of calculating radiation heat transfer (cm**2).

EEBODY Emissivity of body node 'MBODY' (-).

CCBODY Thermal capacity of body node 'MBODY' (W-s/cm**3-K).

VVBODY Volume of body node 'MBODY' (cm**3).

QQBODY Heat source in body cavity 'MBODY'; may be due to radiation, conduction or electrical heating (W).

LIMBODY Delimiting index in DO-loop (= MBODY + 3).

ABODY(I) Surface area of body node I for purposes of calculating radiation exchange (m**2).

EBODY(I) Emissivity of body node I (-).

CBODY(I) Thermal capacity of body node I (W-s/m**3-K).

VBODY(I) Volume of body node I (m**3).

QBODY(I) Heat source in body cavity I; may be due to radiation, conduction or electrical heating (W).

T1BODY(I) Initial temperature of body node I at beginning of solution; must be specified by user by specifying 'HST'. Later set equal to new temperature of node I at end of calculation interval (K).

TGBODY(I) Initial guess of body node I temperature at the end of a calculation time interval; initially set equal to 'HST', but in subsequent time steps it is updated to previous value (K).

COND Value of thermal conductance between body nodes on the same level ('MBODY') in the azimuthal direction (W/K).

GBODY(I,J) Value of thermal conductance between body nodes I and J (W/K).

DBODY(I,J) Value of radiation distribution factor from body node I to body node J (-).

MIBODY Index number of body node "level", or ring (-).

MJBODY Index number of body node "level", or ring (-).

SOURCE(I) Blackbody source-to-body node I distribution factor (-).

BODYDF Value of distribution factor, either body node-to-body node or source-to-body node, read from unit 17 (-).

FTIME Total calculation time;, i.e., duration of the problem (s).

DT Length of cavity calculation time increment; i.e., integration step size (s).

NDT Number of time increments used in cavity calculation (-).

DTBODY Length of body calculation time increment (s).

PI Ratio of circumference to diameter of a circle (-).

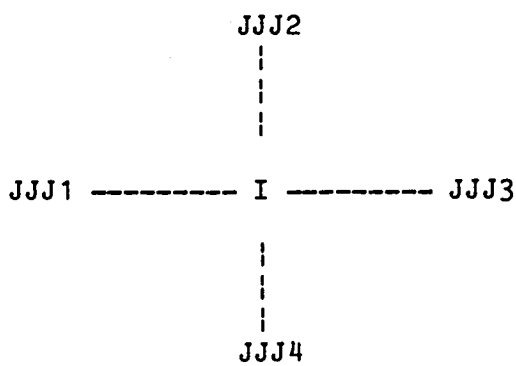
NARSZ Index used in Subroutine GAUSS; represents upper limit of size of matrix that can be inverted by the subroutine; i.e., used as argument of dimension statement in the subroutine. (-).

M2 Number of cavity nodes (-).

ELIMIT RMS error limit (%).

ITRMAX Maximum number of iterations permitted in the cavity temperature distribution calculation loop.

| | |
|---|---|
| ITRMIN | Minimum number of iterations permitted in the cavity temperature distribution calculation loop. |
| AO | Aperture area (mm**2). |
| NTH | Number of circumferential divisions of cavity (-). |
| EC | Emissivity (=absorptivity) of the cavity (-) |
| EB | Emissivity of the back surface of the cavity (-). |
| ETIME | Elapsed time counter (s). |
| QTOT(I) | Average rate of energy transfer from the cavity to Node '13' of the radiometer body during ten successive cavity calculation time increments (W). |
| I1 | Index indicating level of Node I. |
| I2 | Index indicating position of Node I in Level I1. |
| JJJ1 | Index of node "to left" of Node I. |
| JJJ2 | Index of node "above" Node I. |
| JJJ3 | Index of Node "to right" of Node I. |
| JJJ4 | Index of node "below" Node I. |




| | |
|---|---|
| Q1 | Energy transfer rate into Node I from cavity nodes other than the receiving node itself during calculation time interval DT (W). |
| J1 | Index indicating level of Node J. |
| J2 | Index indicating position of node J in Level J1. |

## 4.2. Program Description

Double precision must be specified when this program is compiled on IBM equipment. When other processors are used, the following statement may not be necessary.

```
IMPLICIT REAL*8 (A-H,O-Z)
```

Dimension subscripted variables.

```
      DIMENSION ABODY(104),EBODY(104),CBODY(104),VBODY(104)
      DIMENSION QBODY(104),T1BODY(104),TGBODY(104),TUBODY(104)
      DIMENSION QTOT(4),SOURCE(45),IANS(100)
      DIMENSION DBODY(104,106),GBODY(104,104)
      DIMENSION TO(100),TN(100),AA(100),CC(100),RHS(100)
      DIMENSION Z(100,100),G(100,100)
      DIMENSION DFC(10,101),DFA(4,100,101)
```

Establish COMMON variables.

```
      COMMON /EMSGC/ IER
```

Set constants. (Note: This is where each of these numbers must be changed when different cases are to be run.)

```
      NBODY = 104
      RNBODY = NBODY
      HST = 313.0D0
      TCLAMP = 326.0D0
      HEAT = 400.0D0
      TSOURC = 299.36D0
      GLINK = 0.04D0
      SIG = 5.6693D-08
      NARSZ = 100
      PI = 3.141592D0
```

Read in and write out system input variables.

```
      READ(5,97) A,B,C,N1,N2,NTH,NSHOTN,ABS,REFR,DT,ELIMIT,MAT
      WRITE(6,96) A,B,C,N1,N2,NTH,NSHOTN,ABS,REFR,DT,ELIMIT,MAT
```

The following 'WRITE' statement causes a dimensioned sketch of the cavity to be "drawn" by the line printer.

```
      WRITE(6,94)
```

Initialize radiometer body internode conductances and distribution factors to zero. This is necessary because the solution strategy considers possible interaction between all possible combinations of any two radiometer body nodes. Later we will read in the nonzero values of these two matrices.

```
      NBODY2 = NBODY + 2
      DO 111 I = 1,NBODY
      DO 111 J = I,NBODY2
      IF(J.LE.NBODY) GBODY(I,J) = 0.0D0
      IF(J.LE.NBODY) GBODY(J,I) = GBODY(I,J)
      DBODY(I,J) = 0.0D0
      IF(J.LE.NBODY) DBODY(J,I) = DBODY(I,J)
111   CONTINUE
```

Read in values of constants and properties; initialize the node temperatures for radiometer body.

```
123   CONTINUE
      READ(14,81) MBODY,AABODY,EEBODY,CCBODY,VVBODY,QQBODY
      IF(MBODY.EQ.0) GOTO 124
      LIMBOD = MBODY + 3
      DO 223 I = MBODY,LIMBOD
      ABODY(I) = AABODY*1.0D-4
      EBODY(I) = EEBODY
      CBODY(I) = CCBODY*1.0D6
      VBODY(I) = VVBODY*1.0D-6
      QBODY(I) = QQBODY
      T1BODY(I) = HST
      TGBODY(I) = HST
223   CONTINUE
      GOTO 123
124   CONTINUE
```

Read in nonzero elements of 'GBODY' matrix.

```
211 CONTINUE
    READ(14,82) MBODY,COND
    IF(MBODY.EQ.0) GOTO 212
    LIMBOD = MBODY + 3
    DO 215 I = MBODY,LIMBOD
    J = I + 1
    IF(I.EQ.LIMBOD) J = MBODY
    GBODY(I,J) = COND
    GBODY(J,I) = COND
215 CONTINUE
    GOTO 211
212 CONTINUE
    READ(14,83) MIBODY,MJBODY,GBODY(MIBODY,MJBODY)
    IF(MIBODY.EQ.0) GOTO 213
    GBODY(MJBODY,MIBODY) = GBODY(MIBODY,MJBODY)
    GOTO 212
213 CONTINUE
```

Initialize source-to-instrument distribution factors to zero.

```
    DO 601 I = 1,41
    SOURCE(I) = 0.0D0
601 CONTINUE
```

Read in nonzero values of 'DBODY' matrix.

```
      INN = 0
      INK = 1
403   CONTINUE
      READ(17,92) I, J, BODYDF
      IF(I.EQ.0) GOTO 402
      INN = INN + 1
      IF(I.EQ.1) GOTO 501
      IF(I.LE.13) II = I - 4
      IF(I.EQ.17) II = 17
      IF(I.GE.21.AND.I.LT.33) II = I + 8
      IF(I.GE.33) II = I + 4
      IF(J.LT.5) JJ = 105
      IF(J.GE.5.AND.J.LT.17) JJ = J - 4
      IF(J.GE.17.AND.J.LT.21) JJ = J
      IF(J.GE.21.AND.J.LT.33) JJ = J + 8
      IF(J.GE.33.AND.J.LT.41) JJ = J + 4
      IF(J.EQ.41) JJ = 106
      DBODY(II,JJ) = DBODY(II,JJ) + BODYDF
      GOTO 403
501   CONTINUE
      IF(INN.EQ.5) INK = INK + 4
      IF(INN.EQ.5) INN = 1
      SOURCE(INK) = SOURCE(INK) + BODYDF
      GOTO 403
402   CONTINUE
```

Use symmetry to establish the values of the elements of the 'DBODY' matrix not computed directly.

```
      DO 401 I = 1,104,4
      DO 401 K = 1,104
      I1 = I + 1
      I2 = I + 2
      I3 = I + 3
      L1 = K + 3
      IF(L1.GT.4) L1 = K - 1
      L2 = K + 2
      IF(L2.GT.4) L2 = K - 2
      L3 = K + 1
      IF(L3.GT.4) L3 = K - 3
      DBODY(I1,K) = DBODY(I,L1)
      DBODY(I2,K) = DBODY(I,L2)
      DBODY(I3,K) = DBODY(I,L3)
401   CONTINUE
```

Set upper limit on calculation time; i.e., the duration of the heating or cooling problem being studied. The value used below, 400 s, permits the instrument to come into equilibrium. Also, set number of calculation time intervals.

```
      FTIME = 400.0D0
      NDT = FTIME/DT
```

Set calculation step size for body temperature calculation. Note that this is ten times the calculation time increment for the cavity calculation.

```
      DTBODY = 10.0D0*DT
```

Establish constants and indices.

```
      AO = PI*C*C/100.0D0
      FNTH = NTH
      NN1 = N1
      NN2 = NN1 + N2
      M1 = NN1*NTH
      M2 = NN2*NTH
      FM2 = M2
      MM2 = M2 + 1
      EC = ABS
      ITRMAX = 100
      ITRMIN = 2
      EB = 1.0D0
      ETIME = 0.0D0
      IER = 0
```

Write out node grid and geometry information.

```
      WRITE(6,98) M2,NN2,NTH,NA,NTH,A,B,C
      IF(MAT.EQ.0) WRITE(6,93) ELIMIT
```

Read in values of dimensioned constants. Begin by zeroing the G(I,J), Z(I,J) and RHS(I) matrices.

```
      DO 19 I = 1, M2
      RHS(I) = 0.0D0
      DO 19 J = 1, M2
      G(I,J) = 0.0D0
      Z(I,J) = 0.0D0
   19 CONTINUE
```

Note: The node half-conductances must be retrieved from the lower left-hand triangle of the 'V' matrix created by 'TRW SHAPE'.

```
 10 CONTINUE
    READ(15,7) I, J, XX, YY, ZZ
    IF((I.GT.M2).OR.(I.EQ.0)) GOTO 11
    IF(I.LE.J) GOTO 10
```

If I or J are either one greater than M2, the values of XX and YY will be zero; that is, there are no conductances stored at those locations. We skip the following two statements in that case.

```
    IF(XX.LE.1.0D-10) GOTO 10
    IF(YY.LE.1.0D-10) GOTO 10
    G(I,J) = 1.0D0/XX + 1.0D0/YY
    G(I,J) = 1.0D0/G(I,J)
    G(J,I) = G(I,J)
    GOTO 10
 11 CONTINUE
    READ(10,1) I, AI, CI
    IF(I.EQ.0) GOTO 12
    AA(I) = AI
    CC(I) = CI
    GOTO 11
 12 CONTINUE
    READ(11,2) L, M, I, DFADUM
    IF(L.EQ.0) GOTO 13
    DFA(L,M,I) = DFADUM
    GOTO 12
 13 CONTINUE
    READ(12,3) I, J, DFCIJ
    IF(I.EQ.0) GOTO 14
    DFC(I,J) = DFCIJ
    GOTO 13
 14 CONTINUE
```

Initialize 'old' node temperatures. This establishes the initial conditions for the solution.

```
    DO 15 I = 1, M2
    TO(I) = HST
    TN(I) = TO(I)
 15 CONTINUE
```

Write out initial cavity temperature distribution.

```
    WRITE(6,8) ETIME
    DO 301 I5 = NTH, M2 , NTH
    I9 = I5 - NTH + 1
    WRITE(6,6) (TN(I3), I3 = I9, I5)
    WRITE(16,6) (TN(I3), I3 = I9, I5)
301 CONTINUE
```

Zero the 'QTOT' accumulator.

```
    DO 987 I = 1,4
    QTOT(I) = 0.0D0
987 CONTINUE
```

We "march in time" by letting (if desired) the heat sink and calibration source temperatures change with each time increment DT. We assume that DT is sufficiently small that each incremental change in these temperatures is small.

The source and heat sink temperatures are changed in the 'K' DO-loop (if desired; this is an option not actually used in this example).

```
    KOUNT = 0
    DO 280 K = 1, NDT
```

Compute the new radiometer body temperature distribution every tenth time increment.

```
      KOUNT = KOUNT + 1
      IF(KOUNT.EQ.10) GOTO 678
      GOTO 679
678   CONTINUE
      KOUNT = 0
219   CONTINUE
      DO 230 J = 1,NBODY
      CJ = CBODY(J)*VBODY(J)
      CJ = DTBODY/CJ
      SUM1 = 0.0D0
      SUM2 = 0.0D0
      SIGMA = SIG*AO/10000.0D0
      DO 220 I = 1,NBODY
      IF(I.EQ.J) GOTO 220
      FACTOR = EBODY(I)*SIG*ABODY(I)*DBODY(I,J)
      FACTOR = FACTOR*TGBODY(I)*TGBODY(I)*TGBODY(I)*TGBODY(I)
      FACTOR = FACTOR + GBODY(I,J)*TGBODY(I)
      SUM1 = SUM1 + FACTOR
      SUM2 = SUM2 + GBODY(I,J)
220   CONTINUE
      IF(J.GE.1.AND.J.LT.5)
     &       QBODY(J) = SIGMA*SOURCE(5)*TSOURC**4
      IF(J.GE.5.AND.J.LT.9)
     &       QBODY(J) = SIGMA*SOURCE(9)*TSOURC**4
      IF(J.GE.9.AND.J.LT.13)
     &       QBODY(J) = SIGMA*SOURCE(13)*TSOURC**4
      IF(J.GE.17.AND.J.LT.21)
     &       QBODY(J) = SIGMA*SOURCE(17)*TSOURC**4
      IF(J.GE.29.AND.J.LT.33)
     &       QBODY(J) = SIGMA*SOURCE(21)*TSOURC**4
      IF(J.GE.33.AND.J.LT.37)
     &       QBODY(J) = SIGMA*SOURCE(25)*TSOURC**4
      IF(J.GE.37.AND.J.LT.41)
     &       QBODY(J) = SIGMA*(SOURCE(29)+SOURCE(33))*TSOURC**4
      IF(J.GE.41.AND.J.LT.45)
     &       QBODY(J) = SIGMA*SOURCE(37)*TSOURC**4
      IF(J.GT.20.AND.J.LT.25) QBODY(J) = 3.46D0*TCLAMP
      IF(J.GT.24.AND.J.LT.29) QBODY(J) = 0.22D0*TCLAMP
      IF(J.GT.48.AND.J.LT.53) GOTO 636
      GOTO 637
636   CONTINUE
      ILL = J - 48
      QBODY(J) = QTOT(ILL)/10.0D0 + GLINK*T1BODY(J)
637   CONTINUE
      TOP = CJ*(SUM1 + QBODY(J)) + T1BODY(J)
      BOTTOM = (DBODY(J,J) - 1.0D0)*EBODY(J)*SIG*ABODY(J)
      BOTTOM = BOTTOM*TGBODY(J)*TGBODY(J)*TGBODY(J)
      IF(J.GT.20.AND.J.LT.25) SUM2 = SUM2 + 3.46D0
      IF(J.GT.24.AND.J.LT.29) SUM2 = SUM2 + 0.22D0
      IF(J.GT.48.AND.J.LT.53) SUM2 = SUM2 + GLINK
      BOTTOM = 1.0D0 + CJ*(SUM2 - BOTTOM)
```

```
      TUBODY(J) = TOP/BOTTOM
      IF(J.GT.76.AND.J.LT.81) TUBODY(J) = HST
230   CONTINUE
      SUM = 0.0D0
      AVE = 0.0D0
      DO 240 I = 1,NBODY
      BRMS = TGBODY(I) - TUBODY(I)
      BRMS = BRMS*BRMS
      SUM = SUM + BRMS
      AVE = AVE + TGBODY(I)
240   CONTINUE
      BRMS = SUM/RNBODY
      BRMS = DSQRT(BRMS)
      AVE = AVE/RNBODY
      TEST = BRMS/AVE
      IF(TEST.LE.0.000001) GOTO 241
      DO 250 I = 1,NBODY
      TGBODY(I) = TUBODY(I)
250   CONTINUE
      GOTO 219
241   CONTINUE
      DO 260 I = 1,NBODY
      IF(I.LE.4) QTOT(I) = 0.0D0
      T1BODY(I) = TUBODY(I)
260   CONTINUE
      EETIME = ETIME + DT
      WRITE(6,88) EETIME
      DO 246 IS = 4, 104, 4
      IX = IS - 3
      WRITE(6,66) (T1BODY(I17), I17 = IX, IS)
246   CONTINUE
      QBOD = 0.0D0
      DO 661 I = 1,104
      ALPHA = 0.96D0
      IF(I.LT.13) ALPHA = 0.05D0
      QBOD = QBOD
     &       + ALPHA*ABODY(I)*SIG*DBODY(I,106)*T1BODY(I)**4
661   CONTINUE
      QBOD = QBOD/4.0D0
679   CONTINUE
```

Compute the elapsed time.

```
      ETIME = K*DT
```

The 'ITER' DO-loop loops through the temperature calculations until convergence is obtained.

```
    DO 400 ITER = 1, ITRMAX
    RMS = 0.0D0
    TSUM = 0.0D0
    RMSMAX = 0.0D0
```

The 'I' DO-loop steps through the nodes.

```
    DO 18 I = 1, M2
```

Compute the (Level/Position) coordinates of Node I.

```
    I1 = I - 1
    I1 = I1/NTH
    I2 = I - I1*NTH
    I1 = I1 + 1
```

Compute the indices of the nodes surrounding Node I.

```
    JJJ1 = I - 1
    IF(I2.EQ.1) JJJ1 = JJJ1 + NTH
    JJJ2 = I - NTH
    JJJ3 = I + 1
    IF(I2.EQ.NTH) JJJ3 = JJJ3 - NTH
    JJJ4 = I + NTH
    IF(JJJ2.LE.0) JJJ2 = I
    IF(JJJ4.GT.M2) JJJ4 = I
```

Compute the heat flow rate radiated into each node during the calculation time interval DT.

```
    Q1 = 0.0D0
    DFSUM = 0.0D0
```

Loop through the J nodes which radiate to Node I.

```
    DO 22 J = 1, M2
```

Compute (Level/Position) coordinates of Node J.

```
      J1 = J - 1
      J1 = J1/NTH
      J2 = J - J1*NTH
      J1 = J1 + 1
```

If Node J is in the first position on Level J1, i.e., if J2 = 1, the appropriate distribution factors have been read in; otherwise they must be found by a coordinate transformation.

```
      IF(J2.EQ.1) GOTO 20
```

Coordinate transformation; rotate source Node J to first position on Level J1 and rotate receiver node an equal number of positions on Level I1.

```
      JJ = J1
      II = (I1 - 1)*NTH + I2 - J2 + 1
      IF((I2 - J2).LT.0) II = II + NTH
      GOTO 21
```

The distribution factor has been read in directly.

```
 20 CONTINUE
      JJ = J1
      II = I
```

The distribution factor from Node J to Node I is the same as the distribution factor from Node JJ to Node II due to symmetry.

```
 21 CONTINUE
```

Calculate Z(I,J).

```
    IF(J.NE.I) GOTO 75
    II1 = (I1 - 1)*NTH + 1
    ZIJ = AA(I)*EC*SIG*TO(I)*TO(I)*TO(I)*DFC(I1,II1)
    Z(I,J) = -(EC + EB)*AA(I)*SIG*TO(I)*TO(I)*TO(I)
    Z(I,J) = Z(I,J)  + ZIJ - CC(I)/DT
    Z(I,J) = Z(I,J) - G(JJJ1,I) - G(JJJ2,I)
    Z(I,J) = Z(I,J) - G(JJJ3,I) - G(JJJ4,I)
    GOTO 76
75  Z(I,J) = EC*SIG*TO(J)*TO(J)*TO(J)*AA(J)*DFC(JJ,II)
    Z(I,J) = Z(I,J) + G(I,J)
```

There is no heat transfer by conduction from a node to to itself; thus if J = I, we skip the 'Q1' summation.

```
76  IF(J.EQ.I) GOTO 22
    Q1 = Q1 + Z(I,J)*TN(J)
    DFSUM = DFSUM + DFC(II,JJ)
22  CONTINUE
    Q2 = 1.0D0 - DFSUM
    Q2 = Q2*AA(I)*EC
    Q2 = Q2*SIG*TSOURC*TSOURC*TSOURC*TSOURC
```

The heat input into the first ring of cavity nodes must include that from the F-O-V limiter; thus, if 'I' is less than or equal to 4, we must add the heat 'QBOD' from the F-O-V limiter.

```
    IF(I.LE.4) Q2 = Q2 + QBOD
```

Calculate the quantity RHS(I).

```
    RHS(I) = - AA(I)*EB*SIG*HST*HST*HST*HST
    Q2 = Q2 + HEAT*AA(I)
    RHS(I) = RHS(I) - Q2  - CC(I)*TO(I)/DT
```

Check to see if the matrix inversion method has been chosen; if so, proceed to next node.

```
    IF(MAT.EQ.1) GOTO 888
```

Calculate TNEW for this node if the iterative solution has been chosen.

```
      IF(I.LE.4) GOTO 188
      GOTO 189
188   CONTINUE
      IHLP = I + 48
      TNEW = (Q1 - RHS(I) -
     &       GLINK*T1BODY(IHLP))/(-Z(I,I) - GLINK)
      GOTO 190
189   CONTINUE
      TNEW = (Q1 - RHS(I))/(-Z(I,I))
190   CONTINUE
      TRMSMX = (TNEW - TN(I))*(TNEW - TN(I))
      RMS = RMS + TRMSMX
      IF(TRMSMX.GT.RMSMAX) RMSMAX = TRMSMX
      TSUM = TSUM + TNEW
      TN(I) = TNEW
      GOTO 18
888   CONTINUE
      IF(I.LE.4) GOTO 517
      GOTO 518
517   CONTINUE
      IHLP = I + 48
      RHS(I) = RHS(I) - GLINK*T1BODY(IHLP)
      Z(I,I) = Z(I,I) - GLINK
518   CONTINUE
 18   CONTINUE
```

If the matrix solution method has been chosen, call the matrix inversion Subroutine GAUSS.

```
      IF(MAT.EQ.0) GOTO 73
      CALL GAUSS(Z,RHS,M2,NARSZ,IANS)
      IF(IER.EQ.131) STOP
      IF(IER.EQ.130) STOP
      DO 288 NP = 1, M2
      TN(NP) = RHS(NP)
288   CONTINUE
      GOTO 270
```

Determine if convergence has been obtained.

```
 73 RMS = DSQRT(RMS)
    RMSMAX = DSQRT(RMSMAX)
    ERROR = RMS*100.0/TSUM
    ERRMAX = RMSMAX*100.0*FM2/TSUM
    IF((ERRMAX.LE.ELIMIT).AND.
   &       (ITER.GT.ITRMIN)) GOTO 28
400 CONTINUE
410 CONTINUE
 28 CONTINUE
```

Write the value of the elapsed time (s).

```
270 WRITE(6,8) ETIME
```

Write out current cavity temperature distribution at the end of each integration step.

```
    DO 300 I5 = NTH, M2 , NTH
    I9 = I5 - NTH + 1
    WRITE(6,6) (TN(I3), I3 = I9, I5)
    WRITE(16,6) (TN(I3), I3 = I9, I5)
300 CONTINUE
    DO 299 IJK = 1, M2
    TO(IJK) = TN(IJK)
    IJKL = IJK + 48
    IF(IJK.LE.4) LL = IJK
    IF(IJK.LE.4) QTOT(LL) = QTOT(LL)
   &       + GLINK*(TO(IJK) - T1BODY(IJK))
299 CONTINUE
280 CONTINUE
```

Format statements.

```
 1 FORMAT(5X, I5, 2D14.5)
 2 FORMAT(5X, 3I5, D14.5)
 3 FORMAT(5X, 2I5, 2D14.5)
 4 FORMAT(5X, F10.4)
 5 FORMAT(5X, 2I5, D14.5)
 6 FORMAT(5X, 2(F8.3),2(F8.3),2(F8.3),
  &       2(F8.3),2(F8.3))
 7 FORMAT(5X, 2I5, 3D14.5)
 8 FORMAT(10X, ///, 5X, 'ELAPSED TIME =',
  &       F6.1,' SECOND(S).', //)
 9 FORMAT(5X/5X,'CONVERGENCE NOT OBTAINED,',
  &       'ERROR =',E11.4,' PERCENT'/)
66 FORMAT(45X, 2(F8.3),2(F8.3),2(F8.3),2(F8.3),2(F8.3))
88 FORMAT(44X, 'ELAPSED TIME =',F6.1,' SECOND(S).',//)
81 FORMAT(4X,I3,3X,F4.2,1X,F4.2,2X,F5.3,5X,F5.3,5X,F5.3)
82 FORMAT(4X,I3,3X,F6.4)
83 FORMAT(5X,I2,2X,I3,2X,F5.2)
92 FORMAT(5X, 2I5, D14.5)
93 FORMAT(//5X,'ITERATION MAXIMUM ERROR LIMIT ',
  &       '= ',F10.8,' PERCENT.',/,'1')
94 FORMAT('1',////////////,15X,'TRANSIENT',
  &       'TEMPERATURE DISTRIBUTION OF A',/,15X,
  &       'TRW TYPE INVERTED CONE CAVITY RADIOMETER.',
  &       //,14X,'DEVELOPED BY J. R. MAHAN AND ',
  &       'L. D. ESKIN AT',/,11X,'VIRGINIA ',
  &       'POLYTECHNIC INSTITUTE & STATE UNIVERSITY',
  &       /,17X,'FOR THE NASA LANGLEY RESEARCH CENTER.')
95 FORMAT(10X,'THE SOLUTION HAS SWITCHED ',
  &       'TO THE ITERATION METHOD.')
96 FORMAT(5X,'THE SYSTEM VARIABLES ARE AS ',
  &       'FOLLOWS:' ,//10X,'A = ',F5.2 ,' MM',
  &       /,10X,'B = ',F5.2,' MM',/,10X,'C = ',
  &       F5.2,' MM',/,10X,'N1 = ',I2,/,10X,'N2 '
  &       ,'= ',I2,/,10X,'NTH =',I2,/,10X,'NSHOTN = '
  &       ,I6,/,10X,'ABS = ',F3.1,/,10X'REFR = '
  &       '= ',I6,/,10X,'ABS = ',F3.1,/,10X,'REFR = '
  &       ,F3.1,/,10X,'DT = ',F4.1,/,10X,'ELIMIT = '
  &       ,E8.2,/10X,'TEMPERATURE SOLUTION METHOD ',
  &       ' (0=ITERATION,1=MATRIX) = ',I2)
97 FORMAT(F5.2,F5.2,F5.2,I2,I2,I2,I6,
  &       F3.1,F3.1,F4.1,E8.2,4I1)
```

```
 98 FORMAT(////15X,'NUMBER OF CAVITY NODES = '
   &       ,I3,//,15X,I2,' CAVITY NODE RINGS WITH ',I2,
   &       ' NODES PER RING.',//,1X,
   &       T27,'SYSTEM GEOMETRY',//1X,T41,'____',/
   &       ,1X,T27,'¦<----2C---->¦ ¦ ¦'/,1X,T27,'¦',T40,
   &       '¦ ¦ ¦',/1X,T27,'¦',T40,'¦ A ¦'/,1X,T27,'¦'
   &       ,T40,'¦ ¦ ¦'/,1X,T27,'¦',T40,'¦ V ¦',/'+',T42
   &       ,'_',/1X,T28,'*',T39,'*',T44,'B'/,1X,T29,'*',
   &       T38,'*',T44,'¦'/ ,1X,T30,'*',T37,'*',T44,'¦'/
   &       ,1X,T31,'*',T36,'*',T44,'¦'/ ,1X,T32,'*',T35
   &       ,'*',T44,'¦',/,1X,T33,'*',T34,'*',T44,'V',/
   &       ,'+',T35,'___________',//,5X,'A = ',F5.2,
   &       ' MM',5X,'B = ',F5.2,' MM',5X,'C = ',F5.2
   &       ,' MM',/'1')
 99 FORMAT(5X,/,5X,'THE AVERAGE RMS ERROR = '
   &       ,F10.8,' PERCENT AFTER ',I2, ' ITERATIONS .'
   &       //,5X,'THE MAXIMUM RMS ERROR = ',F10.8,
   &       ' PERCECNT.',//,5X, 'THE HEAT SINK ',
   &       'TEMPERATURE IS ',F5.1,' K.',/)
```

End of 'TRW TEMP'.

```
      STOP
      END
```

## 4.3. Subroutine GAUSS

This is a standard matrix inversion subroutine which could be replaced at the discretion of the user if the argument list of the replacement routine was the same. We present this subroutine without further explanation.

## 4.3.1. Subroutine Listing

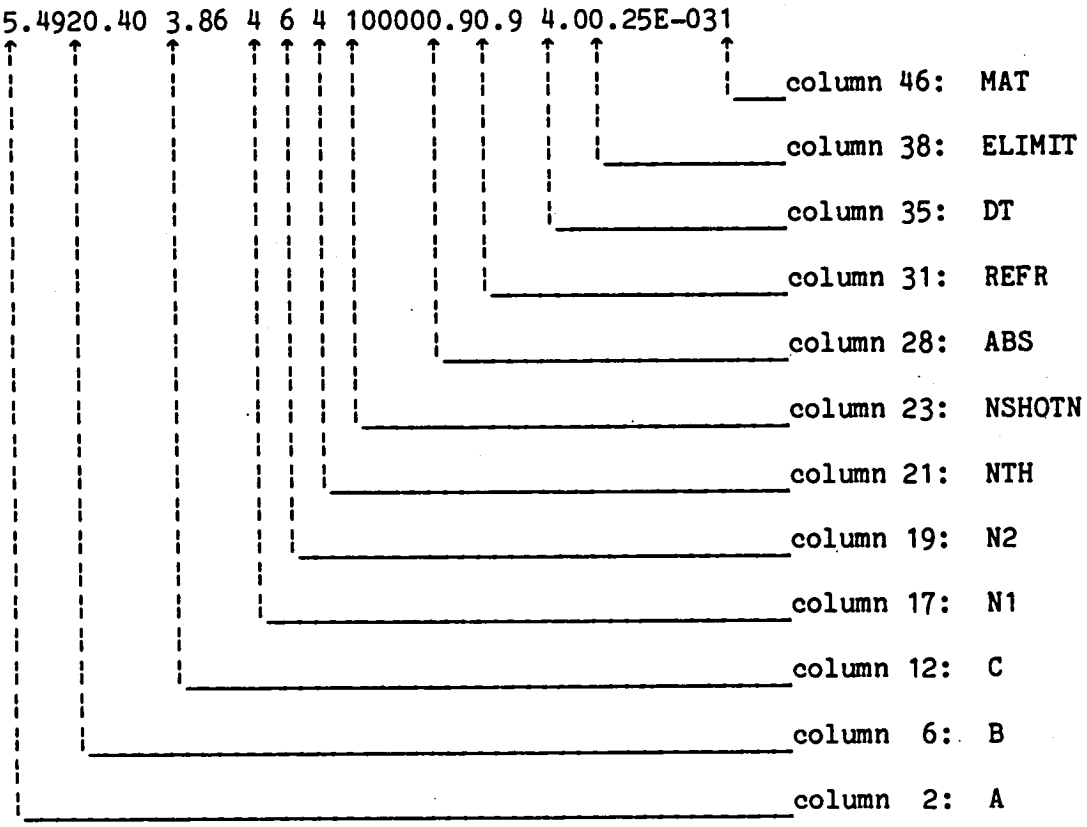
```
      SUBROUTINE GAUSS(A,B,N,NARSZ,IANS)

      IMPLICIT REAL*8(A-H,O-Z), INTEGER(I-N)
      INTEGER IANS(NARSZ)
      REAL*8 A(NARSZ,NARSZ),B(NARSZ)

      COMMON /EMSGC/ IER

      DO 10 I = 1,N
      IANS(I) = I
      AM = DABS(B(I))

      DO 20 J = 1, N
      BM = DABS(A(I,J))
      IF ( BM .GT. AM ) AM = BM
   20 CONTINUE

      IF(AM.EQ.0.0) GOTO 901

   10 CONTINUE

      N1 = N + 1

      DO 100 L = 1,N
      AMX = DABS(A(L,L))
      I = L
      J = L

      DO 110 I1 = L,N

      DO 120 J1 = L,N

      F1 = DABS(A(I1,J1))
      IF(AMX.GE.F1) GOTO 120
      AMX = F1
      I = I1
      J = J1

  120 CONTINUE

  110 CONTINUE

      IF(I.EQ.L) GOTO 140

      DO 130 K = 1,N
      F1 = A(I,K)
      A(I,K) = A(L,K)
      A(L,K) = F1
  130 CONTINUE
```

```
      F1 = B(I)
      B(I) = B(L)
      B(L) = F1

140 CONTINUE

      IF(J.EQ.L) GOTO 160

      DO 150 K = 1,N
      F1 = A(K,J)
      A(K,J) = A(K,L)
      A(K,L) = F1
150 CONTINUE

      III = IANS(J)
      IANS(J) = IANS(L)
      IANS(L) = III

160 CONTINUE

      F1 = A(L,L)
      IF(F1.EQ.0.0) GOTO 902

      DO 170 LP =1,N
      A(L,LP) = A(L,LP)/F1
170 CONTINUE

      B(L) = B(L)/F1

      IF(N.EQ.L) GOTO 200
      N2 = L + 1

      DO 180 M = N2,N

      F1 = A(M,L)
      IF(F1.EQ.0.0) GOTO 182

      DO 185 LP = 1,N
      A(M,LP) = A(M,LP)/F1 - A(L,LP)
185 CONTINUE

      B(M) = B(M)/F1 - B(L)

182 CONTINUE

180 CONTINUE

100 CONTINUE

200 CONTINUE
```

```
      F1 = 0.0
      N3 = N-1

      DO 210 LP = 1,N3
      L = N - LP
      F1 = 0.0
      N4 = L + 1

      DO 220 K = N4,N
      F1 = F1 + B(K)*A(L,K)
220   CONTINUE

      B(L) = B(L) - F1

210   CONTINUE

      DO 250 LP = 1,N
      A(1,LP) = B(LP)
250   CONTINUE

      DO 260 I = 1,N
      IA = IANS(I)
      B(IA) = A(1,I)
260   CONTINUE

      GOTO 999

901   IER = 130

      WRITE(6,666)
666   FORMAT(/,' GAUSS(F):  ZERO ROW INPUT')
      GOTO 999

902   IER = 131

      WRITE(6,777)
777   FORMAT(/,' GAUSS(F):  SINGULAR MATRIX')
999   CONTINUE

      RETURN
      END
```

## 5. TRW DATA

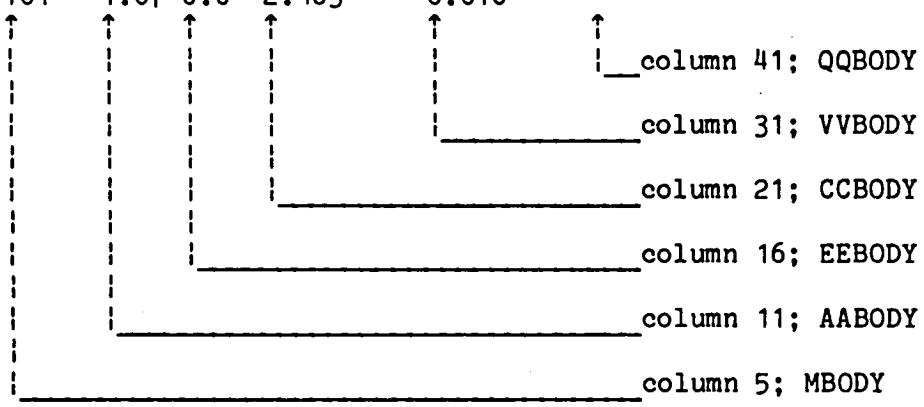
The data read by 'TRW SHAPE', 'TRW NODES' and 'TRW TEMP' must be entered at the appropriate position at the end of each of the three programs. Note that not all of the programs use, or even read, all of the input data. However, the same data file serves all three.




| Variable | Type | Field | Description |
|---|---|---|---|
| A | F5.2 | 1-5 | Height of barrel (mm) |
| B | F5.2 | 6-10 | Height of entire cavity (mm) |
| C | F5.2 | 11-15 | Radius of barrel (mm) |
| N1 | I2 | 16-17 | Number of barrel axial divisions |
| N2 | I2 | 18-19 | Number of cone axial divisions |
| NTH | I2 | 20-21 | Number of circumferential divisions |
| NSHOTN | I6 | 22-27 | Number of volleys fired |
| ABS | F3.1 | 28-30 | Absorptivity of cavity |
| REFR | F3.1 | 31-33 | Reflectivity ratio |
| DT | F4.1 | 34-37 | Cavity calculation time interval |
| ELIMIT | E8.2 | 38-45 | Accuracy criterion using iteration |
| MAT | I1 | 46 | = 1, matrix solution; = 0, iteration |

## 6. BODY DATA

```
  1   2.58 0.05 2.425     1.020     0.000
  5   2.58 0.05 2.425     0.710     0.000
  9   2.58 0.05 2.425     0.570     0.000
 13   0.00 0.0  2.425     0.580
 17   3.05 0.96 2.425     1.900     0.000
 21   8.24 0.0  2.425     9.080
 25   0.00 0.0  2.425     1.980
 29   3.90 0.96 2.425     1.330     0.000
 33   1.56 0.96 2.425     1.200     0.000
 37   1.20 0.96 2.425     0.120     0.000
 41   0.24 0.96 2.425     0.260     0.000
 45   0.46 0.0  2.425     0.530
 49   0.51 0.0  2.463     0.003     0.000
 53   1.22 0.0  2.425     1.970
 57   0.56 0.0  2.463     0.003
 61   0.00 0.0  2.425     1.200
 65   0.40 0.0  3.430     0.240
 69   0.00 0.0  2.425     1.560     0.000
 73   0.52 0.0  3.430     0.310
 77   0.00 0.0  3.430     1.540     0.000
 81   0.51 0.0  3.430     0.310
 85   3.04 0.0  2.425     1.170     0.000
 89   0.41 0.0  3.430     0.540
 93   4.87 0.0  2.425     1.340
 97   8.70 0.0  2.425     0.990
101   1.07 0.0  2.463     0.016
```




| Variable | Type | Field | Description |
|---|---|---|---|
| MBODY | I3 | 5-7 | Body node number |
| AABODY | F4.2 | 11-14 | Body node surface area |
| EEBODY | F4.2 | 16-19 | Body node emissivity |
| CCBODY | F5.3 | 21-25 | Body node thermal capacity |
| VVBODY | F5.3 | 31-35 | Body node volume |
| QQBODY | F5.3 | 41-45 | Body node heat input |

These variables are described in detail in Section 4.1.

```
  1   0.0592
  5   0.0412
  9   0.0320
 13   0.0340
 17   0.1384
 21   0.6240
 25   0.1240
 29   0.1760
 33   0.3440
 37   0.0920
 41   0.2000
 45   0.4000
 49   0.0192
 53   0.8000
 57   0.0208
 61   0.4400
 65   0.6800
 69   0.5600
 73   0.8800
 77   1.0800
 81   0.8800
 85   3.9200
 89   1.5600
 93   6.7600
 97   4.6400
101   1.0800
```

column 11; COND

column 5; MBODY

| Variable | Type | Field | Description |
|---|---|---|---|
| MBODY | I3 | 5-7 | Body node number |
| COND | F6.4 | 11-16 | Conductance between body nodes on MBODY |

| | | |
|---|---|---|
| 1 | 5 | 8.95 |
| 2 | 6 | 8.95 |
| 3 | 7 | 8.95 |
| 4 | 8 | 8.95 |
| 5 | 9 | 6.59 |
| 6 | 10 | 6.59 |
| 7 | 11 | 6.59 |
| 8 | 12 | 6.59 |
| 9 | 13 | 5.32 |
| 10 | 14 | 5.32 |
| 11 | 15 | 5.32 |
| 12 | 16 | 5.32 |
| 13 | 17 | 11.23 |
| 14 | 18 | 11.23 |
| 15 | 19 | 11.23 |
| 16 | 20 | 11.23 |
| 13 | 21 | 2.08 |
| 14 | 22 | 2.08 |
| 15 | 23 | 2.08 |
| 16 | 24 | 2.08 |
| 17 | 21 | 7.20 |
| 18 | 22 | 7.20 |
| 19 | 23 | 7.20 |
| 20 | 24 | 7.20 |
| 17 | 29 | 3.91 |
| 18 | 30 | 3.91 |
| 19 | 31 | 3.91 |
| 20 | 32 | 3.91 |
| 21 | 25 | 4.33 |
| 22 | 26 | 4.33 |
| 23 | 27 | 4.33 |
| 24 | 28 | 4.33 |
| 29 | 33 | 2.63 |
| 30 | 34 | 2.63 |
| 31 | 35 | 2.63 |
| 32 | 36 | 2.63 |
| 33 | 37 | 0.66 |
| 34 | 38 | 0.66 |
| 35 | 39 | 0.66 |
| 36 | 40 | 0.66 |
| 33 | 41 | 0.02 |
| 34 | 42 | 0.02 |
| 35 | 43 | 0.02 |
| 36 | 44 | 0.02 |
| 33 | 45 | 0.04 |
| 34 | 46 | 0.04 |
| 35 | 47 | 0.04 |
| 36 | 48 | 0.04 |
| 33 | 53 | 0.05 |

| | | |
|---|---|---|
| 34 | 54 | 0.05 |
| 35 | 55 | 0.05 |
| 36 | 56 | 0.05 |
| 37 | 41 | 0.00 |
| 38 | 42 | 0.00 |
| 39 | 43 | 0.00 |
| 40 | 44 | 0.00 |
| 41 | 45 | 7.11 |
| 42 | 46 | 7.11 |
| 43 | 47 | 7.11 |
| 44 | 48 | 7.11 |
| 45 | 53 | 4.03 |
| 46 | 54 | 4.03 |
| 47 | 55 | 4.03 |
| 48 | 56 | 4.03 |
| 49 | 57 | 0.03 |
| 50 | 58 | 0.03 |
| 51 | 59 | 0.03 |
| 52 | 60 | 0.03 |
| 53 | 61 | 7.77 |
| 54 | 62 | 7.77 |
| 55 | 63 | 7.77 |
| 56 | 64 | 7.77 |
| 57 | 65 | 0.06 |
| 58 | 66 | 0.06 |
| 59 | 67 | 0.06 |
| 60 | 68 | 0.06 |
| 61 | 65 | 1.99 |
| 62 | 66 | 1.99 |
| 63 | 67 | 1.99 |
| 64 | 68 | 1.99 |
| 61 | 69 | 8.44 |
| 62 | 70 | 8.44 |
| 63 | 71 | 8.44 |
| 64 | 72 | 8.44 |
| 65 | 73 | 3.17 |
| 66 | 74 | 3.17 |
| 67 | 75 | 3.17 |
| 68 | 76 | 3.17 |
| 69 | 73 | 2.59 |
| 70 | 74 | 2.59 |
| 71 | 75 | 2.59 |
| 72 | 76 | 2.59 |
| 69 | 77 | 8.92 |
| 70 | 78 | 8.92 |
| 71 | 79 | 8.92 |
| 72 | 80 | 8.92 |
| 73 | 81 | 2.82 |
| 74 | 82 | 2.82 |
| 75 | 83 | 2.82 |
| 76 | 84 | 2.82 |

```
77   81   4.51
78   82   4.51
79   83   4.51
80   84   4.51
77   85  13.46
78   86  13.46
79   87  13.46
80   88  13.46
81   89   2.06
82   90   2.06
83   91   2.06
84   92   2.06
85   89   9.17
86   90   9.17
87   91   9.17
88   92   9.17
85   93   0.72
86   94   0.72
87   95   0.72
88   96   0.72
89  101   0.03
90  102   0.03
91  103   0.03
92  104   0.03
93   97   2.47
94   98   2.47
95   99   2.47
96  100   2.47
¦    ¦     ¦
¦    ¦     ¦_____________________________column 16; GBODY(MIBODY,MJBODY)
¦    ¦
¦    ¦_____________________________________column 10; MJBODY
¦
¦_______________________________________column  6; MIBODY
```

| Variable | Type | Field | Description |
|---|---|---|---|
| MIBODY | I2 | 6-7 | Body node 'level' index |
| MJBODY | I3 | 10-12 | Body node 'level' index |
| GBODY | F5.2 | 16-20 | Conductance between MIBODY and MJBODY |

| 1. Report No. | 2. Government Accession No. | 3. Recipient's Catalog No. |
|---|---|---|
| NASA CR-172240 | | |

| 4. Title and Subtitle | 5. Report Date |
|---|---|
| TRANSIENT THERMAL MODELING OF THE NONSCANNING ERBE DETECTOR | November 1983 |
| | 6. Performing Organization Code |

| 7. Author(s) | 8. Performing Organization Report No. |
|---|---|
| J. R. Mahan | |
| | 10. Work Unit No. |

| 9. Performing Organization Name and Address | 11. Contract or Grant No. |
|---|---|
| Mechanical Engineering Department<br>Virginia Polytechnic Institute and State University<br>Blacksburg, Virginia 24061 | NAS1-16508 |
| | 13. Type of Report and Period Covered |
| 12. Sponsoring Agency Name and Address | Contractor Report |
| National Aeronautics and Space Administration<br>Washington, DC 20546 | 14. Sponsoring Agency Code |

15. Supplementary Notes

Langley technical monitor: Dwayne E. Hinton
Final Report

16. Abstract

A numerical model has been developed to predict the transient thermal response of the ERBE nonscanning wide field-of-view total radiometer channel. The model, which uses Monte Carlo techniques to characterize the radiative component of heat transfer, is described and a listing of the computer program is provided. Application of the model to simulate the actual blackbody calibration procedure is discussed in detail. Recommendations are made for using the model to establish a real-time flight data interpretation strategy. Indications are also given for modifying the model to include a simulated Earth radiation source field and a filter dome.

| 17. Key Words (Suggested by Author(s)) | 18. Distribution Statement |
|---|---|
| Thermal modeling<br>Monte Carlo techniques<br>radiometry<br>Earth Radiation Budget Experiment (ERBE) | [redacted] Distribution<br>Subject Category 43 |

| 19. Security Classif. (of this report) | 20. Security Classif. (of this page) | 21. No. of Pages | 22. Price |
|---|---|---|---|
| Unclassified | Unclassified | 107 | |

Available: NASA Industrial Applications Centers