Optical Analysis of the ERBE Scanning Thermistor Bolometer Radiometer Using the Monte Carlo Method

by

Jeffrey L. Meekins

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Mechanical Engineering

APPROVED:

J. Robert Mahan, Chairman

Thomas E. Diller

Vatche' Vorperian

August, 1990

Blacksburg, Virginia

Optical Analysis of the ERBE Scanning Thermistor Bolometer Radiometer Using the Monte Carlo Method

by

Jeffrey L. Meekins

J. Robert Mahan, Chairman

Mechanical Engineering

(ABSTRACT)

In 1984, the Earth Radiation Budget Experiment (ERBE) was started by the National Aeronautics and Space Administration (NASA) to provide data to the meteorological community to predict long-term weather and climate trends. Three satellites employing nonscanning active cavity and scanning thermistor bolometer radiometers are orbiting the Earth to monitor its radiative emission. A numerical model has been formulated to better understand the performance of the ERBE scanning radiometer and to aid future radiometric design and calibration procedures.

The Monte Carlo method is applied to the ERBE scanning radiometer to spectrally characterize its optical and radiative performance. The optical analysis reveals that the ERBE scanning radiometer design successfully limits the amount of energy that reaches the active sensor to the designated instrument field-of-view. Distribution factors between the diffuse-specular surfaces of the scanning radiometer are calculated using the Monte Carlo method, and are then used to perform the radiative analysis. This analysis shows that less than three percent of the radiation emitted from the passive surfaces of the radiometer reaches the active sensor, an acceptable level for radiometric instrumentation used in space.

Dedication

To my mother, Maggie, and my father, Robert, for without your love, support and guidance, I may not have written this thesis.

To my brother, Anthony, who shares many of my aspirations. I hope this will inspire you to do your very best always.

Acknowledgments

Thank God from whom all blessings flow. I thank Him for life and for blessing me with intelligent parents who saw the value of education.

I gratefully acknowledge Professor J. Robert Mahan, who sensed that I was capable of switching from electrical to mechanical engineering, and gave me the opportunity to do so. Thank you for your support and guidance during my graduate studies.

I would also like to thank Professor Diller and Professor Vorperian for serving on my advisory committee.

Thanks are extended to NASA for its minority graduate research fellowship support under grant NGT-70088. Many thanks are extended to Mr. Robert B. Lee, III, at NASA Langley Research Center for his friendship and support. Thanks are also owed to Mr. Leonard Kopia, also at NASA Langley, for his technical assistance.

My gratitude is extended to the other graduate students, especially Noureddine Tira, for their help.

A very special *thank you* goes to my friend, Cynthia, who has been more help than she realizes.

And a “thank you” to anyone that I may have forgotten to acknowledge. Please, blame it on my mind, and not my heart.

Table of Contents

List of Illustrations

List of Tables

Nomenclature

A	Area (mm^2)
C_1	Constant used in Planck's spectral distribution (5.9544×10^8 W-$\mu m^4/m^2$)
C_2	Constant used in Planck's spectral distribution (1.4388×10^4 μm-K)
D_{ij}	Distribution factor (-)
N	Number of energy bundles as used in Eq. (3.1) (-)
P	Cumulative energy to active flake as used in Eq. (4.3) (-)
Q	Heat rate (W)
R_{spec}	Specular-to-total reflectivity ratio $(\frac{\rho^s}{\rho^s + \rho^d})$ (-)
T	Temperature (K)

Greek

α	Absorptivity (-)
ε	Emissivity (-)
λ	Wavelength (μm)

ρ	Reflectivity (-)
ρ^d	Diffuse component of reflectivity (-)
ρ^s	Specular component of reflectivity (-)
σ	Stefan-Boltzmann constant (5.6696×10^{-8} W/m^2-K^4)
τ	Transmissivity (-)
θ	Angle of incidence of collimated radiation (deg)

1.0 Introduction

The continuing rivalry between industrialist and environmentalist has excited a new concern in our times... global warming. Increasing levels of atmospheric constituents, such as carbon dioxide, are trapping some of the Earth-emitted longwave radiation causing the so-called greenhouse effect. Conversely, aerosol pollutants and particulate matter trapped in the upper atmosphere reflect incident solar radiation back into space causing global cooling. The net effect, whether it is global warming or cooling, warrants that everyone understand the Earth radiation budget and its potential effect on our climate 10, 20, even 50 years into the future.

The Earth Radiation Budget Experiment (ERBE) was developed by the National Aeronautics and Space Administration (NASA) to accurately measure the Earth radiation budget. For six years, ERBE has been monitoring the Earth radiation budget by satellite, returning information back to Earth to form a data base for use by meteorologists, climatologists, and atmospheric scientists world-wide to predict long-term weather and climate trends. A brief look into the history of ERBE is given to justify the importance and the meaning of the research described in this thesis.

1.1 A Brief History of Earth Radiation Budget Investigations

From 1900 to 1960, astrophysicists and climatologists had a profound impact on the scientific community. They were seeking the relationship between solar radiation and the climate. Before the advent of Earth-orbiting, Earth-observing systems, men like C. G. Abbot, F. E. Fowle, W. H. Dines, A. Danjon, and J. A. London managed to estimate the Earth radiation budget, including albedo[1], using radiation transfer models, a limited climatological data base, and much insight [1]. They realized that the cloud-ocean-atmosphere system of the Earth played an important role in the amount of longwave radiation emitted by the Earth and that it somehow impacted the weather. In fact, the quantity and variability of longwave Earth-emitted radiation directly affects our daily, monthly, and yearly weather patterns. Although the knowledge and technology of that era only allowed scientists like Abbot to extrapolate the effects of this complex, dynamical system, he had a vision that one day the Earth radiation budget could be measured via satellite.

On February 17, 1959, Abbot's vision became reality. The Explorer 6 meteorological satellite was successfully deployed from a Vanguard 2 rocket. The Explorer 6, equipped with flat-plate and hemispherical radiometers to measure the Earth radiation budget, never attained a stable orbit [2]. The radiometric instrumentation aboard this satellite was crude compared with today's standards, but advanced for that time.

Geist concludes that radiometers have been used since 1725 either for a particular application or to satisfy scientific curiosity [3]. The theoretical foundations of chemis-

[1] Albedo is the amount of incident solar radiation reflected by the Earth.

try, physics, and radiation heat transfer were advanced using radiometric instrumentation. So why not use radiometers specifically designed to measure the Earth energy balance to better understand our climate?

The attempt to orbit the Explorer 6 in 1959 began an era of three generations of Earth radiation budget satellite missions between 1960 and 1984. In 1979, twenty years after the first attempt at measuring the Earth radiation budget from space, a team of scientists met and discussed the history of these satellite missions and their experiences with such measurements [2]. Deliberations continued until 1984 when old knowledge, new ideas and a 25-year history of progress finally gave birth to ERBE.

1.2 The Earth Radiation Budget Experiment

The Earth Radiation Budget Experiment (ERBE) consists of three Earth-observing satellites employing radiometric instruments to monitor the Earth-reflected shortwave, Earth-emitted longwave, and Earth-incident solar radiation. A study by Harrison, *et al.* [4] determined the number of satellites and the number, size, and type of radiometric instruments needed for ERBE in order to provide the best spatial resolution and daily sampling necessary to accurately monitor the Earth radiation budget and its variability. The Earth Radiation Budget Experiment is the first satellite mission dedicated to measuring the Earth radiation budget using multiple satellites. On October 8, 1984, the first ERBE satellite, Earth Radiation Budget Satellite (ERBS), was launched from the space shuttle Challenger. The other two satellites designated National Oceanic and Atmospheric Administration (NOAA) 9 and 10 were launched December 12, 1984, and September 17, 1986, respectively. It was decided that each ERBE satellite should fly three narrow field-of-view (NFOV) scanning thermistor bolometer radiometers covering the total, shortwave, and longwave bands. Here shortwave refers to visible, or solar radiation, and longwave refers to Earth-emitted radiation. Each satellite also would carry two wide field-of-view (WFOV) and two medium field-of-view (MFOV) nonscanning active cavity radiometers, and a solar monitor [5]. Luther, *et al.* [6] describe the nonscanning active cavity radiometer and the solar monitor, whose designs are beyond the scope of this thesis. Mahan, *et al.* [7] and Tira, *et al.* [8] have formulated a dynamic electrothermal model of the nonscanning active cavity radiometers. They have found excellent agreement between the results of the model and the operational data of the actual instrument. The ERBE scanning radiometer design is discussed in Chapter 2.

The instruments aboard each satellite complement each other to provide the most accurate measurement of the Earth radiation budget to date. They were designed to measure the Earth radiation budget with only one-percent uncertainty during their projected two-year orbital lifetime [9,10]. The instruments aboard each satellite have outlived their projected lifetime with accuracies approaching the one-percent goal [11]. An article in **Science** reports the apparent success of ERBE instrumentation in monitoring the Earth radiation budget [12]. Inversion methods have been used to relate radiances at satellite altitudes to the radiation emitted from the top of the atmosphere (TOA)[2] where its value is most useful to meteorologists and climatologists [13]. These data have been used to develop models that determine the average monthly radiation budget and its variability on regional (250 × 250 km), zonal (10° Earth central angle = 1000 km diameter), and global scales. The Earth radiation budget map shown in Fig. 1 was produced using models developed for ERBE scanning radiometer data.

---

[2] TOA is defined as the radius of the Earth plus 30 km, encompassing the entire Earth-ocean-atmosphere system.

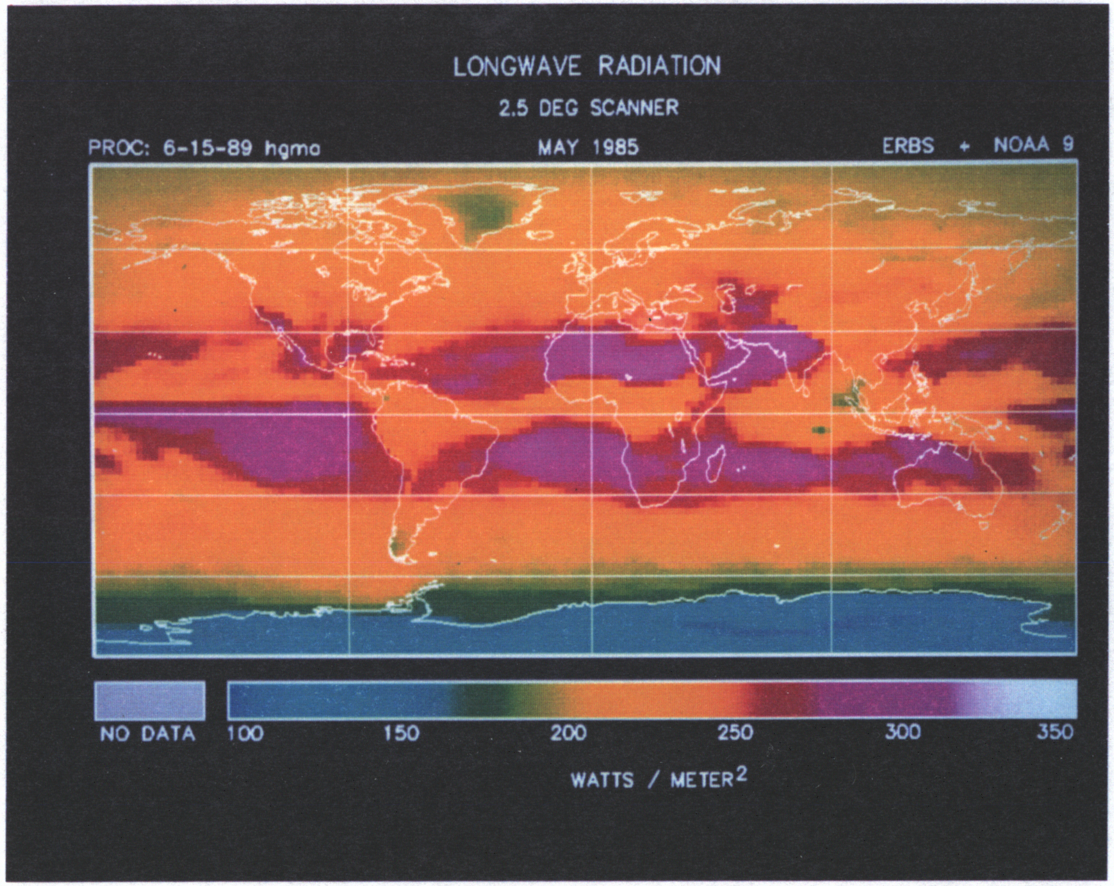


Fig. 1. Map of Longwave Radiative Emission from the Earth (May 1985, ERBS and NOAA 9 satellites, courtesy of Mr. R. B. Lee, III, NASA Langley Research Center).

1.3 The ERBE Scanning Radiometer: Its Mission Purpose

The ERBE scanning radiometer, shown in Fig. 2, detects radiation on regional scales, thereby providing the sampling and spatial resolution necessary to resolve some of the anisotropy of Earth-emitted radiation. Clouds account for two-thirds of the reflected solar radiation and one-half of the emitted longwave radiation [10] contributing to the anisotropic radiation. Clouds also contribute to Earth radiation budget variability because cloud cover is always changing.

Correctly identifying different types of Earth scenes (land, ocean, desert, etc.) is crucial to providing accurate Earth radiation budget data to the scientific community. Clouds make Earth-scene identification difficult because they scatter and refract much of the radiation leaving the Earth. Not only does the scanning radiometer contribute to the objectives outlined in Section 1.2, but ERBE scanning radiometer data are also used to define bidirectional reflectance models that aid in cloud identification and allow scientists to better identify the underlying Earth scene [13,14].

1.4 Afterthought

After years of study, ERBE has provided new insight and understanding about clouds and the Earth radiation budget. Subsequently, with the advent of new ideas and technology, NASA has responded with plans to launch another experiment in 1997 dubbed, CERES (pronounced "siris"), for Clouds and the Earth's Radiant Energy System. This investigation will focus on clouds and their elusive contribution to the Earth radiation budget. This experiment will use radiometers similar to the ERBE scanning radiometer.

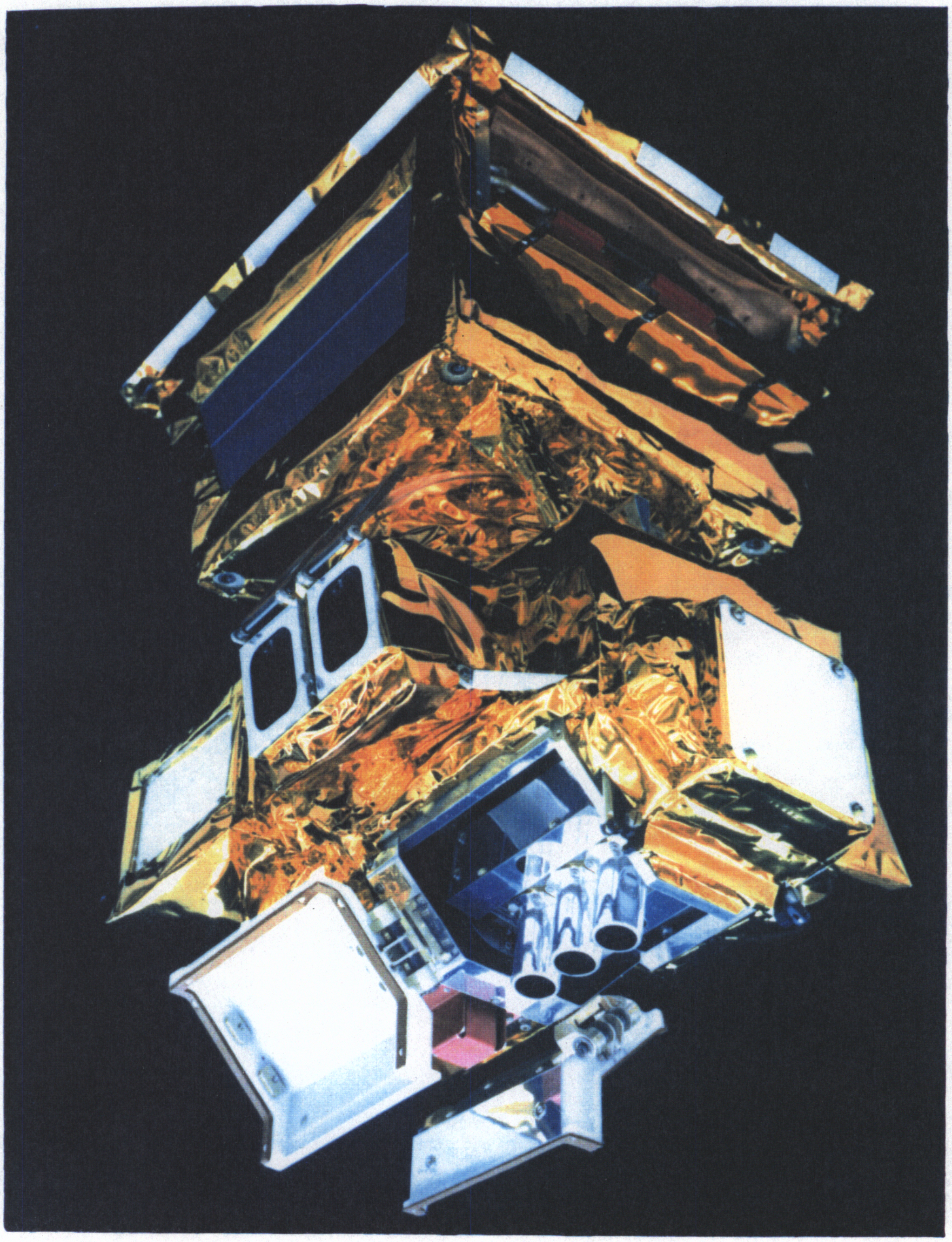


Fig. 2. The ERBE Scanning Radiometer (Courtesy of NASA Langley Research Center).

Practically, the CERES scanning radiometers will provide more than just Earth radiation budget measurements. The data these instruments provide will help resolve the hypothesis of global warming. For this reason, the instrumentation used in this investigation must perform as expected. These radiometers must reliably obtain data with the accuracy necessary to properly monitor the Earth radiation budget.

The main thrust of this research is to confirm that the ERBE scanning radiometers, and eventually the CERES scanning radiometers, operate as expected. Detailed radiometer modeling is essential to successful radiometer design and operation. The model described in this thesis will aid future scanning radiometer design, evaluate its calibration procedures, and provide the foundation for a completely dynamic, electrothermal model of such instruments.

2.0 Radiometer Optical Design

Figure 3 shows a diagram of the ERBE scanning radiometer. The scan head houses three telescopes boresighted to receive radiation from the same Earth scene, or pixel. Each telescope, or channel, is 25.4 mm in diameter, approximately 127 mm long, and monitors a different wavelength band of radiation emitted or reflected by the pixel. It has been established that the radiation emitted or reflected by a pixel can be separated into shortwave and longwave broadbands without any loss of information or any significant amount of energy leaking from one band into the other [13]. Therefore, two channels containing optical bandpass filters separate the radiant energy from the pixel into its shortwave and longwave components. The shortwave channel of the ERBE scanning radiometer contains two Suprasil-W2 silica glass windows that pass radiation between 0.2 and 5.0 μm. The longwave channel contains a diamond window painted with several coatings that pass radiation between 5.0 and 50.0 μm. The third channel is the total channel which receives all radiation from the pixel between 0.2 and 100.0 μm.

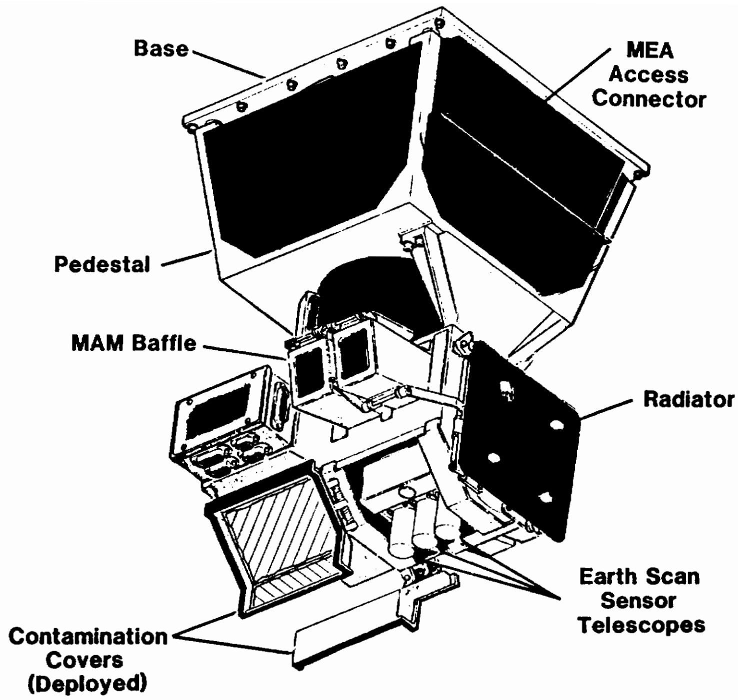


Fig. 3. The ERBE Scanning Radiometer Diagram (Courtesy of NASA Langley Research Center) [14].

Each channel (telescope) is an f/1.84 optical system based on Cassegrain's[3] design which uses two aspherical mirrors (primary and secondary). Implementing this design contributes greatly to a smaller instrument package. The basic Cassegrain telescope concept is shown in Fig. 4. The ERBE telescope is shown in Fig. 5. The mirrors of the ERBE telescope are aluminized Schott glass FK-52 with a special coating which protects the mirror surface from ultraviolet radiation. The coating is used because ultraviolet radiation degrades the finish of the mirror surface which ultimately changes the reflectivity of the mirrors. The primary mirror, held in place by a threaded detector header, or assembly, is a 15.74-mm O.D. and 5.84-mm I.D. concave paraboloid with a curvature of 25.53 mm. The secondary mirror, held in place by a three-spoke mount, is a 6.86-mm O.D. convex hyperboloid with a curvature of 23.50 mm. Remarkably, the diameter of the secondary mirror is about the diameter of the eraser on a No. 2 pencil! This mirror combination yields an effective focal length of 23.03 mm in a 7.5-mm long space, which is the distance between the vertex of each mirror.

The optics module shown in Fig. 6, which houses the ERBE telescope optics, is constructed of Al-6061, an aluminum alloy, coated mainly with Chemglaze™ containing microballoons. This coating has been selected because of its ability to absorb most of the radiation that enters the ERBE telescope off the optical axis. Chemglaze™, which is a black spacecraft coating with an approximate total absorptivity of 0.9, has been textured with microscopic glass balloons to encourage specular reflection of unabsorbed off-axis radiation. This coating, coupled with the geometry of the optics module, reduces scattering and severely attenuates any off-axis incident radiation.

---

[3] Invented in 1672 by French physician and inventor, N. Cassegrain.

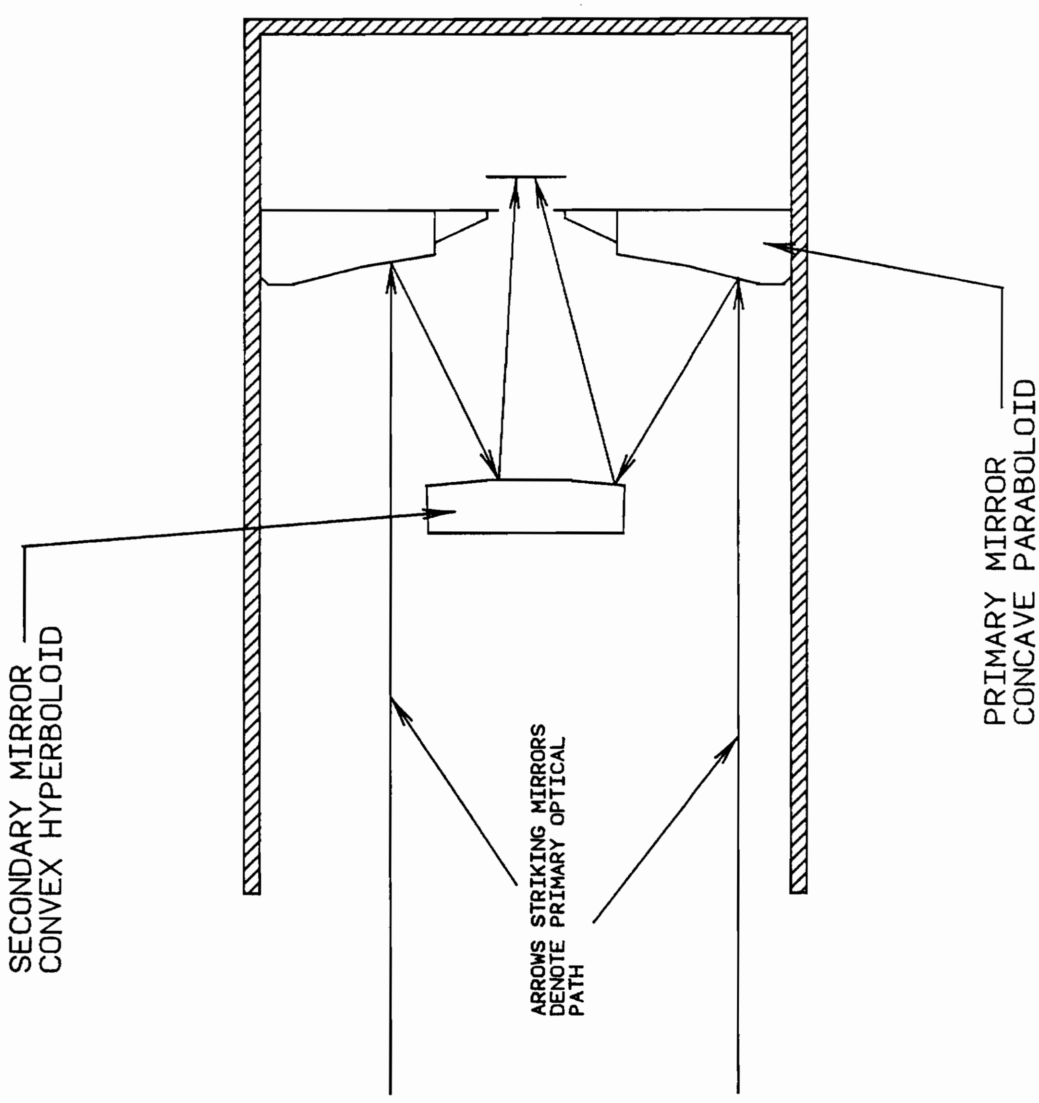


Fig. 4. The Cassegrain Telescope Concept.

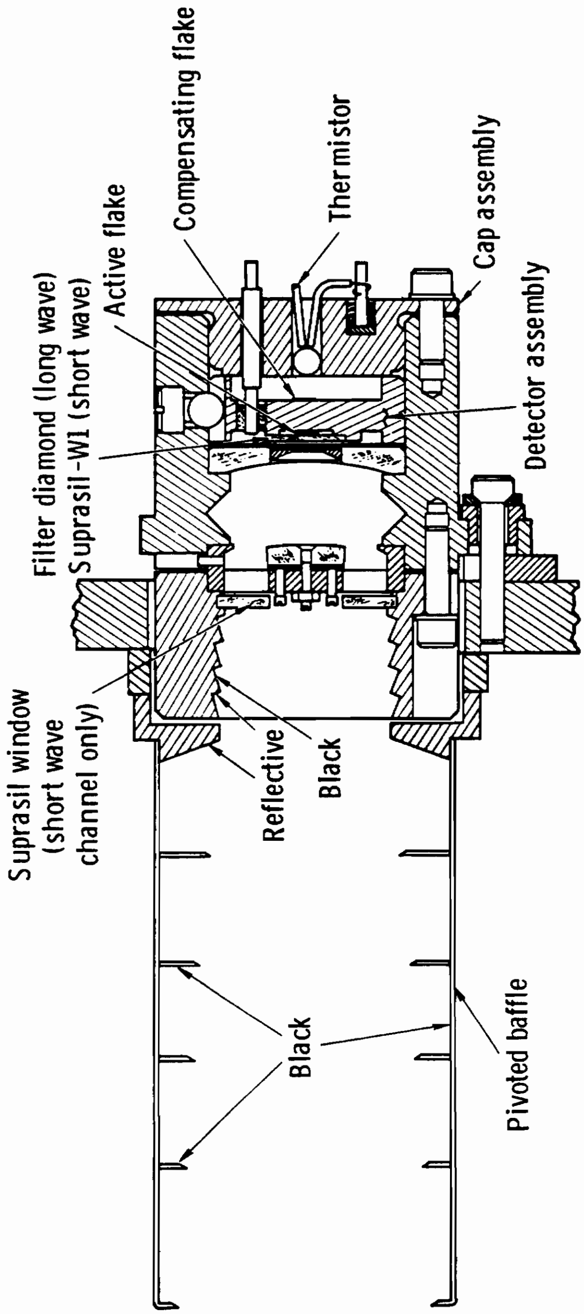
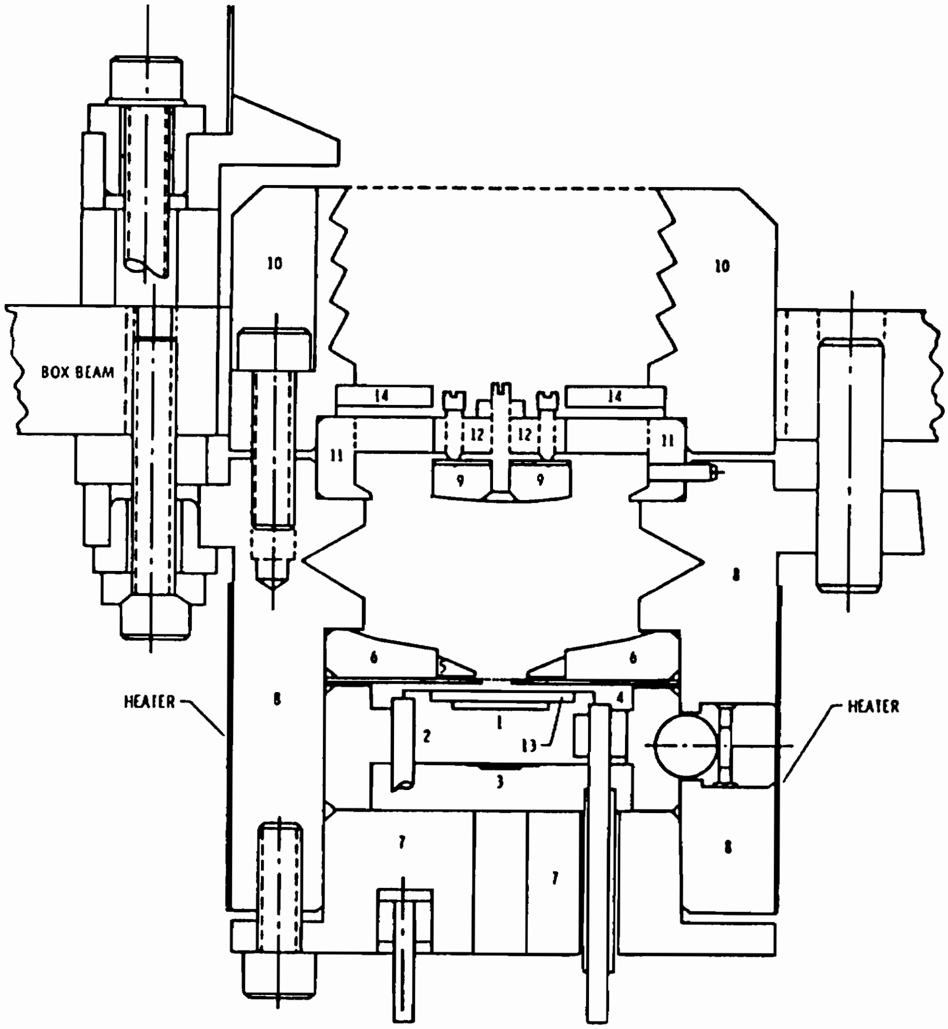


Fig. 5. The ERBE Scanning Radiometer Telescope (Courtesy of NASA Langley Research Center) [14].



1. Active Flake
2. Detector Header and Heat Sink
3. Compensating Flake
4. Field Stop
5. Primary Insert
6. Primary Mirror
7. Detector Header Cap
8. Detector Housing
9. Secondary Mirror
10. Reflector Cap
11. Secondary Mirror Mount
12. Secondary Mirror Mount
13. Suprasil or Diamond Filter
14. Suprasil Filter

Fig. 6. The ERBE Scanning Radiometer Optics Module [15].

The reflector cap, shown in Fig. 6, consists of intersecting conical sections. Those conical sections facing the aperture of the optics module are uncoated polished aluminum to better reflect incoming radiation not aligned with the optical axis. Those conical sections facing away from the aperture have been coated with Chemglaze™ to encourage absorption of radiation that is reflected from the surfaces of the optics module. The V-groove in the detector housing, also shown in Fig. 6, is an application of research performed by Sparrow and Jonsson [16]. Their research showed that a purely specular V-groove absorbs diffuse radiation better than a purely diffuse V-groove. Because the V-groove has also been coated with Chemglaze™ containing microballoons, most of the radiation entering the groove will be absorbed. However, if any radiation entering the groove is not absorbed initially, it will suffer several specular reflections, because of the glass balloons, before exiting the groove. Since the coating has a total absorptivity of about 0.9, essentially all radiation reflected within the V-groove will be absorbed on its surfaces.

Any radiation entering the optics module normal to its aperture is reflected by the mirrors through an opening in the center of the primary mirror, as shown in Fig. 4. The radiant energy gathered by the radiometer that is reflected through this opening is incident to a bolometer mounted behind the primary mirror. Historically, the first bolometer was manufactured by S. P. Langley in 1881. It was made of iron strips mounted on ebonite, which is a hard, black rubber. An article in **Nature** reports that Langley's invention was a "most interesting and most promising instrument of research" [17]. Although it was Langley who invented the bolometer, it was an electrically calibrated bolometer developed in 1893 by K. Angstrom which, with small enhancements, is still presently used in meteorological radiometry [3].

On Langley's bolometer, the iron strips were the active sensing elements. On the ERBE scanning radiometer, the active sensor of the bolometer is a thermistor flake. The thermistor, developed during the 1940's by Bell Laboratories, is characterized by its fast response time and compatible electrical properties. The thermistor flake, or active flake as shown in Fig. 6, used in the ERBE scanning radiometer is a semiconducting material made by sintering a mixture of manganese, cobalt, and nickel. The active flake has been coated with 3M Black Velvet™ paint so that more of the radiation incident to the surface of the flake is absorbed. The total absorptivity of this coating is approximately 0.9.

The active flake, which has a surface area of 4.0 mm^2 and is 10.0 μm thick, is mounted on a mylar™ thermal impedance. This assembly is then mounted on an aluminum substrate, as shown in Fig. 7, that is 2.8 mm thick. The aluminum substrate acts as a heat sink for the active flake. The active flake is made thin to respond quickly to varying radiation heat fluxes. When it is irradiated, its electrical resistivity changes according to its temperature change. To monitor its change in resistivity, the active flake is placed in a balanced bridge network with a compensating, or reference flake. The reference flake has properties and dimensions as close as possible to the properties of the active flake. Two fixed resistors complete the bridge circuit.

The compensating flake, which is shielded from any radiation entering the radiometer, as shown in Fig. 6, behaves as a thermometer for the detector header, also shown in Fig. 6. Its temperature is maintained at 311 K by a heater that also maintains the temperature of the active flake around 311 K. However, if the temperature of the detector header changes, it will not be detected by the active flake as a change in the incoming signal because the reference flake will respond accordingly. The bridge deflection, which is the difference between the resistance of the active and compensating

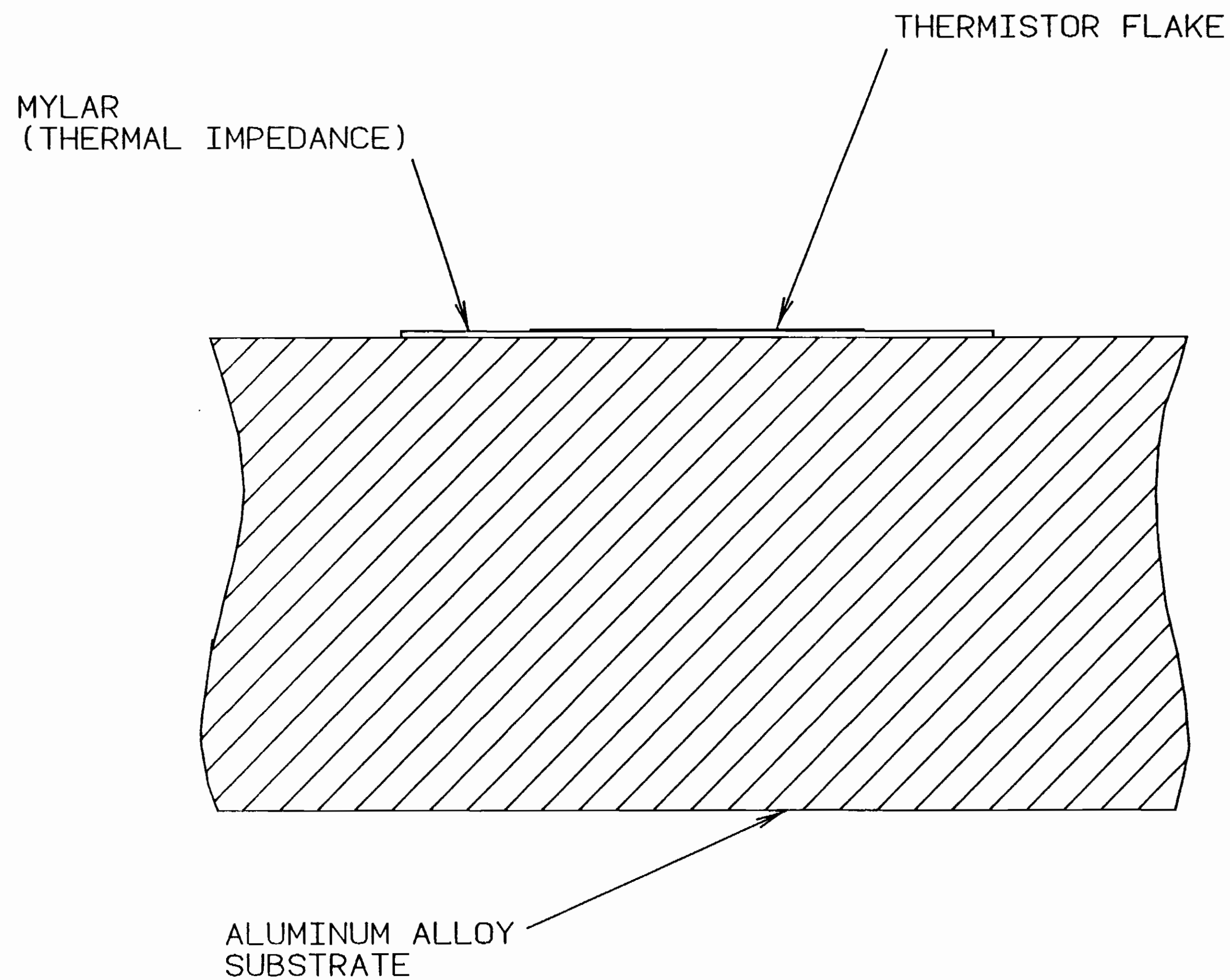


Fig. 7. The Active Flake Mounted on the Aluminum Substrate Heat Sink.

flakes, is then indicative only of the radiant energy sensed by the active flake. References 18, 19, and 20 contain further information on the theory and operation of thermistor bolometers.

Approximately 1.2 mm above the active flake, between the primary mirror and the detector header, is the field stop shown in Fig. 6. The field stop is a copper plate 1.4 mm thick that has electrodeposited nickel on the side that faces the active flake to minimize thermal emission from this component that may be sensed by the active flake. At the center of the field stop is a hexagonal aperture whose shape was derived by Huck to offset aliasing errors caused by scan and data sampling rates [21]. The dimensions of the aperture were chosen to limit the angle between the optical axis and the radiation entering the ERBE telescope that is incident to the active flake to ± 1.5 deg along the scan direction, and to ± 2.25 deg along the satellite ground track, as shown in Fig. 8. The field stop is described in this chapter not only because it is an important component in the optical design of the ERBE telescope, but also because Chapter 4 will reveal an interesting fact about the effect the field stop has on collimated radiation entering the aperture of the telescope.

For the interested reader, References 22 and 23 contain more information on radiometry and radiometer design. The next chapter focuses on the model that has been formulated to perform the research documented in this thesis.

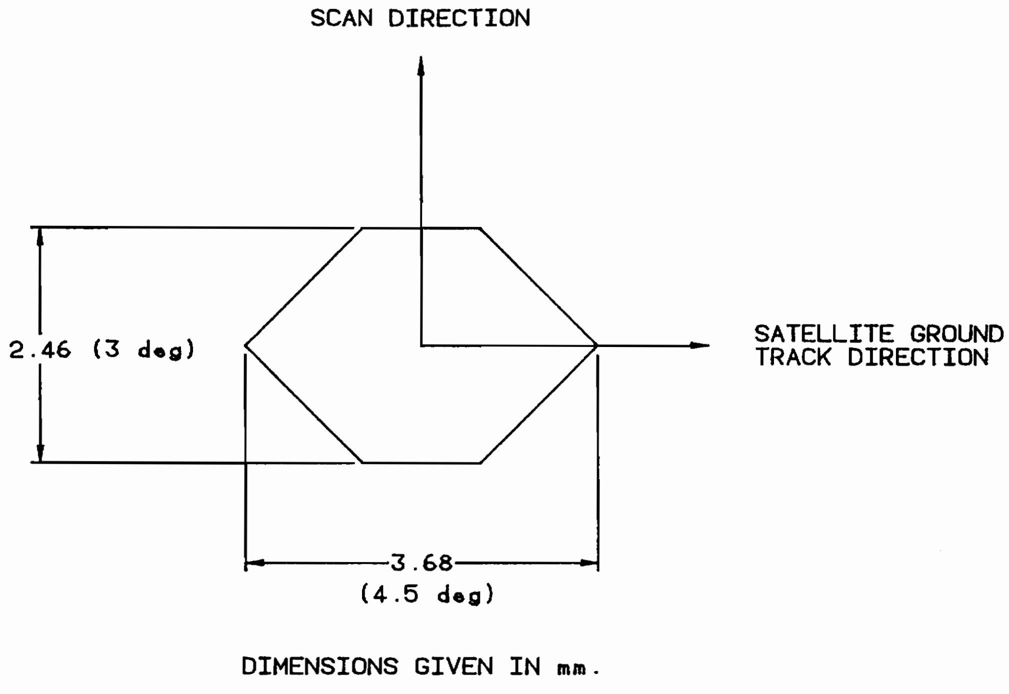


Fig. 8. The Field Stop Aperture.

3.0 Model Formulation

A Monte Carlo-based numerical model has been formulated to study the optical and radiative characteristics of the ERBE scanning radiometer. The word "optical" in this context refers to how the optics of the ERBE telescope transfer collimated or diffuse radiation incident to the instrument aperture to the active flake. The word "radiative" refers to the thermal emissions from different components of the telescope that reach the active flake. The model has been programmed in FORTRAN for portability since it will be used in a variety of computational environments. Figure 9 represents the ERBE scanning radiometer optics module as it is defined numerically using planar and quadric surfaces well-known in analytic geometry. This representation is used as the Monte Carlo domain. Many surfaces of the optics module could have been modeled more simply using valid engineering assumptions; however, the author strove for the highest practical resolution. The complexity of the enclosure made it difficult to describe mathematically, but the flexibility of the resulting model easily justifies the effort involved in its formulation. All dimensions and surface properties are user-defined in the program. A brief outline of the Monte Carlo method and how it applies to the radiative analysis is given first followed by a description of how it applies to the optical analysis.

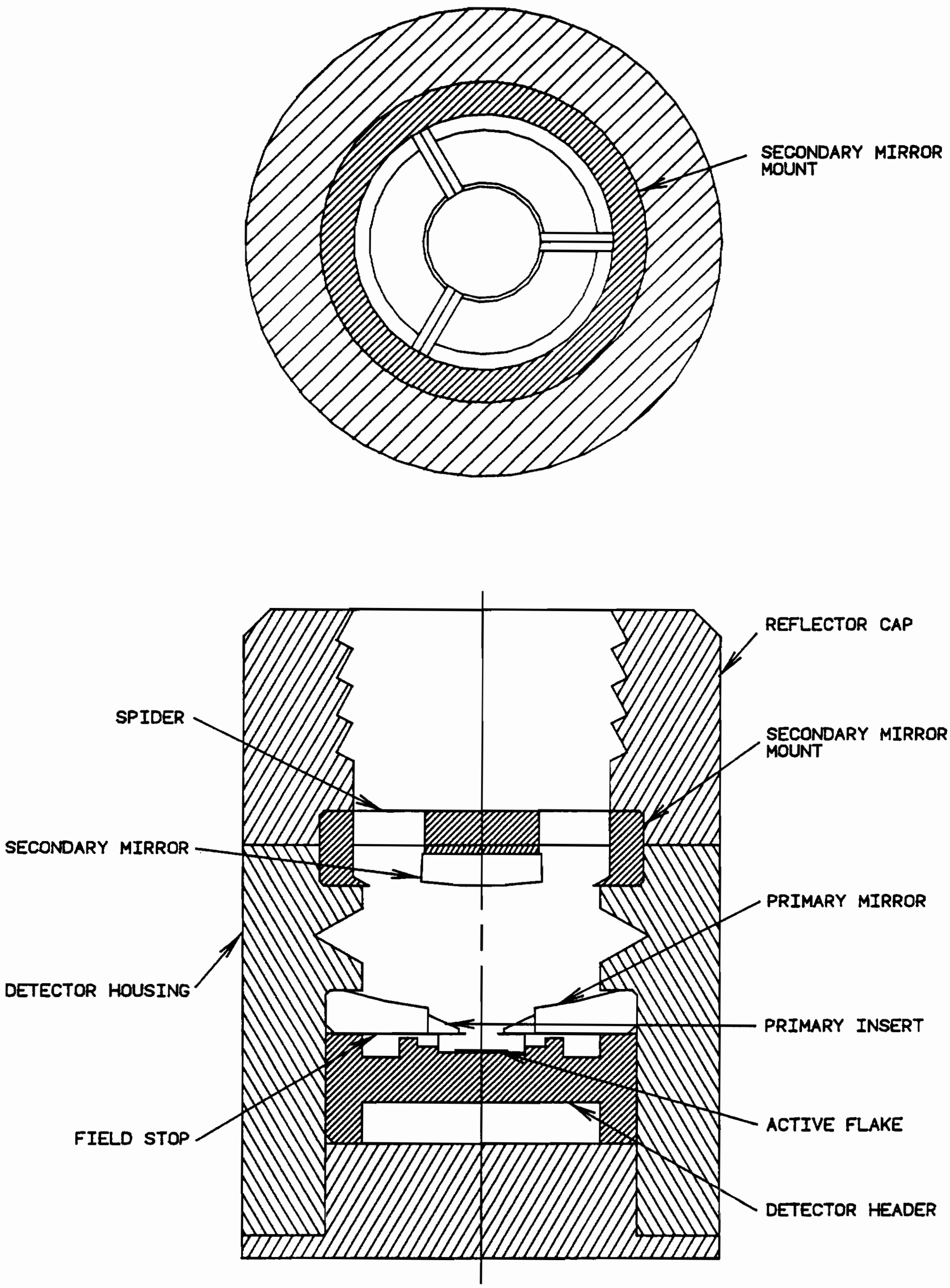


Fig. 9. Numerically Generated Representation of the ERBE Scanning Radiometer Optics Module.

3.1 The Radiative Model Formulation

3.1.1 The Distribution Factor

Determining radiative exchange between the surfaces of an axisymmetric enclosure has always been a formidable task for radiation heat transfer practitioners, especially when curved surfaces are involved in the calculation. Add the fact that no "real" surface is purely diffuse or purely specular and the problem is complicated still further. In his discussion of a paper by Sparrow *et al.* [24], Seban suggests that reflections from most real surfaces can be treated as the sum of a diffuse component and a specular component. This idea led to the formulation of several types of radiant interchange factors used to describe radiative exchange between diffuse-specular surfaces of an enclosure. The *exchange factor* described by Sarofim and Hottel [25] is perhaps the most popular of these. Other factors for radiative analysis and their application may be found in standard radiation heat transfer texts such as References 26 and 27.

For all but the most simple enclosures, obtaining the exchange factors between their surfaces, especially if the surfaces are curved, is extremely tedious. When varying surface properties are introduced, it is virtually impossible. However, Mahan and Eskin [28] introduce what they call the *distribution factor*. It can be calculated for any irregularly-shaped, asymmetric, diffuse-specular enclosure *with* varying surface properties. The distribution factor is similar to Gebhart's *absorption factor* [29] and Toor and Viskanta's *extended absorption factor* [30]; however, the former considers diffuse reflection only and the latter accounts for directional emission and bidirectional reflection. Clearly, the absorption factor is not good enough to provide the flexibility desired when modeling real surfaces because only diffuse reflections are considered, and practical applications for the extended absorption factor are rare because directional surface properties are often difficult to accurately obtain. Therefore, the distribution factor is used in the radiative analysis of the ERBE scanning radiometer.

The distribution factor, D_{ij}, is defined as the fraction of energy emitted diffusely from surface *i* that is absorbed by surface *j* by direct radiation and by all possible diffuse and specular reflections. It has the following useful properties [28]:

$$\sum_{j=1}^{N} D_{ij} = 1.0\,, \qquad i = 1, 2, \ldots, N\,, \tag{3.1}$$

$$\varepsilon_i A_i D_{ij} = \varepsilon_j A_j D_{ji}\,, \tag{3.2}$$

and

$$\sum_{i=1}^{N} \varepsilon_i A_i D_{ij} = \varepsilon_j A_j\,, \qquad j = 1, 2, \ldots, N\,, \tag{3.3}$$

where N is the number of surfaces, or elements, into which the enclosure is divided, ε is the total hemispherical emissivity of a given surface, and A is its area. Equation (3.1) is the summation rule, or conservation of energy, which states that all the energy emitted from surface *i* must be absorbed somewhere. Equation (3.2) is the reciprocity relation and Eq. (3.3) is a consequence of Eqs. (3.1) and (3.2) obtained by summing both sides of Eq. (3.2) over *i* for all N surfaces of the enclosure and then applying Eq. (3.1) to the result. These relations may be used to check the consistency of the values obtained for the distribution factors.

The difficulty encountered when calculating the distribution factors is directly related to the complexity of the geometry of the enclosure, and to any variation in the surface properties of the enclosure. For enclosures as complex as the optics module of the ERBE scanning radiometer, the only practicable method that may be used to calculate the distribution factors is the Monte Carlo method.

3.1.2 The Monte Carlo Method

The Monte Carlo method is a statistical, probabilistic, numerical scheme used to simulate random processes that have a certain expected outcome. The Monte Carlo method has been applied to radiation heat transfer problems to determine the radiative exchange between surfaces, as described by Corlett [31] and Howell [32]. The complex integration inherent in thermal radiation problems when determining the radiation distribution between the surfaces of geometrically-complex enclosures can only be resolved numerically. The Monte Carlo method makes this possible.

Applying the Monte Carlo method to radiation heat transfer problems to determine the distribution factors for an enclosure hinges on two assumptions:

1. The total energy of a radiant heat source, or the total energy emitted from a surface, can be divided into an arbitrary number of discrete energy bundles each having an equal fraction of the total energy of the source.

2. The dispersion of those energy bundles within the enclosure can be determined by interpreting the optical properties of that enclosure in terms of the laws of probability.

When these assumptions can be justified, the location of emission and the fate of a given energy bundle is determined as follows:

Step 1. Selecting the location of emission:

The location of emission (ϕ,z) on a conic surface is randomly selected by

$$z = z_o + R_z \Delta z \tag{3.4}$$

and

$$\phi = 2\pi R_\phi , \tag{3.5}$$

where z_o represents a reference z location. For surfaces modeled by a disk or annulus, the location of emission (r,ϕ) is randomly selected using Eq. (3.5) and

$$r = r_o + \sqrt{R_r}\, \Delta r , \tag{3.6}$$

where r_o denotes a reference radius. The x- and y-coordinates of the emission point are easily obtained using the circumferential angle, ϕ. The symbols R_z , R_ϕ , and R_r represent pseudo-random numbers uniformly distributed between 0 and 1, inclusive. The author notes here that the accuracy of the Monte Carlo method often depends on the randomness of the pseudo-random number generator being used. Reference 33 documents the ISML subroutine RNUN used in this application. It has proved to be an excellent pseudo-random number generator in previous applications of the Monte Carlo method implemented on the IBM mainframe computers at Virginia Polytechnic Institute and State University [34,35].

Step 2. Selecting a direction for diffuse emission or reflection:

Reference 26 shows how the direction of the diffuse emission or reflection can be obtained by generating two pseudo-random numbers R_θ and R_ϕ to select a cone angle by

$$\theta = \sin^{-1}\sqrt{R_\theta} \tag{3.7}$$

and a circumferential angle by Eq. (3.5) with respect to the local spherical coordinate system at the location of emission found in **Step 1**. Once obtained, the direction of the diffuse emission must be converted to the global Cartesian coordinate system. This involves calculating unit normal and unit tangent vectors to the surface at the location of emission. See subprogram VECTOR in Appendix A. Having the direction of the diffuse emission or reflection, each surface of the enclosure is then tested to determine upon which surface the energy bundle impinges.

Alternative approach to Step 2:

An alternative approach to **Step 2** has been devised for determining the direction of a diffuse emission. At the location of emission, point labeled $\mathbf{P_1}$ in Fig. 10, construct a unit sphere tangent to the given surface, as shown in the same figure. The unit normal is calculated for that surface at the location of emission and the center of the unit sphere lies at the end of the unit normal. The direction of the diffuse emission or reflection can be obtained by randomly selecting a point along the diameter of the sphere that is perpendicular to the surface by

$$z_s = 2R_z - 1\,, \tag{3.8}$$

obtaining the cone angle by

$$\theta = \cos^{-1}(z_s)\,, \tag{3.9}$$

and then finding the circumferential angle by Eq. (3.5). Finding the cone angle as defined by Eq. (3.9) eliminates having to calculate the unit tangent vectors to a surface, a calculation that is quite complicated for conic sections. Once obtained, θ and ϕ can be used to calculate the x-, y-, and z-coordinates (point P_2 in Fig. 10) of the diffuse emission in the local spherical coordinate system. Converting these coordinates to the global Cartesian coordinate system is the vector addition of the unit normal and the vector obtained in the local spherical coordinate system. The direction of the diffuse emission, $\overline{\mathbf{P_1P_2}}$, is the difference between the position vector, $\mathbf{r}_2$, to the point on the sphere and the position vector, $\mathbf{r}_1$, to the emission point, as shown in Fig. 10.

The proof that the cone angle found using a unit sphere tangent to the location of emission is the same as the one calculated using Eq. (3.7) is given below:

Proof:

From Eqs. (3.8) and (3.9),

$$R_z = \frac{\cos\theta + 1}{2}\,. \tag{3.10}$$

Using half-angle and trigonometric formulae,

$$R_z = \cos^2\frac{\theta}{2} = 1 - \sin^2\frac{\theta}{2}\,; \tag{3.11}$$

therefore,

$$\sin^2\frac{\theta}{2} = 1 - R_z\,. \tag{3.12}$$

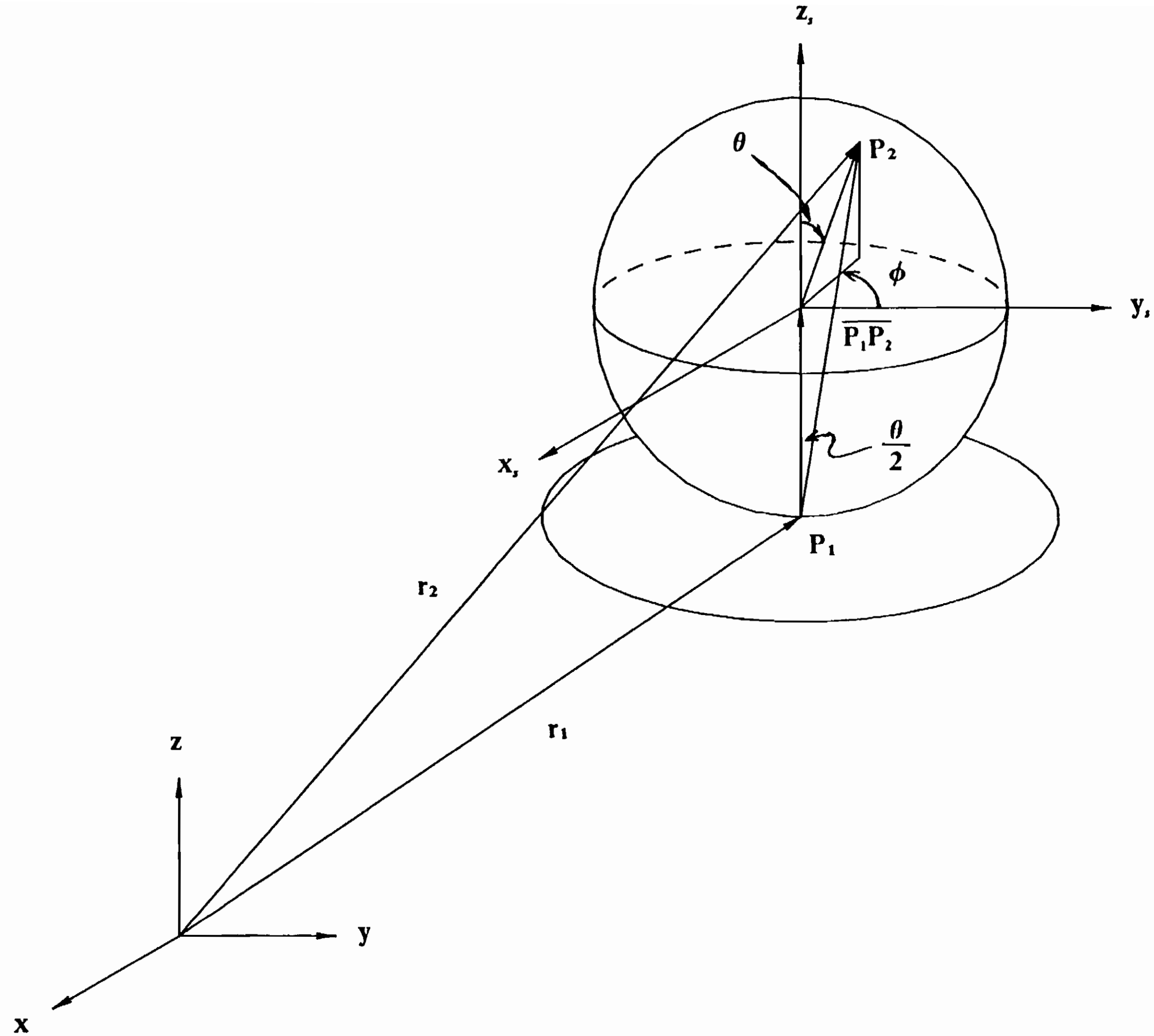


Fig. 10. Choosing the Direction of a Diffuse Emission or Reflection Using the Unit Tangent Sphere Method.

Since R_z is a random number between 0 and 1, 1 - R_z is also a random number between 0 and 1.

Therefore, substituting R_z' for $1 - R_z$,

$$\sin^2 \frac{\theta}{2} = R_z' , \tag{3.13a}$$

or

$$\sin \frac{\theta}{2} = \sqrt{R_z'} . \tag{3.13b}$$

Since θ assumes values between 0 deg and 180 deg on the unit sphere, $\frac{\theta}{2}$ in Eq. (3.13b) signifies that the cone angle calculated using the unit sphere maps to the hemispherical space above the point of emission, as shown in Fig. 11. This is exactly what Eq. (3.7) does.

Step 3. Determining whether an incident energy bundle is absorbed or reflected:

Once the location of incidence is found, a pseudo-random number, R_a , is generated to determine whether the energy bundle is absorbed or reflected by the surface it strikes. If $R_a \leq \alpha$, where α is the total absorptivity of the surface, the energy bundle is absorbed and another energy bundle is emitted as described in **Step 1** and **Step 2**. If $R_a > \alpha$, the energy bundle is reflected and execution continues with **Step 4**.

Step 4. Determining whether the reflection is specular or diffuse:

A random number, R_r, is generated and compared to the specular-to-total reflectivity ratio ($R_{spec} = \frac{\rho^s}{\rho^s + \rho^d}$) of the surface. If $R_r \leq R_{spec}$, then the energy bundle is specularly reflected. Its direction is obtained by using the laws of specular reflection

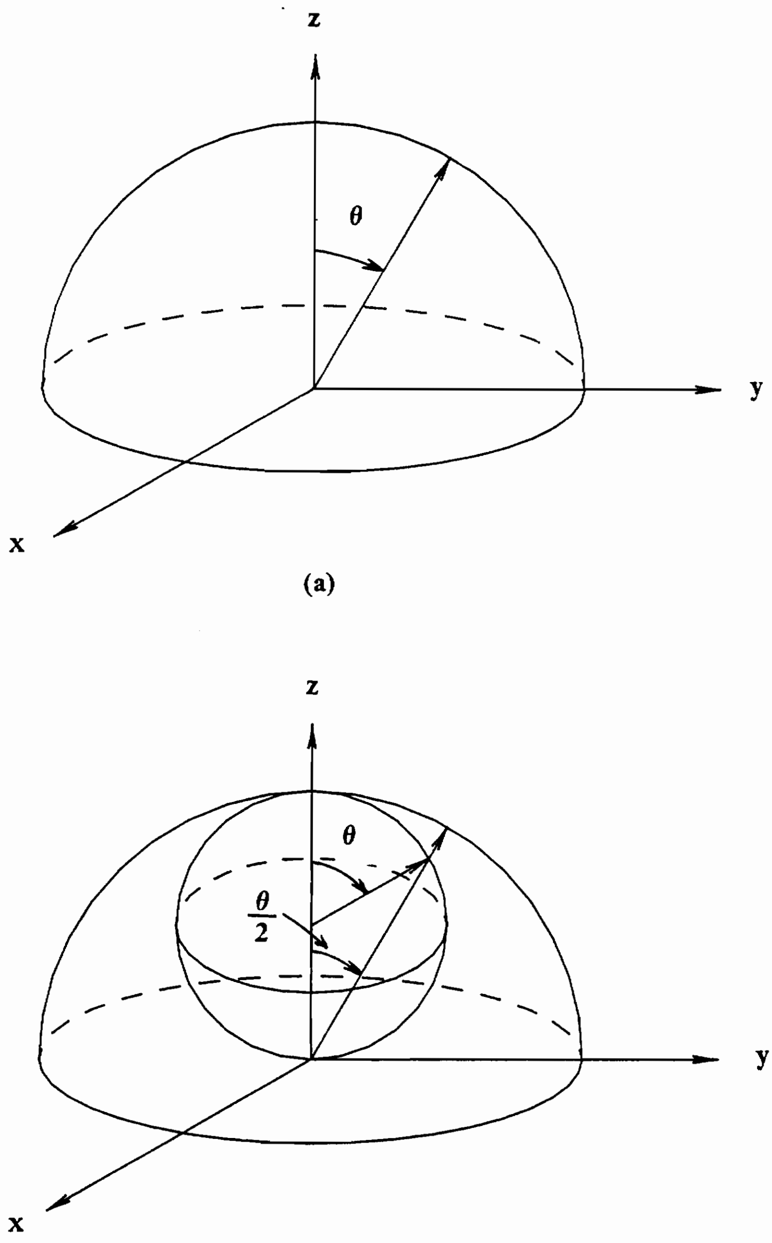


Fig. 11. Choosing the Cone Angle for a Diffuse Emission or Reflection:
(a) using the $\sin^{-1}\sqrt{R_\theta}$ method and (b) using the unit tangent sphere method.

which require that the reflection angle be equal to the incident angle, and that both angles be in the same plane. If $R_r > R_{spec}$, then a diffuse reflection occurs and the direction of the energy bundle is obtained as outlined in **Step 2**.

Once all energy bundles emitted from a given surface, *i*, are accounted for, the distribution factor can be obtained from that surface to each surface, *j*, of the enclosure by

$$D_{ij} = \frac{n_{ij}}{N_i} , \tag{3.14}$$

where n_{ij} is the number of energy bundles emitted from surface *i* that are *absorbed* by surface *j*, and N_i is the total number of energy bundles emitted from surface *i*.

Once the distribution factors have been obtained, they are easily implemented to calculate the radiative heat transfer, Q_{ij}, to surface *j* due to thermal emission from surface *i* by

$$Q_{ij} = \sigma \varepsilon_i A_i T_i^4 D_{ij} . \tag{3.15}$$

In Eq. (3.15), ε_i, A_i and T_i represent the total hemispherical emissivity, area and temperature of surface *i*, respectively, and σ is the Stefan-Boltzmann constant defined in the Nomenclature. Equation (3.15) is used in the radiative analysis to compute the amount of radiation emitted from the surfaces of the ERBE telescope that is absorbed by the active flake.

3.1.3 Application to the ERBE Scanning Radiometer

The model of the optics module of the ERBE scanning radiometer, shown in Fig. 9, has been divided into 37 elements. Each element represents a disk, annulus, or conic section that has been used to construct a component of the model. These elements are then used to estimate the distribution factors for the optics module. To avoid statistically overweighing any surface by emitting the same number of energy bundles from the larger elements as from the smaller elements, the number of energy bundles emitted from any given element has been weighted by the surface area of that element. That is, the number of energy bundles emitted from the smallest element is multiplied by the surface area of any given element and then divided by the surface area of the smallest element in the module. To illustrate, suppose 100 energy bundles are emitted from the element within the module that has the smallest surface area. Thermal emission from an element with ten times the surface area of the element with the smallest surface area in the module would be grossly undersampled by 100 energy bundles. Therefore, to ensure proper sampling of the larger element, the number of energy bundles emitted from that element would be ten times the number of them emitted from the smaller element, or 1000.

For the calculations whose results are presented in this thesis, 5000 energy bundles have been emitted from the smallest element in the optics module to determine the distribution factors for the enclosure. The number of energy bundles emitted from the other 36 elements has been weighted by the appropriate surface area ratio. Computing the distribution factors for the optics module requires two hours of CPU time on an IBM 3090 mainframe computer. The number of energy bundles emitted from the smallest area within the module, 5000, provides a good tradeoff between Monte Carlo convergence and computing time: the observed improvement in the value of the distribution factors when doubling the number of energy bundles emitted does not justify doubling the computer time required to compute those values.

Equation (3.3) is used to check the consistency of the values obtained for the distribution factors. Then, the percent error in the distribution factors estimated for the j^{th} element of the model is calculated by manipulating Eq. (3.3) to obtain

$$\text{percent error} = \left[1 - \frac{\sum_{i=1}^{N} \varepsilon_i A_i D_{ij}}{\varepsilon_j A_j} \right] \times 100\,(\%)\,. \qquad (3.16)$$

The largest error estimate for any element is 3.6 percent. This error reflects the statistical nature of the Monte Carlo method. The model program listings that implement the Monte Carlo method and their flowchart are given in Appendix A.

3.2 The Optical Model Formulation

The optical model formulation is similar to the radiative model fomulation with the primary difference being that the distribution factors are not calculated. The Monte Carlo method is often referred to as a ray tracing method in this type of application. It has proved to be an excellent way of determining how rays of light, or radiation in this case, are transferred through an optical system [36]. It is used here to describe how radiation is transferred through the Cassegrain optical system of the ERBE telescope to the active flake. Energy bundles emitted from an imaginary surface at the aperture of the ERBE scanning radiometer optics module are traced as outlined in Section 3.1.2.

3.3 Wavelength-Dependent Model Formulation

Unlike the ERBE nonscanning active cavity radiometer, the ERBE scanning radiometer is not a spectrally flat instrument [35]. The primary and secondary mirror spectral reflectivity, as shown in Fig. 12, and the shortwave and longwave filter spectral transmissivities, as shown in Figs. 13 and 14, influence the spectral response of the instrument. These spectral characteristics must be accounted for in order to accurately model the ERBE scanning radiometer or to evaluate calibration procedures. The model documented in this thesis has both gray and spectral analysis capabilities, but the gray assumption has been and should only be used to characterize the optical performance of the ERBE scanning radiometer. All radiative analyses have been performed using the spectral model.

The wavelength-dependent formulation of the ERBE scanning radiometer model uses the method outlined by Siegel and Howell [26]. A gray radiant source with an assumed Planckian distribution is introduced at the aperture of the instrument. When an emission point is located at the aperture of the instrument as described in **Step 1** of Section 3.1.2, a wavelength is then arbitrarily assigned to that energy bundle according to the temperature of the source of radiation. A value for the blackbody fraction, $F_{0-\lambda T}$ as defined in Table A-5 of Reference 26, is randomly selected using

$$F_{0-\lambda T} = R_{\lambda}, \tag{3.17}$$

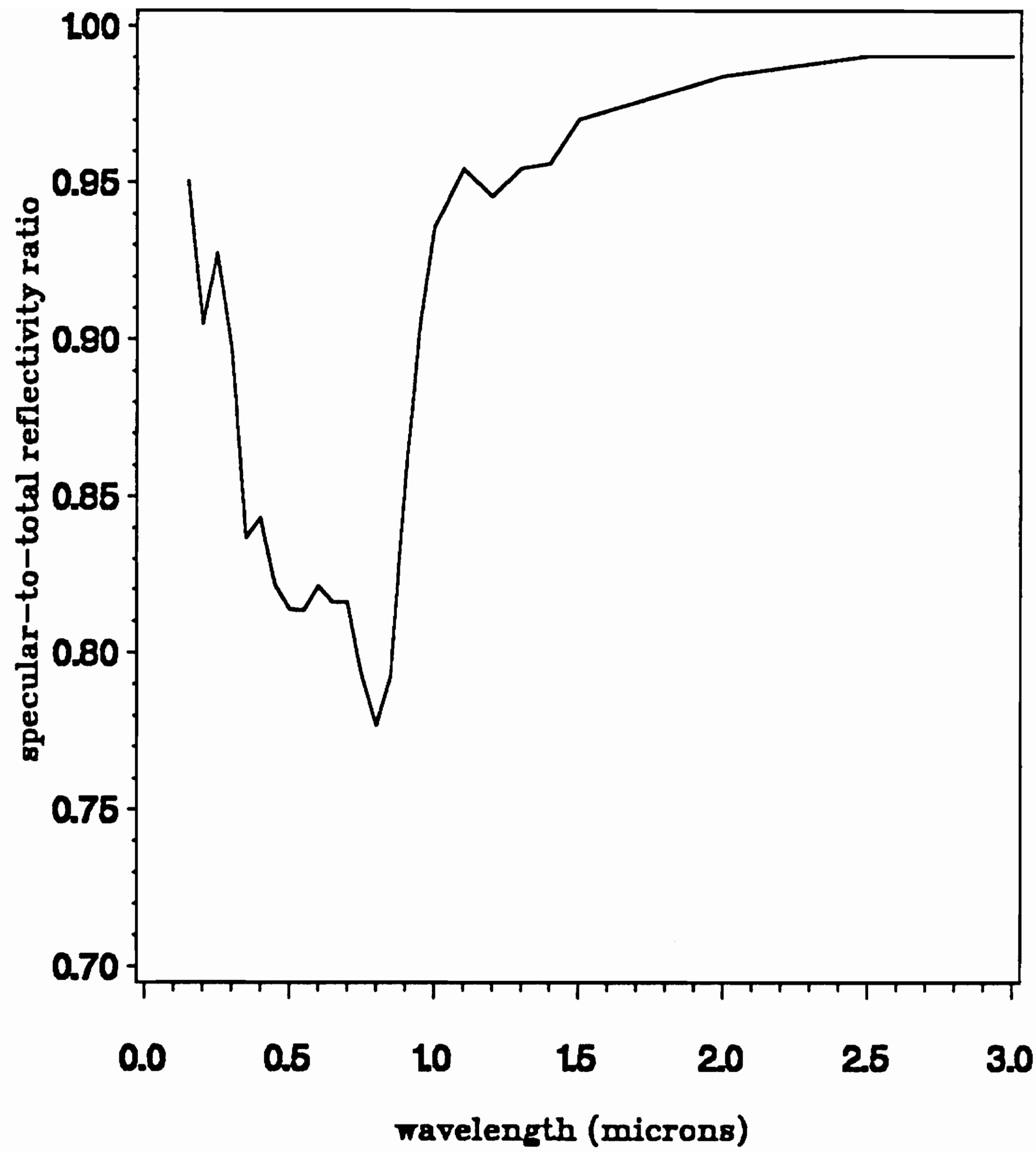


Fig. 12. Approximate Spectral Specular-to-Total Reflectivity Ratio of the Primary and Secondary Mirrors (Courtesy of NASA Langley Research Center).

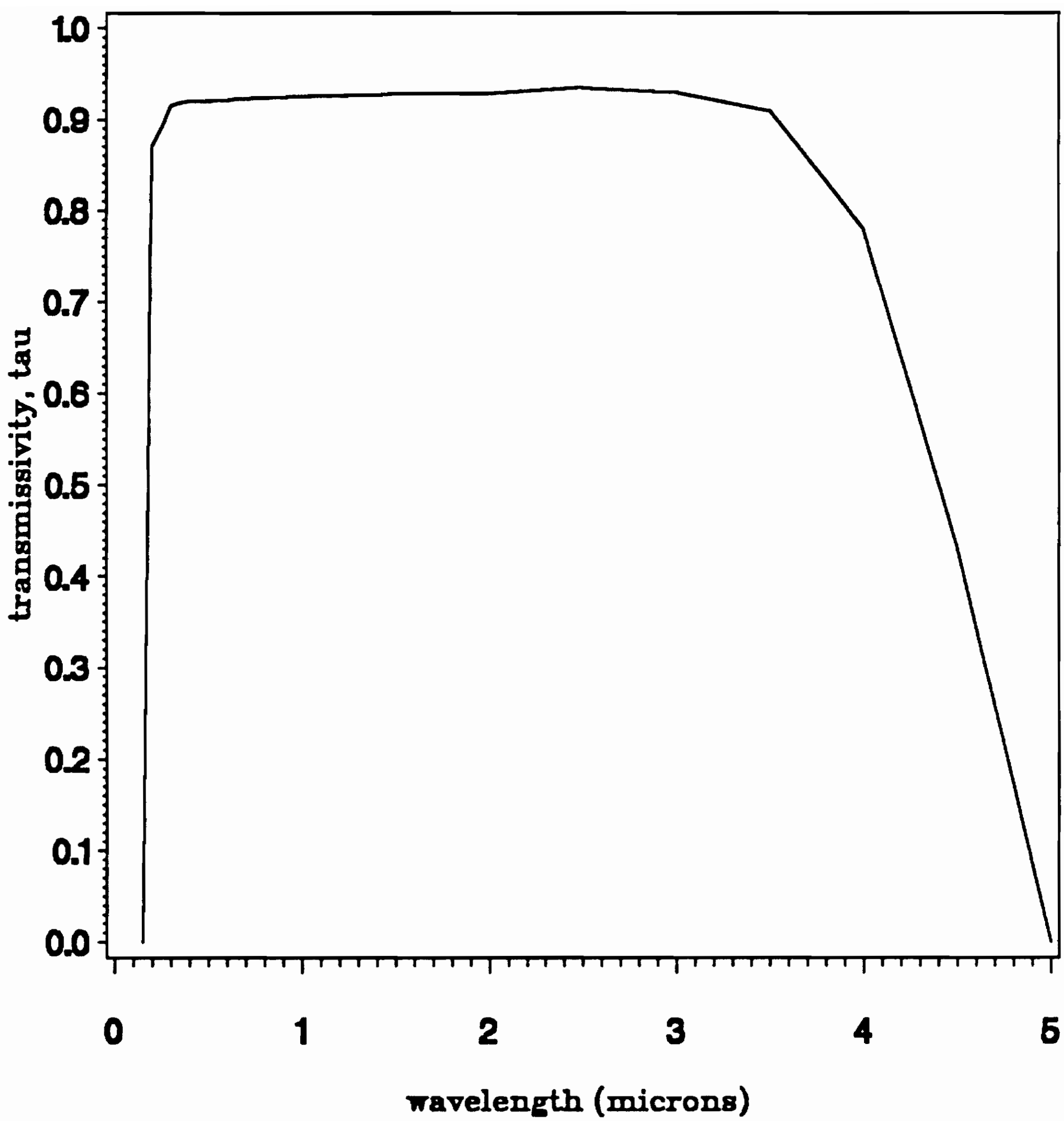


Fig. 13. Approximate Spectral Transmissivity of the Suprasil-W Filter (Courtesy of NASA Langley Research Center).

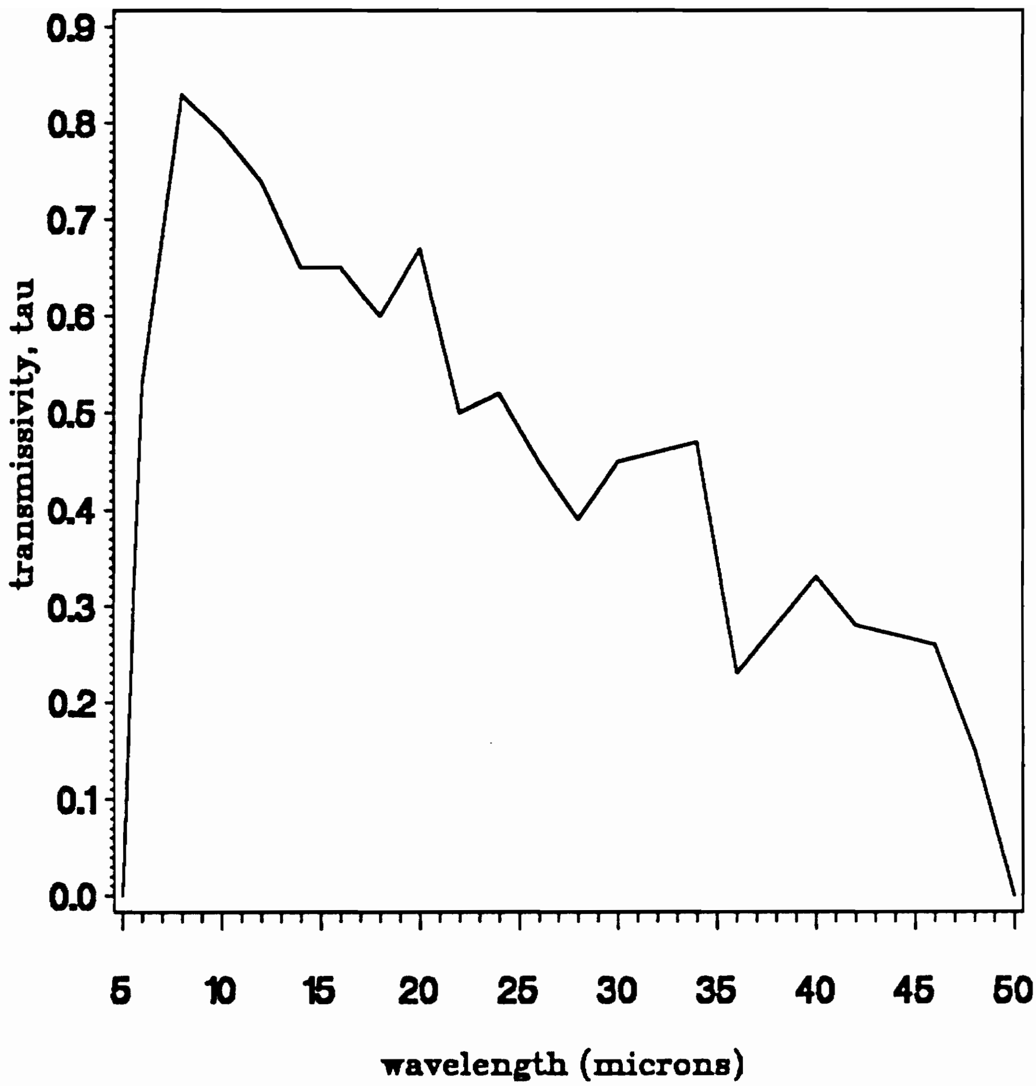


Fig. 14. Approximate Spectral Transmissivity of the Diamond Filter (Courtesy of NASA Langley Research Center).

where R_λ is a pseudo-random number uniformly distributed between 0 and 1, inclusive. Then, the appropriate value of the wavelength-temperature product, λT, corresponding to this blackbody fraction can be found iteratively by applying the Newton-Raphson method to

$$\int_0^{\lambda T} \frac{2\pi C_1 e^{-C_2/\lambda T}}{(\lambda T)^5 (1 - e^{-C_2/\lambda T})} d(\lambda T) - F_{0-\lambda T} = 0 , \tag{3.18}$$

where λT is the root of that equation. The integral in Eq. (3.18) is evaluated using the Romberg integration technique (see subprogram ROMB in Appendix A). The values of the constants C_1 and C_2 are given in the Nomenclature. Once λT is found, the wavelength, λ, can be calculated by dividing λT by the temperature, T, of the source.

Since the transmissivity, τ, of the filters and the reflectivity, ρ, of the mirrors are wavelength-dependent, these values must be determined from the characteristics shown in Figs. 12, 13, and 14. All other surfaces of the optics module are assumed to be gray. The characteristics in Figs. 12, 13, and 14 have been tabulated to permit comparison of the wavelength of the energy bundle emitted with the tabulated wavelengths that correspond to the value of reflectivity of the mirrors and to the value of transmissivity of the filters at that wavelength. To illustrate, consider an energy bundle at a given wavelength which strikes the filter in the longwave channel of the ERBE scanning radiometer. The wavelength of the energy bundle is successively compared to the tabulated wavelengths until it matches one of the tabulated values or falls between two of them. If the wavelength of the energy bundle falls between two of the tabulated values, the value of spectral transmissivity, τ_λ, is linearly interpolated. A random number is then generated and compared with this value of spectral transmissivity. If the random number is less than this value, the energy bundle is transmitted. Otherwise, the energy bundle is reflected. See subprogram FILTER in Appendix A.

The method outlined above is also used to obtain the value of the specular-to-total reflectivity ratio, R_{spec}, of the mirrors when an energy bundle at a given wavelength is incident upon their surfaces. Once this value is obtained, a random number is generated and compared to it. If the random number is less than this value, the energy bundle is specularly reflected. Otherwise, the energy bundle is reflected diffusely. See subprograms XMIT and RHOSPEC in Appendix A.

4.0 Results and Discussion

4.1 Optical Model Accuracy

The accuracy of the optical model was partially accessed based on the known optical properties of a Cassegrain telescope. after manipulating some equations from References 37 and 38, the optical focal point for the f/1.84 optical system of the ERBE telescope, which has an equivalent focal length of 23.03 mm, may be expressed as

$$FP = \frac{FL_{sec}}{m}(m+1)(m-1) - FL_{pri}, \tag{4.1}$$

where

FP is the focal point.

FL_{sec} is the focal length of the secondary mirror.

FL_{pri} is the focal length of the primary mirror.

and *m* is the magnification where

$$m = \frac{\text{equivalent focal length}}{FL_{pri}} . \tag{4.2}$$

The focal point, which lies 0.65 mm behind the field stop, is calculated as a reference location to test whether an image projected onto the active flake is inverted as it passes through the focal point. Uniformly collimated radiation is introduced normal to the aperture of the optics module. The collimated radiation behaves like sunlight and casts the shadow of the secondary mirror mount onto the active flake. The surfaces of the optics module are assumed to be perfect absorbers in order to eliminate rays that may follow nonoptical paths to the active flake. This ensures a sharper image of the secondary mirror mount on the flake. Unless otherwise mentioned, whenever the surfaces of the optics module are referred to as "perfect absorbers" or as "perfectly absorbing", this means that all surfaces of the optics module are perfectly absorbing ($\alpha = 1$) except the mirrors, which are assumed to be perfectly specular reflectors ($R_{spec} = 1$). The active flake is then hypothetically placed directly behind the field stop, which is 0.65 mm in front of the optical focal point. Figure 15 shows the shadow of the three-spoked secondary mirror mount that has been cast onto the active flake. If the active flake is placed behind the optical focal point then the Cassegrain telescope, behaving exactly like a convex lens, inverts the image as it passes through the focal point. Accordingly, Fig. 16 shows an inverted image of the secondary mirror mount that has been cast onto the active flake. Here the active flake is approximately 2.0 mm behind the field stop. Image symmetry and inversion prove that the optics of the system was modeled correctly.

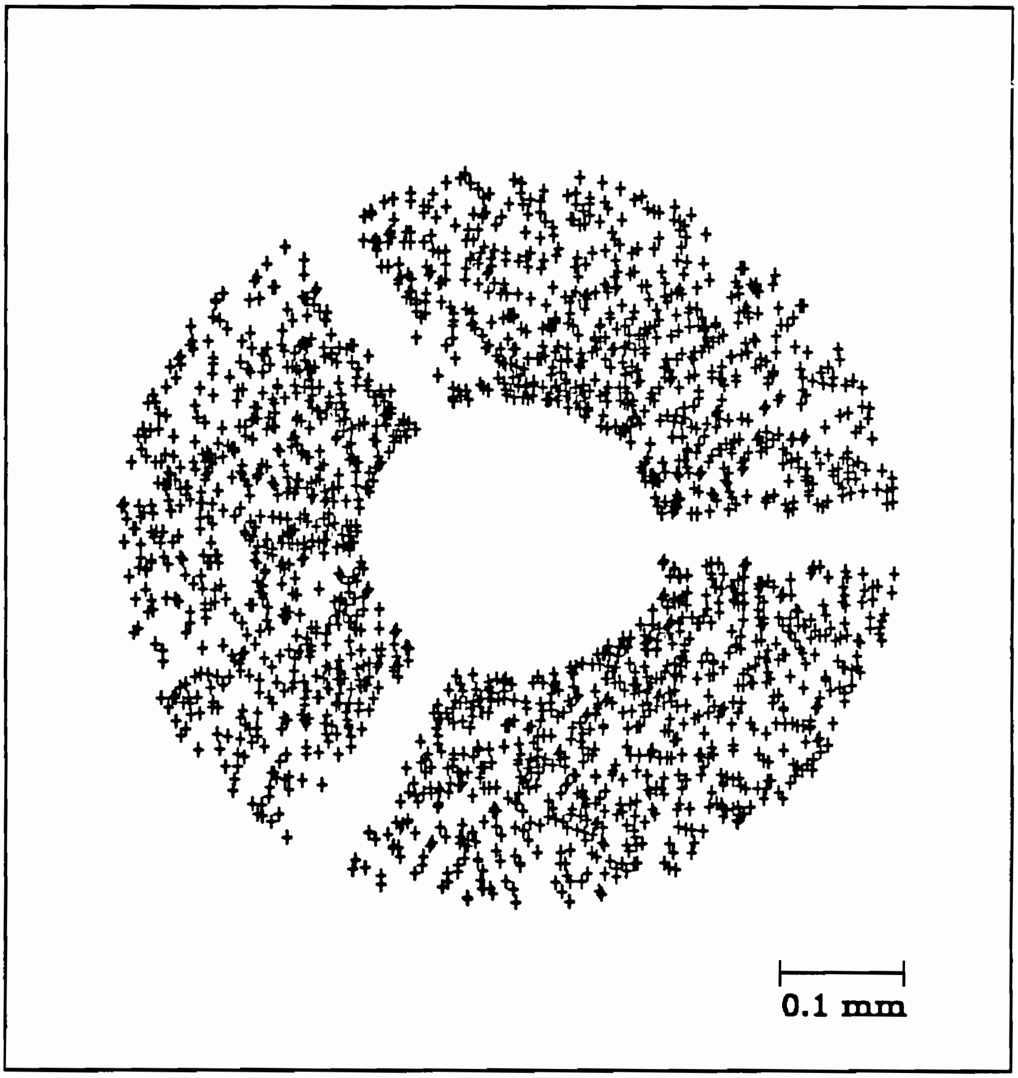


Fig. 15. Image of Secondary Mirror Mount Projected on the Active Flake When Placed 0.65 mm in Front of the Optical Focal Point.

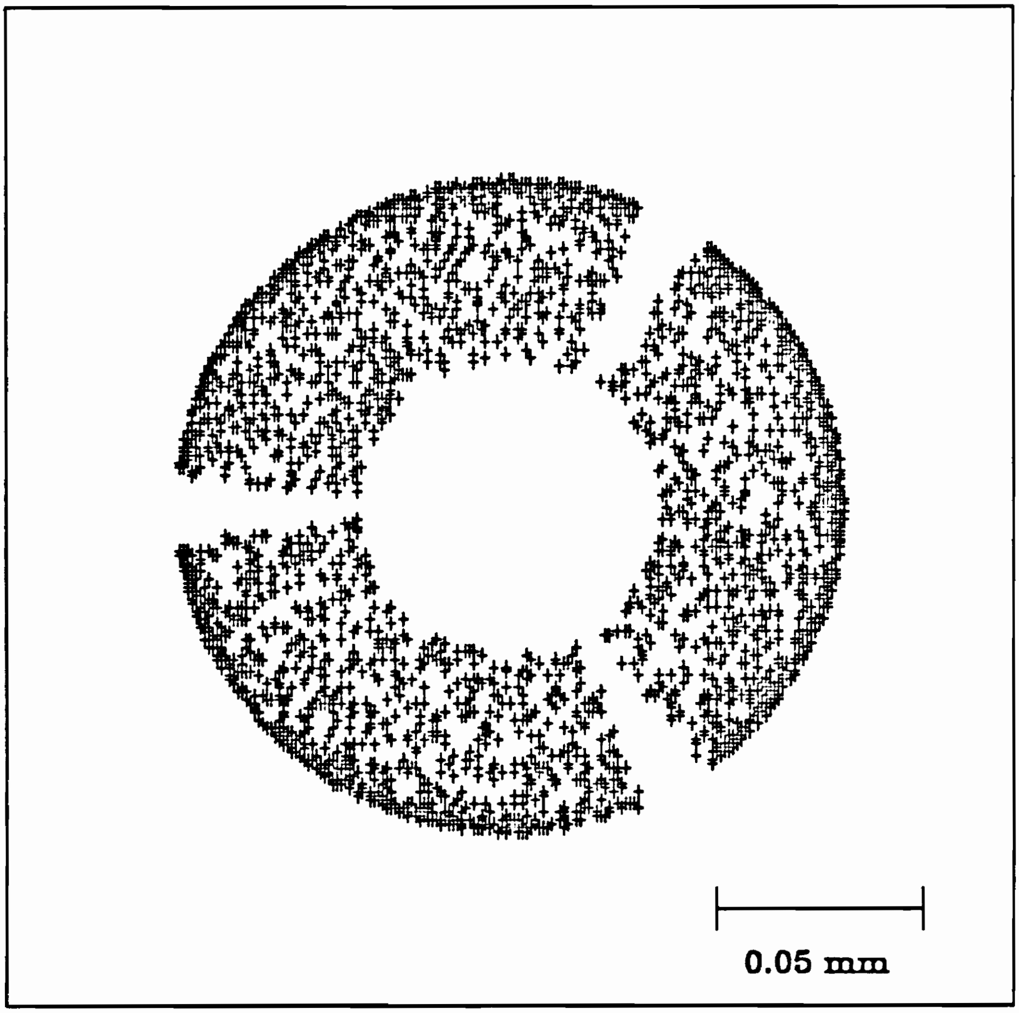


Fig. 16. Image of Secondary Mirror Mount Projected on the Active Flake When Placed Approximately 1.35 mm Behind the Optical Focal Point.

4.2 The Optical Analysis

4.2.1 The Optical Analysis Using Gray Characterizations

A telescope collects radiation, or light, from a point source (collimated radiation), such as a distant star, and focuses that energy, usually for observation. If a diffuse source is observed, then the telescope should be a good collimator. The ERBE telescope has been designed to be a good collimator. Some of the thermal control coatings used on the ERBE scanning radiometer and their approximate total absorptivities are given in Table 1. Unless otherwise mentioned, all results presented in this chapter have been obtained using these values.

Different numbers of energy bundles have been diffusely emitted from an imaginary surface stretched across the aperture of the optics module (open end shown in Fig. 9) to provide a measure of just how well the ERBE telescope collimates radiation. The cone angle of the diffusely emitted radiation, calculated using Eq. (3.9), has been limited to 14 deg. This limit is determined by computing the angle between the optical axis and a line drawn from the center of the aperture of the optics module to the outer edge of the primary mirror. This is equivalent to allowing diffuse radiation to enter a cylindrical tube that is long enough to allow only that radiation within a cone angle of 14 deg to pass through it. This tube is then brought directly in contact with the aperture of the optics module. The cone angle for all diffuse sources used in the analyses presented in this chapter has been limited in this manner to obtain a better sampling of the number of energy bundles that strike the active flake. Table 2 shows the fraction of diffuse-gray energy entering the aperture of the optics module that strikes the active flake.

The values in the second column of Table 2 were estimated using the Monte Carlo method for the values of total absorptivity presented in Table 1. The values in the third column were estimated assuming that the surfaces of the optics module are perfectly absorbing. The difference between the second and third columns is used to assess the effect the optics module has on radiation that may reach the active flake along nonoptical paths.

Consider the last row of Table 2. The second column shows that only 2.54 percent of the 100,000 diffusely emitted energy bundles reach the active flake. Clearly, the ERBE telescope is an excellent collimator. The third column shows that the number of energy bundles that arrive at the flake, assuming that the surfaces of the optics module are perfect absorbers, drops to 2.50 percent. That means that only 0.04 percent of the energy that enters the "real" diffuse-specular optics module arrives at the active flake along nonoptical paths, thus providing a measure of how well the optics module masks the active flake from randomly reflected radiation.

Suppose the radiation from a given pixel is diffuse, which is approximately true for most Earth scenes, especially in the longwave band. Figure 17 then shows the radiation distribution on the active flake while viewing a diffuse source. Notice that the outline of the distribution resembles the aperture of the field stop shown in Fig. 8. Recall that the field stop is approximately 1.2 mm above the active flake. Some radiation incident to the field stop aperture at angles to the optical axis will actually strike the active flake outside of the projection of the field stop aperture on the active flake. Thus, the outline of the radiation distribution on the active flake is not a one-to-one projection of the field stop aperture. Regardless, as described in Chapter 2, the field stop *is* successfully limiting the field-of-view of the instrument to 3 deg $\times$ 4.5 deg. However, its role is severely diminished when collimated radiation enters the aperture of the instrument. In fact, the field-of-view is then mostly limited by the optical characteristics of the Cassegrain telescope.

The surfaces of the optics module are assumed to be perfectly absorbing to provide an ideal Cassegrain telescope. Figure 18 shows the radiation distribution on the active flake for collimated radiation incident normal to the aperture of the optics module. The image is clearly symmetric and the spokes of the secondary mirror mount are clearly visible. Figures 19 through 23 show how the image quickly becomes distorted as the angle between the incident radiation and the optical axis increases. This is a trait of Cassegrain telescopes. The angle of collimated radiation incident to the aperture of the instrument used to create the images in these figures is measured in a direction along the satellite ground track. Figure 8 shows how the satellite ground track is defined with respect to the field stop aperture.

In Fig. 19, the radiation enters the telescope at an angle of one degree with respect to the optical axis. Some asymmetry is visible in this image even for such a small departure from normal incidence. The asymmetry of the image increases, as shown in Fig. 20, for collimated radiation incident at an angle of 1.25 deg. Figures 21 and 22, for radiation incident at angles of 1.5 deg and 1.75 deg, show strong image asymmetry and significant reduction of the incident radiation reaching the active flake. Finally, for radiation entering the aperture of the optics module at an angle of only 2.0 deg, Fig. 23 shows a marked reduction of energy incident to the active flake.

In Figs. 19 and 20, the asymmetry of the image is caused by the angle of incident radiation coupled with the manner in which the secondary mirror and its mount shadow the active flake. However, Figs. 21, 22, and 23 show that *less* energy arrives at the flake,

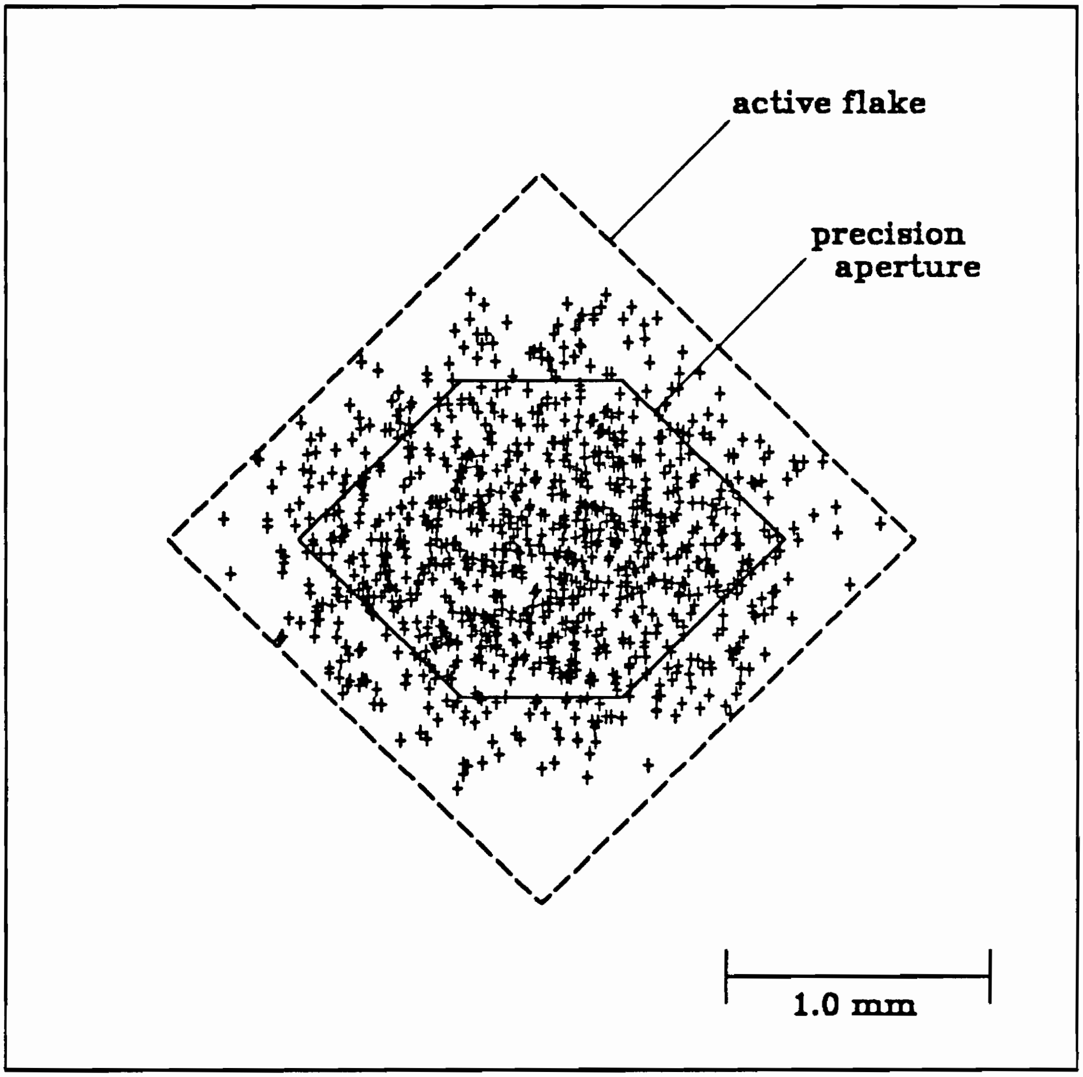


Fig. 17. Radiation Distribution on Active Flake for Diffuse Radiation Incident to the Instrument Aperture.

primarily because much of the energy is reflected back out of the telescope or absorbed on the walls of the optics module. It is possible, or even likely, that the signal attenuation caused by the secondary mirror mount will be different for different types of sources (Earth scene and calibration sources). If so, the shadow of the secondary mirror mount cast onto the active flake, clearly visible in Figs. 18 through 23, should not be ignored.

The results in Figs. 18 through 23 are summarized in Fig. 24, which shows the fraction of energy from a source of collimated radiation entering the instrument aperture (open end shown in Fig. 5) that arrives at the active flake as a function of the angle of incidence. The absorptivities in Table 1 were used to calculate the results given in Fig. 24. This figure provides a measure of the optical performance of the actual instrument. The maximum amount of energy that reaches the active flake at a given angle, in this case, one degree, is used to normalize the energy arriving at the active flake at all angles. The maximum value does not occur at 0 deg as expected because of the Monte Carlo approximation. This figure clearly shows the fraction of energy that reaches the active flake drops from 87 percent for collimated radiation incident normal to the instrument aperture to 37 percent for an angle of 2.0 deg off the optical axis in the satellite ground track direction. Similarly, for the same range of angles, the fraction of energy incident to the active flake drops from 91 percent for collimated radiation incident normal to the instrument aperture to nine percent for radiation incident at an angle of 2.0 deg off the optical axis in the scan direction, as shown in Fig. 25. The absorptivities in Table 1 were used to obtain the results presented in this figure also. Figure 8 shows how the scan and satellite ground track directions are defined with respect to the field stop aperture.

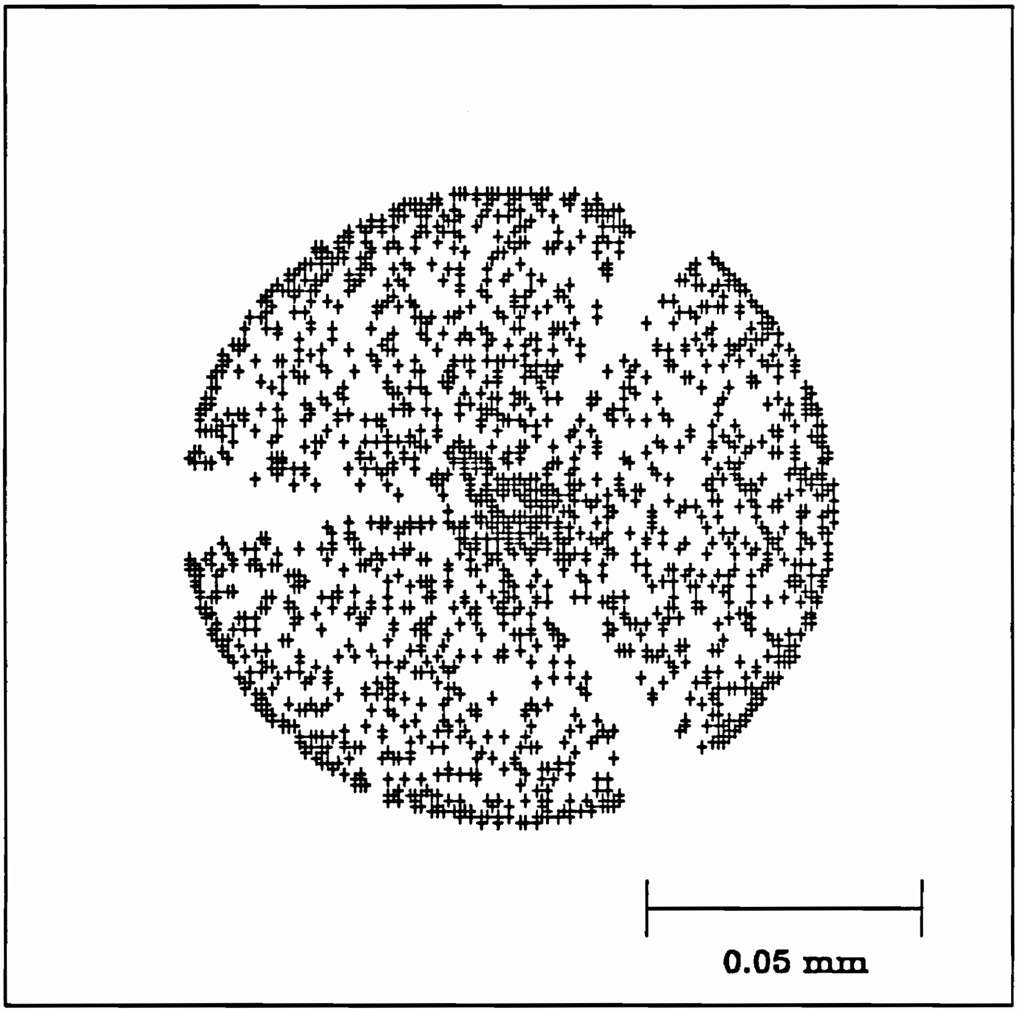


Fig. 18. Radiation Distribution on Active Flake for Collimated Radiation Incident to the Instrument Aperture at an angle of 0.0 deg (43.6 percent of energy incident to instrument aperture reaches the flake).

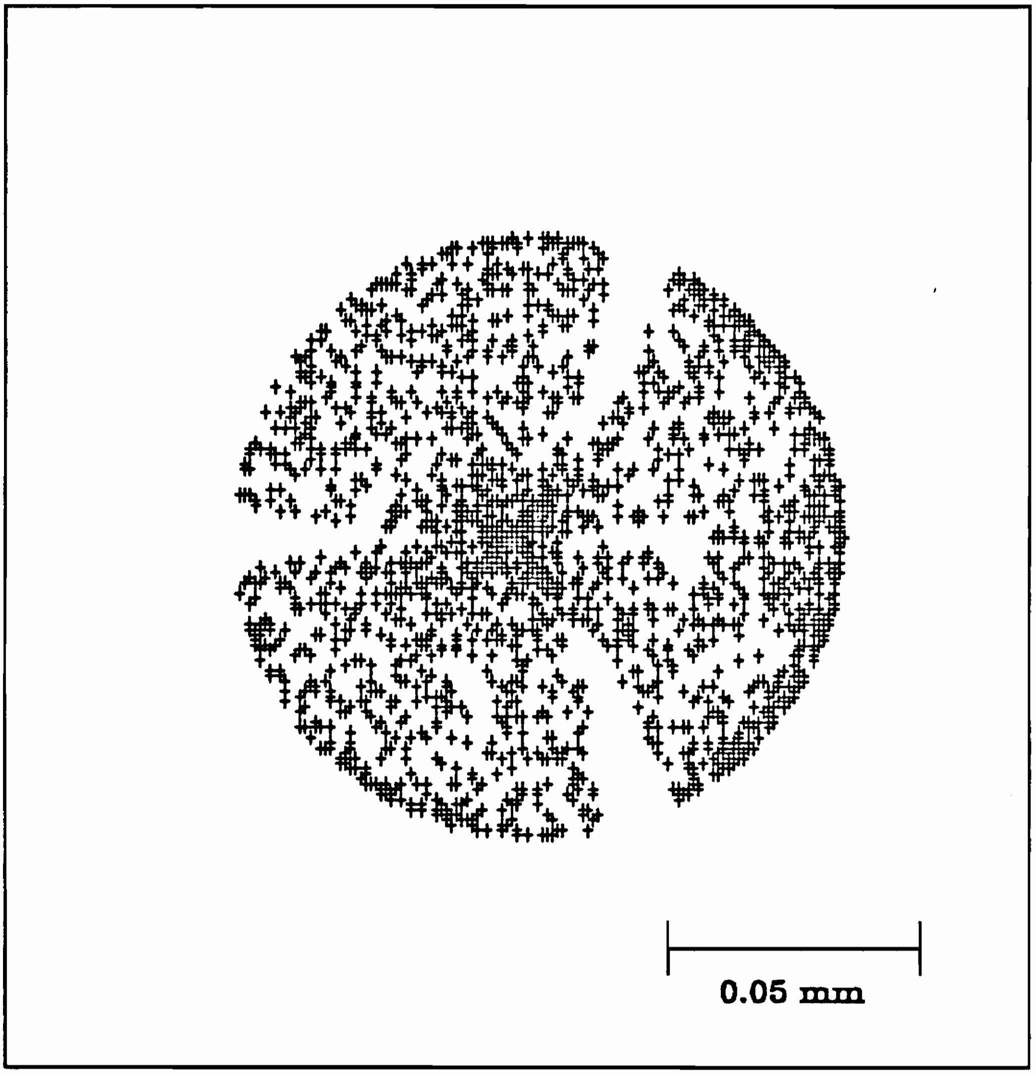


Fig. 19. Radiation Distribution on Active Flake for Collimated Radiation Incident to the Instrument Aperture at an angle of 1.0 deg (42.6 percent of energy incident to instrument aperture reaches the flake).

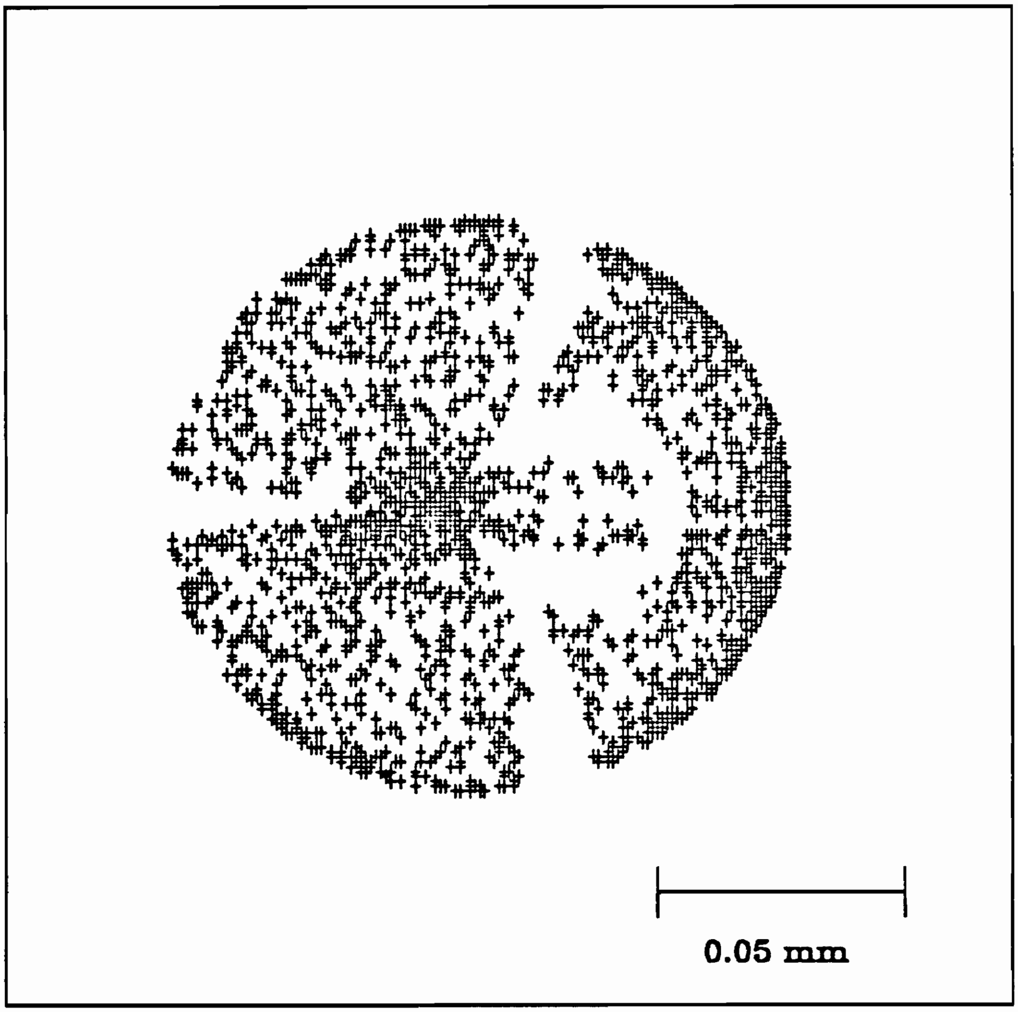


Fig. 20. Radiation Distribution on Active Flake for Collimated Radiation Incident to the Instrument Aperture at an angle of 1.25 deg (41.0 percent of energy incident to instrument aperture reaches the flake).

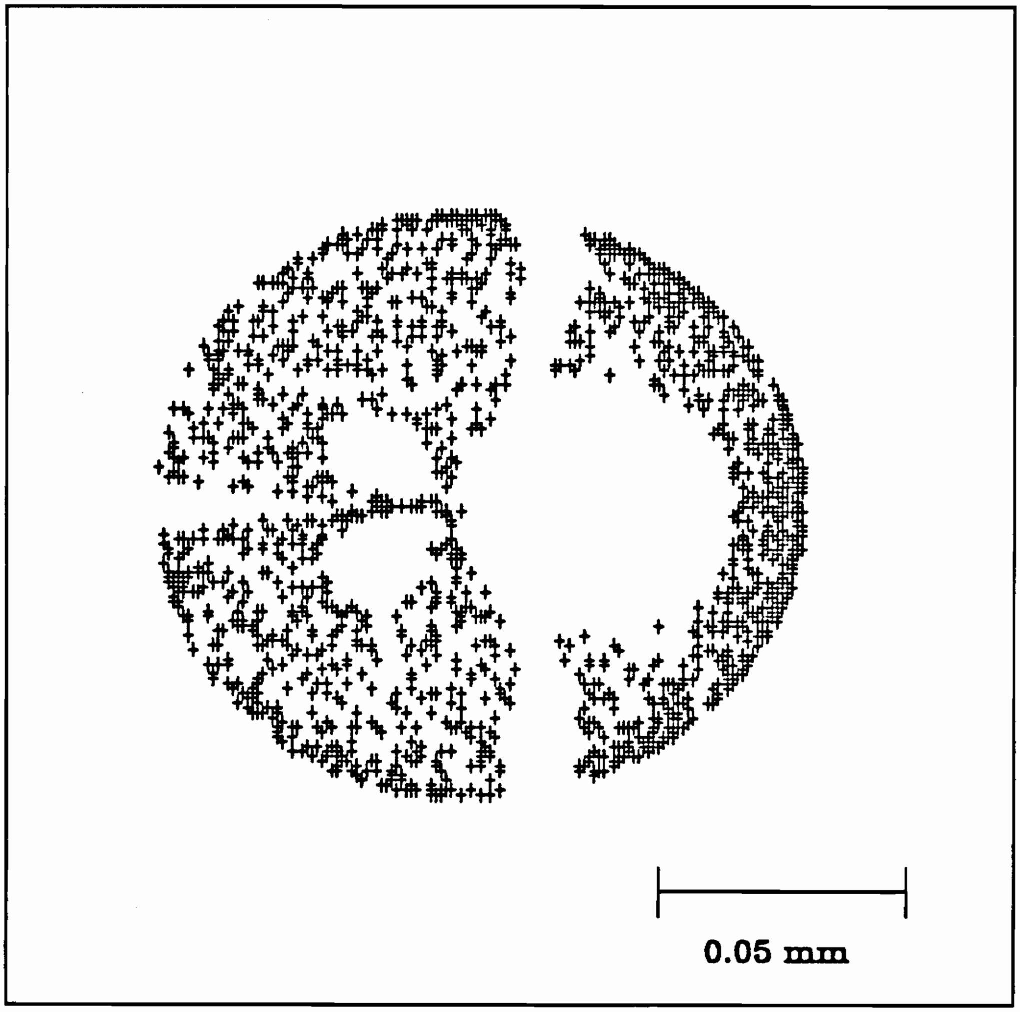


Fig. 21. Radiation Distribution on Active Flake for Collimated Radiation Incident to the Instrument Aperture at an angle of 1.5 deg (33.4 percent of energy incident to instrument aperture reaches the flake).

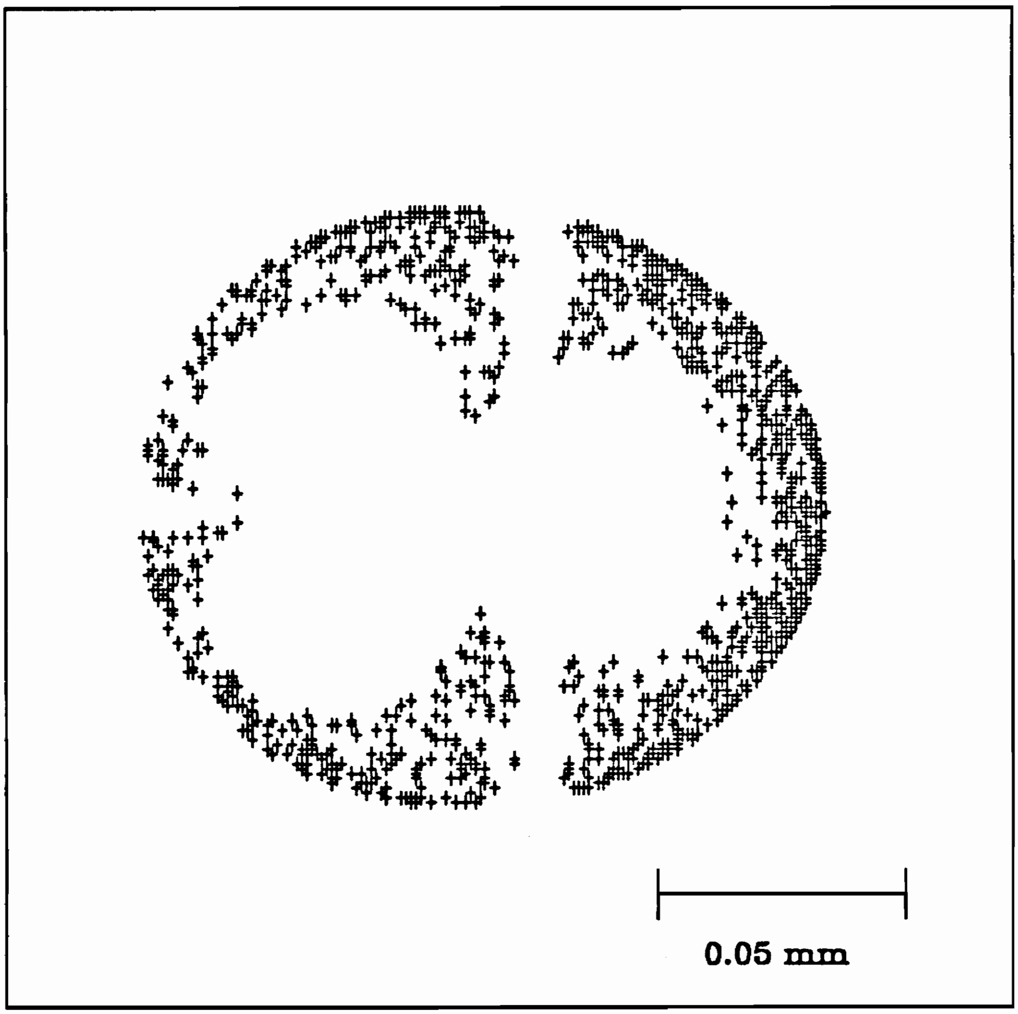


Fig. 22. Radiation Distribution on Active Flake for Collimated Radiation Incident to the Instrument Aperture at an angle of 1.75 deg (24.1 percent of energy incident to instrument aperture reaches the flake).

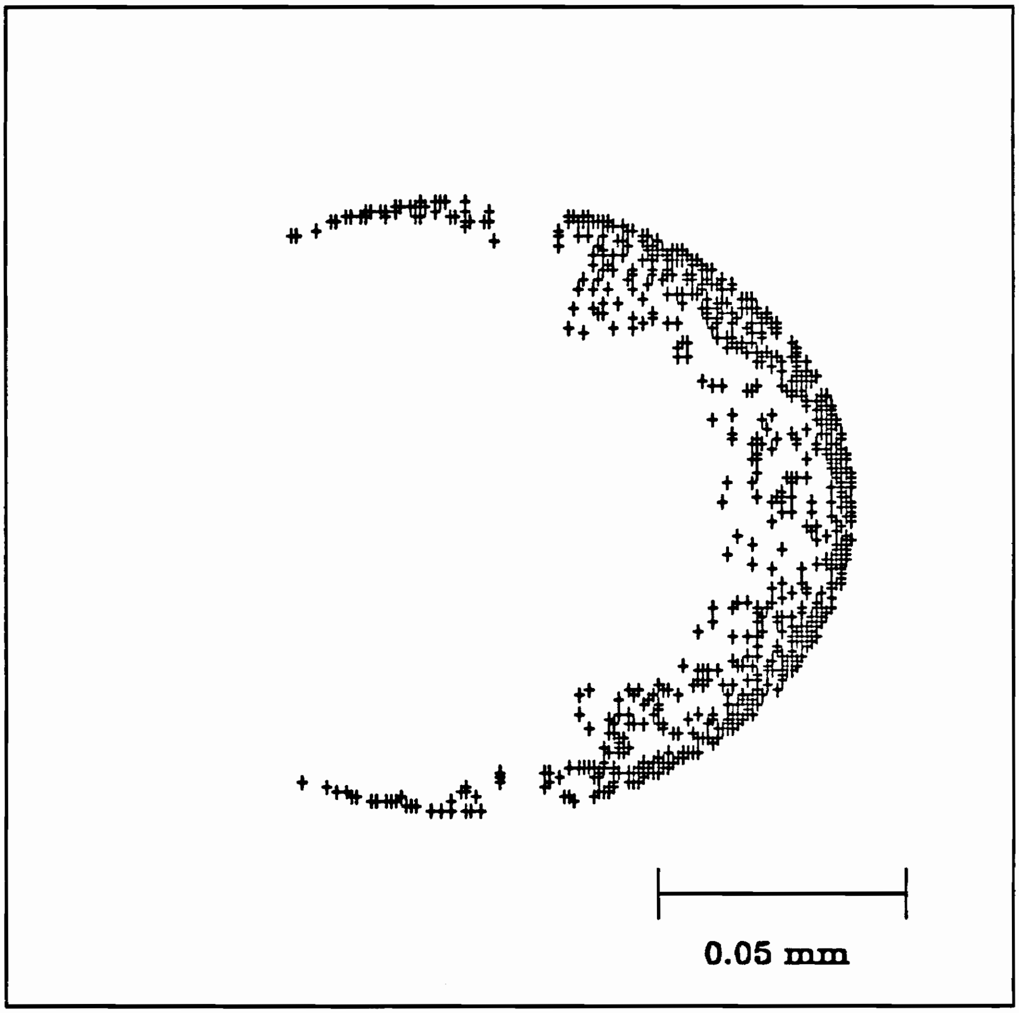


Fig. 23. Radiation Distribution on Active Flake for Collimated Radiation Incident to the Instrument Aperture at an angle of 2.0 deg (15.9 percent of energy incident to instrument aperture reaches the flake).

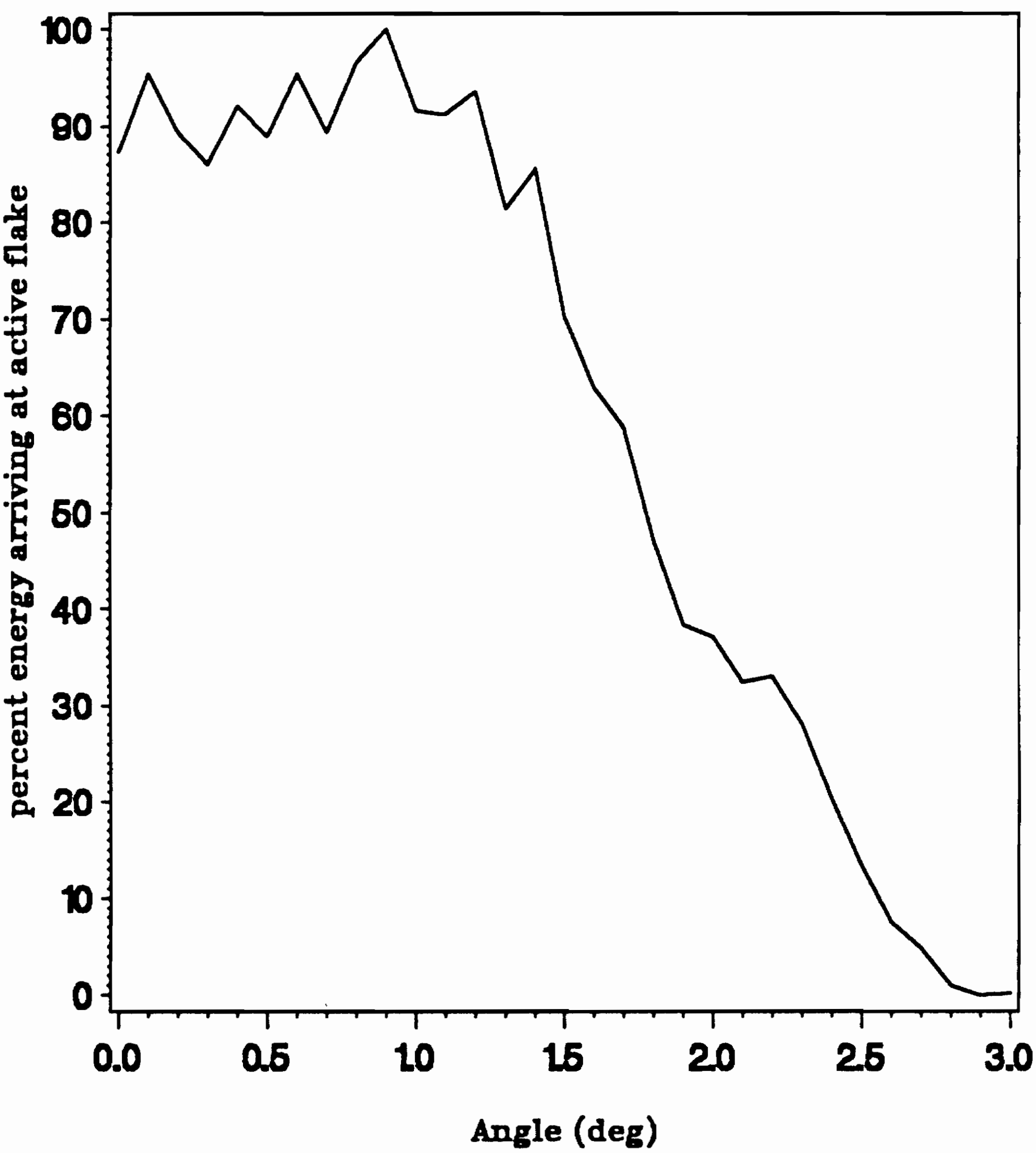


Fig. 24. Energy Arriving at Active Flake as a Function of the Incident Angle of Collimated Radiation: Satellite Ground Track Direction.

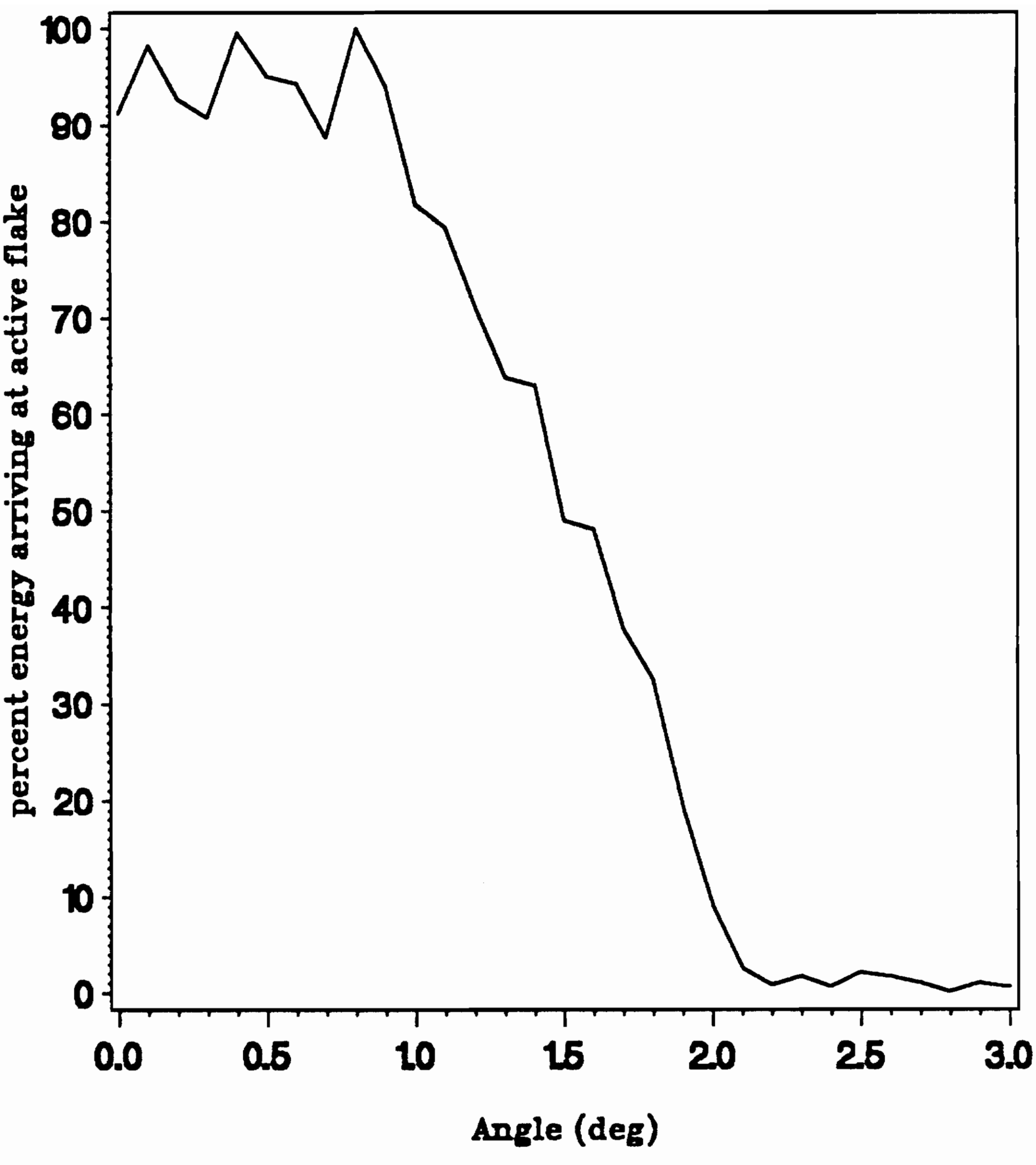


Fig. 25. Energy Arriving at Active Flake as a Function of the Incident Angle of Collimated Radiation: Scan Direction.

Figures 26 and 27 show the fraction of energy, $P(\theta)$, from a source of collimated radiation accumulating on the active flake for all angles, θ, between 0 deg (on-axis) and 3.0 deg, as defined by

$$P(\theta) = \frac{\int_0^{\theta} f(\theta') \cos \theta' d\theta'}{\int_0^{3} f(\theta') \cos \theta' d\theta'} . \tag{4.3}$$

In Eq. (4.3), $f(\theta')$ is the function plotted in Figs. 24 and 25, and the $\cos \theta'$ weighting factor projects the area of the aperture normal to the incoming beam of collimated radiation. These integrals were evaluated using the Romberg integration technique given in subroutine ROMB, Appendix A.

Figure 26 shows that negligible energy, less than five percent, reaches the active flake for collimated radiation incident to the instrument aperture (open end shown in Fig. 5) at an angle outside of 2.0 deg in the satellite ground track direction. The same is true along the scan direction for radiation incident to the instrument aperture outside of 1.8 deg, as shown in Fig. 27. Once again, Fig. 8 shows how the scan and satellite ground track directions are defined with respect to the aperture of the field stop. These results show that the ERBE scanning radiometer is highly sensitive to the direction of incoming radiation. This is why the ERBE scanning radiometer provides data that are good for estimating directional quantities like albedo and bidirectional reflectance, and good for resolving the anisotropy of Earth-emitted radiation.

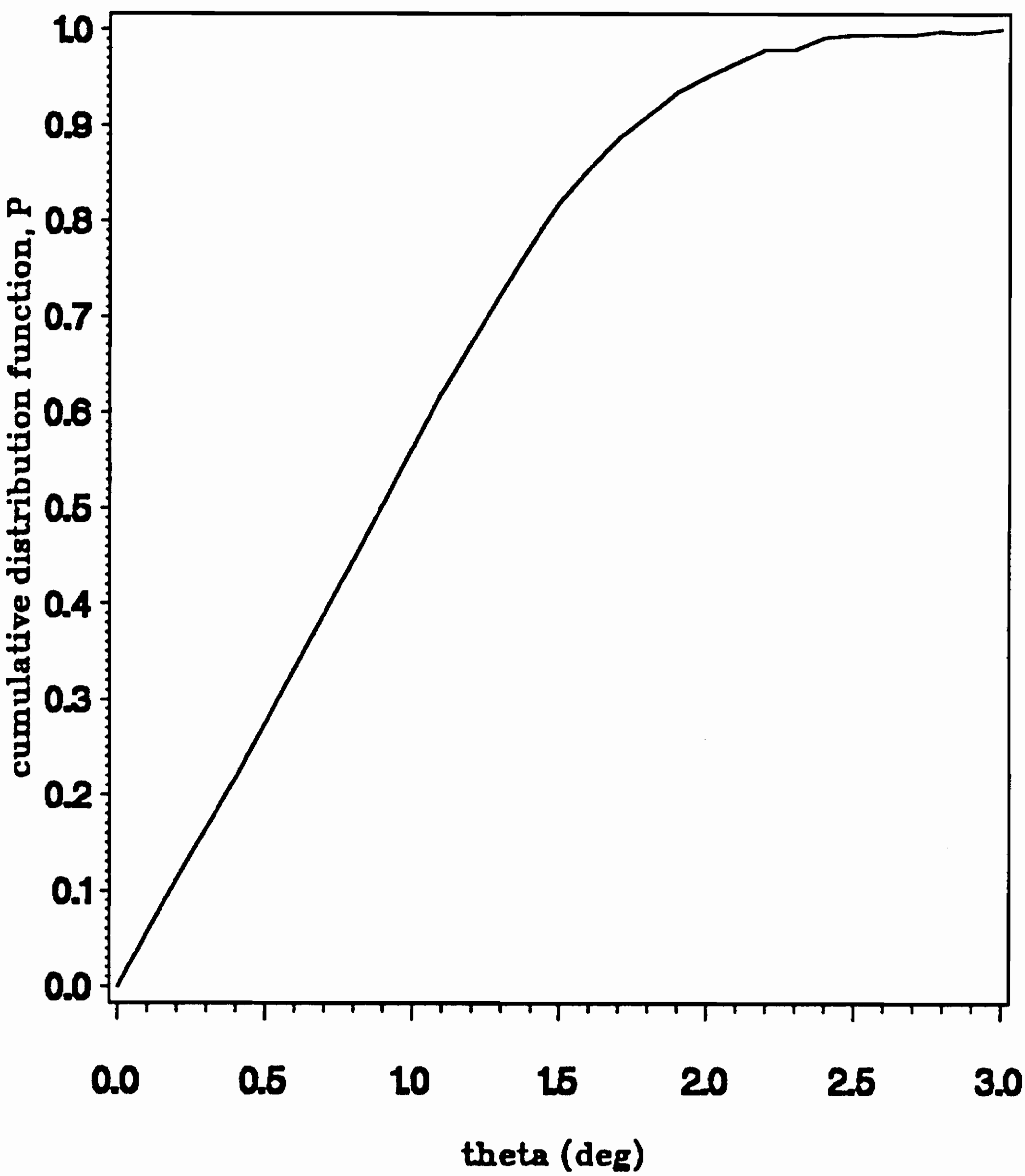


Fig. 26. Cumulative Energy Incident to Active Flake As Given by Equation (4.3): Satellite Ground Track Direction.

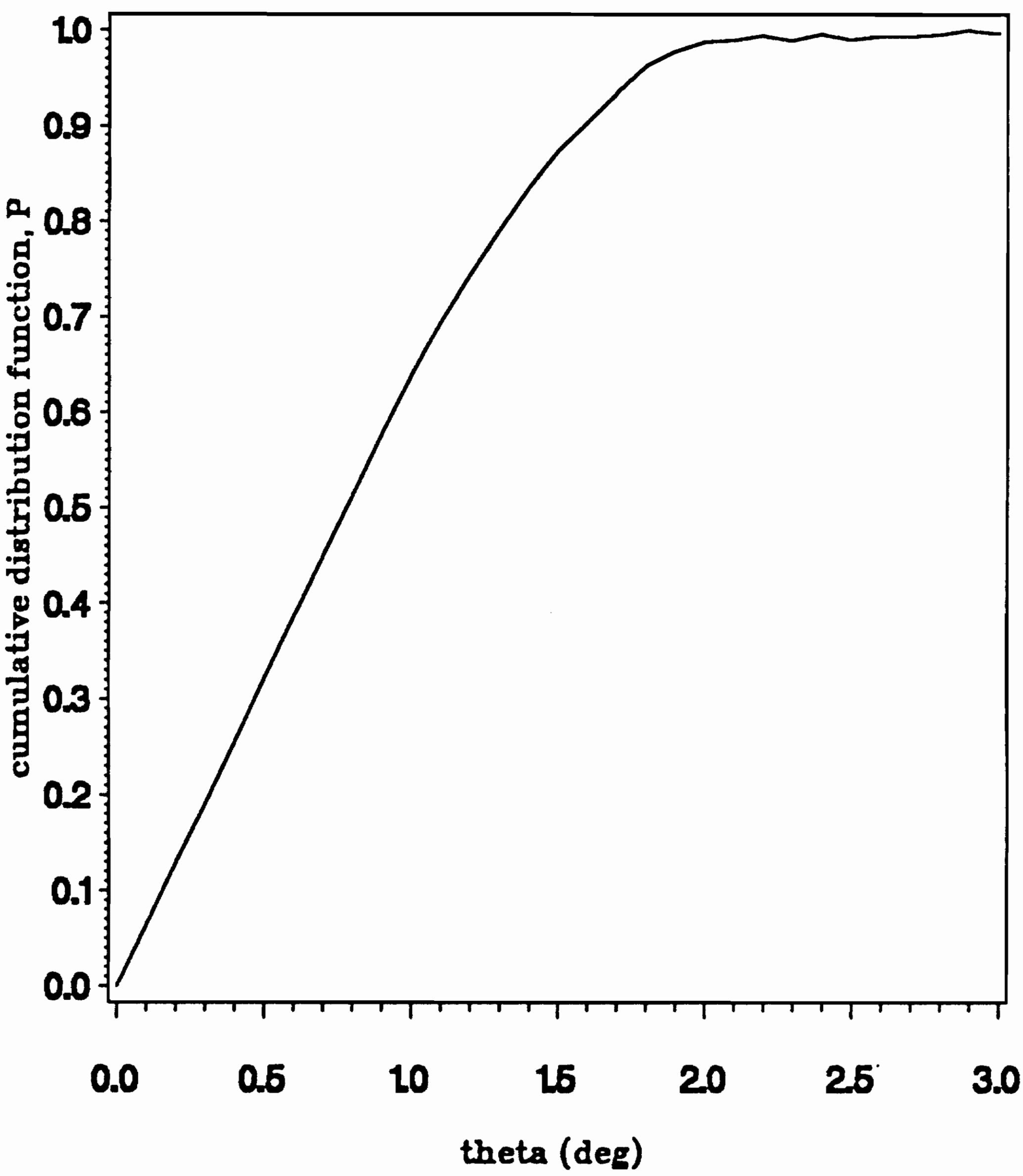


Fig. 27. Cumulative Energy Incident to Active Flake As Given by Equation (4.3): Scan Direction.

4.2.2 *The Optical Analysis Using Wavelength-Dependent (Spectral) Characterizations*

The optical analysis assuming gray characteristics was performed assuming perfectly specularly reflecting gray mirrors ($R_{spec} = 1$). This assumption is valid for estimating the optical performance of the ERBE scanning radiometer, or for characterizing its behavior when irradiated at long wavelengths, but it is not valid when the instrument is irradiated at short wavelengths. As mentioned in Chapter 2, the mirrors are coated to lessen the effects of ultraviolet radiation on the finish of the mirror surface, but this coating is not a good specular reflector of short wavelength radiation. Figure 12 shows that the spectral specular-to-total reflectivity ratio of the mirrors declines significantly around 0.4 μm. This suggests that a wavelength-dependent analysis would provide a better description of the optical performance of the ERBE scanning radiometer.

To conduct the wavelength-dependent analysis, two diffuse sources at different temperatures have been used to simulate the instrument calibration procedures. One source, at 5800 K, represents a solar calibration through the Mirror Attenuator Mosaic, or MAM, of the ERBE scanning instrument. The MAM is a baffled solar attenuator that uses a polished aluminum mirror with small spherical indentations to diffuse and attenuate the rays of the sun before they enter the aperture of the telescope. The MAM must be used during solar calibration to attenuate the solar radiation because thermistor flakes are easily sunburned. Direct exposure to sunlight focused by the telescope would burn a hole in the active flake, much like a magnifying glass held to the sun on a clear day would burn a hole in a sheet of paper. The other source represents the onboard internal blackbody calibration source at 300 K.

Table 3 shows the fraction of diffuse energy entering the aperture of the ERBE scanning radiometer optics module that arrives at the active flake. Again, the cone angle of the diffuse radiation has been limited to 14 deg. Consider the last row of Table 3. During the shortwave (5800 K source) calibration sequence, only 0.007 percent of the energy from the shortwave source of radiation is transmitted by the diamond filter of the longwave channel, as shown in the third column of the table. Conversely, during the longwave (300 K source) calibration sequence, only 0.008 percent of the energy from the longwave source is transmitted by the Suprasil filters of the shortwave channel, as shown in the last column of the table. During the shortwave calibration sequence, only the total and shortwave channels are calibrated. Similarly, during the longwave calibration sequence, only the total and longwave channels are calibrated. Lee states that there is no need to calibrate the longwave channel during the shortwave calibration sequence and vice versa [39]. Based on the filter transmissivity characteristics shown in Figs. 13 and 14, the results outlined above clearly support this statement.

Observe the third and fourth lines of the last row in Table 3. Ideally, the sum of the shortwave and longwave channel measurements should equal the total channel measurement. This can only be accomplished if the transmissivity of the filters is unity over their wavelengths of transmission. However, this is not the case, as shown in Figs. 13 and 14, especially for the diamond filter whose transmissivity spectrum is shown in Fig. 14. Fortunately, this loss of longwave radiation information does not compromise the objectives of ERBE since 60 to 80 percent of the longwave Earth-emitted radiation is in the 8 to 12 μm wavelength band [40]. However, scientists would like this measurement to be as accurate as possible so they can compare it with the ERBE nonscanning active cavity radiometer measurements. This permits a certain amount of redundancy.

Tables 4 and 5 show, respectively, the fractions of energy entering the aperture of the optics module from the 5800 K and 300 K diffuse sources that arrive at the active flake for the total channel of the ERBE scanning instrument. The results given in the second column of each table were calculated using the values of total absorptivity given in Table 1. The results given in the third column were calculated assuming that all surfaces of the optics module are perfectly absorbing and that the mirrors have the spectral specular-to-total reflectivity ratio characteristic given in Fig. 12. The last column in each table is the difference between the second and third columns. It is a measure of how much energy reaches the active flake by nonoptical paths.

Consider the last row of Tables 4 and 5. In Table 4, two percent of the energy from the 5800 K (shortwave) source reaches the active flake using the values of total absorptivity given in Table 1. In Table 5, 2.55 percent of the energy from the 300 K (longwave) source reaches the active flake for the same values of absorptivity. However, the fraction of energy that reaches the active flake assuming that the surfaces of the optics module are perfect absorbers, drops to 1.89 percent for the 5800 K source and to 2.50 percent for the 300 K source. Obviously, more energy reaches the active flake when the total channel is viewing a longwave source of radiation than when viewing a shortwave source. This is because the mirrors have a higher specular-to-total reflectivity ratio at longer wavelengths, as shown in Fig. 12. Thus, more radiation is able to follow the theoretical optical path shown in Fig. 4.

The difference between the values in the second and third columns of Tables 4 and 5 is 0.11 percent and 0.05 percent, respectively, as shown in the last column of each table. This suggests that more energy following nonoptical paths reaches the active flake when the instrument is viewing a diffuse source of shortwave radiation than when viewing a diffuse source of longwave radiation. This is consistent with the fact that more shortwave radiation than longwave radiation is diffusely reflected from the mirrors (refer to Fig. 12). The numbers shown in the last column of Tables 4 and 5 indeed show that some energy reaches the active flake by nonoptical paths. However, these numbers are small compared to the numbers in the third columm of each table, which show the amount of radiation that actually reaches the flake by the theoretical optical path. Only 5.5 percent (0.11 ÷ 2.00 × 100) of the energy from the 5800 K source arrives at the active flake by nonoptical paths. Similarly, only 2.0 percent (0.05 ÷ 2.55 × 100) of the energy from the 300 K source arrives at the active flake by nonoptical paths. True to its design, the optics module is excellent at keeping radiation randomly reflected from the surfaces of the module from reaching the active flake.

Consider again the effects of ultraviolet radiation on the mirrors of the total channel, and on the mirrors and filters of the shortwave channel, of the ERBE scanning radiometer. The entire spectral specular-to-total reflectivity ratio characteristic, shown in Fig. 12, and the entire spectral transmissivity characteristic of the Suprasil-W filter, shown in Fig. 13, were lowered 15 percent in one-percent decrements to simulate the effects of ultraviolet radiation on the mirrors and filters of the ERBE scanning radiometer. A 5800 K source of collimated radiation, which resembles solar radiation perhaps attenuated and reflected by a cloud, is introduced normal to the instrument aperture. The results of the simulation are shown in Figs. 28 and 29.

Figures 28 and 29 show that less energy reaches the active flake, as expected. In Fig. 28, a one-percent decrease in the reflectivity of the mirrors causes two percent less energy to reach the active flake. This deficiency increases to 26.8 percent for a 15-percent decrease in the specular-to-total reflectivity ratio of the mirrors. Similarly, a

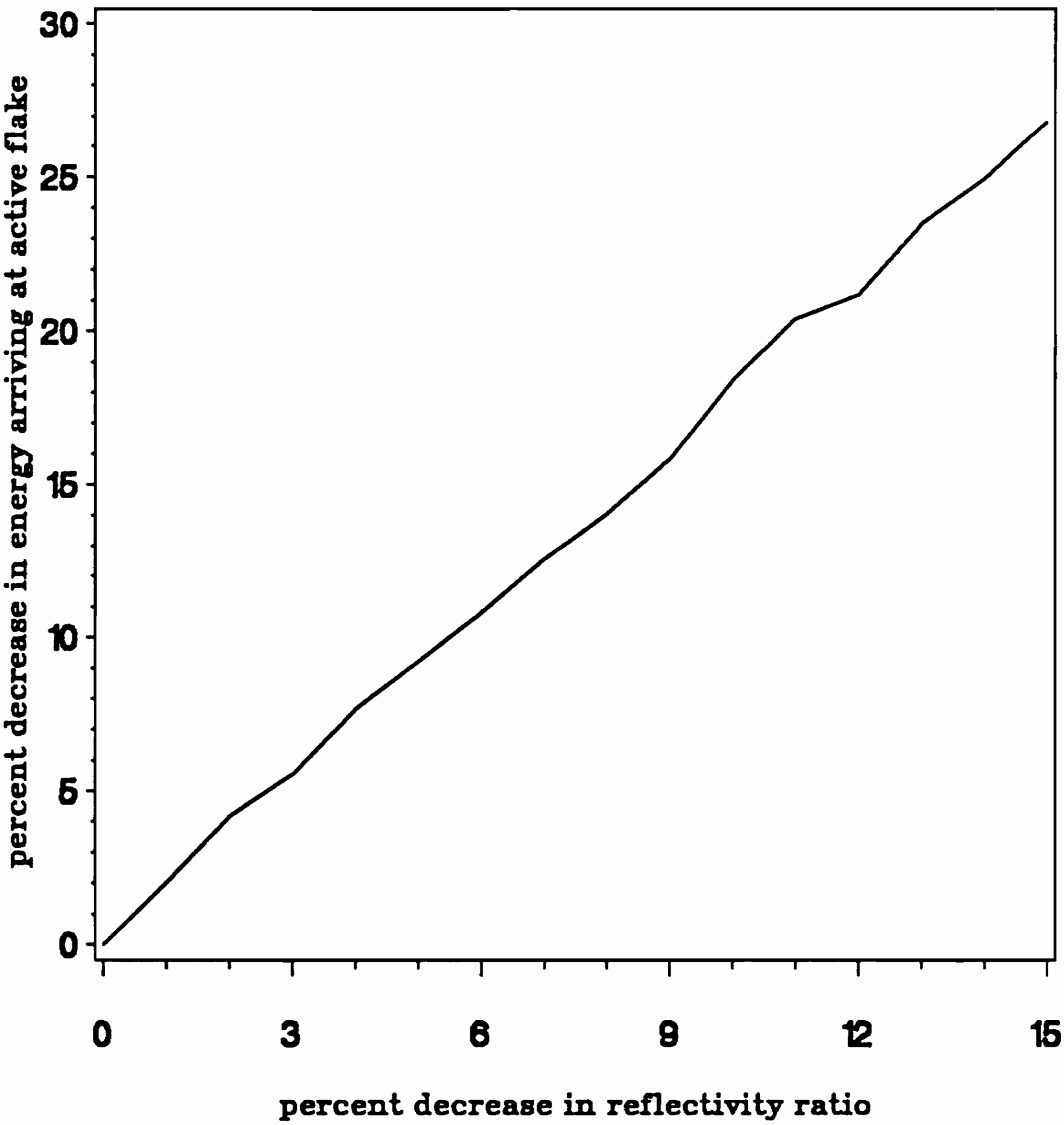


Fig. 28. Energy Arriving at Active Flake as a Function of Percent Decrease in the Spectral Specular-to-Total Reflectivity Ratio Spectrum of both the Primary and Secondary Mirrors.

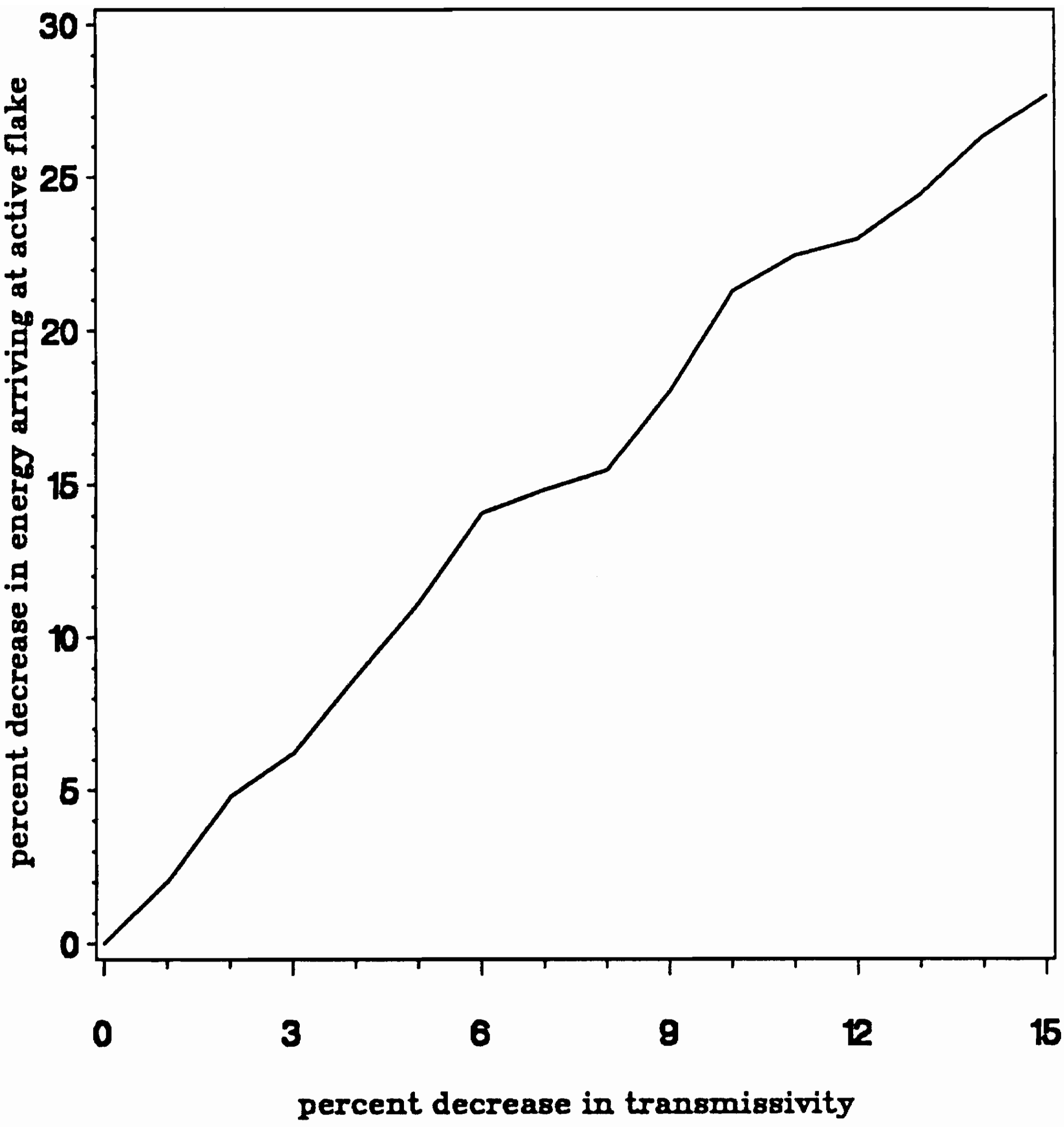


Fig. 29. Energy Arriving at Active Flake as a Function of Percent Decrease in the Spectral Transmissivity Spectrum of the Suprasil-W Filter.

one-percent decrease in the transmissivity of the Suprasil filters causes two percent less energy to reach the active flake. This shortfall increases to 27.7 percent less energy for a 15-percent change in the transmissivity of the filters, as shown in Fig. 29.

The theory regarding energy transfer through media that can absorb, reflect, and transmit radiation [26] predicts that the amount of energy entering the instrument aperture that reaches the active flake, assuming that the primary and secondary mirrors are the reflecting media, can be approximated by

$$Q \simeq \rho^2. \tag{4.4}$$

In Eq. (4.4), ρ is the reflectivity of the mirrors and Q is the amount of energy reaching the active flake. The reflectivity is squared because there are two mirrors in each channel of the ERBE scanning radiometer. Similarly, since there are two Suprasil filters in the shortwave channel of the ERBE scanning radiometer, the amount of energy entering the instrument aperture that arrives at the active flake can be approximated by

$$Q \simeq \tau^2. \tag{4.5}$$

In Eq. (4.5), τ is the transmissivity of the filters and Q is the amount of energy reaching the active flake.

Taking the derivative of both sides of Eqs. (4.4) and (4.5) with respect to ρ and τ, respectively, separating variables, and then dividing the results by Eqs. (4.4) and (4.5), respectively, yields

$$\frac{dQ}{Q} \simeq 2\frac{d\rho}{\rho}, \tag{4.6}$$

and

$$\frac{dQ}{Q} \simeq 2\frac{d\tau}{\tau}. \tag{4.7}$$

Equations (4.6) and (4.7) reveal that for each percentage decrement in the reflectivity of the mirrors or the transmissivity of the filters, the percent decrease in the amount of energy reaching the active flake is approximately twice that decrement. The slope of the curves plotted in Figs. 28 and 29 is nearly two as Eqs. (4.6) and (4.7) predict. Both curves are approximately linear, which suggests once again that the optics module is good at absorbing any radiation diffusely reflected from the mirrors or any radiation reflected by the Suprasil filters. The curves are not smooth because of the Monte Carlo approximation. However, for a decrease in the specular-to-total reflectivity ratio of the mirrors or transmissivity of the filters of only one percent, two percent less energy arrives at the flake, as shown in these figures. This surpasses the one-percent uncertainty goal for ERBE measurements. Therefore, the effects of ultraviolet radiation cannot be ignored and should be given careful consideration when assessing any degradation in the optical performance of this instrument.

4.2.3 A Brief Comparison of the Gray and Spectral Optical Analyses

It is important to realize that the wavelength-dependent model must be used when characterizing the optical performance of the ERBE scanning radiometer, especially when shortwave radiation enters the instrument aperture. The results in Tables 2 and 5 compare well because the results in Table 2 were obtained in the gray analysis assuming perfectly specularly reflecting mirrors, and the results in Table 5 were obtained in the longwave spectral analysis using mirrors that are highly specular when irradiated at long wavelengths ($R_{spec} = 0.99$). However, when comparing Tables 2 and 4, less energy reaches the active flake when the mirrors are irradiated at short wavelengths, as shown in Table 4. Compare the last row and second column of Tables 2 and 4. The fraction of energy that reaches the active flake drops from 2.55 percent in Table 2, using the gray assumption, to two percent in Table 4 using the spectral analysis. This is because the reflectivity of the mirrors declines significantly around 0.4 μm, and more shortwave radiation is diffusely reflected from the mirrors. The gray assumption does not properly account for this difference.

4.2.4 Assessing Monte Carlo Convergence

In Tables 2 through 5, different numbers of energy bundles have been emitted to assess the convergence of the Monte Carlo method. When doubling the number of diffusely emitted energy bundles from 50,000 to 100,000 in each table, negligible changes occur in the results. At least three minutes of CPU time are required for all cases where 100,000 energy bundles are diffusely emitted into the aperture of the instrument. When doubling the number of diffusely emitted energy bundles from 100,000 to 200,000, the negligible changes in the results do not justify doubling the computing time. Because only small variations occur between the values given in the tables for different numbers of energy bundles emitted, it may reasonably be assumed that the Monte Carlo method solution has converged.

4.3 The Radiative Analysis

The radiative analysis is performed using distribution factors as discussed in Chapter 3. They are used to determine how much thermal energy emitted from the components of the ERBE telescope reaches the active flake. The thermal emissions from these components are compared to the maximum radiative power entering the instrument aperture that reaches the active flake. That would be from a source of collimated radiation. Therefore, Figs. 30 through 37 have been composed using a source of collimated radiation at 300 K introduced normal to the aperture of the optics module of the total channel. The noise levels that the ERBE telescope contribute to the signal arriving at the flake have been obtained for source temperatures between 250 K and 350 K, and for all components of the instrument, assuming they are isothermal. The component temperatures are assumed to be within ±25 percent of the heater-controlled (311 K) operating temperature of the active flake. The noise levels presented in these figures do *not* include radiation that enters through the instrument aperture and reaches the active flake along nonoptical paths. The noise energy has been nondimensionalized using the signal energy of the source of collimated radiation at 300 K.

Figures 30 and 31 show the nondimensional noise energy that arrives at the active flake from the pivoted baffle and the reflector cap of the ERBE telescope, respectively. Figure 5 shows where these components are located with respect the active flake. Figures 32 and 33 show the noise energy arriving at the active flake from the secondary mirror mount and the detector housing of the ERBE telescope, respectively. Figure 6 shows where these components are located with respect to the active flake. Each of these components have mostly black surfaces with high emittance; however, the secondary mirror mount and the detector housing are closer to the active flake than the pivoted baffle or the reflector cap. Therefore, they have a greater influence on the noise level at the active flake.

Figure 34 shows the nondimensional noise energy that arrives at the active flake from the primary insert, shown in Fig. 6. The information in this figure is significant because all of the noise energy that reaches the active flake from this component must first be reflected from other surfaces of the optics module. The numbers are very low, which means that most of the radiation emitted from the primary insert is absorbed on the walls of the optics module or escapes through the instrument aperture.

Figures 35 and 36 quantify thermal emissions from the field stop and the detector header, respectively. These two components are the closest to the flake, as shown in Fig. 6, but do not contribute the highest amount of noise energy because the side of the field stop facing the active flake is electrodeposited nickel and the detector header is polished aluminum. Each of these component finishes have a low emissivity, and therefore, contribute only a small amount of thermal noise energy to the active flake.

Figure 37 shows the amount of thermal energy emitted from the active flake that is reflected back to it (self-contamination). Because the active flake is practically surrounded by a specular enclosure, some radiation emitted from it is reflected back to it. Only an extremely small amount of radiation is reflected back to the active flake from the secondary mirror that sits directly above it. The secondary mirror is highly specular to longwave radiation, and most of the radiation from the active flake that strikes it will be reflected to the primary mirror and then reflected out of the telescope.

It is easy to observe, referring to Figs. 30 through 37, that most of the thermal noise energy incident to the active flake comes from the secondary mirror mount and the detector housing, as shown in Figs. 32 and 33. These are the largest, primarily black components with high emittance closest to the active flake. Only extremely small amounts of radiation from the pivoted baffle and the primary insert actually reach the active flake (refer to Figs. 30 and 31). The amounts of thermal noise energy from the reflector cap, field stop, detector header, and the active flake lie between these two extremes. The ERBE telescope components have been ranked in Table 6 from the component contributing the most thermal noise to the active flake to the component contributing the least amount.

Table 6 also presents the amount of noise energy the active flake is subjected to assuming that the ERBE telescope is isothermal at 311 K. Collimated radiation at 255 K, the effective radiating temperature of the Earth, has been used as the source. This particular case may be typical of a normal Earth scan sequence. The total noise contribution to the active flake is 2.2 percent of the incoming signal where 1.1 percent of that noise comes from the secondary mirror mount and the detector housing. The noise level from other components of the telescope are also presented in Table 6 for this special case. Although the total noise level is remarkably low for an instrument as complex as the ERBE telescope, especially in view of the large number of black surfaces, the noise energy from the secondary mirror mount and the detector housing, in this representative case, surpass the one-percent uncertainty that the ERBE measurements are to achieve. Therefore, this demonstrates how important it is to calibrate these instruments before each Earth scan to ensure that the instrument is "zeroed" so that noise energy is not mistaken as signal energy.

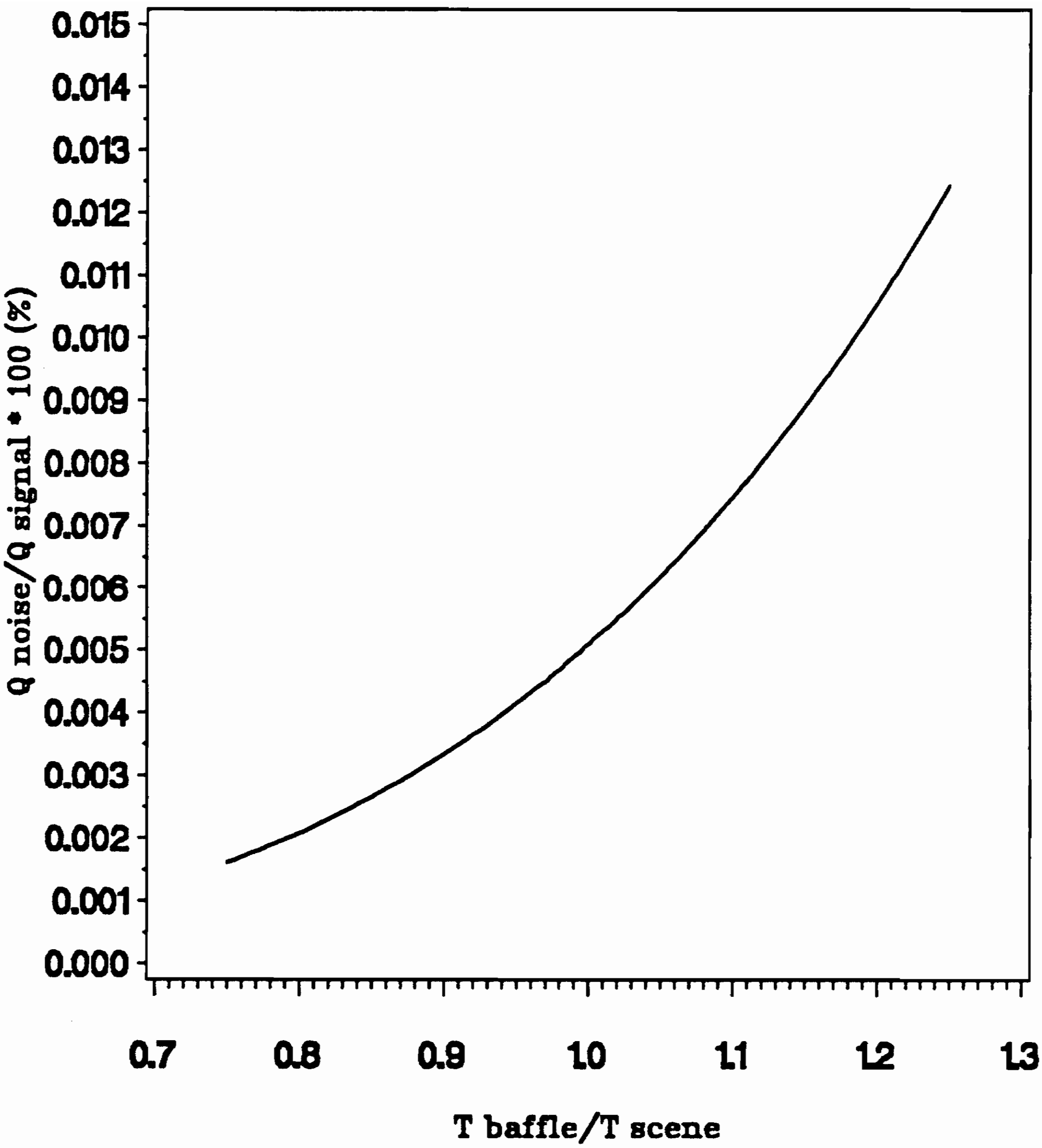


Fig. 30. Ratio of Thermal Emission from Pivoted Baffle to Signal Energy Arriving at the Active Flake.

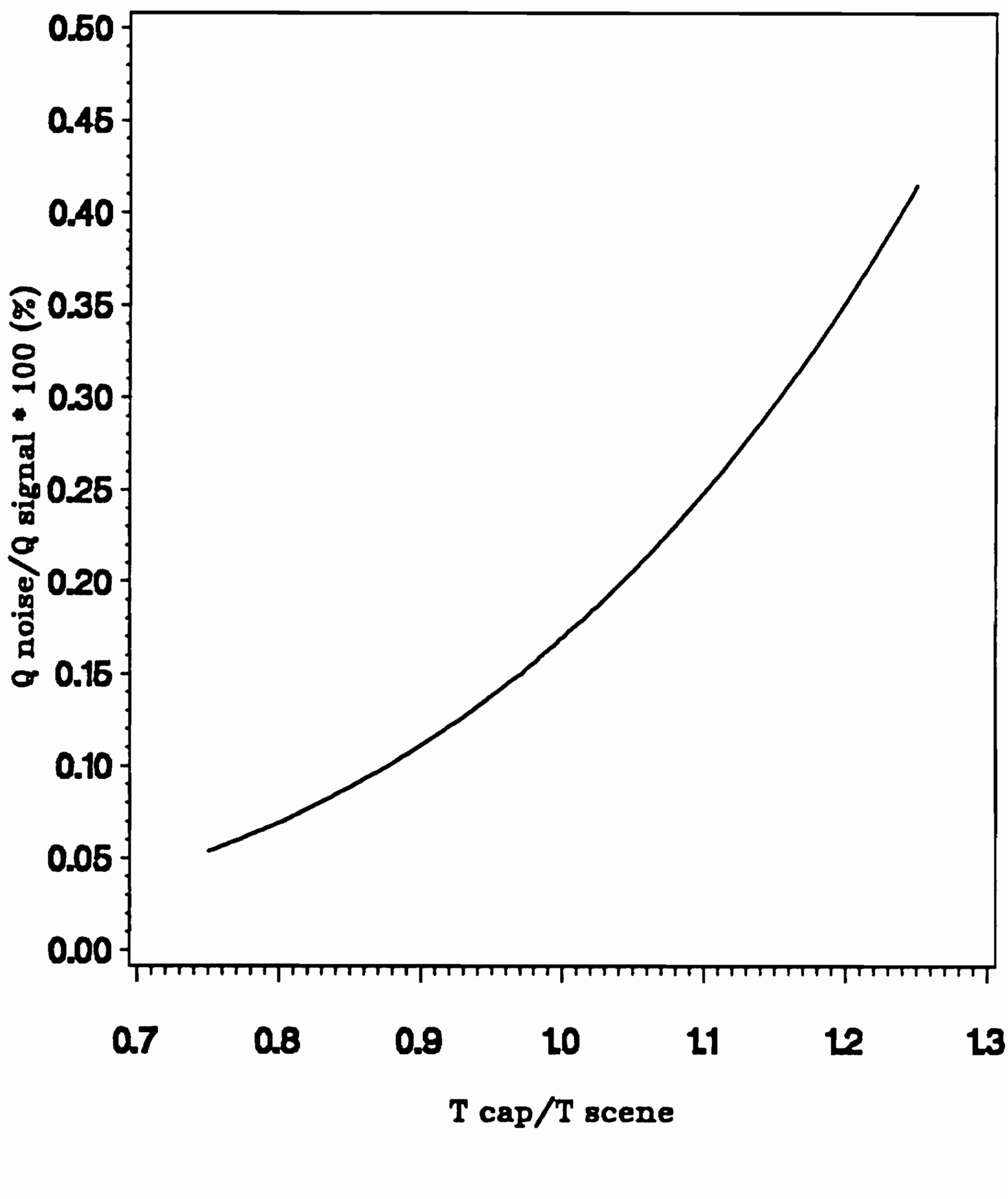


Fig. 31. Ratio of Thermal Emission from Reflector Cap to Signal Energy Arriving at the Active Flake.

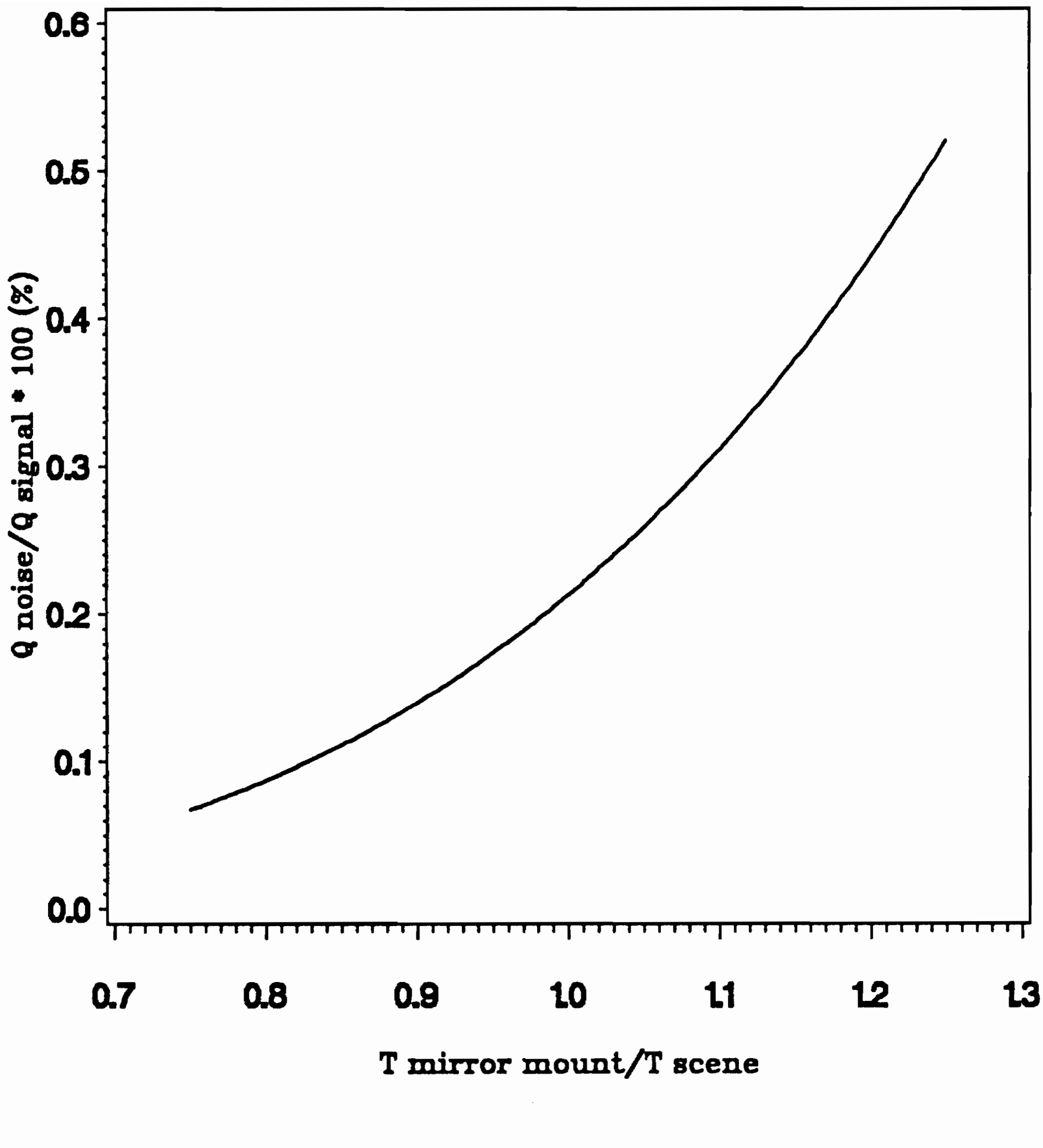


Fig. 32. Ratio of Thermal Emission from Secondary Mirror Mount to Signal Energy Arriving at the Active Flake.

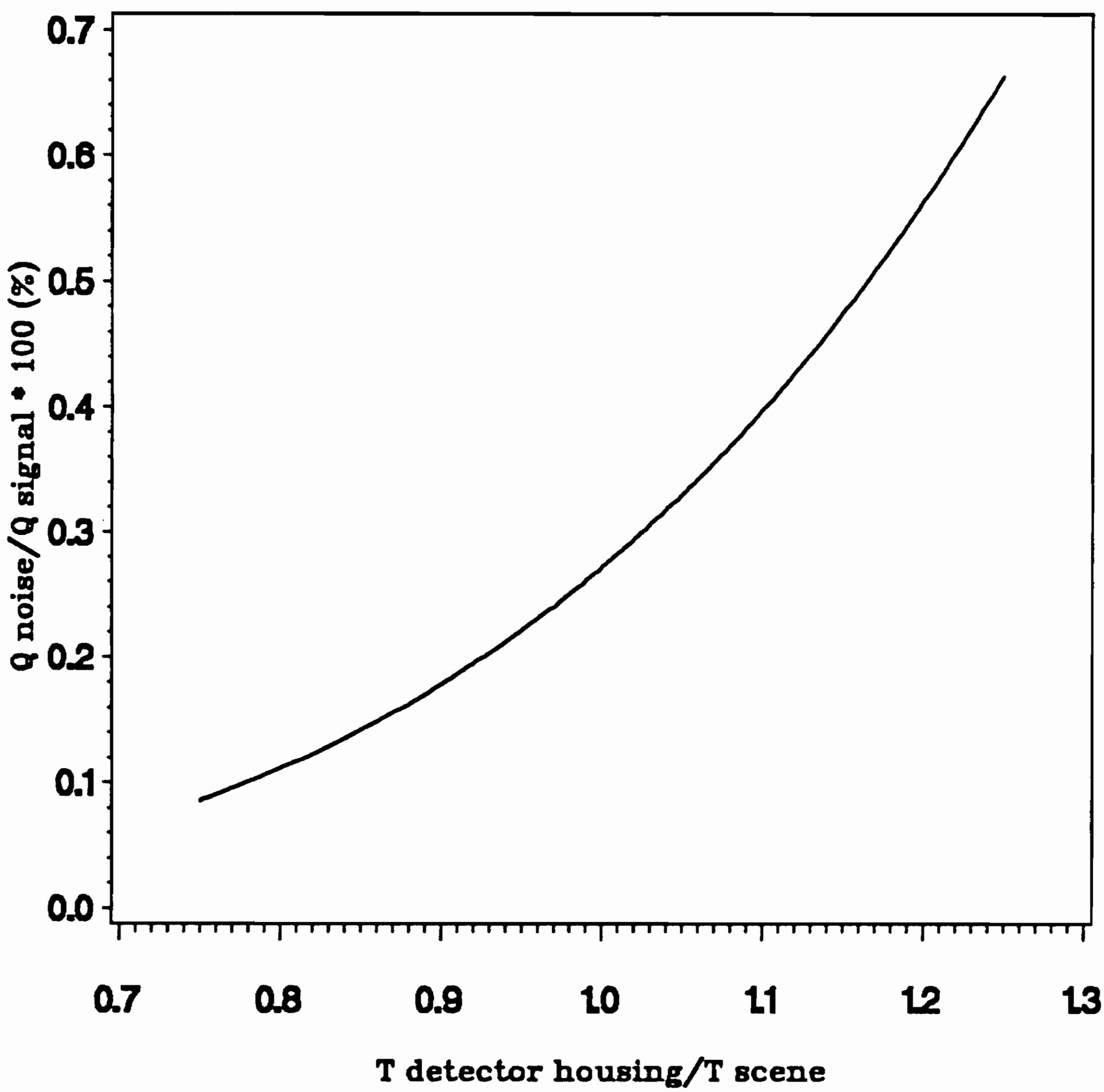


Fig. 33. Ratio of Thermal Emission from Detector Housing to Signal Energy Arriving at the Active Flake.

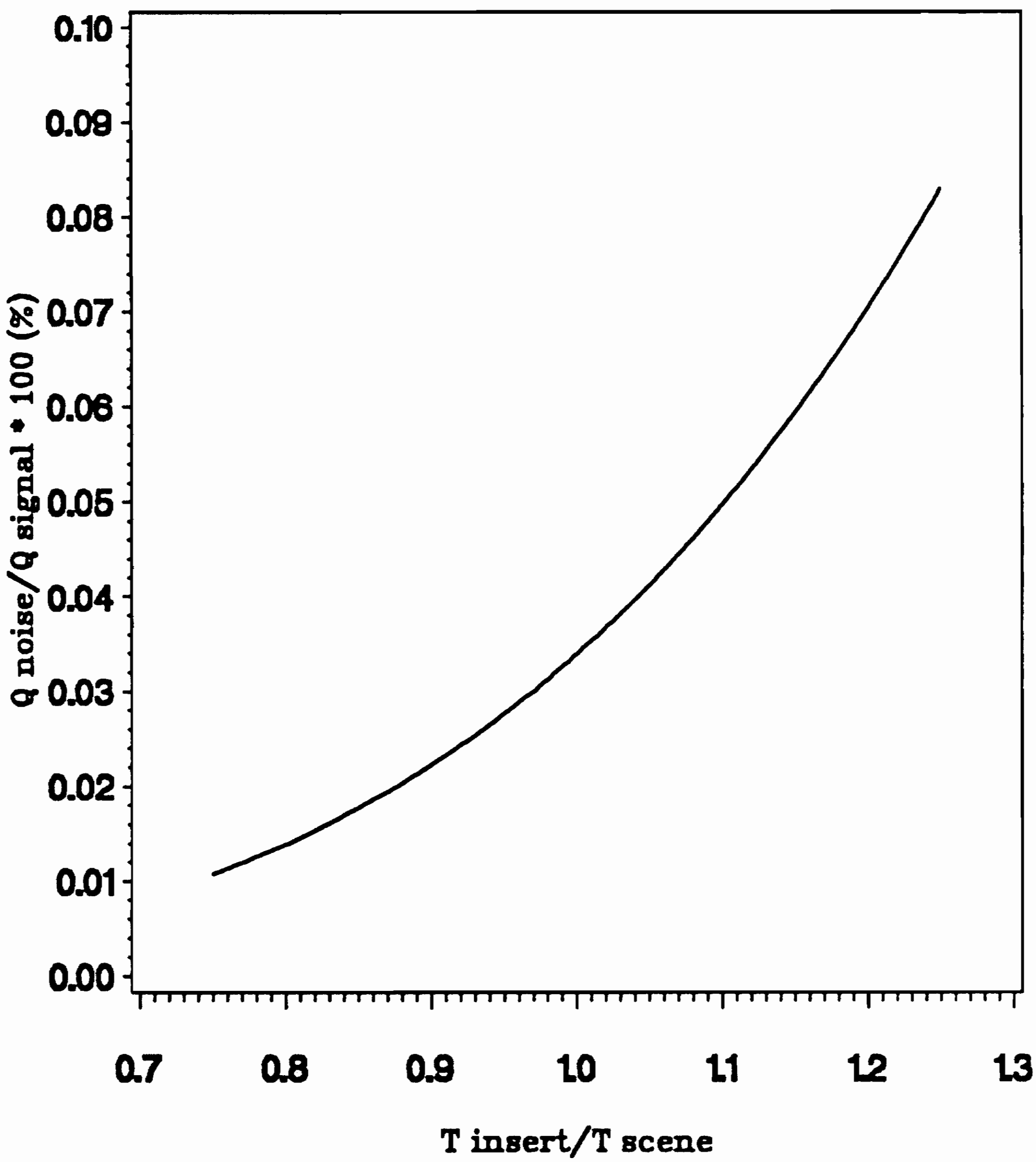


Fig. 34. Ratio of Thermal Emission from Primary Insert to Signal Energy Arriving at the Active Flake.

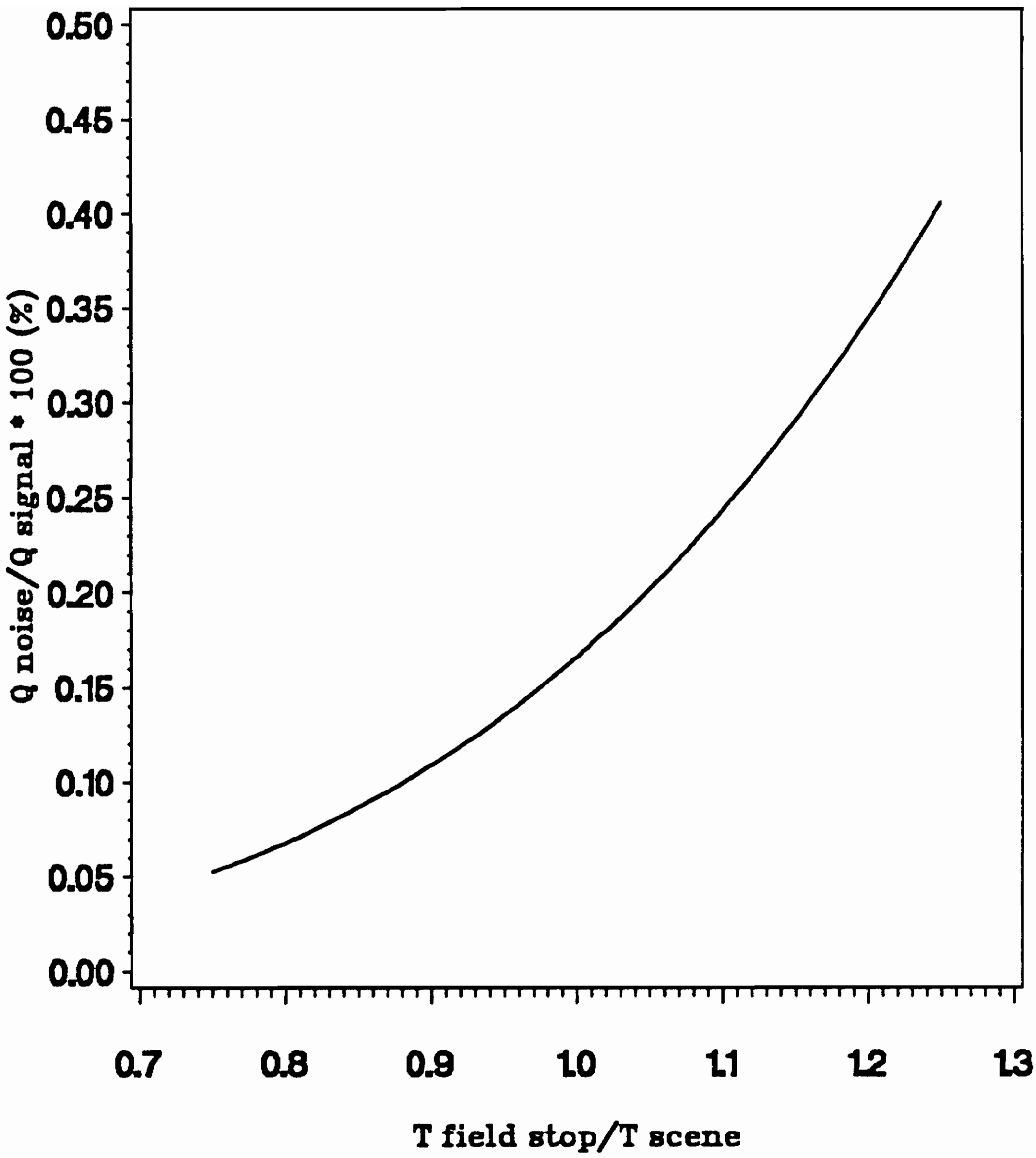


Fig. 35. Ratio of Thermal Emission from Field Stop to Signal Energy Arriving at the Active Flake.

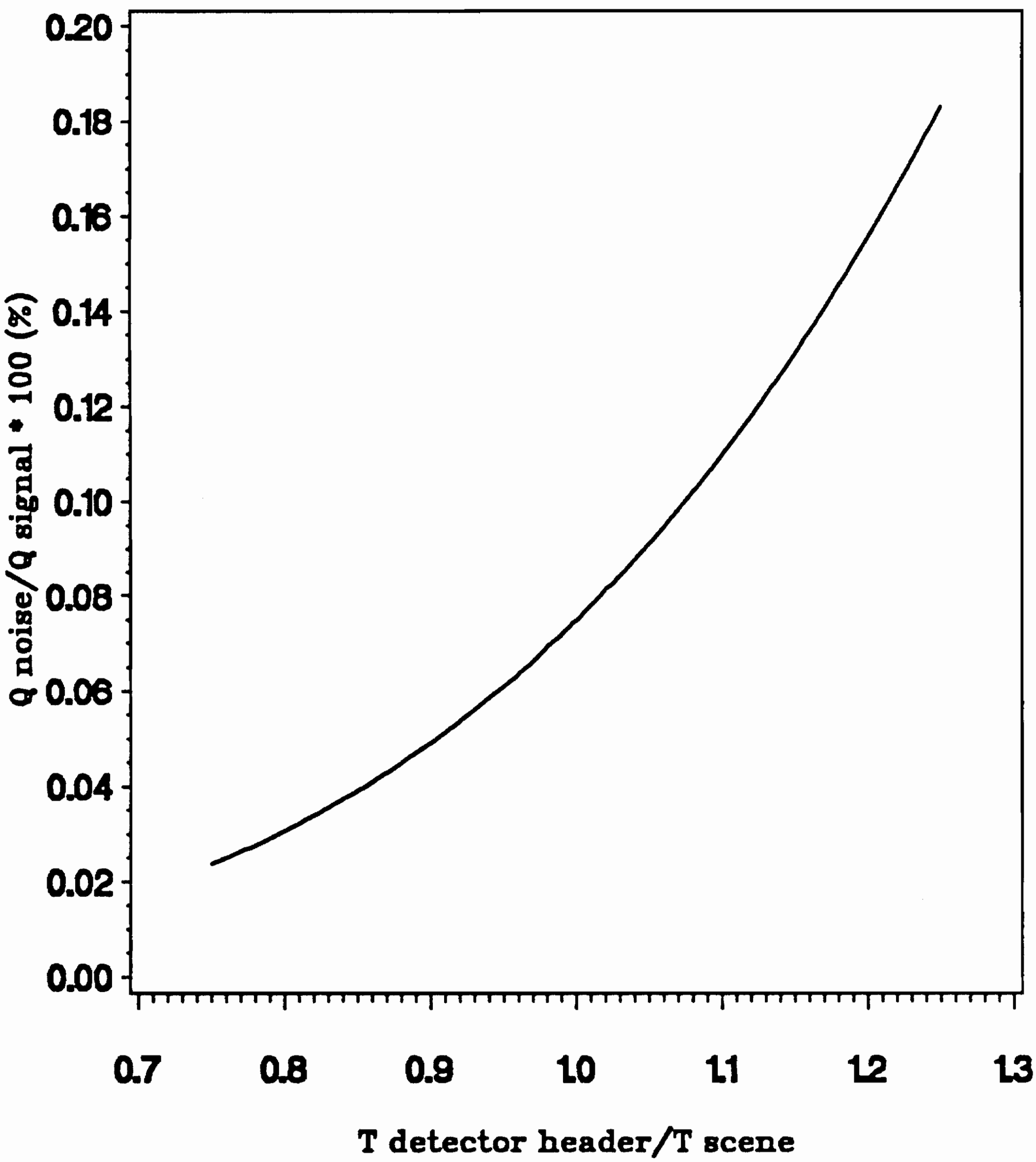


Fig. 36. Ratio of Thermal Emission from Detector Header to Signal Energy Arriving at the Active Flake.

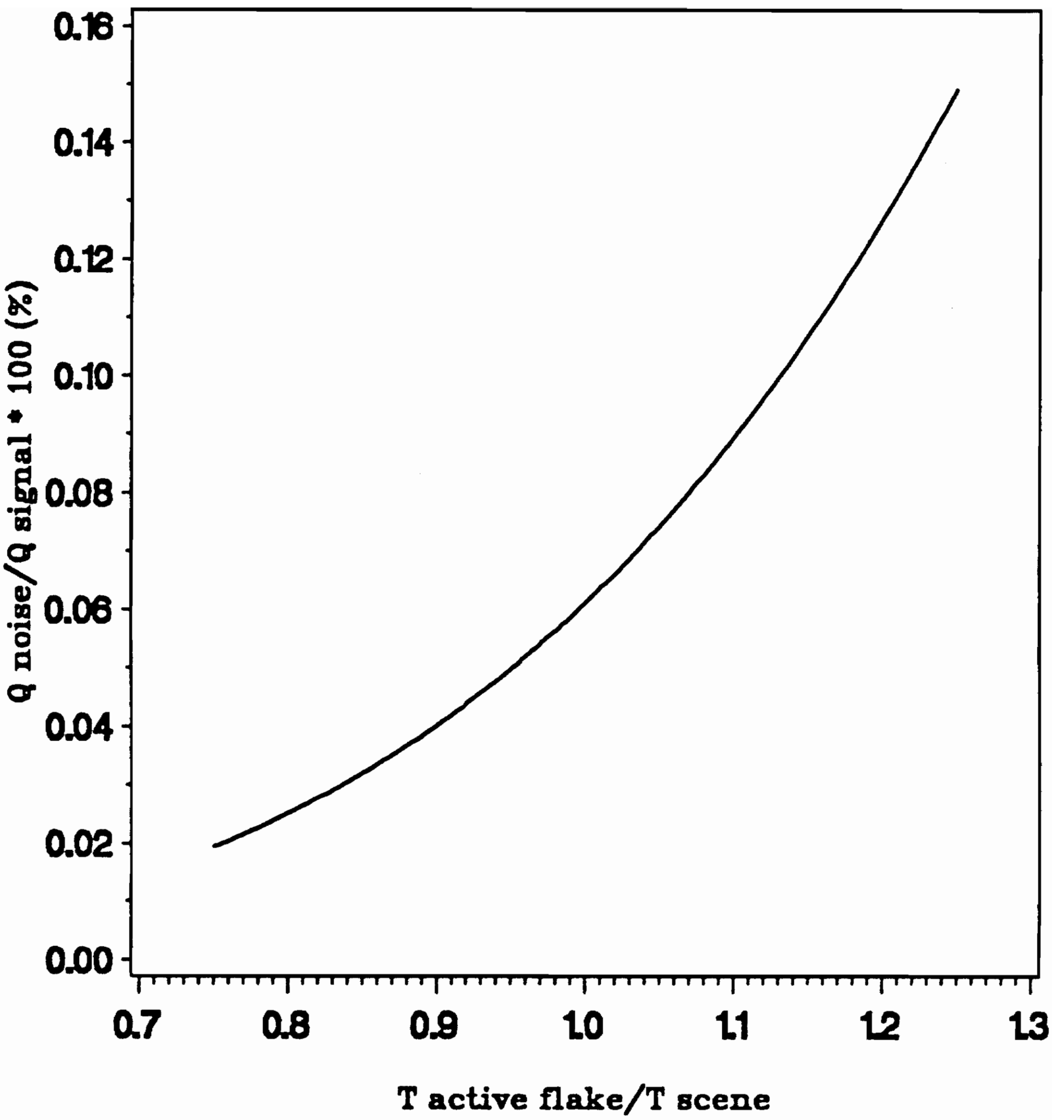


Fig. 37. Ratio of Thermal Emission from Active Flake to Signal Energy Arriving at the Active Flake (Self-Contamination).

Table 1. Approximate Total Absorptivity of Some Thermal Control Coatings and Finishings Used on the ERBE Scanning Radiometer Optics Module.

Thermal Control Coating	α
Chemglaze™	.90
Copper Black	.90
Electrodeposited Nickel	.30
Polished Aluminum	.15
3M™ Black Velvet	.90

Table 2. Fraction of Diffuse-Gray Energy Incident to the Total Channel Aperture That Arrives at the Active Flake.

Total number of energy bundles emitted	Percent energy arriving at active flake		
	See below[1] (%)	$\alpha = 1.0$ (%)	Δ%[2]
10,000	2.56	2.50	0.06
50,000	2.51	2.51	0.00
100,000	2.54	2.50	0.04

[1] Results in this column were obtained using the optical properties given in Table 1.
[2] Δ% = the difference between the values in the second and third columns.

Table 3. Fraction of Energy Arriving at Active Flake from Two Diffuse Sources at Different Temperatures.

Percent energy bundles arriving at flake			
channel	No. of energy bundles emitted	5800 K source Percent at flake	300 K source Percent at flake
SW LW SW+LW Total	10,000	1.67 0.012 1.682 2.15	0.008 1.44 1.448 2.46
SW LW SW+LW Total	50,000	1.61 0.008 1.618 2.06	0.008 1.49 1.498 2.54
SW LW SW+LW Total	100,000	1.66 0.007 1.667 2.00	0.008 1.52 1.528 2.55

Table 4. Fraction of Diffuse Energy Incident to the Total Channel Aperture That Arrives at the Active Flake (spectral analysis, 5800 K).

Total number of energy bundles emitted	Percent energy arriving at active flake		
	See below[1] (%)	$\alpha = 1.0$ (%)	$\Delta\%$[2]
10,000	2.15	1.93	0.22
50,000	2.06	1.89	0.17
100,000	2.00	1.89	0.11

Table 5. Fraction of Diffuse Energy Incident to the Total Channel Aperture That Arrives at the Active Flake (spectral analysis, 300 K).

Total number of energy bundles emitted	Percent energy arriving at active flake		
	See below[1] (%)	$\alpha = 1.0$ (%)	$\Delta\%$[2]
10,000	2.46	2.37	0.09
50,000	2.54	2.46	0.08
100,000	2.55	2.50	0.05

[1] Results in this column were obtained using the optical properties given in Table 1.
[2] $\Delta\%$ = the difference between the values in the second and third columns.

Table 6. Thermal Emission (Noise) from the ERBE Telescope to the Active Flake.

Telescope Component	Q_{noise} (mW)[1]	Noise (%)[2]
Detector Housing	0.109	0.602
Secondary Mirror Mount	0.085	0.470
Reflector Cap	0.069	0.381
Field Stop	0.066	0.365
Detector Header	0.030	0.166
Active Flake (Self Contamination)	0.024	0.133
Primary Insert	0.014	0.077
Pivoted Baffle	0.002	0.011
Total Noise Contribution	0.399	2.205

[1] with the component at 311 K.

[2] $\text{Noise} = \dfrac{Q_{noise}}{Q_{signal}} \times 100\ (\%)$

where Q_{signal} is the amount of signal energy that reaches the active flake (18.1 mW).

5.0 Conclusions and Recommendations

5.1 Conclusions

Based on the results presented in this thesis, it may be concluded that:

1. The optics module design successfully limits the amount of radiation entering through the aperture that reaches the active flake along nonoptical paths.

2. Thermal emissions from the scanning radiometer assembly to the active flake may attain 2.2 percent of the incoming signal, based on a 255 K collimated source of blackbody radiation.

3. The field stop aperture has little effect in limiting collimated radiation incident to the instrument aperture.

4. A wavelength-dependent model should be used for all optical analyses.

5. The effects of ultraviolet radiation on the mirrors of the total and shortwave channels and on the Suprasil filters of the shortwave channel should not be ignored. These effects could explain any observed degradation in the optical performance of these scanning channels.

5.2 Recommendations

1. This model should facilitate the CERES scanning radiometer design.

2. The radiative model presented in this thesis should be combined with a dynamic conduction and electrical model to provide a complete instrument dynamic simulation model.

3. The complete dynamic electrothermal model should be used to verify that the instrument data sampling rate and scan rate produce the desired overlap in successive scans.

4. The complete dynamic electrothermal model should be used to determine how the scanning radiometer responds to different types of radiant sources, that is, responds to Earth scenes versus calibration sources.

References

1. Hunt, G. E., R. Kandel, and A. T. Mecherikunnel, *A History of Presatellite Investigations of the Earth's Radiation Budget*, **Reviews of Geophysics**, Vol. 24, No. 2, May 1986, pp. 351-356.

2. House, F. B., A. Gruber, G. E. Hunt, and A. T. Mecherikunnel, *History of Satellite Missions and Measurements of the Earth Radiation Budget (1957-1984)*, **Reviews of Geophysics**, Vol. 24, No. 2, May 1986, pp. 357-377.

3. Geist, J., *Trends in the Development of Radiometry*, **Optical Engineering**, Vol. 15, No. 6, 1976, pp 537-539.

4. Harrison, E. F., D. R. Brooks, and G. G. Gibson, *Mission Analysis to Define Satellite Orbits for Earth Radiation Budget Measurements*, AIAA Paper No. 76-811, August 1976.

5. Barkstrom, B. R., and G. L. Smith, *The Earth Radiation Budget Experiment: Science and Implementation*, **Reviews of Geophysics**, Vol. 24, No. 2, May 1986, pp. 379-390.

6. Luther, M. R., J. E. Cooper, and G. R. Taylor, *The Earth Radiation Budget Experiment Nonscanner Instrument*, **Reviews of Geophysics**, Vol. 24, No. 2, May 1986, pp. 391-396.

7. Mahan, J. R., N. E. Tira, R. B. Lee, III, and R. J. Keynton, *Comparison of the Measured and Predicted Response of the Earth Radiation Budget Experiment Active Cavity Radiometer During Solar Observations*, **Applied Optics**, Vol. 28, No. 7, April 1989, pp. 1327-1337.

8. Tira, N. E., J. R. Mahan, and R. B. Lee, III, *Dynamic Electrothermal Model for the ERBE nonscanning radiometer with application to solar observation and thermal noise*, **Optical Engineering**, Vol. 29, No. 4, April 1990, pp. 351-358.

9. Barkstrom, B. R., *The Earth Radiation Budget Experiment (ERBE)*, **Bulletin of the American Meteorological Society**, Vol. 65, No. 11, Nov. 1984, pp. 1170-1185.

10. Barkstrom, B. R., and J. B. Hall, Jr., *The Earth Radiation Budget Experiment (ERBE): An Overview*, **Journal of Energy**, Vol. 6, No. 2, 1982, pp. 141-146.

11. Lee, III, R. B., M. A. Gibson, S. Thomas, J. R. Mahan, J. L. Meekins, and N. E. Tira, *Earth Radiation Budget Experiment (ERBE) Scanner Radiometric Calibration Results*, **Proceedings of SPIE Technical Symposium on Aerospace Sensing**, Orlando, FL, Apr. 16-20, 1990.

12. Ramanathan, V., R. D. Cess, E. F. Harrison, P. Minnis, B. R. Barkstrom, E. Ahmad, and D. Hartmann, *Cloud-radiative forcing and climate: results from the Earth Radiation Budget Experiment*, **Science**, Vol. 243, No. 1, Jan. 1989, pp. 57-63.

13. Smith, G. L., R. N. Green, E. Raschke, L. M. Avis, J. T. Suttles, B. A. Wielicki, and R. Davies, *Inversion Methods for Satellite Studies of the Earth's Radiation Budget: Development of Algorithms of the ERBE Mission*, **Reviews of Geophysics**, Vol. 24, No. 2, May 1986, pp. 407-421.

14. Kopia, L. P., *The Earth Radiation Budget Experiment Scanner Instrument*, **Reviews of Geophysics**, Vol. 24, No. 2, May 1986, pp. 397-406.

15. Halyo, N., D. K. Pandey, and D. B. Taylor, *Modeling and Characterization of the Earth Radiation Budget Experiment (ERBE) Nonscanner and Scanner Sensors*, NASA CR-181818, March 1989.

16. Sparrow, E. M., and V. K. Jonsson, *Absorption of Thermal Radiation in V-Groove Cavities*, **International Journal of Heat and Mass Transfer**, Vol. 5, 1962, pp. 1111-1115.

17. S. P. T., *The Bolometer*, **Nature**, Vol. 25, No. 1, Nov. 3, 1881, pp. 14-16.

18. Brattain, W. H., and J. A. Becker, *Thermistor Bolometers*, **Journal of the Optical Society of America**, Vol. 36, 1946, p. 354.

19. Jones, R. C., *The General Theory of Bolometer Performance*, **Journal of the Optical Society of America**, Vol. 43, No. 1, Jan. 1953, pp. 1-14.

20. Wormser, E. M., *Properties of Thermistor Infrared Detectors*, **Journal of the Optical Society of America**, Vol. 43, No. 1, Jan. 1953, pp. 15-21.

21. Huck, F. O., N. Halyo, and S. K. Park, *Aliasing and Blurring in 2-D Sampled Imagery*, **Applied Optics**, Vol. 19, No. 13, pp. 2174-2180.

22. Boyd, R. W., **Radiometry and the Detection of Optical Radiation**, John Wiley & Sons, Inc., New York, 1983.

23. Wyatt, Clair L., **Radiometric System Design**, MacMillan Publishing Co., New York, 1987.

24. Sparrow, E. M., E. R. G. Eckert, and V. K. Jonsson, *An Enclosure Theory for Radiative Exchange Between Specularly and Diffusely Reflecting Surfaces*, **Journal of Heat Transfer**, Vol. 84, 1962, pp. 294-300.

25. Sarofim, A. F., and H. C. Hottel, *Radiative Exchange Among Non-Lambert Surfaces*, **Journal of Heat Transfer**, Vol. C88, 1966, pp. 37-44.

26. Siegel, R., and J. R. Howell, **Thermal Radiation Heat Transfer**, Second Edition, McGraw-Hill, New York, 1987.

27. Sparrow, E. M., and R. D. Cess, **Radiation Heat Transfer**, Augmented Edition, McGraw-Hill, New York, 1987.

28. Mahan, J. R., and L. D. Eskin, *The Radiation Distribution Factor--Its Calculation Using Monte Carlo and an Example of its Application*, **First UK National Heat Transfer Conference**, July 4-6, 1984, Leeds, Yorkshire, England, pp. 1001-1012.

29. Gebhart, B., **Heat Transfer**, Second Edition, McGraw-Hill, New York, 1971, pp. 150-158.

30. Toor, J. S., and R. Viskanta, *A Numerical Experiment of Radiant Heat Interchange by the Monte Carlo Method*, **International Journal of Heat and Mass Transfer**, Vol. 11, May 1968, pp. 883-897.

31. Corlett, R. C., *Direct Monte Carlo Calculation of Radiative Heat Transfer in a Vacuum*, **Journal of Heat Transfer**, Vol. 88, Nov. 1966, pp. 379-390.

32. Howell, J. R., *Application of Monte Carlo to Heat Transfer Problems*, **Advances in Heat Transfer**, Vol. 5, Academic Press, New York, 1968, pp. 1-54.

33. Anon., International Statistical and Mathematical Library (ISML) Users' Manual, Vol. 3, April 1987, pp. 947-951.

34. Eskin, L. D., **Application of the Monte Carlo Method to the Transient Thermal Modeling of a Diffuse-Specular Radiometer Cavity**, M. S. Thesis, Mechanical Engineering Department, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1981.

35. Tira, N. E., **Dynamic Simulation of Solar Calibration of the Total, Earth-Viewing Channel of the Earth Radiation Budget Experiment (ERBE)**, M. S. Thesis, Mechanical Engineering Department, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1987.

36. Chou, T. S., *A Monte Carlo Approach to Optical Analysis*, **Optical Engineering**, Vol. 13, No. 4, pp. 299-302, 1974.

37. Brown, S., **All About Telescopes**, Second Edition, Edmund Scientific Co., New Jersey, 1975.

38. Page, T., and L. Williams, eds., **Telescopes: How to Make and Use Them,** Macmillan Publishing Company, New York, 1966, pp. 33-35, 237-242.

39. Lee, III, R. B., B. R. Barkstrom, N. Halyo, M. A. Gibson, and L. M. Avis, *Characterizations of the Earth Radiation Budget Experiment (ERBE) Scanning Radiometers,* **Proceedings of SPIE Technical Symposium on Aerospace Sensing,** Orlando, FL, Apr. 17-21, 1989.

40. Ramanathan, V., **Second CERES Science Team Meeting, NASA Langley Research Center,** Hampton, VA, Nov. 6-8, 1989.

41. Henninger, J. H., *Solar Absorptance and Thermal Emittance of Some Common Spacecraft Thermal-Control Coatings,* **NASA Reference Publication 1121,** 1984.

Appendix A. Computer Program Listing

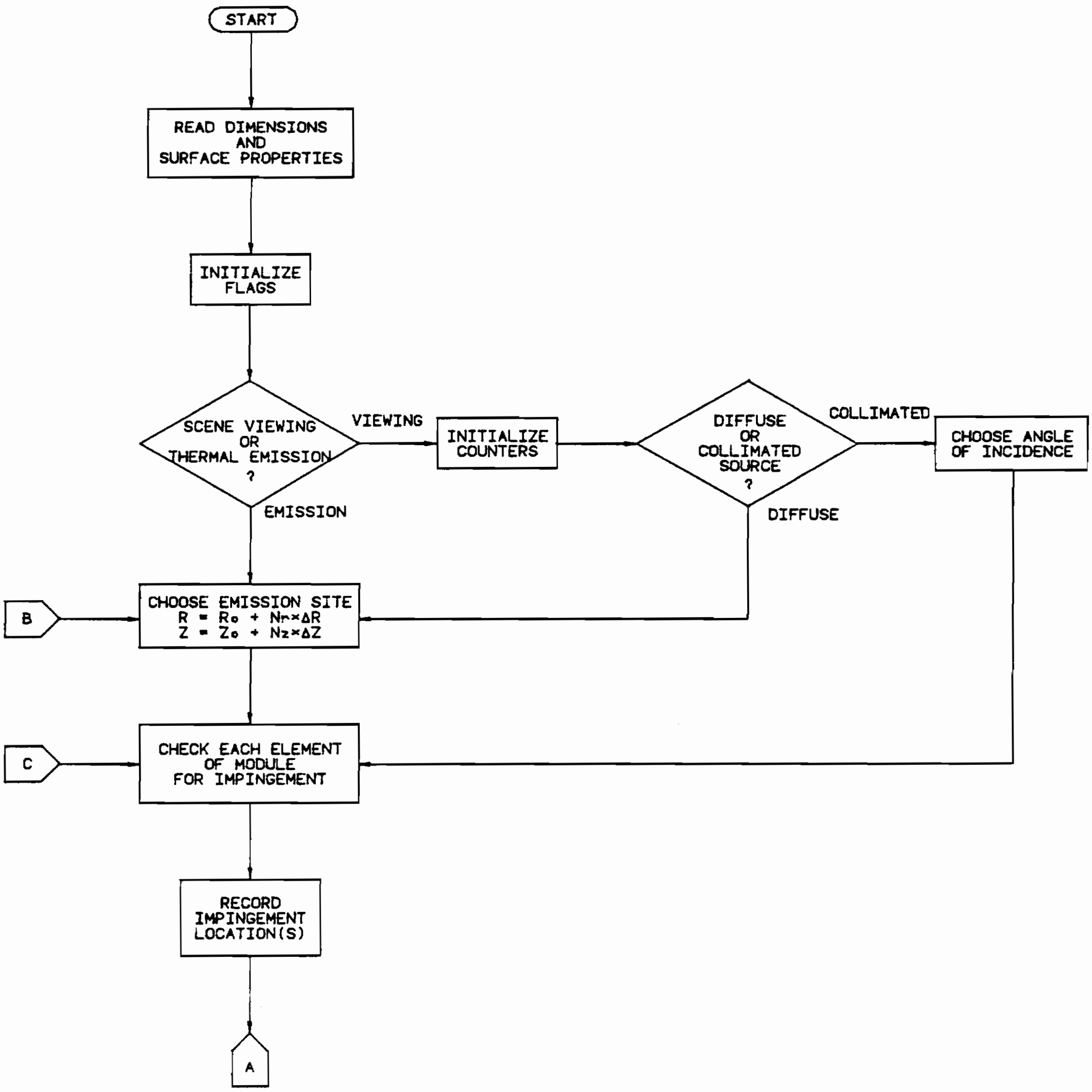


Fig. A. 1. Flowchart of Monte Carlo implementation for calculating D_{ij}.

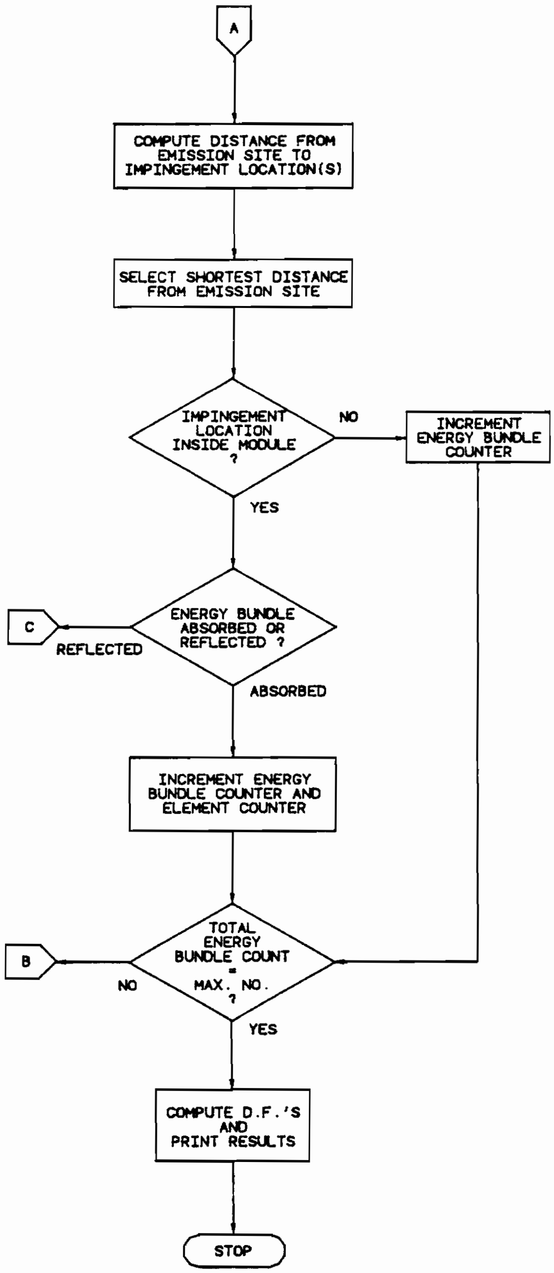


Fig. A. 2. Flowchart of Monte Carlo implementation for calculating D_{ij} (cont.).

```
****************************************************************
*                                                              *
*      All programs and subprograms written and debugged by:   *
*                                                              *
*                      Jeffrey L. Meekins                      *
*                                                              *
*                              at                              *
*                                                              *
*        Virginia Polytechnic Institute & State University     *
*                Mechanical Engineering Department             *
*                   Blacksburg, Virginia 24061                 *
*                         August, 1990                         *
*                                                              *
****************************************************************
*--------------------------------------------------------------*
****************************************************************
*                                                              *
*                          PROGRAM DFS                         *
*                                                              *
* This program calculates the distribution factors for the     *
* ERBE scanning thermistor bolometer radiometer.  The number of *
* energy bundles emitted from a particular part within the optics *
* module is based on the ratio of the surface area of that part *
* to the surface area of the smallest part in the optics module. *
*                                                              *
****************************************************************
*
      PROGRAM DFS
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Specify size and type of storage parameters.
*
      REAL*8 KAPPA1,KAPPA2,DF(40,40),AREA(37),APEX(8),DFC(40)
      REAL*8 XN(20),YN(20),ZN(20),X1N(20),Y1N(20),Z1N(20)
*
* Place all common variables in one, unique storage block.
*
      COMMON /GEOM/ H1,D1,D2,THETA1,C1,C2,C3,C4,C5,H2,D3,D4,D5
      COMMON /GEOM/ DEL1,DEL2,THETA2,PHI,H3,HL1,HL2,D6,THETA3
      COMMON /GEOM/ H4,HCURVE,D7,KAPPA1,H5,D8,KAPPA2,THETA4,H6,D9
      COMMON /GEOM/ THETA5,DIM1,DIM2,HREF,DREF
      COMMON /GEOM1/ H7,D10,FLAKDIM,ABS11,REFR11

      COMMON /DFSAREA/ AREA
      COMMON /RAND/ IN
      COMMON /COORDS/ XN,YN,ZN,X1N,Y1N,Z1N
      COMMON /DISTR/ DF,II
      COMMON /CHANNEL/ ALAMBDA,NCHANEL,NFLTRSN,NSPCTRM
      COMMON /SPEC/ NUMHITS,MIRTYPE
*
* Initialize necessary data.
*
      IN = 201
      MSHOTS = 5000
      NSPCTRM = 0
      MIRTYPE = 0
      PI = DACOS(-1.0D0)

      DO 111 I=1,40
      DO 112 J=1,40
  112   DF(I,J) = 0.0D0
        DFC(I) = 0.0D0
  111 CONTINUE
*
```

```
* Initialize arrays used to store the coordinates of energy bundles
* impinging a surface within the optics module for use in subprogram
* COORD.
*
      DO 2 J=1,20
        XN(J) = 0.0D0
        YN(J) = 0.0D0
        ZN(J) = 0.0D0
        X1N(J) = 0.0D0
        Y1N(J) = 0.0D0
        Z1N(J) = 0.0D0
    2 CONTINUE
*
* Call subroutines to initialize necessary variables.
*
      CALL INPUT
      CALL EQAREA
      CALL RNSET (5520470)
*
* Choose the part of the optics module with the smallest area.
* This will be used to determine the area ratio needed to
* compensate for the number of energy bundles emitted from each
* part of the optics module.
*
      SMLAREA = AREA(1)
      DO 3 I=1,37
    3   IF(AREA(I).LT.SMLAREA) SMLAREA = AREA(I)
*
******************** Reflector cap ******************
*
* Initialize data for the equation of a cone.
*
      ALPHA2 = THETA1*PI/180.0D0
      ALPHA1 = PI/2.0D0 - ALPHA2

      V1 = DTAN(ALPHA1)*DTAN(ALPHA1)
      V2 = DTAN(ALPHA2)*DTAN(ALPHA2)
*
* Compute vertices of cones of reflector cap.
*
      DO 5 I=1,4
        APEX(I) = (D2/2.0D0)/DTAN(ALPHA1) + HREF
    5   APEX(I+4) = (-D2/2.0D0)/DTAN(ALPHA2) + HREF

      APEX(1) = APEX(1) + C1
      APEX(2) = APEX(2) + C2
      APEX(3) = APEX(3) + C3
      APEX(4) = APEX(4) + C4
      APEX(5) = APEX(5) + C1
      APEX(6) = APEX(6) + C2
      APEX(7) = APEX(7) + C3
      APEX(8) = APEX(8) + C4
*
* The DO 15 loop randomly finds points of emission on the reflector
* cap.
*
* Compute the number of shots necessary for accurate distribution
* factors.
*
      AREASUM = 0.D0
      RB = D2/2.D0

      DO 10 I=1,8
   10   AREASUM = AREASUM + AREA(I)

      NSHOTS = INT(MSHOTS * AREASUM/SMLAREA)
*
```

```
* Compute height and angular coordinates of emission site randomly.
*
      DO 15 J=1,NSHOTS
        CALL RANDOM (R)
        Z1 = R*C5
*
        DO 16 I=1,8
          IF(I.EQ.4) THEN
            RA = D4/2.D0
          ELSE
            RA = D1/2.D0
          ENDIF

          IF(I.GE.5) THEN
            RADIUS = DTAN(ALPHA2)*(Z1-APEX(I))
          ELSE
            RADIUS = DTAN(ALPHA1)*(APEX(I)-Z1)
          ENDIF

          IF(RADIUS.GE.RA.AND.RADIUS.LE.RB) GO TO 17
  16    CONTINUE
  17    CONTINUE
        II = I
        DFC(I) = DFC(I) + 1.D0

        CALL RANDOM (R)
        THETA = 2.0D0 * PI * R
*
* Compute the x and y coordinates corresponding to the radius
* selected.
*
        X1 = RADIUS*DCOS(THETA)
        Y1 = RADIUS*DSIN(THETA)
*
* Find corresponding unit normal to the surface at that point.
*
        IF(I.GE.5) THEN
          UN = DSQRT(X1*X1 + Y1*Y1 + V2*V2*(Z1-APEX(I))*(Z1-APEX(I)))
          UNX = X1/UN
          UNY = Y1/UN
          UNZ = -V2*(Z1-APEX(I))/UN
        ELSE
          UN = DSQRT(X1*X1 + Y1*Y1 + V1*V1*(Z1-APEX(I))*(Z1-APEX(I)))
          UNX = X1/UN
          UNY = Y1/UN
          UNZ = -V1*(Z1-APEX(I))/UN
        ENDIF
*
* Find energy bundle direction.
*
        CALL VECTOR (X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
        CALL OPTICS (X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)

   15 CONTINUE
*****************************************************************
* The rest of this program is based on the logic outlined in    *
* the comment statements above for each part of the optics module. *
*****************************************************************
*
********* Cylindrical surface of reflector cap *********
*
      NSHOTS = INT(MSHOTS*AREA(9)/SMLAREA)
      DFC(9) = DFLOAT(NSHOTS)
      II = 9

      DO 20 I=1,NSHOTS
        CALL RANDOM (R)
```

```
        THETA = 2.0D0*PI*R
        CALL RANDOM (R)
        Z1 = HREF+C5 + R*(H1-C5)

        X1 = D4/2.0D0 * DCOS(THETA)
        Y1 = D4/2.0D0 * DSIN(THETA)

        UNX = DCOS(THETA)
        UNY = DSIN(THETA)
        UNZ = 0.0

        CALL VECTOR (X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
        CALL OPTICS(X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
   20 CONTINUE
*
************** Hub of secondary mirror mount (top) ****************
*
      NSHOTS = INT(MSHOTS * AREA(10)/SMLAREA)
      DFC(10) = DFLOAT(NSHOTS)
      II = 10

      UNX = 0.0D0
      UNY = 0.0D0
      UNZ = 1.0D0

      DO 25 I=1,NSHOTS
        CALL RANDOM (R)
        RADIUS = D3/2.0D0 * DSQRT(R)
        CALL RANDOM (R)
        THETA = 2.0D0 * PI * R
        Z1 = HREF + H1

        X1 = RADIUS * DCOS(THETA)
        Y1 = RADIUS * DSIN(THETA)

        CALL VECTOR(X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
        CALL OPTICS(X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
   25 CONTINUE
*
*************** Spider legs (top) **************
*
      ZA = HREF + H1
      ZB = HREF + H1 + DEL2
      RADII = ((99.5D0/100.D0)*D4 - D3)/2.0D0
      PSI = PHI * PI/180.0D0
      N = NINT(2.0D0*PI/PSI)
      NSHOTS = INT(MSHOTS * AREA(11)/SMLAREA)

      DO 30 I=0,N-1
        II = I + 11
        DFC(II) = DFLOAT(NSHOTS)
        ANGLE = DFLOAT(I)*PSI
        XA = D3/2.0D0 * DCOS(ANGLE)
        XB = D4/2.0D0 * DCOS(ANGLE)
        YA = D3/2.0D0 * DSIN(ANGLE)
        YB = D4/2.0D0 * DSIN(ANGLE)
        IF(ANGLE.EQ.90.D0) THEN
          YC = YA
          YD = YB
        ELSE
          YC = YA + (DEL1/2.0D0)/DABS(DCOS(ANGLE))
          YD = YB - (DEL1/2.0D0)/DABS(DCOS(ANGLE))
        ENDIF

        COEF1 = (YB-YA) * (ZB-ZA)
        COEF2 = -(XB-XA) * (ZB-ZA)
        COEF3 = (XB-XA) * (YC-YA)
```

```
        COEF4 = COEF1
        COEF5 = COEF2
        COEF6 = (XB-XA)*(YD-YA) - (XB-XA)*(YB-YA)

      DO 35 J=1,NSHOTS
        CALL RANDOM (R)
        XPRIME = D3/2.0D0 + R*RADII
        CALL RANDOM (R)
        YPRIME = -DEL1/2.0D0 + R * DEL1
*
* When finding the unit normal to the legs of the spider, be sure
* to rotate the unit Y vector along with the axis rotation.
*
        X1 = XPRIME*DCOS(ANGLE) - YPRIME*DSIN(ANGLE)
        Y1 = YPRIME*DCOS(ANGLE) + XPRIME*DSIN(ANGLE)

        IF(DCOS(ANGLE).GE.0.0D0) THEN
        IF(YPRIME.GE.0.0D0) THEN
          Z1 = (COEF1*(XA-X1) + COEF2*(YA-Y1))/COEF3 + ZA
          UNX = COEF1/DSQRT(COEF1**2 + COEF2**2 + COEF3**2)
          UNY = COEF2/DSQRT(COEF1**2 + COEF2**2 + COEF3**2)
          UNZ = COEF3/DSQRT(COEF1**2 + COEF2**2 + COEF3**2)
        ELSE
          Z1 = (COEF1*(XA-X1) + COEF2*(YA-Y1))/COEF6 + ZA
          UNX = COEF1/DSQRT(COEF1**2 + COEF2**2 + COEF6**2)
          UNY = -COEF2/DSQRT(COEF1**2 + COEF2**2 + COEF6**2)
          UNZ = -COEF6/DSQRT(COEF1**2 + COEF2**2 + COEF6**2)
        ENDIF
        ENDIF

        IF(DCOS(ANGLE).LT.0.0D0) THEN
        IF(YPRIME.LE.0.0D0) THEN
          Z1 =  (COEF1*(XA-X1) + COEF2*(YA-Y1))/COEF3 + ZA
          UNX = -COEF1/DSQRT(COEF1**2 + COEF2**2 + COEF3**2)
          UNY = -COEF2/DSQRT(COEF1**2 + COEF2**2 + COEF3**2)
          UNZ = -COEF3/DSQRT(COEF1**2 + COEF2**2 + COEF3**2)
        ELSE
          Z1 =  (COEF1*(XA-X1) + COEF2*(YA-Y1))/COEF6 + ZA
          UNX = COEF1/DSQRT(COEF1**2 + COEF2**2 + COEF6**2)
          UNY = COEF2/DSQRT(COEF1**2 + COEF2**2 + COEF6**2)
          UNZ = COEF6/DSQRT(COEF1**2 + COEF2**2 + COEF6**2)
        ENDIF
        ENDIF

        CALL VECTOR(X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
        CALL OPTICS(X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
   35 CONTINUE
   30 CONTINUE
*
************ Spider legs (bottom) **************
*
      NSHOTS = INT(MSHOTS * AREA(14)/SMLAREA)
      UNX = 0.0D0
      UNY = 0.0D0
      UNZ = -1.0D0

      DO 40 I=0,N-1
        II = I + 14
        DFC(II) = DFLOAT(NSHOTS)
        ANGLE = DFLOAT(I)*PSI
      DO 45 J=1,NSHOTS
        CALL RANDOM (R)
        XPRIME = D3/2.0D0 + R*RADII
        CALL RANDOM (R)
        YPRIME = -DEL1/2.0D0 + R * DEL1
```

```
        X1 = XPRIME*DCOS(ANGLE) - YPRIME*DSIN(ANGLE)
        Y1 = YPRIME*DCOS(ANGLE) + XPRIME*DSIN(ANGLE)
        Z1 = ZB

        CALL VECTOR(X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
        CALL OPTICS(X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
   45 CONTINUE
   40 CONTINUE
*
********** Perimeter of hub of secondary mirror mount ************
*
      NSHOTS = INT(MSHOTS * AREA(17)/SMLAREA)
      II = 17
      DFC(17) = DFLOAT(NSHOTS)
      UNZ = 0.0D0
      RADIUS = D3/2.0D0
      DO 50 I=1,NSHOTS
   51   CONTINUE
        CALL RANDOM (R)
        THETA = 2.0D0 * PI * R
        CALL RANDOM (R)
        Z1 = HREF + H1 + R*(H2 - H5)

        X1 = RADIUS*DCOS(THETA)
        Y1 = RADIUS*DSIN(THETA)
*
* Cannot emit where spoke of secondary mirror mount is attached to its
* hub.
*
        ZTEST = Z1 - HREF - H1
        IF(ZTEST.GE.0.D0.AND.ZTEST.LE.DEL1) THEN
          YTEST =  DEL2 * ZTEST / (DEL1* 2.D0)
          DO 52 J=0,N-1
            ANGLE = DFLOAT(J) * PSI
            YPRIME = Y1*DCOS(ANGLE) - X1*DSIN(ANGLE)
            IF(DABS(YPRIME).LE.YTEST) GO TO 51
   52     CONTINUE
        ENDIF

        UNX = -X1/RADIUS
        UNY = -Y1/RADIUS
        CALL VECTOR(X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
        CALL OPTICS(X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
   50 CONTINUE
*
************ Perimeter of secondary mirror ************
*
      ALPHA = THETA4 * PI/180.0D0
      VERTEX = (D8/2.0D0)/DTAN(ALPHA)
      HH = (-D8*D8/4.D0)/(2.D0*KAPPA2)
      CAPEX = HH - VERTEX
      HMIRROR = HH + H5
      V = DTAN(ALPHA)*DTAN(ALPHA)
      RAD = (HREF + H1 + H2 - H5 - CAPEX) * DTAN(ALPHA) - D3/2.D0

      NSHOTS = INT(MSHOTS * AREA(18)/SMLAREA)
      II = 18
      DFC(18) = DFLOAT(NSHOTS)

      KSHOTS = NINT(0.9 * DFLOAT(NSHOTS))

      DO 55 I=1,NSHOTS
        IF(I.LT.KSHOTS) THEN
          CALL RANDOM (R)
          Z1 = HREF + H1 + H2 - H5 + R*HMIRROR
          CALL RANDOM (R)
          THETA = 2.0D0 * PI * R
```

```
          RADIUS = (Z1-CAPEX)*DTAN(ALPHA)
        ELSE
          CALL RANDOM (R)
          RADIUS = D3/2.D0 + R*RAD
          CALL RANDOM (R)
          THETA = 2.0D0 * PI * R
          Z1 = HREF + H1 + H2 - H5
        ENDIF

        X1 = RADIUS*DCOS(THETA)
        Y1 = RADIUS*DSIN(THETA)

        IF(I.LT.KSHOTS) THEN
          UN = DSQRT(X1*X1 + Y1*Y1 + V*V*(Z1-CAPEX)*(Z1-CAPEX))
          UNX = -X1/UN
          UNY = -Y1/UN
          UNZ = V*(Z1-CAPEX)/UN
        ELSE
          UNX = 0.0D0
          UNY = 0.0D0
          UNZ = 1.0D0
        ENDIF

        CALL VECTOR(X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
        CALL OPTICS(X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
   55 CONTINUE
*
*********** Perimeter of secondary mirror mount ***********
*
      ANGLE = THETA2 * PI/180.0D0
      HTAPER = ((D4-D5)/2.0D0) * DTAN(ANGLE)
      RADIUS = D4/2.D0

      NSHOTS = INT(MSHOTS * AREA(19)/SMLAREA)
      II = 19
      DFC(19) = DFLOAT(NSHOTS)

      DO 60 I=1,NSHOTS
   61   CONTINUE
        CALL RANDOM (R)
        THETA = 2.0D0 * PI * R
        CALL RANDOM (R)
        Z1 = HREF + H1 + R*(H2-HTAPER)

        X1 = RADIUS * DCOS(THETA)
        Y1 = RADIUS * DSIN(THETA)
*
* Cannot emit where spoke of secondary mirror mount is attached to its
* perimeter.
*
        ZTEST = Z1 - HREF - H1
        IF(ZTEST.GE.0.D0.AND.ZTEST.LE.DEL1) THEN
          YTEST =  DEL2 * ZTEST / (DEL1* 2.D0)
          DO 62 J=0,N-1
            ANGLE2 = DFLOAT(J) * PSI
            YPRIME = Y1*DCOS(ANGLE2) - X1*DSIN(ANGLE2)
            IF(DABS(YPRIME).LE.YTEST) GO TO 61
   62     CONTINUE
        ENDIF

        UNX = DCOS(THETA)
        UNY = DSIN(THETA)
        UNZ = 0.0D0

        CALL VECTOR(X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
        CALL OPTICS(X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
```

```
   60 CONTINUE
*
********** Conical surface of secondary mirror mount **********
*
      ALPHA = PI/2.0D0 - ANGLE
      CAPEX = (D4/2.0D0)/DTAN(ALPHA) + HREF + H1 + H2 - HTAPER
      V = DTAN(ALPHA)*DTAN(ALPHA)

      NSHOTS = INT(MSHOTS * AREA(20)/SMLAREA)
      II = 20
      DFC(20) = DFLOAT(NSHOTS)
      DO 65 I=1,NSHOTS
        CALL RANDOM (R)
        THETA = 2.0D0 * PI * R
        CALL RANDOM (R)
        Z1 = HREF + H1 + H2 - R*HTAPER

        RADIUS = DTAN(ALPHA)*(CAPEX-Z1)
        X1 = RADIUS*DCOS(THETA)
        Y1 = RADIUS*DSIN(THETA)

        UN = DSQRT(X1*X1 + Y1*Y1 + V*V*(Z1-CAPEX)*(Z1-CAPEX))
        UNX = X1/UN
        UNY = Y1/UN
        UNZ = -V*(Z1-CAPEX)/UN

        CALL VECTOR(X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
        CALL OPTICS(X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
   65 CONTINUE
*
*************** Surface of secondary mirror *************
*
      NSHOTS = INT(MSHOTS * AREA(21)/SMLAREA)
      II = 21
      DFC(21) = DFLOAT(NSHOTS)
      DELZ = ((D8*D8/4.D0)/(2.D0*KAPPA2))
      VERTEX = HREF + H1 + H2

      DO 70 I=1,NSHOTS
        CALL RANDOM (R)
        Z1 = VERTEX - R*DELZ
        CALL RANDOM (R)
        THETA = 2.0D0 * PI * R

        RADIUS = DSQRT(2.D0*(VERTEX-Z1)*KAPPA2)
        X1 = RADIUS * DCOS(THETA)
        Y1 = RADIUS * DSIN(THETA)

        UN = DSQRT(4.0D0*(X1*X1 + Y1*Y1 + KAPPA2*KAPPA2))
        UNX = -2.0D0*X1/UN
        UNY = -2.0D0*Y1/UN
        UNZ = -2.0D0*KAPPA2/UN
        CALL VECTOR(X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
        CALL OPTICS(X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
   70 CONTINUE
*
*************** Detector housing ***************
*
* Cylinder 1
*
      NSHOTS = INT(MSHOTS * AREA(22)/SMLAREA)
      II = 22
      DFC(22) = DFLOAT(NSHOTS)
      RADIUS = D6/2.D0
      DO 75 I=1,NSHOTS
        CALL RANDOM (R)
        THETA = 2.0D0 * PI * R
```

```
        CALL RANDOM (R)
        Z1 = VERTEX + R*(H3-HL1-HL2)

        X1 = RADIUS * DCOS(THETA)
        Y1 = RADIUS * DSIN(THETA)

        UNX = DCOS(THETA)
        UNY = DSIN(THETA)
        UNZ = 0.0D0

        CALL VECTOR(X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
        CALL OPTICS(X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
   75 CONTINUE
*
* Cone 1
*
      OMEGA = (THETA3/2.0D0) * PI/180.0D0
      ALPHA = PI/2.0D0 - OMEGA
      RRCONE = (HL1/2.0D0)/DTAN(OMEGA)
      RCONE = D6/2.0D0 + RRCONE
      HCONE = HREF+H1+H2+H3-HL2-HL1/2.0D0
      VERTEX = RCONE/DTAN(ALPHA)
      V = DTAN(ALPHA)*DTAN(ALPHA)
      CAPEX = HCONE - VERTEX

      NSHOTS = INT(MSHOTS * AREA(23)/SMLAREA)
      II = 23
      DFC(23) = DFLOAT(NSHOTS)
      DO 80 I=1,NSHOTS
        CALL RANDOM (R)
        Z1 = HCONE - R*HL1/2.D0
        CALL RANDOM (R)
        THETA = 2.0D0 * PI * R

        RADIUS = DTAN(ALPHA)*(Z1-CAPEX)
        X1 = RADIUS * DCOS(THETA)
        Y1 = RADIUS * DSIN(THETA)

        UN = DSQRT(X1*X1 + Y1*Y1 + V*V*(Z1-CAPEX)*(Z1-CAPEX))
        UNX = X1/UN
        UNY = Y1/UN
        UNZ = -V*(Z1-CAPEX)/UN

        CALL VECTOR(X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
        CALL OPTICS(X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
   80 CONTINUE
*
* Cone 2
*
      CAPEX = HCONE + VERTEX

      NSHOTS = INT(MSHOTS * AREA(24)/SMLAREA)
      II = 24
      DFC(24) = DFLOAT(NSHOTS)
      DO 85 I=1,NSHOTS
        CALL RANDOM (R)
        Z1 = HCONE + R*HL1/2.D0
        CALL RANDOM (R)
        THETA = 2.0D0 * PI * R

        RADIUS = DTAN(ALPHA)*(CAPEX-Z1)
        X1 = RADIUS * DCOS(THETA)
        Y1 = RADIUS * DSIN(THETA)

        UN = DSQRT(X1*X1 + Y1*Y1 + V*V*(Z1-CAPEX)*(Z1-CAPEX))
        UNX = X1/UN
        UNY = Y1/UN
```

```
          UNZ = -V*(Z1-CAPEX)/UN

          CALL VECTOR(X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
          CALL OPTICS(X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
   85 CONTINUE
*
* Cylinder 2
*
        NSHOTS = INT(MSHOTS * AREA(25)/SMLAREA)
        II = 25
        DFC(25) = DFLOAT(NSHOTS)
        RADIUS = D6/2.D0
        DO 90 I=1,NSHOTS
          CALL RANDOM (R)
          THETA = 2.0D0 * PI * R
          CALL RANDOM (R)
          Z1 = HREF + H1 + H2 + H3 - R*HL2

          X1 = RADIUS * DCOS(THETA)
          Y1 = RADIUS * DSIN(THETA)

          UNX = DCOS(THETA)
          UNY = DSIN(THETA)
          UNZ = 0.0D0

          CALL VECTOR(X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
          CALL OPTICS(X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
   90 CONTINUE
*
*************** Surface of primary mirror *************
*
        NSHOTS = INT(MSHOTS * AREA(26)/SMLAREA)
        II = 26
        DFC(26) = DFLOAT(NSHOTS)
        RPRABLA = DSQRT(HCURVE*2.D0*KAPPA1)
        DELZ = ((D7*D7/4.D0)/(2.D0*KAPPA1))
        DELZ = HCURVE - DELZ
        HA = HREF + H1 + H2 + H3
        VERTEX = HA + HCURVE

        DO 95 I=1,NSHOTS
          CALL RANDOM (R)
          Z1 = HA + R*DELZ
          CALL RANDOM (R)
          THETA = 2.0D0 * PI * R

          RADIUS = DSQRT(2.D0*(VERTEX-Z1)*KAPPA1)
          X1 = RADIUS * DCOS(THETA)
          Y1 = RADIUS * DSIN(THETA)

          UN = DSQRT(4.0D0*(X1*X1 + Y1*Y1 + KAPPA1*KAPPA1))
          UNX = 2.0D0*X1/UN
          UNY = 2.0D0*Y1/UN
          UNZ = 2.0D0*KAPPA1/UN

          CALL VECTOR(X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
          CALL OPTICS(X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
   95 CONTINUE
*
****************** Primary insert ****************
*
        ALPHA = PI/2.0D0 - THETA5*PI/180.0D0
        HB = ((D7-D9)/2.D0)*(DTAN(THETA5*PI/180.D0))
        CAPEX = (D7/2.0D0)/DTAN(ALPHA) + HREF+H1+H2+H3+H4+HCURVE-H6
        V = DTAN(ALPHA)*DTAN(ALPHA)

        NSHOTS = INT(MSHOTS * AREA(27)/SMLAREA)
```

```
      II = 27
      DFC(27) = DFLOAT(NSHOTS)

      DO 100 I=1,NSHOTS
        CALL RANDOM (R)
        THETA = 2.0D0 * PI * R
        CALL RANDOM (R)
        Z1 = HREF+H1+H2+H3+H4+HCURVE-H6 + R*HB

        RADIUS = DTAN(ALPHA)*(CAPEX-Z1)
        X1 = RADIUS*DCOS(THETA)
        Y1 = RADIUS*DSIN(THETA)

        UN = DSQRT(X1*X1 + Y1*Y1 + V*V*(Z1-CAPEX)*(Z1-CAPEX))
        UNX = X1/UN
        UNY = Y1/UN
        UNZ = -V*(Z1-CAPEX)/UN

        CALL VECTOR(X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
        CALL OPTICS(X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
  100 CONTINUE
*
*************** Field stop *******************
*
      ANGLE1 = PI/2.0D0 - DATAN((DIM1-DIM2)/DIM2)

      DO 110 I=1,2
        II = 27 + I
        NSHOTS = 2 * INT(MSHOTS * AREA(II)/SMLAREA)
        UNX = 0.0D0
        UNY = 0.0D0
      DO 115 J=1,NSHOTS
        CALL RANDOM (R)
        THETA = 2.0D0 * PI * R

        IF(THETA.GE.0.0D0.AND.THETA.LE.ANGLE1.OR.THETA.GE.PI.AND.
     &     THETA.LE.ANGLE1+PI) THEN
          X = DIM1 / (2.0D0*(1.0D0 + DTAN(THETA)))
          Y = DTAN(THETA) * X
          RADMIN = DSQRT(X*X + Y*Y)
          GO TO 1
        ENDIF

        IF(THETA.GE.PI-ANGLE1.AND.THETA.LE.PI.OR.THETA.GE.
     &     2.D0*PI-ANGLE1.AND.THETA.LE.2.D0*PI) THEN
          X = DIM1 / (2.0D0*(DTAN(THETA) - 1.0D0))
          Y = DTAN(THETA) * X
          RADMIN = DSQRT(X*X + Y*Y)
          GO TO 1
        ENDIF

        IF(THETA.GE.ANGLE1.AND.THETA.LE.PI-ANGLE1.OR.THETA.GE.
     &     PI+ANGLE1.AND.THETA.LE.2.D0*PI-ANGLE1) THEN
          Y = DIM2/2.0D0

          IF(THETA.EQ.PI/2.0D0) THEN
            RADMIN = Y
            GO TO 1
          ELSE
            X = Y/DTAN(THETA)
            RADMIN = DSQRT(X*X + Y*Y)
            GO TO 1
          ENDIF
        ENDIF

    1   CALL RANDOM (R)
        IF(I.EQ.1) THEN
```

```
          RADIUS = DSQRT(R) * D9/2.D0
          IF(RADIUS.LT.RADMIN) GO TO 115
          DFC(28) = DFC(28) + 1.D0
          UNZ = 1.0D0
        ELSE
          RADIUS = DSQRT(R) * D10/2.D0
          IF(RADIUS.LT.RADMIN) GO TO 115
          DFC(29) = DFC(29) + 1.D0
          UNZ = -1.0D0
        ENDIF

        X1 = RADIUS * DCOS(THETA)
        Y1 = RADIUS * DSIN(THETA)
        Z1 = HREF+H1+H2+H3+H4+HCURVE

        CALL VECTOR(X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
        CALL OPTICS(X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
  115 CONTINUE
  110 CONTINUE
*
******************* Active Flake *******************
*
      NSHOTS = INT(MSHOTS * AREA(30)/SMLAREA)
      DFC(30) = DFLOAT(NSHOTS)
      II = 30

      UNX = 0.0D0
      UNY = 0.0D0
      UNZ = 1.0D0
      ANGLE = DSQRT(0.5D0)

      DO 120 I=1,NSHOTS
        CALL RANDOM (R)
        XPRIME = -FLAKDIM/2.0D0 + R*FLAKDIM
        CALL RANDOM (R)
        YPRIME = -FLAKDIM/2.0D0 + R*FLAKDIM
        Z1 = HREF+H1+H2+H3+H4+HCURVE+H7

        X1 = ANGLE * (XPRIME - YPRIME)
        Y1 = ANGLE * (XPRIME + YPRIME)

        CALL VECTOR(X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
        CALL OPTICS(X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
  120 CONTINUE
*
*********** Mirror Mount Rim ***************
*
      NSHOTS = INT(AREA(31)/SMLAREA * MSHOTS)
      DFC(31) = DFLOAT(NSHOTS)
      II = 31

      UNX = 0.0D0
      UNY = 0.0D0
      UNZ = -1.0D0

      DO 125 I=1,NSHOTS
        CALL RANDOM (R)
        RADIUS = D5/2.D0 + R*((D6-D5)/2.D0)
        CALL RANDOM (R)
        THETA = 2.0D0 * PI * R

        X1 = RADIUS * DCOS(THETA)
        Y1 = RADIUS * DSIN(THETA)
        Z1 = HREF + H1 + H2
        CALL VECTOR (X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
        CALL OPTICS (X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
```

```
  125 CONTINUE
*
*********** Detector housing (cont.) ********
*
      NSHOTS = INT(AREA(32)/SMLAREA * MSHOTS)
      DFC(32) = DFLOAT(NSHOTS)
      II = 32

      UNX = 0.0D0
      UNY = 0.0D0
      UNZ = -1.0D0

      DO 130 I=1,NSHOTS
        CALL RANDOM (R)
        RADIUS = D6/2.D0 + R*(RPRABLA-D6/2.D0)
        CALL RANDOM (R)
        THETA = 2.0D0 * PI * R

        X1 = RADIUS * DCOS(THETA)
        Y1 = RADIUS * DSIN(THETA)
        Z1 = HREF + H1 + H2 + H3
        CALL VECTOR (X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
        CALL OPTICS (X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)

  130 CONTINUE
*
********** Cylindrical surface of primary mirror *****************
*
      NSHOTS = INT(MSHOTS * AREA(33)/SMLAREA)
      DFC(33) = DFLOAT(NSHOTS)
      II = 33

      HA = HREF+H1+H2+H3+HCURVE-(D7*D7/(8.D0*KAPPA1))
      HB = H4-H6+D7*D7/(8.D0*KAPPA1)
      RADIUS = D7/2.0D0

      DO 135 I=1,NSHOTS
        CALL RANDOM (R)
        THETA = 2.0D0 * PI * R
        CALL RANDOM (R)
        Z1 = HA + R*HB

        X1 = RADIUS*DCOS(THETA)
        Y1 = RADIUS*DSIN(THETA)

        UNX = X1/RADIUS
        UNY = Y1/RADIUS
        UNZ = 0.0D0
        CALL VECTOR(X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
        CALL OPTICS(X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
  135 CONTINUE
*
********** Cylindrical surface of primary insert *****************
*
      NSHOTS = INT(MSHOTS * AREA(34)/SMLAREA)
      DFC(34) = DFLOAT(NSHOTS)
      II = 34

      HB = H6 - ((D7-D9)/2.D0)*(DTAN(THETA5*PI/180.D0))
      HA = HREF+H1+H2+H3+H4+HCURVE-HB
      RADIUS = D9/2.0D0

      DO 140 I=1,NSHOTS
        CALL RANDOM (R)
        THETA = 2.0D0 * PI * R
        CALL RANDOM (R)
        Z1 = HA + R*HB
```

```
        X1 = RADIUS*DCOS(THETA)
        Y1 = RADIUS*DSIN(THETA)

        UNX = X1/RADIUS
        UNY = Y1/RADIUS
        UNZ = 0.0D0
        CALL VECTOR(X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
        CALL OPTICS(X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
  140 CONTINUE
*
**************** Aperture *****************
*
      NSHOTS = INT(AREA(35)/SMLAREA * MSHOTS)
      DFC(35) = DFLOAT(NSHOTS)
      II = 35

      RC = (HREF-APEX(5)) * DTAN(THETA1*PI/180.D0)
      UNX = 0.0D0
      UNY = 0.0D0
      UNZ = -1.0D0

      DO 145 I=1,NSHOTS
        CALL RANDOM (R)
        RADIUS = RC * DSQRT(R)
        CALL RANDOM (R)
        THETA = 2.0D0 * PI * R

        X1 = RADIUS * DCOS(THETA)
        Y1 = RADIUS * DSIN(THETA)
        Z1 = HREF
        CALL VECTOR (X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
        CALL OPTICS (X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)

  145 CONTINUE
*
************* Detector header cylinder ******************
*
      NSHOTS = INT(MSHOTS * AREA(36)/SMLAREA)
      II = 36
      DFC(36) = DFLOAT(NSHOTS)
      DO 150 I=1,NSHOTS
        CALL RANDOM (R)
        THETA = 2.0D0 * PI * R
        CALL RANDOM (R)
        Z1 = HREF + H1 + H2 + H3 + H4 + HCURVE + R*H7

        X1 = D10/2.0D0 * DCOS(THETA)
        Y1 = D10/2.0D0 * DSIN(THETA)

        UNX = DCOS(THETA)
        UNY = DSIN(THETA)
        UNZ = 0.0D0

        CALL VECTOR(X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
        CALL OPTICS(X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
  150 CONTINUE
*
******************* Flake Substrate ********************
*
      NSHOTS = INT(MSHOTS * AREA(37)/SMLAREA)
      II = 37
      DFC(37) = DFLOAT(NSHOTS)
      DO 155 J=1,NSHOTS
        CALL RANDOM (R)
        THETA = 2.0D0 * PI * R
```

```
      IF(THETA.GE.0.0D0.AND.THETA.LE.PI/2.D0.OR.THETA.GE.
     &   PI.AND.THETA.LE.3.D0*PI/2.D0) THEN
        IF(THETA.EQ.PI/2.D0.OR.THETA.EQ.3.D0*PI/2.D0) THEN
          X = 0.0D0
          Y = FLAKDIM/DSQRT(2.D0)
          GO TO 160
        ENDIF
        X = DIM1 / (2.0D0*(1.0D0 + DTAN(THETA)))
        Y = DTAN(THETA) * X
        RADMIN = DSQRT(X*X + Y*Y)
        GO TO 160
      ENDIF

      IF(THETA.GE.PI/2.D0.AND.THETA.LE.PI.OR.THETA.GE.
     &    3.D0*PI/2.D0.AND.THETA.LE.2.D0*PI) THEN
        IF(THETA.EQ.PI/2.D0.OR.THETA.EQ.3.D0*PI/2.D0) THEN
          X = 0.0D0
          Y = -FLAKDIM/DSQRT(2.D0)
          GO TO 160
        ENDIF
        X = DIM1 / (2.0D0*(DTAN(THETA) - 1.0D0))
        Y = DTAN(THETA) * X
        RADMIN = DSQRT(X*X + Y*Y)
        GO TO 160
      ENDIF

 160  CONTINUE
      CALL RANDOM (R)
      UNX = 0.0D0
      UNY = 0.0D0
      UNZ = 1.0D0
      RADIUS = RADMIN + DSQRT(R)*(D10/2.0D0 - RADMIN)

      X1 = RADIUS * DCOS(THETA)
      Y1 = RADIUS * DSIN(THETA)
      Z1 = HREF+H1+H2+H3+H4+HCURVE+H7

      CALL VECTOR(X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
      CALL OPTICS(X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
 155 CONTINUE
*
* Calculate the distribution factors.
*
     DO 200 I=1,37
     DO 200 J=1,37
 200  DF(I,J) = DF(I,J)/DFC(I)
*
* Write the distribution factors to file DISTR FCTR.
*
     CALL SUMCHK
     CALL DFSCHK
     WRITE(10,500)((I,J,DF(I,J),J=1,37),I=1,37)
 500 FORMAT(1X,2I4,2X,D15.10,5X,2I4,2X,D15.10)

     STOP
     END



*************************************************************
*                                                           *
*                      PROGRAM WAVELGN                      *
*                                                           *
* This is the driving program for the Monte Carlo method used to *
* to determine the distribution of radiation within the optics  *
```

```
* module of the ERBE scanning radiometer as a function of        *
* wavelength or assuming gray surfaces.                          *
*                                                                *
******************************************************************
*
      PROGRAM WAVELGN
*
* Request double precision variables.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Define storage parameters and variable types.
*
      INTEGER*2 FLAG
      INTEGER*4 SHTCNT
      REAL*8 KAPPA1,KAPPA2,DF(40,40),AREA(37),APEX(8),POINT(2,3000)
      REAL*8 XN(20),YN(20),ZN(20),X1N(20),Y1N(20),Z1N(20)
*
* Place frequently used variables in a unique common storage location.
*
      COMMON /GEOM/ H1,D1,D2,THETA1,C1,C2,C3,C4,C5,H2,D3,D4,D5
      COMMON /GEOM/ DEL1,DEL2,THETA2,PHI,H3,HL1,HL2,D6,THETA3
      COMMON /GEOM/ H4,HCURVE,D7,KAPPA1,H5,D8,KAPPA2,THETA4,H6,D9
      COMMON /GEOM/ THETA5,DIM1,DIM2,HREF,DREF

      COMMON /MATL/ ABS1,ABS2,ABS3,ABS4,ABS5,ABS6,ABS7,ABS8,ABS9
      COMMON /MATL/ ABS10,REFR1,REFR2,REFR3,REFR4,REFR5,REFR6
      COMMON /MATL/ REFR7,REFR8,REFR9,REFR10

      COMMON /RAND/ IN
      COMMON /PLOT/ POINT,NSPLOT(25,25)
      COMMON /COORDS/ XN,YN,ZN,X1N,Y1N,Z1N
      COMMON /DISTR/ DF,II
      COMMON /CHANNEL/ ALAMBDA,NCHANEL,NFLTRSW,NSPCTRM
      COMMON /SPEC/ NUMHITS,MIRTYPE
*
* Initialize all variables (dimensions).
*
      PI = DACOS(-1.D0)
      CALL INPUT
      CALL RNSET(5520470)
*
* Select type of incident radiation (collimated beam or diffuse)
*
      WRITE(*,*)' Enter the number corresponding to the type'
      WRITE(*,*)' of radiation you wish to be incident on the'
      WRITE(*,*)' scanner optics module.'
      WRITE(*,*)' 0 = diffuse'
      WRITE(*,*)' 1 = collimated radiation at user defined angle'

      READ(*,*) NRADTYP
*
* If a collimated beam of radiation is chosen, prompt the user for
* the angle of incidence with respect to the normal of the aperture
* of the optics module.
*
      IF(NRADTYP.EQ.1) THEN
        WRITE(*,*)' Specify the angle of the incident collimated'
        WRITE(*,*)' beam of radiation.'

        WRITE(*,*)' Angle of incidence (deg) ='
        READ(*,*) PPHI1

        THETA = PPHI1 * PI/180.0D0

      ENDIF
*
```

```
* Specify type of analysis to be performed.
*
      WRITE(*,*)' Specify the type of analysis to be performed'
      WRITE(*,*)' 0 = gray'
      WRITE(*,*)' 1 = spectral'

      READ(*,*) NSPCTRM
*
* Select radiometric channel.
*
      WRITE(*,*)' Enter the number corresponding to which'
      WRITE(*,*)' channel of the scanner to be modeled.'
      WRITE(*,*)' If you selected gray analysis, choose total'
      WRITE(*,*)' channel only.'
      WRITE(*,*)' 0 = total channel'
      WRITE(*,*)' 1 = longwave channel'
      WRITE(*,*)' 2 = shortwave channel'

      READ(*,*) NCHANEL
*
* Select type of mirrors. (100% specular or wavelength-dependent)
*
      WRITE(*,*)' Select the type of mirrors used in your analysis'
      WRITE(*,*)' If you selected gray analysis, choose user '
      WRITE(*,*)' defined specularity (0) only.'
      WRITE(*,*)' 0 = user-defined specularity'
      WRITE(*,*)' 1 = wavelength dependent'
      READ(*,*) MIRTYPE
*
* Enter the number of energy bundles to be emitted from the
* aperture of the optics module.
*
      WRITE(*,*)' Enter the number of energy bundles to be emitted'
      WRITE(*,*)' from the aperture of the optics module (integer).'
      READ(*,*) NSHOTS
*
* Enter the temperature of the observed scene.
*
      IF(NSPCTRM.EQ.1) THEN
        WRITE(*,*) 'Enter the temperature of the scene.'
        READ(*,*) TEMP
      ENDIF
*
* Initialize necessary data.
*
      IN = 201
      II = 1
      SHTCNT = 0
      ABSSHT = 0.D0
*
* Initialize arguments to be sent to subprograms.
*
      DO 5 I=1,40
      DO 5 J=1,40
    5   DF(I,J) = 0.D0

      DO 10 I=1,2
        DO 10 J=1,3000
   10     POINT(I,J) = 0.0D0

      DO 20 J=1,20
        XN(J) = 0.0D0
        YN(J) = 0.0D0
        ZN(J) = 0.0D0
        X1N(J) = 0.0D0
        Y1N(J) = 0.0D0
        Z1N(J) = 0.0D0
```

```
   20 CONTINUE
*
* Monitor energy bundle counter.
*
    1 CONTINUE
      IF(SHTCNT.EQ.NSHOTS) GO TO 2
*
* Find point of emission.
*
      CALL RANDOM (R)
      RCTRD = DREF/2.0D0 * DSQRT(R)
      CALL RANDOM (R)
      GAMMA = 2.0D0 * PI * R
*
* In cartesian coordinates, points on a circle are defined as:
*
      X1 = RCTRD * DCOS(GAMMA)
      Y1 = RCTRD * DSIN(GAMMA)
      Z1 = 0.0D0
*
* Choose a direction for the emitted energy bundle by imposing a unit
* sphere tangent to the point on the optics module aperture.
*
      IF(NRADTYP.EQ.0) THEN
        UNX = 0.0D0
        UNY = 0.0D0
        UNZ = -1.0D0

        CALL VECTOR1 (X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
      ENDIF
*
* If collimated radiation enters the aperture, pick the entrance
* angle.
*
      IF(NRADTYP.EQ.1) THEN
        ZS = DCOS(THETA)
        XS = DSIN(THETA)
        YS = 0.0D0

        X2 = X1 + XS
        Y2 = Y1 + YS
        Z2 = Z1 + ZS

        VOX = X2 - X1
        VOY = Y2 - Y1
        VOZ = Z2 - Z1
      ENDIF
*
* This gives two points in the global coordinate system which may be
* used to write the equation of a line associated with the emitted
* energy bundle.
*
* Pick a wavelength for the energy bundle.
*
      IF(NSPCTRM.EQ.1) CALL BBFCN (ALAMBDA,TEMP)
*
* Determine whether energy bundle enters aperture of optics
* module.
*
      T = (HREF-Z1)/VOZ
      TX = T*VOX + X1
      TY = T*VOY + Y1

      RADIUS = DSQRT(TX*TX + TY*TY)
      IF(RADIUS.GE.DREF/2.0D0) GO TO 99
*
* Solve the set of simultaneous equations generated between the
```

```
* intersection of the line associated with the emitted energy bundle
* and the surfaces of the optics module.
*
      CALL OPTICS (X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
   99 CONTINUE
      SHTCNT = SHTCNT + 1
      GO TO 1
*
* Print results
*
    2 CONTINUE
      CALL PRINT (ABSSHT,NSHOTS)
      WRITE(2,100) (POINT(1,J),POINT(2,J),J=1,3000)
*
********************** Format Statements ***************************
*
  100 FORMAT(1X,8F8.3)
  200 FORMAT(1X,I3,2X,E13.7,2X,I3,2X,E13.7)
      STOP
      END




********************************************************************
*                                                                  *
*                         SUBROUTINE INPUT                         *
*                                                                  *
* This subroutine queries the user for all applicable dimensions   *
* associated with the optics module of the ERBE scanning radiometer. *
*                                                                  *
********************************************************************
*
      SUBROUTINE INPUT
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Specify size and type of storage parameters.
*
      DIMENSION A(62)
      REAL*8 KAPPA1,KAPPA2
      CHARACTER*1 CHAR
*
* Place all dimensions in one storage unit.
*
      COMMON /GEOM/ H1,D1,D2,THETA1,C1,C2,C3,C4,C5,H2,D3,D4,D5
      COMMON /GEOM/ DEL1,DEL2,THETA2,PHI,H3,HL1,HL2,D6,THETA3
      COMMON /GEOM/ H4,HCURVE,D7,KAPPA1,H5,D8,KAPPA2,THETA4,H6,D9
      COMMON /GEOM/ THETA5,DIM1,DIM2,HREF,DREF
      COMMON /GEOM1/ H7,D10,FLAKDIM,ABS11,REFR11

      COMMON /MATL/ ABS1,ABS2,ABS3,ABS4,ABS5,ABS6,ABS7,ABS8,ABS9
      COMMON /MATL/ ABS10,REFR1,REFR2,REFR3,REFR4,REFR5,REFR6
      COMMON /MATL/ REFR7,REFR8,REFR9,REFR10
*
* Read all data into an array necessary for making any changes in the
* dimensions of the optics module.
*
      DO 15 I=1,62
   15   READ(7,*) A(I)
*
* Initialize computed go to statement.
*
```

```
      M = 1
*
* Go back to top of data file.
*
      REWIND 7
*
* Read data of a previous run for each integral component of the
* optics module.
*
***** reflector cap *****

   25 CONTINUE
      READ(7,*) H1,D1,D2,THETA1,C1,C2,C3,C4,C5
      READ(7,*) ABS1,REFR1,ABS2,REFR2

***** secondary mirror mount *****

      READ(7,*) H2,D3,D4,D5,DEL1,DEL2,THETA2,PHI
      READ(7,*) ABS3,REFR3,ABS4,REFR4

***** detector housing *****

      READ(7,*) H3,HL1,HL2,D6,THETA3
      READ(7,*) ABS5,REFR5

***** primary mirror *****

      READ(7,*) H4,HCURVE,D7,KAPPA1
      READ(7,*) ABS6,REFR6

***** secondary mirror *****

      READ(7,*) H5,D8,KAPPA2,THETA4
      READ(7,*) ABS7,REFR7

***** primary insert *****

      READ(7,*) H6,D9,THETA5
      READ(7,*) ABS8,REFR8

***** field stop *****

      READ(7,*) DIM1,DIM2
      READ(7,*) ABS9,REFR9,ABS10,REFR10
      READ(7,*) HREF,DREF

***** active flake *****

      READ(7,*) H7,D10,FLAKDIM,ABS11,REFR11
*
* Return control back to proper executable statement.
*
      GO TO (90,100,200,300,400,500,600,700,800) M
   90 CONTINUE
*
* Query user for data changes from a previous run.
*
      WRITE(6,6)
      READ(5,1) CHAR
      CHAR = 'N'
      IF(CHAR.EQ.'N'.OR.CHAR.EQ.'n') RETURN
*
* Write data to screen and prompt user for any necessary changes.
*
  100 WRITE(6,10) H1,D1,D2,THETA1,C1,C2,C3,C4,C5,ABS1,REFR1,ABS2,REFR2
      CALL CHECK (CHAR,A)
      IF(CHAR.EQ.'Y'.OR.CHAR.EQ.'y') THEN
```

```
      M = 2
      REWIND 7
      GO TO 25
    ELSE
      CONTINUE
    ENDIF

200 WRITE(6,20) H2,D3,D4,D5,DEL1,DEL2,THETA2,PHI,ABS3,REFR3,ABS4,REFR4
    CALL CHECK (CHAR,A)
    IF(CHAR.EQ.'Y'.OR.CHAR.EQ.'y') THEN
      M = 3
      REWIND 7
      GO TO 25
    ELSE
      CONTINUE
    ENDIF

300 WRITE(6,30) H3,HL1,HL2,D6,THETA3,ABS5,REFR5
    CALL CHECK (CHAR,A)
    IF(CHAR.EQ.'Y'.OR.CHAR.EQ.'y') THEN
      M = 4
      REWIND 7
      GO TO 25
    ELSE
      CONTINUE
    ENDIF

400 WRITE(6,40) H4,HCURVE,D7,KAPPA1,ABS6,REFR6
    CALL CHECK (CHAR,A)
    IF(CHAR.EQ.'Y'.OR.CHAR.EQ.'y') THEN
      M = 5
      REWIND 7
      GO TO 25
    ELSE
      CONTINUE
    ENDIF

500 WRITE(6,50) H5,D8,KAPPA2,THETA4,ABS7,REFR7
    CALL CHECK (CHAR,A)
    IF(CHAR.EQ.'Y'.OR.CHAR.EQ.'y') THEN
      M = 6
      REWIND 7
      GO TO 25
    ELSE
      CONTINUE
    ENDIF

600 WRITE(6,60) H6,D9,THETA5,ABS8,REFR8
    CALL CHECK (CHAR,A)
    IF(CHAR.EQ.'Y'.OR.CHAR.EQ.'y') THEN
      M = 7
      REWIND 7
      GO TO 25
    ELSE
      CONTINUE
    ENDIF

700 WRITE(6,70) DIM1,DIM2,ABS9,REFR9,ABS10,REFR10
    CALL CHECK (CHAR,A)
    IF(CHAR.EQ.'Y'.OR.CHAR.EQ.'y') THEN
      M = 8
      REWIND 7
      GO TO 25
    ELSE
      CONTINUE
    ENDIF
```

```
  800 WRITE(6,80) H7,D10,FLAKDIM,ABS11,REFR11
      CALL CHECK (CHAR,A)
      IF(CHAR.EQ.'Y'.OR.CHAR.EQ.'y') THEN
        M = 9
        REWIND 7
        GO TO 25
      ELSE
        CONTINUE
      ENDIF
*
* Go back to top of data file.
*
      REWIND 7
      RETURN

********************** Format statements *************************
    1 FORMAT(A1)
    2 FORMAT(5X,'Enter the number in parenthesis of the variable',/,
     &5x,'you would like to change and enter the corrected value.',/)
    5 FORMAT(5X,'Would you like to make any changes (Y/N) ?',/)
    6 FORMAT(5X,'Would you like to review the input data from a',/,
     &5x,'previous run (Y/N)?')
   10 FORMAT(5X,'THE SYSTEM VARIABLES ARE AS FOLLOWS:',//,
     &15X,'***** reflector cap *****',//,
     &10X,'(1) overall reflector cap height          =',F6.2,' (mm)',/,
     &10X,'(2) ref. diameter (inner,conical surfaces) =',F6.2,' (mm)',/,
     &10X,'(3) ref. diameter (outer,conical surfaces) =',F6.2,' (mm)',/,
     &10X,'(4) ref. angle of conical surfaces         =',F6.2,' (deg)',/
     &10X,'height of each conical surface along cap:',//,
     &10X,'(5) cone height 1                          =',F5.2,' (mm)',/,
     &10X,'(6) cone height 2                          =',F5.2,' (mm)',/,
     &10X,'(7) cone height 3                          =',F5.2,' (mm)',/,
     &10X,'(8) cone height 4                          =',F5.2,' (mm)',/,
     &10X,'(9) cone height 5                          =',F5.2,' (mm)',//
     &10X,'The value of the absorptivity and reflectivity for the',/,
     &10X,'reflective conical surfaces are:',//,
     &10X,'(10) abs =',F5.2,5X,'(11) refl =',F5.2,//,
     &10X,'The value of the absorptivity and reflectivity for the',/,
     &10X,'black conical surfaces are:',//,
     &10X,'(12) abs =',F5.2,5X,'(13) refl =',F5.2,/)
   20 FORMAT(5X,'THE SYSTEM VARIABLES ARE AS FOLLOWS:',//,
     &15X,'***** secondary mirror mount *****',//,
     &10X,'(14) overall mirror mount height        =',F6.2,' (mm)',/,
     &10X,'(15) inner ref. diameter (spider side)  =',F6.2,' (mm)',/,
     &10X,'(16) outer ref. diameter (spider side)  =',F6.2,' (mm)',/,
     &10X,'(17) outer ref. diameter (taper side)   =',F6.2,' (mm)',/,
     &10X,'(18) thickness of a spider leg          =',F6.2,' (mm)',/,
     &10X,'(19) depth of a spider leg              =',F6.2,' (mm)',/,
     &10X,'(20) angle of taper                     =',F6.2,' (deg)',/
     &10X,'(21) ref. angle between each spider leg =',F7.2,' (deg)',//
     &10X,'The value of the absorptivity and reflectivity for the',/,
     &10X,'reflective surfaces are:',//,
     &10X,'(22) abs =',F5.2,5X,'(23) refl =',F5.2,//,
     &10X,'The value of the absorptivity and reflectivity for the',/,
     &10X,'black surfaces are:',//,
     &10X,'(24) abs =',F5.2,5X,'(25) refl =',F5.2,/)
   30 FORMAT(5X,'THE SYSTEM VARIABLES ARE AS FOLLOWS:',//,
     &15X,'***** detector housing *****',//,
     &10x,'(26) overall housing height (detail -g-) =',F6.2,' (mm)',/,
     &10X,'(27) cylindrical surface length         =',F6.2,' (mm)',/,
     &10X,'(28) conical surface length             =',F6.2,' (mm)',/,
     &10X,'(29) ref. diameter                      =',F6.2,' (mm)',/,
     &10X,'(30) ref. angle (conical surface)       =',F6.2,' (deg)',//,
     &10X,'The value of the absorptivity and reflectivity for the',/,
     &10X,'detector housing surfaces are:',//,
     &10X,'(31) abs =',F5.2,5X,'(32) refl =',F5.2,/)
   40 FORMAT(5X,'THE SYSTEM VARIABLES ARE AS FOLLOWS:',//,
```

```
     &15X,'***** primary mirror *****',//,
     &10X,'(33) base ht. below curvature  =',F6.2,' (mm)',/,
     &10X,'(34) height of curvature       =',F6.2,' (mm)',/,
     &10X,'(35) inner diameter            =',F6.2,' (mm)',/,
     &10X,'(36) radius of curvature       =',F6.2,' (mm)',//,
     &10X,'The value of the absorptivity and reflectivity for the',/,
     &10X,'mirrored surface are:',//,
     &10X,'(37) abs =',F5.2,5X,'(38) refl =',F5.2,/)
   50 FORMAT(5X,'THE SYSTEM VARIABLES ARE AS FOLLOWS:',//,
     &15X,'***** secondary mirror *****',//,
     &10X,'(39) overall mirror height         =',F6.2,' (mm)',/,
     &10X,'(40) diameter of mirrored surface  =',F6.2,' (mm)',/,
     &10X,'(41) radius of curvature           =',F6.2,' (mm)',/,
     &10X,'(42) ref. angle of mirror taper    =',F6.2,' (deg)',//,
     &10X,'The value of the absorptivity and reflectivity for the',/,
     &10X,'secondary mirror are:',//,
     &10X,'(43) abs =',F5.2,5X,'(44) refl =',F5.2,/)
   60 FORMAT(5X,'THE SYSTEM VARIABLES ARE AS FOLLOWS:',//,
     &15X,'***** primary insert *****',//,
     &10X,'(45) height of insert                   =',F6.2,' (mm)',/,
     &10X,'(46) inner diameter                     =',F6.2,' (mm)',/,
     &10X,'(47) angle measure (conical surface)  =',F6.2,' (deg)',//,
     &10X,'The value of the absorptivity and reflectivity for the',/,
     &10X,'insert are:',//,
     &10X,'(48) abs =',F5.2,5X,'(49) refl =',F5.2,/)
   70 FORMAT(5X,'THE SYSTEM VARIABLES ARE AS FOLLOWS:',//,
     &15X,'***** field stop *****',//,
     &10X,'(50) aperture measure normal to scan   =',F5.2,' (mm)',/,
     &10X,'(51) aperture measure along scan dir.  =',F5.2,' (mm)',//,
     &10X,'The value of the absorptivity and reflectivity for the',/,
     &10X,'reflective surfaces are:',//,
     &10X,'(52) abs =',F5.2,5X,'(53) refl =',F5.2,//,
     &10X,'The value of the absorptivity and reflectivity for the',/,
     &10X,'black surfaces are:',//,
     &10X,'(54) abs =',F5.2,5X,'(55) refl =',F5.2,/)
   80 FORMAT(5X,'THE SYSTEM VARIABLES ARE AS FOLLOWS:',//,
     &15X,'***** active flake *****',//,
     &10X,'(58) distance, active flake behind field stop =',F6.2,'(mm)',
     &/,10X,'(59) diameter, detector header cyl. (app.) =',F6.2,'(mm)',
     &/,10X,'(60) size of active flake =',F6.2,'(mm edge)',
     &//,10X,'The value of the absorptivity and reflectivity for the
     &flake is:',//,
     &10X,'(61) abs =',F5.2,5X,'(62) refr =',F5.2,/)
   99 FORMAT( )
      END



      SUBROUTINE CHECK (CHAR,A)
*
* This subroutine prompts the user for changes in system variables.
*
      REAL*8 A(62)
      CHARACTER*1 CHAR

      WRITE(6,5)
      READ(5,1) CHAR
      IF(CHAR.EQ.'Y'.OR.CHAR.EQ.'y') THEN
        WRITE(6,2)
        READ(5,*) I,A(I)
        REWIND 7
        WRITE(7,10) (A(I),I=1,62)
      ENDIF

      RETURN
```

```
*********************** Format statements ***************************
    1 FORMAT(A1)
    2 FORMAT(5X,'Enter the number in parenthesis of the variable',/,
     &5x,'you would like to change and enter the corrected value.',/)
    5 FORMAT(5x,'Would you like to make any changes (Y/N) ?',/)
   10 FORMAT(F6.2)

      END




*******************************************************************
*                                                                 *
*                       SUBROUTINE EQAREA                         *
*                                                                 *
* This subprogram computes the surface area of each part of the   *
* optics module.                                                  *
*                                                                 *
*******************************************************************
*
      SUBROUTINE EQAREA
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Define size and type of storage variables.
*
      REAL*8 AREA(37),KAPPA1,KAPPA2
*
* Place all common variables in one, unique storage block.
*
      COMMON /DFSAREA/ AREA

      COMMON /GEOM/ H1,D1,D2,THETA1,C1,C2,C3,C4,C5,H2,D3,D4,D5
      COMMON /GEOM/ DEL1,DEL2,THETA2,PHI,H3,HL1,HL2,D6,THETA3
      COMMON /GEOM/ H4,HCURVE,D7,KAPPA1,H5,D8,KAPPA2,THETA4,H6,D9
      COMMON /GEOM/ THETA5,DIM1,DIM2,HREF,DREF
      COMMON /GEOM1/ H7,D10,FLAKDIM,ABS11,REFR11

      PI = DACOS(-1.0D0)
*
* Initialize area array.
*
      DO 30 I=1,37
   30   AREA(I) = 0.0D0
*_________________________________________________________________
* Compute the surface area of each part of the optics module.    |
*________________________________________________________________|
*
* Reflector cap.
*
      CEXT = C2 - 2.0D0*C1
      AREA(1) = DSQRT((D2/2.0D0 - D1/2.0D0)**2 + CEXT*CEXT)
      AREA(1) = AREA(1) * PI*((D1 + D2)/2.0D0)
      AREA(2) = AREA(1)
      AREA(3) = AREA(1)
      AREA(4) = DSQRT(((D2-D4)/2.0D0)**2 + (C5-C4)*(C5-C4))
      AREA(4) = AREA(4) * PI*((D2+D4)/2.0D0)
      AREA(5) = DSQRT(((D2 - D1)/2.0D0)**2 + C1*C1)
      AREA(5) = AREA(5) * PI *((D1+D2)/2.0D0)

      AREA(6) = AREA(5)
      AREA(7) = AREA(5)
      AREA(8) = AREA(5)
```

```
      AREA(9) = PI*D4*(H1-C5)
*
* Top of mirror mount.
*
      AREA(10) = PI*D3*D3/4.0D0
*
* Spider legs.
*
      AREA(11) = D4-D3
      AREA(11) = AREA(11) * DSQRT(DEL1*DEL1/4.0D0 + DEL2*DEL2)
      AREA(12) = AREA(11)
      AREA(13) = AREA(11)
      AREA(14) = (D4-D3)/2.0D0 * DEL1
      AREA(15) = AREA(14)
      AREA(16) = AREA(14)
*
* Hub of secondary mirror mount and perimeter of secondary mirror.
*
      AREA(17) = PI*D3*(H2-H5) - 3.D0/2.D0*(DIM1*DIM2)
      HMIRROR = DSQRT(KAPPA2*KAPPA2 - D8*D8/4.0D0) + H5 - KAPPA2
      CAL = (D8/2.0D0) / DTAN(THETA4*PI/180.0D0)
      APEX = HREF + H1 + H2 - CAL
      HA = HREF + H1 + H2 - H5
      R1 = (HA-APEX)*DTAN(THETA4*PI/180.0D0)
      R2 = D3/2.0D0
      R3 = D8/2.0D0
      AREA(18) = DSQRT((R3-R1)*(R3-R1) + HMIRROR*HMIRROR)
      AREA(18) = AREA(18)*PI*(R3 + R1)
      AREA(18) = AREA(18) + PI*(R1*R1 - R2*R2)
*
* Secondary mirror mount, perimeter and conical surface.
*
      HTAPER = ((D4-D5)/2.0D0) * DTAN(THETA2*PI/180.0D0)
      AREA(19) = PI*D4*(H2-HTAPER) - 3.D0/2.D0*(DIM1*DIM2)
      AREA(20) = DSQRT(((D4-D5)/2.0D0)**2 + HTAPER*HTAPER)
      AREA(20) = AREA(20)*PI*((D4+D5)/2.0D0)
      AREA(31) = PI*(D6*D6 - D5*D5)/4.0D0
*
* Surface of secondary mirror (approximate as circular area).
*
      AREA(21) = PI*D8*D8/4.0D0
*
* Detector housing.
*
      AREA(22) = PI*D6*(H3-HL1-HL2)
      RCONE = HL1/2.0D0/DTAN(THETA3/2.0D0*PI/180.D0)
      AREA(23) = PI*(D6 + RCONE)
      AREA(23) = AREA(23) * DSQRT(RCONE*RCONE + HL1*HL1/4.0D0)
      AREA(24) = AREA(23)
      AREA(25) = PI*D6*HL2
      RA = DSQRT(HCURVE*2.0D0*KAPPA1)
      AREA(32) = PI*(RA*RA - D6*D6/4.0D0)
*
* Surface area of primary mirror (approximate as a cone).
*
      RB = D7/2.0D0
      HEIGHT = HCURVE - (RB*RB)/(2.D0*KAPPA1)
      AREA(26) = 2.D0* KAPPA1 * PI * HEIGHT
      AREA(33) = PI*D7*(H4-H6+(D7*D7/(8.D0*KAPPA1)))
*
* Primary insert.
*
      HEIGHT = ((D7-D9)/2.D0)*DTAN(THETA5*PI/180.0D0)
      AREA(27) = DSQRT(((D7-D9)/2.0D0)**2 + HEIGHT*HEIGHT)
      AREA(27) = AREA(27) * PI*((D7+D9)/2.0D0)
```

```
      AREA(34) = PI*D9*(H6-HEIGHT)
*
* Field stop.
*
      AREA(28) = PI*D9*D9/4.0D0 - DIM1*DIM1/2.0D0
      AREA(28) = AREA(28) + ((DIM1-DIM2)**2) / 2.0D0
      AREA(29) = PI*D10*D10/4.0D0 - DIM1*DIM1/2.0D0
      AREA(29) = AREA(29) + ((DIM1-DIM2)**2) / 2.0D0
*
* Active Flake.
*
      AREA(30) = FLAKDIM*FLAKDIM
*
* Aperture.
*
      AREA(35) = DREF * DREF * PI /4.0D0
*
* Detector header cylinder.
*
      AREA(36) = PI*D10*H7
*
* Flake substrate.
*
      AREA(37) = PI*D10*D10/4.D0 - AREA(30)

      RETURN
      END




*******************************************************************
*                                                                 *
*                       SUBROUTINE OPTICS                         *
*                                                                 *
* This subprogram calls subroutines that monitors each part       *
* of the optics module.  The subroutines determine whether an     *
* emitted energy bundle is reflected or absorbed by that part     *
* part of the module.                                             *
*                                                                 *
*******************************************************************
*
      SUBROUTINE OPTICS (X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Specify parameter size and type.
*
      INTEGER*2 FLAG
*
* Place all frequently used variables in one, unique storage location.
*
      COMMON /FLAGS/ NFLAG,NTEST,NN
      COMMON /CHANNEL/ ALAMBDA,NCHANEL,NFLTRSM,NSPCTRM
      COMMON /SPEC/ NUMHITS,MIRTYPE
*
* Set necessary flags.
*
      FLAG = 1
      NUMHITS = 0

    1 CONTINUE
      NN = 0
      NTEST = 0
```

```
      NFLAG = 0
*
* Determine if radiation hits the reflector cap.
*
    2 CALL CAP (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
      IF(NTEST.EQ.1) GO TO 1
      IF(FLAG.EQ.0) RETURN
*
* Determine if radiation hits the top of the secondary mirror mount.
*
    3 CALL SPIDER (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
      IF(NTEST.EQ.1) GO TO 1
      IF(FLAG.EQ.0) RETURN
*
* Determine if radiation hits filter of shortwave channel.
*
   15 IF(NCHANEL.EQ.2.AND.NSPCTRM.EQ.1) THEN
        NFLTRSW = 1
        CALL FILTER (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
        IF(NTEST.EQ.1) GO TO 1
      ENDIF
*
* If radiation does not hit the top of secondary mirror mount, determine
* if it hits the perimeter of the secondary mirror or the hub of its
* mount.
*            ***** secondary mirror mount *****
*
    4 CALL MIDCYL (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
      IF(NTEST.EQ.1) GO TO 1
      IF(FLAG.EQ.0) RETURN
*
*               ***** secondary mirror *****
*
    5 CALL MIRROR (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
      IF(NTEST.EQ.1) GO TO 1
      IF(FLAG.EQ.0) RETURN
*
* If radiation does not hit any of the centrally located optics,
* determine if radiation hits the inner perimeter of the secondary
* mirror mount.
*
    6 CALL MOUNT (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
      IF(NTEST.EQ.1) GO TO 1
      IF(FLAG.EQ.0) RETURN
*
* Continue search along optics module wall.  Determine whether ray
* hits the cylindrical surfaces of the detector housing.
*
    7 CALL DETCYL (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
      IF(NTEST.EQ.1) GO TO 1
      IF(FLAG.EQ.0) RETURN
*
* Determine if an emitted energy bundle hits the conical surfaces
* of the detector housing.
*
    8 CALL CONE (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
      IF(NTEST.EQ.1) GO TO 1
      IF(FLAG.EQ.0) RETURN
*
* Search the primary mirror for energy bundle interception.
*
    9 CALL PRIMIR (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
      IF(NTEST.EQ.1) GO TO 1
      IF(FLAG.EQ.0) RETURN
*
* Search the insert of the primary mirror.
*
```

```
   10 CALL INSERT (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
      IF(NTEST.EQ.1) GO TO 1
      IF(FLAG.EQ.0) RETURN
*
* Search the secondary mirror for energy bundle interception.
*
   11 CALL SECMIR (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
      IF(NTEST.EQ.1) GO TO 1
      IF(FLAG.EQ.0) RETURN
*
* Check the field stop and the active flake.
*
   12 CALL FOV (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
      IF(NTEST.EQ.1) GO TO 1
      IF(FLAG.EQ.0) RETURN
*
* Determine if radiation strikes filter (LW or SW).
*
   16 IF(NCHANEL.NE.0.AND.NSPCTRM.EQ.1) THEN
        NFLTRSW = 2
        CALL FILTER (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
        IF(NTEST.EQ.1) GO TO 1
      ENDIF
*
* Check the detector header cylinder and flake substrate.
*
   17 CALL HEADER (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
      IF(NTEST.EQ.1) GO TO 1
      IF(FLAG.EQ.0) RETURN
*
* A flag is set that is sent to each subroutine to direct that
* subroutine selected by subprogram COORDS to test that particular
* surface for energy bundle absorption or reflection.
*
      NFLAG = 1
*
* Another flag is set to tell subprogram COORD that the last surface
* has been tested for energy bundle intersection and to begin computing
* the distance from the emission site to any points of possible
* impingement by the energy bundle.
*
      NN = 20
      CALL COORD (X1,Y1,Z1,X,Y,Z)
*
* An integer NN is returned from COORD to direct program to the
* subroutine that contains the coordinates of the point of incidence
* on a given surface that had the shortest distance from the emission
* site.  In that subprogram, it is determined whether the energy
* bundle is absorbed or reflected.
*
      GO TO (2,3,3,4,5,6,7,8,9,10,11,12,9,10,15,16,17) NN
      END




**********************************************************************
*                                                                    *
*                          SUBROUTINE CAP                            *
*                                                                    *
* This subprogram monitors the reflector cap of the ERBE scanning    *
* radiometer.  If an emitted energy bundle is intercepted by this    *
* part of the optics module, its absorption or reflection is         *
* accounted for.                                                     *
*                                                                    *
```

```
**********************************************************************
*
      SUBROUTINE CAP (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Define size and type of storage variables.
*
      INTEGER*2 FLAG
      REAL*8 KAPPA1,KAPPA2,APEX(8),V(2),DF(40,40)
*
* Place all common variables in one, unique storage block.
*
      COMMON /GEOM/ H1,D1,D2,THETA1,C1,C2,C3,C4,C5,H2,D3,D4,D5
      COMMON /GEOM/ DEL1,DEL2,THETA2,PHI,H3,HL1,HL2,D6,THETA3
      COMMON /GEOM/ H4,HCURVE,D7,KAPPA1,H5,D8,KAPPA2,THETA4,H6,D9
      COMMON /GEOM/ THETA5,DIM1,DIM2,HREF,DREF

      COMMON /MATL/ ABS1,ABS2,ABS3,ABS4,ABS5,ABS6,ABS7,ABS8,ABS9
      COMMON /MATL/ ABS10,REFR1,REFR2,REFR3,REFR4,REFR5,REFR6
      COMMON /MATL/ REFR7,REFR8,REFR9,REFR10

      COMMON /FLAGS/ NFLAG,NTEST,NN
      COMMON /DISTR/ DF,II
*
* Initialize necessary variables.
*
      NTEST = 0
      IF(NFLAG.EQ.1) GO TO 2
      NCALL = 0
      PI = DACOS(-1.0D0)
      ALPHA2 = THETA1 * PI/180.0D0
      ALPHA1 = PI/2.0D0 - ALPHA2
      RB = D2/2.0D0
*
* Loop through each conical surface of the reflector cap to determine
* the vertex of each cone and define each in reference to the
* coordinate axes.
*
      DO 10 I=1,4
        APEX(I) = RB/DTAN(ALPHA1) + HREF
   10   APEX(I+4) = -RB/DTAN(ALPHA2) + HREF

      APEX(1) = APEX(1) + C1
      APEX(2) = APEX(2) + C2
      APEX(3) = APEX(3) + C3
      APEX(4) = APEX(4) + C4
      APEX(5) = APEX(5) + C1
      APEX(6) = APEX(6) + C2
      APEX(7) = APEX(7) + C3
      APEX(8) = APEX(8) + C4
*
* Define variables necessary for DO loop and mathematical equation
* for a cone.
*
      V(1) = DTAN(ALPHA1) * DTAN(ALPHA1)
      V(2) = DTAN(ALPHA2) * DTAN(ALPHA2)
      RC = (HREF-APEX(5))*DTAN(ALPHA2)
      M = 1
      N = 4
*_________________________________________________________________
*                                                                 |
* This loop determines whether a reflected ray exits the optics   |
* module.                                                         |
*_________________________________________________________________|
```

```
*
* Determine whether ray is actually coming from inside the optics
* module.
*
      IF(Z1.GT.HREF) THEN
        T = (HREF-Z1)/VOZ
        X = T*VOX + X1
        Y = T*VOY + Y1
        Z = HREF
*
* The nature of this program makes it necessary to be sure a ray
* doesn't actually penetrate a given surface to be incident on
* another.  This has been taken care of by using the dot product
* of the directional vector, T, and the reflection/emission vector
* VOX (or VOY/VOZ).
*
        VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
        VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
        VMAG1 = DSQRT(VMAG1)
        VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
        VMAG2 = DSQRT(VMAG2)
*
* Determine if the dot product is 1 or -1.  If -1, the reflected ray
* passed through the plane tangent to the incident surface at the
* point of incidence.  If 1, the incident ray is reflected in the
* proper direction and program execution continues.
*
        IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) GO TO 111
*
* Determine if reflected ray leaves the optics module.
*
        RADIUS = DSQRT(X*X + Y*Y)
        IF(RADIUS.LT.RC) THEN
          Z = HREF
          MM = 10
          CALL CAPCHK (X1,Y1,Z1,X,Y,Z,MM)
          NCALL = NCALL + 1
        ENDIF
      ENDIF
*_____________________________________________________________
*                                                             |
* Loop complete.                                              |
*_____________________________________________________________|
*
* Determine if ray is incident on any part of the reflector cap.
* The DO 20 and DO 30 loops test each conical surface of the
* reflector cap to determine whether an energy bundle impinges
* the reflector cap or not.
*
  111 CONTINUE
      DO 20 K=1,2
      DO 30 I=M,N
        A = VOX*VOX + VOY*VOY - VOZ*VOZ*V(K)
        B = 2.0D0*X1*VOX + 2.0D0*Y1*VOY + 2.0D0*APEX(I)*VOZ*V(K)
        B = B- 2.0D0*Z1*VOZ*V(K)
        C = X1*X1 + Y1*Y1
        C = C - V(K)*(Z1*Z1 + APEX(I)*APEX(I) - 2.0D0*APEX(I)*Z1)
*
* If a reflected ray passes on the outside of a cone mathematically,
* the argument of the square root in the quadratic equation will be
* negative.  Since this can not happen physically within the optics
* module, it is ignored and program execution continues.
*
        ARG = B*B - 4.0D0*A*C
        IF(ARG.LE.1.D-10) GO TO 30
*
```

```
* Determine the ray (vector) direction.
*
      T=-B + DSQRT(ARG)
*
* The DO 40 loop simply computes the x, y, and z coordinates of a
* possible intersection by looping through the plus/minus term of
* the quadratic equation.
*
     DO 40 J=1,2
      T = T/(2.0D0*A)
      X = T*VOX + X1
      Y = T*VOY + Y1
      Z = T*VOZ + Z1
*
* If the coordinates of this intersection coincide with the previous
* intersection, change the sign in the quadratic equation and
* continue program execution.
*
      IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     &   DABS(Z-Z1).LT.1.D-10) THEN
        IF(J.EQ.2) GO TO 30
        T = -B - DSQRT(ARG)
        GO TO 40
      ENDIF

      VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
      VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
      VMAG1 = DSQRT(VMAG1)
      VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
      VMAG2 = DSQRT(VMAG2)

      IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) THEN
        IF(J.EQ.2) GO TO 30
        T = -B - DSQRT(ARG)
        GO TO 40
      ENDIF

      RADIUS = DSQRT(X*X + Y*Y)
*
* The last conical section of the reflector cap toward the active
* flake is longer than the others, this is taken into account in
* this IF-THEN-ELSE loop.
*
      IF(I.EQ.4.OR.I.EQ.5) THEN
        IF(I.EQ.4) RA = D4/2.0D0
        IF(I.EQ.5) RA = RC
      ELSE
        RA = D1/2.0D0
      ENDIF
*
* If an intersection occurs within the specified limits of the
* conical surfaces of the reflector cap, exit DO loops and record
* point of incidence and starting point of energy bundle in
* subprogram CAPCHK.
*
      IF(M.EQ.1) THEN
        IF(RADIUS.GE.RA.AND.RADIUS.LE.RB.AND.Z.LT.APEX(I)) THEN
          MM = I
          CALL CAPCHK (X1,Y1,Z1,X,Y,Z,MM)
          NCALL = NCALL + 1
          GO TO 30
        ENDIF
      ELSE
        IF(RADIUS.GE.RA.AND.RADIUS.LE.RB.AND.Z.GT.APEX(I).AND.
     &     Z.GT.HREF) THEN
          MM = I
          CALL CAPCHK (X1,Y1,Z1,X,Y,Z,MM)
```

```
            NCALL = NCALL + 1
            GO TO 30
          ENDIF
        ENDIF
*
* If no intersections occur on this conic section, continue looping.
*
        IF(J.EQ.2) GO TO 30
*
* Change the sign of the root term and continue looping.
*
        T = -B -DSQRT(ARG)
   40 CONTINUE
   30 CONTINUE
*
* Change the direction of the conical sections toward the flake.
*
        M = 5
        N = 8
   20 CONTINUE
*
**************** cylindrical surface of reflector cap ***************
*
* If no intersections occur along the conical surfaces of the
* reflector cap, check the cylindrical surface of the cap.
*
      A = VOX*VOX + VOY*VOY
      B = 2.0D0*(X1*VOX + Y1*VOY)
      C = X1*X1 + Y1*Y1 - (D4*D4/4.0D0)
*
* If a ray is parallel to the axis of the cylindrical surface, no
* intersection can possibly occur.
*
      IF(DABS(A).LE.1.D-10) GO TO 1

      ARG = B*B - 4.0D0*A*C
*
* If the discriminate is negative, ray passes, mathematically,
* on the outside of this cylindrical surface.
*
      IF(ARG.LE.1.D-10) GO TO 1
*
* Determine the ray(vector) direction.
*
      T = -B + DSQRT(ARG)
*
* The DO 5 loop determine a possible intersection with the
* cylindrical surface of the reflector cap.  Twice through
* the loop represents the plus/minus term of the quadratic
* formula.
*
      DO 5 I=1,2
        T = T/(2.0D0*A)
        X = T*VOX + X1
        Y = T*VOY + Y1
        Z = T*VOZ + Z1

        IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     &    DABS(Z-Z1).LT.1.D-10) THEN
          IF(I.EQ.2) GO TO 1
          T = -B - DSQRT(ARG)
          GO TO 5
        ENDIF

        VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
        VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
        VMAG1 = DSQRT(VMAG1)
```

```
        VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
        VMAG2 = DSQRT(VMAG2)

        IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) THEN
          IF(I.EQ.2) GO TO 1
          T = -B - DSQRT(ARG)
          GO TO 5
        ENDIF
*
* If ray intersects cylindrical surface within specified limits,
* record the coordinates of the point of incidence and the ray's
* starting point.
*
        IF(Z.GE.HREF+C5.AND.Z.LE.HREF+H1) THEN
          MM = 9
          CALL CAPCHK (X1,Y1,Z1,X,Y,Z,MM)
          NCALL = NCALL + 1
        ENDIF
*
        IF(I.EQ.2) GO TO 1

        T = -B - DSQRT(ARG)

    5 CONTINUE
*
* Assign values of absorptivity and reflectivity to the part of
* the reflector cap impinged by an energy bundle.  Also determine
* the unit normal at the point of incidence for use by subprograms
* VECTOR and REFLECT.
*
    1 CONTINUE
      IF(NCALL.EQ.0) THEN
        FLAG = 1
        RETURN
      ENDIF

      MM = 15
      CALL CAPCHK (X1,Y1,Z1,X,Y,Z,MM)

      IF(MM.EQ.10) THEN
        MM = 35
        ABS = 1.0
      ENDIF

      IF(MM.EQ.9) THEN
        UNX = 2.0D0 * X/D4
        UNY = 2.0D0 * Y/D4
        UNZ = 0.0D0
        ABS = ABS2
        REFR = REFR2
      ENDIF

      IF(MM.GE.1.AND.MM.LE.4) THEN
        ABS = ABS1
        REFR = REFR1
        UN = DSQRT(X*X + Y*Y + V(1)*V(1)*(Z-APEX(MM))*(Z-APEX(MM)))
        UNX = X / UN
        UNY = Y / UN
        UNZ = -V(1)*(Z-APEX(MM)) / UN
      ENDIF

      IF(MM.GE.5.AND.MM.LE.8) THEN
        ABS = ABS2
        REFR = REFR2
        UN = DSQRT(X*X + Y*Y + V(2)*V(2)*(Z-APEX(MM))*(Z-APEX(MM)))
        UNX = X/ UN
        UNY = Y / UN
```

```
        UNZ = -V(2)*(Z-APEX(MM)) / UN
      ENDIF
*
* Record the point of incidence of emitted energy bundle and its
* starting point.  Set flag and return to subprogram OPTICS.
*
      NN = 1
      CALL COORD (X1,Y1,Z1,X,Y,Z)
      FLAG = 1
      RETURN
*
* If subprogram COORD determines an intersection on the reflector
* cap to be the shortest distance between the emission site and
* site of incidence, program execution is returned here to determine
* whether the ray is absorbed or reflected.
*
    2 CONTINUE
      CALL RANDOM (R)
*
* A random number is chosen and compared to the absorptivity of
* the point of incidence on the reflector cap. If the random
* number chosen is less than or equal to the value of absorptivity
* corresponding to that part of the reflector cap, the incident ray
* is absorbed.  If not,...
*
      IF(R.LE.ABS) THEN
        DF(II,MM) = DF(II,MM) + 1.D0
        FLAG = 0
        RETURN
      ENDIF
      NTEST = 1
*
* ...another random number is chosen to determine how the incident
* ray is reflected.  If the random number is less than or equal to
* the value of the reflectivity of that part of the reflector cap,
* the ray is specularly reflected.  If not, a diffuse reflection
* occurs.
*
      CALL RANDOM (R)
      IF(R.LE.REFR) THEN
        CALL REFLECT(X1,Y1,Z1,X,Y,Z,UNX,UNY,UNZ,VOX,VOY,VOZ)
      ELSE
        X1 = X
        Y1 = Y
        Z1 = Z
        CALL VECTOR (X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
      ENDIF

      RETURN
      END




*******************************************************************
*                                                                 *
*                        SUBROUTINE CAPCHK                        *
*                                                                 *
* This subroutine records the coordinates of any emitted energy   *
* bundles incident on any surface of the reflector cap.  Since an *
* emitted energy bundle may hit several surfaces mathematically,  *
* it may only hit one surface physically.  Thus, the distance     *
* formula is used to determine the shortest distance to one of    *
* the recorded coordinates of a given surface, given the          *
* coordinates of the energy bundle's emission site.               *
*                                                                 *
```

```
************************************************************************
*
      SUBROUTINE CAPCHK (X1,Y1,Z1,X,Y,Z,NN)
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Define storage variables, size and type.
*
      REAL*8 XN(15),YN(15),ZN(15),X1N(15),Y1N(15),Z1N(15),DIST(15)
*
* This is a flag to compute the distance from the emission site to the
* coordinates of surfaces that have intercepted the emitted energy
* bundle.
*
      IF(NN.EQ.15) GO TO 1
*
* Record coordinates where energy bundle is intercepted and which
* surface intercepts it (variable NN is a number designated to a
* particular part of the optics module).
*
      XN(NN) = X
      YN(NN) = Y
      ZN(NN) = Z
      X1N(NN) = X1
      Y1N(NN) = Y1
      Z1N(NN) = Z1
      RETURN
*
* Execution continues to determine shortest distance from emission
* site to intercepting surface.
*
    1 CONTINUE

      JJ = 0
      DO 10 I=1,15
        DISTNCE = (XN(I)-X1N(I))*(XN(I)-X1N(I))
        DISTNCE = DISTNCE + (YN(I)-Y1N(I))*(YN(I)-Y1N(I))
        DISTNCE = DISTNCE + (ZN(I)-Z1N(I))*(ZN(I)-Z1N(I))
        DISTNCE = DSQRT(DISTNCE)
*
* If no coordinates are recorded, skip that particular part of the
* optics module.
*
        IF(DISTNCE.LE.1.E-8) THEN
          GO TO 10
        ELSE
          JJ = JJ + 1
          IF(JJ.EQ.1) DISTOLD = DISTNCE
*
* Record coordinates of surface with shortest distance from emission
* site.  Return to subprogram OPTICS.
*
          IF(DISTNCE.LE.DISTOLD) THEN
            NN = I
            X = XN(I)
            Y = YN(I)
            Z = ZN(I)
            X1 = X1N(I)
            Y1 = Y1N(I)
            Z1 = Z1N(I)
            DISTOLD = DISTNCE
          ENDIF
        ENDIF
   10 CONTINUE
*
```

```
* Reset array of coordinates to zero.
*
      DO 20 J=1,15
        XN(J) = 0.0D0
        YN(J) = 0.0D0
        ZN(J) = 0.0D0
        X1N(J) = 0.0D0
        Y1N(J) = 0.0D0
        Z1N(J) = 0.0D0
   20 CONTINUE

      RETURN
      END
```

```
*********************************************************************
*                                                                   *
*                          SUBROUTINE SPIDER                        *
*                                                                   *
* This subprogram monitors the top of the secondary mirror mount    *
* (spider).  If an emitted energy bundle is intercepted by this     *
* part of the optics module, its absorption or reflection is        *
* accounted for.                                                    *
*                                                                   *
*********************************************************************
*
      SUBROUTINE SPIDER (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Specify size and type of flags and storage parameters.
*
      INTEGER*2 FLAG
      REAL*8 KAPPA1,KAPPA2,DF(40,40)
*
* Place all common variables in one, unique storage block.
*
      COMMON /GEOM/ H1,D1,D2,THETA1,C1,C2,C3,C4,C5,H2,D3,D4,D5
      COMMON /GEOM/ DEL1,DEL2,THETA2,PHI,H3,HL1,HL2,D6,THETA3
      COMMON /GEOM/ H4,HCURVE,D7,KAPPA1,H5,D8,KAPPA2,THETA4,H6,D9
      COMMON /GEOM/ THETA5,DIM1,DIM2,HREF,DREF

      COMMON /MATL/ ABS1,ABS2,ABS3,ABS4,ABS5,ABS6,ABS7,ABS8,ABS9
      COMMON /MATL/ ABS10,REFR1,REFR2,REFR3,REFR4,REFR5,REFR6
      COMMON /MATL/ REFR7,REFR8,REFR9,REFR10

      COMMON /FLAGS/ NFLAG,NTEST,NN
      COMMON /DISTR/ DF,II

      NTEST = 0
      IF(NFLAG.EQ.1) GO TO 2
      NTAG = 0
*
* Determine if energy bundle strikes the hub (center) of secondary
* mirror mount.
*
      IF(Z1.LT.HREF+H1) THEN
        T = ((HREF+H1) - Z1)/VOZ
        X = T*VOX + X1
        Y = T*VOY + Y1
        Z = HREF + H1
```

```
      IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     &   DABS(Z-Z1).LT.1.D-10) GO TO 111
*
* The nature of this program makes it necessary to be sure a ray
* doesn't actually penetrate a given surface to be incident on
* another.  This has been taken care of by using the dot product
* of the directional vector, T, and the reflection/emission vector
* VOX(or VOY/VOZ).
*
      VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
      VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
      VMAG1 = DSQRT(VMAG1)
      VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
      VMAG2 = DSQRT(VMAG2)
*
* Determine if the dot product is 1 or -1.  If -1, the reflected ray
* passed through the plane tangent to the incident surface at the
* point of incidence.  If 1, the incident ray is reflected in the
* proper direction and program execution continues.
*
      IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) GO TO 111

      RADIUS = DSQRT(X*X + Y*Y)
      TRAD = D3/2.0D0

      IF(RADIUS.LE.TRAD) THEN
        UNX = 0.0D0
        UNY = 0.0D0
        UNZ = 1.0D0
        ABS = ABS3
        REFR = REFR3
        NN = 2
        GO TO 1
      ENDIF
     ENDIF
*
  111 CONTINUE
     PI = DACOS(-1.0D0)
     ZA = HREF + H1
     ZB = HREF + H1 + DEL2
     PSI = PHI * PI/180.0D0
     N = INT(2.0D0 * PI / PSI)
*
* Compute the coefficients of the equations of the intersecting
* planes for each leg of the secondary mirror mount.
*
     DO 20 I=0,N-1
      ANGLE = DFLOAT(I)*PSI
      XA = D3/2.0D0 * DCOS(ANGLE)
      XB = D4/2.0D0 * DCOS(ANGLE)
      YA = D3/2.0D0 * DSIN(ANGLE)
      YB = D4/2.0D0 * DSIN(ANGLE)
      IF(DCOS(ANGLE).EQ.0.0D0) THEN
        YC = YA
        YD = YB
      ELSE
        YC = YA + (DEL1/2.0D0)/DABS(DCOS(ANGLE))
        YD = YB - (DEL1/2.0D0)/DABS(DCOS(ANGLE))
      ENDIF
*
*    ***** plane 1 *****
*

      COEF1 = (YB-YA) * (ZB-ZA)
      COEF2 = -(XB-XA) * (ZB-ZA)
      COEF3 = (XB-XA) * (YC-YA)
```

```
      R1 = COEF1*VOX + COEF2*VOY + COEF3*VOZ
      S1 = COEF1*(XA-X1) + COEF2*(YA-Y1) + COEF3*(ZA-Z1)
*
*    ***** plane 2 *****
*
      COEF4 = COEF1
      COEF5 = COEF2
      COEF6 = (XB-XA)*(YD-YA) - (XB-XA)*(YB-YA)

      R2 = COEF4*VOX + COEF5*VOY + COEF6*VOZ
      S2 = COEF4*(XA-X1) + COEF5*(YA-Y1) + COEF6*(ZA-Z1)
*
* Determine whether energy bundle is incident on this leg of the
* spider.
*
      IF(Z1.GT.ZB) THEN
        T = (ZB-Z1)/VOZ
        X = T*VOX + X1
        Y = T*VOY + Y1
        Z = ZB

      IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     &  DABS(Z-Z1).LT.1.D-10) GO TO 3

      VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
      VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
      VMAG1 = DSQRT(VMAG1)
      VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
      VMAG2 = DSQRT(VMAG2)

      IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) GO TO 3
      IF(DCOS(ANGLE).GE.0.0D0.AND.X.LT.0.0D0) GO TO 3
      IF(DCOS(ANGLE).LT.0.0D0.AND.X.GT.0.0D0) GO TO 3
      IF(DSQRT(X*X + Y*Y).LE.D3/2.0D0) GO TO 3
*
* Rotate the coordinate axes onto the legs of the spider, to
* determine where reflected rays hit the underside of the
* secondary mirror mount.
*
      XPRIME = X*DCOS(ANGLE) + Y*DSIN(ANGLE)
      YPRIME = -X*DSIN(ANGLE) + Y*DCOS(ANGLE)

      IF(YPRIME.GE.-DEL1/2.0D0.AND.YPRIME.LE.DEL1/2.0D0) THEN
        UNX = 0.0D0
        UNY = 0.0D0
        UNZ = -1.0D0
        ABS = ABS2
        REFR = REFR2
        NN = 3
        MM = I+1
        NTAG = 1
        GO TO 1
      ELSE
        GO TO 3
      ENDIF
    ENDIF

  3 CONTINUE
    T = S1/R1

    DO 30 J = 1,2
      X = T*VOX + X1
      Y = T*VOY + Y1
      Z = T*VOZ + Z1

      IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     &  DABS(Z-Z1).LT.1.D-10) THEN
```

```
          IF(J.EQ.2) GO TO 30
          T = S2/R2
          GO TO 30
        ENDIF

        VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
        VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
        VMAG1 = DSQRT(VMAG1)
        VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
        VMAG2 = DSQRT(VMAG2)

        IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) THEN
          IF(J.EQ.2) GO TO 30
          T = S2/R2
          GO TO 30
        ENDIF
*
* Test the actual boundaries of the spider for interception of
* energy bundle.
*
        IF(Z.GE.ZA.AND.Z.LE.ZB) THEN

          IF(DCOS(ANGLE).GE.0.0D0.AND.X.LT.0.0D0) THEN
            IF(J.EQ.2) GO TO 30
            T = S2/R2
            GO TO 30
          ENDIF

          IF(DCOS(ANGLE).LT.0.0D0.AND.X.GT.0.0D0) THEN
            IF(J.EQ.2) GO TO 30
            T = S2/R2
            GO TO 30
          ENDIF

          IF(DSQRT(X*X + Y*Y).LE.D3/2.0D0) THEN
            IF(J.EQ.2) GO TO 30
            T = S2/R2
            GO TO 30
          ENDIF
*
* Compute the unit normal to the spider leg that intercepts
* the energy bundle.
*
        IF(DCOS(ANGLE).GE.0.0D0) THEN
          IF(J.EQ.1) THEN
            UNX = COEF1/DSQRT(COEF1**2 + COEF2**2 + COEF3**2)
            UNY = COEF2/DSQRT(COEF1**2 + COEF2**2 + COEF3**2)
            UNZ = COEF3/DSQRT(COEF1**2 + COEF2**2 + COEF3**2)
          ELSE
            UNX = COEF1/DSQRT(COEF1**2 + COEF2**2 + COEF6**2)
            UNY = -COEF2/DSQRT(COEF1**2 + COEF2**2 + COEF6**2)
            UNZ = -COEF6/DSQRT(COEF1**2 + COEF2**2 + COEF6**2)
          ENDIF
          ENDIF

          IF(DCOS(ANGLE).LT.0.0D0) THEN
          IF(J.EQ.1) THEN
            UNX = -COEF1/DSQRT(COEF1**2 + COEF2**2 + COEF3**2)
            UNY = -COEF2/DSQRT(COEF1**2 + COEF2**2 + COEF3**2)
            UNZ = -COEF3/DSQRT(COEF1**2 + COEF2**2 + COEF3**2)
          ELSE
            UNX = COEF1/DSQRT(COEF1**2 + COEF2**2 + COEF6**2)
            UNY = COEF2/DSQRT(COEF1**2 + COEF2**2 + COEF6**2)
            UNZ = COEF6/DSQRT(COEF1**2 + COEF2**2 + COEF6**2)
          ENDIF
        ENDIF
```

```
          ABS = ABS3
          REFR = REFR3
          NN = 3
          MM = I+1
          GO TO 1
        ENDIF
      T = S2/R2

   30 CONTINUE
*
* Go to the next leg of the spider to see if energy bundle strikes it.
*
   20 CONTINUE
      FLAG = 1
      RETURN
*
* Record the point of incidence of emitted energy bundle and its
* starting point.  Set flag and return to subprogram OPTICS.
*

    1 CONTINUE
      CALL COORD (X1,Y1,Z1,X,Y,Z)
      FLAG = 1
      RETURN

    2 CONTINUE
      IF(NN.EQ.2) THEN
*
* If subprogram COORD determines an intersection with the spider
* to be the shortest distance between the emission site and
* site of incidence, program execution is returned here to determine
* whether the ray is absorbed or reflected.
*
*
* A random number is chosen and compared to the absorptivity of the
* spider.  If the random number chosen is less than or equal to the
* value of absorptivity corresponding to that part of the spider,
* the incident ray is absorbed.  If not,...
*
        CALL RANDOM (R)
        IF(R.LE.ABS) THEN
          DF(II,10) = DF(II,10) + 1.D0
          FLAG = 0
          RETURN
        ENDIF
      ELSE
        CALL RANDOM (R)
        IF(R.LE.ABS) THEN
          IF(NTAG.EQ.1.AND.Z1.GT.ZB) THEN
            DF(II,MM+13) = DF(II,MM+13) + 1.D0
          ELSE
            DF(II,MM+10) = DF(II,MM+10) + 1.D0
          ENDIF
          FLAG = 0
          RETURN
        ENDIF
      ENDIF

      FLAG = 1
      NTEST = 1
*
* ...another random number is chosen to determine how the incident
* ray is reflected.  If the random number is less than or equal to
* the value of the reflectivity of the top of the mirror mount,
* the ray is specularly reflected.  If not, a diffuse reflection
* occurs.
*
```

```
      CALL RANDOM (R)
      IF(R.LE.REFR) THEN
        CALL REFLECT(X1,Y1,Z1,X,Y,Z,UNX,UNY,UNZ,VOX,VOY,VOZ)
      ELSE
        X1 = X
        Y1 = Y
        Z1 = Z
        CALL VECTOR (X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
      ENDIF

      RETURN
      END




**********************************************************************
*                                                                    *
*                        SUBROUTINE MIDCYL                           *
*                                                                    *
* This subprogram monitors the outer perimeter of the hub of the     *
* secondary mirror mount.  If an emitted energy bundle is            *
* intercepted by this part of the optics module, its absorption      *
* or reflection is accounted for.                                    *
*                                                                    *
**********************************************************************
*
      SUBROUTINE MIDCYL (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Specify storage parameters, size and type.
*
      INTEGER*2 FLAG
      REAL*8 KAPPA1,KAPPA2,T(2),DF(40,40)
*
* Place all common variables in one, unique storage block.
*
      COMMON /GEOM/ H1,D1,D2,THETA1,C1,C2,C3,C4,C5,H2,D3,D4,D5
      COMMON /GEOM/ DEL1,DEL2,THETA2,PHI,H3,HL1,HL2,D6,THETA3
      COMMON /GEOM/ H4,HCURVE,D7,KAPPA1,H5,D8,KAPPA2,THETA4,H6,D9
      COMMON /GEOM/ THETA5,DIM1,DIM2,HREF,DREF

      COMMON /MATL/ ABS1,ABS2,ABS3,ABS4,ABS5,ABS6,ABS7,ABS8,ABS9
      COMMON /MATL/ ABS10,REFR1,REFR2,REFR3,REFR4,REFR5,REFR6
      COMMON /MATL/ REFR7,REFR8,REFR9,REFR10

      COMMON /FLAGS/ NFLAG,NTEST,NN
      COMMON /DISTR/ DF,II
*
* Initialize necessary data.
*
      NTEST = 0
      RAD1 = D3/2.0D0
      IF(NFLAG.EQ.1) GO TO 1
      HA = HREF + H1
      HB = HA + H2 - H5
*
* Determine if ray is incident on the outer perimeter of the hub
* of the secondary mirror mount.
*
      A = VOX*VOX + VOY*VOY
      B = 2.0D0*X1*VOX + 2.0D0*Y1*VOY
```

```
      C = X1*X1 + Y1*Y1 - RAD1*RAD1
*
* If radiation is incident perpendicular to the aperture of the
* optics module it cannot possibly hit the cylindrical portion
* of the secondary mirror mount.
*
      IF(DABS(A).LE.1.E-8) THEN
        FLAG = 1
        RETURN
      ENDIF
*
* Compute the discriminate of the quadratic equation.
*
      ARG = B*B - 4.0D0*A*C
*
* If a reflected ray passes on the outside of a cylinder,
* the discriminate will be negative.
* The argument is ignored and program execution continues.
*
      IF(ARG.LT.0.0D0) THEN
        FLAG = 1
        RETURN
      ENDIF
*
* Determine the ray (vector) direction.
*
      T(1) = (-B + DSQRT(ARG))/(2.0D0*A)
      T(2) = (-B - DSQRT(ARG))/(2.0D0*A)
*
      DO 10 I=1,2
        X = T(I)*VOX + X1
        Y = T(I)*VOY + Y1
        Z = T(I)*VOZ + Z1
*
* The distance from the shot origin to the mirror mount is computed
* to make sure shot hits the side of the mirror mount closest to the
* shot origin.
*
        IF(I.EQ.1) THEN
          DIST1 = DSQRT((X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1))
        ELSE
          DIST2 = DSQRT((X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1))
        ENDIF

   10 CONTINUE

      IF(DIST1.LE.DIST2) THEN
        TSHOT = T(1)
      ELSE
        TSHOT = T(2)
      ENDIF
*
* Compute the coordinates of the point of incidence.
*
    2 CONTINUE
      X = TSHOT*VOX + X1
      Y = TSHOT*VOY + Y1
      Z = TSHOT*VOZ + Z1
*
* If the coordinates of this intersection coincide with the
* previous intersection, return to subprogram OPTICS and
* continue program execution.
*
      IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     & DABS(Z-Z1).LT.1.D-10) THEN
        FLAG = 1
        RETURN
```

```
      ENDIF
*
* The nature of this program makes it necessary to be sure a ray
* doesn't actually penetrate a given surface to be incident on
* another.  This has been taken care of using the dot product.
*
      VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
      VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
      VMAG1 = DSQRT(VMAG1)
      VMAG2 = VOX*VOX + VOY*VOY+ VOZ*VOZ
      VMAG2 = DSQRT(VMAG2)
*
* Determine if the dot product is 1 or -1.  If -1, the reflected ray
* passed through the plane tangent to the incident surface.  Return
* to the subprogram OPTICS.  If 1, the incident ray is reflected in
* the proper direction and program execution continues.
*
      IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) THEN
        FLAG = 1
        RETURN
      ENDIF
*
* If an intersection occurs on the perimeter of the hub of the
* mirror mount, record point of incidence and starting point of
* energy bundle in subprogram COORD.  Set flag and return to
* subprogram OPTICS.
*
      IF(Z.GE.HA.AND.Z.LE.HB) THEN
        NN = 4
        CALL COORD(X1,Y1,Z1,X,Y,Z)
        FLAG = 1
        RETURN
      ELSE
        FLAG = 1
        RETURN
      ENDIF
*
* If subprogram COORD determines an intersection on the perimeter of
* the hub to be the shortest distance between the emission site and
* site of incidence, program execution is returned here to determine
* whether the ray is absorbed or reflected.
*
    1 CONTINUE
      CALL RANDOM (R)
*
* A random number is chosen and compared to the absorptivity of
* the perimeter of the secondary mirror mount's hub.  If the random
* number chosen is less than or equal to the value of absorptivity
* corresponding to this part of the mirror mount, the incident ray
* is absorbed.  If not,...
*
      IF(R.LE.ABS4) THEN
        DF(II,17) = DF(II,17) + 1.D0
        FLAG = 0
        RETURN
      ELSE
        UNX = -X/RAD1
        UNY = -Y/RAD1
        UNZ = 0.0D0
        FLAG = 1
      ENDIF

      NTEST = 1
*
* ...another random number is chosen to determine how the incident
* ray is reflected.  If the random number is less than or equal to
* the value of the reflectivity of this part of the mirror mount,
```

```
* the ray is specularly reflected.  If not, a diffuse reflection
* occurs.
*
      CALL RANDOM (R)
      IF(R.LE.REFR4) THEN
        CALL REFLECT (X1,Y1,Z1,X,Y,Z,UNX,UNY,UNZ,VOX,VOY,VOZ)
      ELSE
        X1 = X
        Y1 = Y
        Z1 = Z
        CALL VECTOR (X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
      ENDIF

      RETURN
      END




**********************************************************************
*                                                                    *
*                         SUBROUTINE MIRROR                          *
*                                                                    *
* This subprogram monitors the perimeter of the secondary mirror.    *
* If an emitted energy bundle is intercepted by this part of the     *
* optics module, its absorption or reflection is accounted for.      *
*                                                                    *
**********************************************************************
*
      SUBROUTINE MIRROR (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Define storage variables, size and type.
*
      INTEGER*2 FLAG
      REAL*8 KAPPA1,KAPPA2,T(2),DF(40,40)
*
* Place all common variables in one, unique storage block.
*
      COMMON /GEOM/ H1,D1,D2,THETA1,C1,C2,C3,C4,C5,H2,D3,D4,D5
      COMMON /GEOM/ DEL1,DEL2,THETA2,PHI,H3,HL1,HL2,D6,THETA3
      COMMON /GEOM/ H4,HCURVE,D7,KAPPA1,H5,D8,KAPPA2,THETA4,H6,D9
      COMMON /GEOM/ THETA5,DIM1,DIM2,HREF,DREF

      COMMON /MATL/ ABS1,ABS2,ABS3,ABS4,ABS5,ABS6,ABS7,ABS8,ABS9
      COMMON /MATL/ ABS10,REFR1,REFR2,REFR3,REFR4,REFR5,REFR6
      COMMON /MATL/ REFR7,REFR8,REFR9,REFR10
      COMMON /FLAGS/ NFLAG,NTEST,NN
      COMMON /DISTR/ DF,II
*
* Initialize necessary variables.
*
      NTEST = 0
      IF(NFLAG.EQ.1) GO TO 1
      PI = DACOS(-1.0D0)
      PSI = THETA4 * PI/180.0D0
      RB  = D8/2.0D0
      CAL = RB / DTAN(PSI)
      V = DTAN(PSI) * DTAN(PSI)
*
* Compute the distance of the edge of the secondary mirror from
* the coordinate axes.
*
```

```
      HB = (-RB * RB / (2.D0*KAPPA2)) + HREF + H1 + H2
      APEX = HB - CAL
      HA = HREF + H1 + H2 - H5
*
* Determine if ray is incident on the perimeter of the mirror.
*
      A = VOX*VOX + VOY*VOY - VOZ*VOZ*V
      B = 2.0D0*X1*VOX + 2.0D0*Y1*VOY + 2.0D0*APEX*VOZ*V
      B = B - 2.0D0*Z1*VOZ*V
      C = X1*X1 + Y1*Y1 - V*(Z1*Z1 + APEX*APEX - 2.0D0*APEX*Z1)
*
* Compute discriminate of the quadratic formula.
*
      ARG = B*B - 4.0D0*A*C
*
* If the discriminate is negative, the emitted energy bundle passed
* the secondary mirror (cone).  A flag is set and execution returns
* to subprogram OPTICS.
*
      IF(ARG.LT.0.0D0) THEN
        FLAG = 1
        RETURN
      ENDIF
*
* Determine the ray (vector) direction.
*
      T(1) = (-B + DSQRT(ARG))/(2.0D0*A)
      T(2) = (-B - DSQRT(ARG))/(2.0D0*A)
*
* The DO 10 loop simply computes the x, y, and z coordinates of a
* possible intersection by looping through the plus/minus term of
* the quadratic equation.
*
      DO 10 I=1,2
        X = T(I)*VOX + X1
        Y = T(I)*VOY + Y1
        Z = T(I)*VOZ + Z1
*
* The distance from the shot origin to the mirror mount is computed
* to make sure shot hits the side of the mirror mount closest to the
* shot origin.
*
        IF(I.EQ.1) THEN
          DIST1 = DSQRT((X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1))
        ELSE
          DIST2 = DSQRT((X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1))
        ENDIF

   10 CONTINUE

      IF(DIST1.LE.DIST2) THEN
        TSHOT = T(1)
      ELSE
        TSHOT = T(2)
      ENDIF
*
* Compute the coordinates of the point of incidence.
*
    2 CONTINUE
      X = TSHOT*VOX + X1
      Y = TSHOT*VOY + Y1
      Z = TSHOT*VOZ + Z1

      IF(DABS(VOX).LE.1.E-8.AND.DABS(VOY).LE.1.E-8) THEN
      IF(Z.LT.APEX) THEN
        IF(TSHOT.EQ.T(1)) THEN
          TSHOT = T(2)
```

```
        ELSE
          TSHOT = T(1)
        ENDIF
        GO TO 2
      ENDIF
      ENDIF
*
* If the coordinates of this intersection coincide with the previous
* intersection...
*
      IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     &  DABS(Z-Z1).LT.1.D-10) GO TO 3
*
* The nature of this program makes it necessary to be sure a ray
* doesn't actually penetrate a given surface to be incident on
* another.  This has been taken care of using the dot product.
*
      VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
      VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
      VMAG1 = DSQRT(VMAG1)
      VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
      VMAG2 = DSQRT(VMAG2)
*
* Determine if the dot product is 1 or -1.  If -1, the reflected
* ray passed through the plane tangent to the incident surface.
* If 1, the incident ray is reflected in the proper direction
* and program execution continues.
*
      IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) GO TO 3
*
* If an intersection occurs within the specified limits of the
* perimeter of the secondary mirror, record point of incidence
* and starting point of energy bundle in subprogram COORD.  Set
* flag and exit subroutine.
*
      IF(Z.GE.HA.AND.Z.LE.HB) THEN
        NN = 5
        CALL COORD (X1,Y1,Z1,X,Y,Z)
*
* Compute unit normal to surface.  Used by subprograms REFLECT and
* VECTOR.
*
        UN = DSQRT(X*X + Y*Y + V*V*(Z-APEX)*(Z-APEX))
        UNX = -X / UN
        UNY = -Y / UN
        UNZ = V*(Z-APEX) / UN
        FLAG = 1
        RETURN
      ENDIF
*
* Dimensions of mirror do not match that of the mount.  There is a
* ridge between the mount and the mirror on the spider side.  This is
* taken into account in this loop.
*
    3 CONTINUE
      IF(Z1.LT.HA) THEN
        TA = (HA-Z1)/VOZ
        X = TA*VOX + X1
        Y = TA*VOY + Y1
        Z = HA

        IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     &    DABS(Z-Z1).LT.1.D-10) THEN
          FLAG = 1
          RETURN
        ENDIF
```

```
        VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
        VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
        VMAG1 = DSQRT(VMAG1)
        VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
        VMAG2 = DSQRT(VMAG2)
        IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) THEN
          FLAG = 1
          RETURN
        ENDIF

        R1 = (HA-APEX)*DTAN(PSI)
        R2 = D3/2.0D0
        RADIUS = DSQRT(X*X + Y*Y)
*
* Determine if the energy bundle intercepts the ridge on the
* secondary mirror.
*
        IF(RADIUS.GT.R2.AND.RADIUS.LT.R1.AND.Z.GT.APEX) THEN
          NN = 5
          CALL COORD (X1,Y1,Z1,X,Y,Z)
          UNX = 0.0D0
          UNY = 0.0D0
          UNZ = 1.0D0
          FLAG = 1
          RETURN
        ENDIF
      ENDIF
*
* If no intersections occurs, set flag and return to subprogram OPTICS.
*
      FLAG = 1
      RETURN
*
* If subprogram COORD determines an intersection on the perimeter of
* the mirror to be the shortest distance between the emission site and
* site of incidence, program execution is returned here to determine
* whether the ray is absorbed or reflected.
*
*
* A random number is chosen and compared to the absorptivity of
* the paint on the perimeter of the secondary mirror. If the random
* number chosen is less than or equal to the value of absorptivity
* corresponding to the perimeter of the mirror, the incident ray
* is absorbed.  If not,...
*
    1 CONTINUE
      CALL RANDOM (R)

      IF(R.LE.ABS4) THEN
        DF(II,18) = DF(II,18) + 1.D0
        FLAG = 0
        RETURN
      ENDIF
      NTEST = 1
*
* ...another random number is chosen to determine how the incident
* ray is reflected.  If the random number is less than or equal to
* the value of the reflectivity of the perimeter of the mirror,
* the ray is specularly reflected.  If not, a diffuse reflection
* occurs.
*
      CALL RANDOM (R)
        IF(R.LE.REFR4) THEN
          CALL REFLECT (X1,Y1,Z1,X,Y,Z,UNX,UNY,UNZ,VOX,VOY,VOZ)
        ELSE
          X1 = X
          Y1 = Y
```

```
          Z1 = Z
          CALL VECTOR (X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
        ENDIF

      RETURN
      END




************************************************************************
*                                                                      *
*                          SUBROUTINE MOUNT                            *
*                                                                      *
* This subprogram monitors the inner perimeter of the secondary        *
* mirror mount.  If an emitted energy bundle is intercepted by this    *
* part of the optics module, its absorption or reflection is           *
* accounted for.                                                       *
*                                                                      *
************************************************************************
*
      SUBROUTINE MOUNT (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Define storage variables, size and type.
*
      INTEGER*2 FLAG
      REAL*8 KAPPA1,KAPPA2,DF(40,40)
*
* Place all common variables in one, unique storage block.
*
      COMMON /GEOM/ H1,D1,D2,THETA1,C1,C2,C3,C4,C5,H2,D3,D4,D5
      COMMON /GEOM/ DEL1,DEL2,THETA2,PHI,H3,HL1,HL2,D6,THETA3
      COMMON /GEOM/ H4,HCURVE,D7,KAPPA1,H5,D8,KAPPA2,THETA4,H6,D9
      COMMON /GEOM/ THETA5,DIM1,DIM2,HREF,DREF

      COMMON /MATL/ ABS1,ABS2,ABS3,ABS4,ABS5,ABS6,ABS7,ABS8,ABS9
      COMMON /MATL/ ABS10,REFR1,REFR2,REFR3,REFR4,REFR5,REFR6
      COMMON /MATL/ REFR7,REFR8,REFR9,REFR10

      COMMON /DISTR/ DF,II
      COMMON /FLAGS/ NFLAG,NTEST,NN
*
* Initialize necessary variables.
*
      NTEST = 0
      IF(NFLAG.EQ.1) GO TO 1
      NTAG = 0
      PI = DACOS(-1.0D0)
      PSI = THETA2 * PI/180.0D0
      HA = HREF + H1
      HMOUNT = HA + H2
      RA = D5/2.0D0
      RB = D4/2.0D0
      RC = D6/2.0D0
      HTAPER = (RB-RA)*DTAN(PSI)
      ALPHA = PI/2.0D0 - PSI
      APEX = RB/DTAN(ALPHA) + HMOUNT - HTAPER
      V1 = DTAN(ALPHA) * DTAN(ALPHA)
*
* Determine if an energy bundle strikes the edge of the mount
* that extends over the detector housing.
*
```

```
      IF(Z1.GT.HMOUNT) THEN
        T = (HMOUNT-Z1)/VOZ
        X = T*VOX + X1
        Y = T*VOY + Y1
        Z = HMOUNT
*
* If the coordinates of this intersection coincide with the previous
* intersection, change the sign in the quadratic equation and
* continue program execution.
*
        IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     &    DABS(Z-Z1).LT.1.D-10) GO TO 4
*
* The nature of this program makes it necessary to be sure a ray
* doesn't actually penetrate a given surface to be incident on
* another.  This has been taken care of using the dot product.
*
        VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
        VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
        VMAG1 = DSQRT(VMAG1)
        VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
        VMAG2 = DSQRT(VMAG2)
*
* Determine if the dot product is 1 or -1.  If -1, the reflected ray
* passed through the plane tangent to the incident surface.  Return
* to the subprogram OPTICS.  If 1, the incident ray is reflected in
* the proper direction and program execution continues.
*
        IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) GO TO 4
*
* Continue computation.
*
        RADIUS = DSQRT(X*X + Y*Y)
        IF(RADIUS.GE.RA.AND.RADIUS.LE.RC) THEN
          NTAG = 1
          UNX = 0.0D0
          UNY = 0.0D0
          UNZ = -1.0D0
          REFR= REFR3
          GO TO 3
        ENDIF
      ENDIF
*
* Determine if ray is incident on the conical surface of the
* secondary mirror mount.  The DO 10 loop tests the conical surface
* of the mount to determine whether an energy bundle is incident
* or not.
*
*
*    ***** conical surface of secondary mirror mount *****
*
    4 CONTINUE
      A = VOX*VOX + VOY*VOY - VOZ*VOZ*V1
      B = 2.0D0*X1*VOX + 2.0D0*Y1*VOY + 2.0D0*APEX*VOZ*V1
      B = B - 2.0D0*Z1*VOZ*V1
      C = X1*X1 + Y1*Y1- V1*(Z1*Z1 + APEX*APEX - 2.0D0*APEX*Z1)
*
* Compute the discriminate of the quadratic equation.
*
      ARG = B*B - 4.0D0*A*C
*
* If the discriminate of the quadratic equation is negative,
* the reflected ray passes outside the quadric surface.
* This surface is ignored and program execution continues.
*
      IF(ARG.LE.1.D-10) GO TO 2
*
```

```
* Determine the ray (vector) direction.
*
      T = -B + DSQRT(ARG)
*
* The DO 10 loop simply computes the x, y, and z coordinates of a
* possible intersection by looping through the plus/minus term of
* the quadratic equation.
*
      DO 10 I=1,2
        T = T/(2.0D0*A)

        X = T*VOX + X1
        Y = T*VOY + Y1
        Z = T*VOZ + Z1
*
* If the coordinates of this intersection coincide with the previous
* intersection, change the sign in the quadratic equation and
* continue program execution.
*
        IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     &    DABS(Z-Z1).LT.1.D-10) THEN
          IF(I.EQ.2) GO TO 2
          T = -B - DSQRT(ARG)
          GO TO 10
        ENDIF
*
* The nature of this program makes it necessary to be sure a ray
* doesn't actually penetrate a given surface to be incident on
* another.  This has been taken care of using the dot product.
*
        VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
        VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
        VMAG1 = DSQRT(VMAG1)
        VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
        VMAG2 = DSQRT(VMAG2)
*
* Determine if the dot product is 1 or -1.  If -1, the reflected ray
* passed through the plane tangent to the incident surface.  Return
* to the subprogram OPTICS.  If 1, the incident ray is reflected in
* the proper direction and program execution continues.
*
        IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) THEN
          IF(I.EQ.2) GO TO 2
          T = -B - DSQRT(ARG)
          GO TO 10
        ENDIF

        RADIUS = DSQRT(X*X + Y*Y)
*
* If an intersection occurs within the specified limits of the
* conical surface of the mirror mount, exit DO loop and record
* point of incidence and starting point of energy bundle in
* subprogram COORD.
*

        IF(RADIUS.GE.RA.AND.RADIUS.LE.RB.AND.Z.LT.APEX) THEN
          UN = DSQRT(X*X + Y*Y + V1*V1*(Z-APEX)*(Z-APEX))
          UNX = X / UN
          UNY = Y / UN
          UNZ = -V1*(Z-APEX) / UN
          REFR = REFR5
          MM = 2
          GO TO 3
        ENDIF
*
* If no intersections occurs, exit loop and continue execution.
*
```

```
      IF(I.EQ.2) GO TO 2
*
* Change the sign of the root of the discriminate and continue
* looping.
*
      T = -B - DSQRT(ARG)
   10 CONTINUE
*
* If reflected ray did not hit conical surface of the mount,
* then determine whether or not ray hits the inner cylindrical
* perimeter of the mirror mount.
*
*
*    ***** cylindrical surface of secondary mirror mount *****
*
    2 CONTINUE
      A = VOX*VOX + VOY*VOY
      B = 2.0D0*(X1*VOX + Y1*VOY)
      C = X1*X1 + Y1*Y1 - (D4*D4/4.0D0)
*
* If a ray enters the optics module parallel to the axis of the
* instrument, it can not possibly intersect the cylindrical surface
* of the mirror mount.
*
      IF(DABS(A).LE.1.E-8) THEN
        FLAG = 1
        RETURN
      ENDIF
*
* Compute the discriminate of the quadratic equation.
*
      ARG = B*B - 4.0D0*A*C
*
      IF(ARG.LE.1.D-10) THEN
        FLAG = 1
        RETURN
      ENDIF
*
* Determine the ray(vector) direction.
*
      T = -B + DSQRT(ARG)
*
* The DO 20 loop determines a possible intersection with the
* cylindrical surface of the mirror mount.  Twice through
* the loop represents the plus/minus term of the quadratic
* formula.
*
      DO 20 J=1,2
        T = T/(2.0D0*A)

        X = T*VOX + X1
        Y = T*VOY + Y1
        Z = T*VOZ + Z1

        IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     &     DABS(Z-Z1).LT.1.D-10) THEN
          IF(J.EQ.2) THEN
            FLAG = 1
            RETURN
          ENDIF
          T = -B - DSQRT(ARG)
          GO TO 20
        ENDIF

        VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
        VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
        VMAG1 = DSQRT(VMAG1)
```

```
        VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
        VMAG2 = DSQRT(VMAG2)

        IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) THEN
          IF(J.EQ.2) THEN
            FLAG = 1
            RETURN
          ENDIF
          T = -B - DSQRT(ARG)
          GO TO 20
        ENDIF
*
* If ray intersects the cylindrical surface within specified limits,
* record the coordinates of the point of incidence and the ray's
* starting point.
*
        IF(Z.GE.HA.AND.Z.LE.HMOUNT-HTAPER) THEN
          UNX = X/(D4/2.0D0)
          UNY = Y/(D4/2.0D0)
          UNZ = 0.0
          REFR = REFR5
          MM = 1
          GO TO 3
        ENDIF
*
* If no intersection occurs, set flag and return to subprogram
* OPTICS.
*
        IF(J.EQ.2) THEN
          FLAG = 1
          RETURN
        ENDIF

        T = -B - DSQRT(ARG)

   20 CONTINUE
*
* Record the point of incidence of emitted energy bundle and its
* starting point.  Set flag and return to subprogram OPTICS.
*
    3 CONTINUE
      NN = 6
      CALL COORD (X1,Y1,Z1,X,Y,Z)
      FLAG = 1
      RETURN
*
* If subprogram COORD determines an intersection on the secondary
* mirror mount to be the shortest distance between the emission site
* and site of incidence, program execution is returned here to
* determine whether the ray is absorbed or reflected.
*
*
* A random number is chosen and compared to the absorptivity of
* the inner perimeter of the secondary mirror mount. If the random
* number chosen is less than or equal to the value of absorptivity
* corresponding to the secondary mirror mount, the incident ray
* is absorbed.  If not,...
*
    1 CONTINUE
      CALL RANDOM (R)
      IF(NTAG.EQ.1.AND.R.LE.ABS3) THEN
        DF(II,31) = DF(II,31) + 1.D0
        FLAG=0
        RETURN
      ENDIF

      IF(R.LE.ABS4) THEN
```

```
        DF(II,18+MM) = DF(II,18+MM) + 1.D0
        FLAG = 0
        RETURN
      ENDIF

      CALL RANDOM (R)
      NTEST = 1
*
* ...another random number is chosen to determine how the incident
* ray is reflected.  If the random number is less than or equal to
* the value of the reflectivity of the secondary mirror mount,
* the ray is specularly reflected.  If not, a diffuse reflection
* occurs.
*
      IF(R.LE.REFR) THEN
        CALL REFLECT (X1,Y1,Z1,X,Y,Z,UNX,UNY,UNZ,VOX,VOY,VOZ)
      ELSE
        X1 = X
        Y1 = Y
        Z1 = Z
        CALL VECTOR (X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
      ENDIF

      RETURN
      END




*******************************************************************
*                                                                 *
*                       SUBROUTINE DETCYL                         *
*                                                                 *
* This subprogram monitors the cylindrical surfaces of the detector *
* housing of the ERBE scanning radiometer.  If an emitted energy  *
* bundle is intercepted by this part of the optics module, its    *
* absorption or reflection is accounted for.                      *
*                                                                 *
*******************************************************************
*
      SUBROUTINE DETCYL (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Define storage variables, size and type.
*
      INTEGER*2 FLAG
      REAL*8 KAPPA1,KAPPA2,DF(40,40)
*
* Place all common variables in one, unique storage block.
*
      COMMON /GEOM/ H1,D1,D2,THETA1,C1,C2,C3,C4,C5,H2,D3,D4,D5
      COMMON /GEOM/ DEL1,DEL2,THETA2,PHI,H3,HL1,HL2,D6,THETA3
      COMMON /GEOM/ H4,HCURVE,D7,KAPPA1,H5,D8,KAPPA2,THETA4,H6,D9
      COMMON /GEOM/ THETA5,DIM1,DIM2,HREF,DREF

      COMMON /MATL/ ABS1,ABS2,ABS3,ABS4,ABS5,ABS6,ABS7,ABS8,ABS9
      COMMON /MATL/ ABS10,REFR1,REFR2,REFR3,REFR4,REFR5,REFR6
      COMMON /MATL/ REFR7,REFR8,REFR9,REFR10
      COMMON /FLAGS/ NFLAG,NTEST,NN

      COMMON /DISTR/ DF,II
*
```

```
* Initialize necessary variables.
*
      NTEST = 0
      IF(NFLAG.EQ.1) GO TO 1
      NTAG = 0
      PI = DACOS(-1.0D0)
      RA = DSQRT(HCURVE * 2.D0*KAPPA1)
      HMOUNT = HREF +H1+H2
      HB = HMOUNT+H3
      HDET1 = HB - HL1 - HL2
      HDET2 = HB - HL2
*
* Determine if an energy bundle strikes the secondary mirror mount
* overhang.
*
      IF(Z1.GT.HB) THEN
        T = (HB-Z1)/VOZ
        X = T*VOX + X1
        Y = T*VOY + Y1
        Z = HB
*
* The nature of this program makes it necessary to be sure a ray
* doesn't actually penetrate a given surface to be incident on
* another.  This has been taken care of using the dot product.
*
        VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
        VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
        VMAG1 = DSQRT(VMAG1)
        VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
        VMAG2 = DSQRT(VMAG2)
*
* Determine if the dot product is 1 or -1.  If -1, the reflected ray
* passed through the plane tangent to the incident surface.  Return
* to the subprogram OPTICS.  If 1, the incident ray is reflected in
* the proper direction and program execution continues.
*
        IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) GO TO 111
*
        RADIUS = DSQRT(X*X+Y*Y)
*
* Determine if ray hits the primary mirror overhang.
*
        IF(RADIUS.GT.D6/2.0D0.AND.RADIUS.LE.RA) THEN
          NN = 7
          CALL COORD (X1,Y1,Z1,X,Y,Z)
          UNX = 0.0D0
          UNY = 0.0D0
          UNZ = -1.0D0
          ABS = ABS1
          REFR = REFR1
          NTAG = 1
          FLAG = 1
          RETURN
        ENDIF
      ENDIF
*
* Determine if ray is incident on the cylindrical surfaces of the
* detector housing.
*
*    ***** cylindrical surface of detector housing *****
*
  111 CONTINUE
      A = VOX*VOX + VOY*VOY
      B = 2.0D0*(X1*VOX + Y1*VOY)
      C = X1*X1 + Y1*Y1 - (D6*D6/4.0D0)
*
* Compute the discriminate of the quadratic formula.
```

```
      ARG = B*B - 4.0D0*A*C
*
* If a reflected ray passes on the outside of a cylinder,
* the discriminate of the quadratic equation will be negative.
* Since this can not happen physically within the optics
* module. It is ignored and program execution continues.
*
      IF(ARG.LE.1.D-10) THEN
        FLAG = 1
        RETURN
      ENDIF
*
* Collimated radiation entering the optics module parallel to its
* axis cannot hit the surface of a cylinder; therefore, a flag is
* set and execution is returned to subprogram OPTICS.
*
      IF(DABS(A).LE.1.E-8) THEN
        FLAG = 1
        RETURN
      ENDIF
*
* Determine the ray (vector) direction.
*
      T = -B + DSQRT(ARG)
*
* The DO 10 loop simply computes the x, y, and z coordinates of a
* possible intersection by looping through the plus/minus term of
* the quadratic equation.
*
      DO 10 I=1,2
        T = T/(2.0D0*A)

        X = T*VOX + X1
        Y = T*VOY + Y1
        Z = T*VOZ + Z1
*
* If the coordinates of this intersection coincide with the previous
* intersection, change the sign on the root in the quadratic equation
* and continue program execution.
*
        IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     &    DABS(Z-Z1).LT.1.D-10) THEN
          IF(I.EQ.2) THEN
            FLAG = 1
            RETURN
          ENDIF
          T = -B - DSQRT(ARG)
          GO TO 10
        ENDIF
*
* The nature of this program makes it necessary to be sure a ray
* doesn't actually penetrate a given surface to be incident on
* another.  This has been taken care of using the dot product.
*
        VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
        VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
        VMAG1 = DSQRT(VMAG1)
        VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
        VMAG2 = DSQRT(VMAG2)
*
* Determine if the dot product is 1 or -1.  If -1, the reflected ray
* passed through the plane tangent to the incident surface.  Compute
* opposite direction of ray.  If 1, the incident ray is reflected in
* the proper direction and program execution continues.
*
        IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) THEN
```

```
          IF(I.EQ.2) THEN
            FLAG = 1
            RETURN
          ENDIF
          T = -B - DSQRT(ARG)
          GO TO 10
        ENDIF
*
* If an intersection occurs within the specified limits of the
* cylindrical surfaces of the detector housing, record
* point of incidence and starting point of energy bundle in
* subprogram COORD.
*
        IF(Z.GE.HMOUNT.AND.Z.LE.HDET1) THEN
          NN = 7
          CALL COORD (X1,Y1,Z1,X,Y,Z)
          UNX = X/(D6/2.0D0)
          UNY = Y/(D6/2.0D0)
          UNZ = 0.0
          ABS = ABS5
          REFR = REFR5
          FLAG = 1
          RETURN
        ENDIF

        IF(Z.GE.HDET2.AND.Z.LE.HB) THEN
          NN = 7
          CALL COORD (X1,Y1,Z1,X,Y,Z)
          UNX = X/(D6/2.0D0)
          UNY = Y/(D6/2.0D0)
          UNZ = 0.0
          ABS = ABS5
          REFR = REFR5
          FLAG = 1
          RETURN
        ENDIF

        IF(I.EQ.2) THEN
          FLAG = 1
          RETURN
        ENDIF
*
* Change the sign on the root and continue looping.
*
        T = -B - DSQRT(ARG)
   10 CONTINUE
*
* If subprogram COORD determines an intersection on the detector
* housing to be the shortest distance between the emission site and
* site of incidence, program execution is returned here to determine
* whether the ray is absorbed or reflected.
*
    1 CONTINUE
      CALL RANDOM (R)
*
* A random number is chosen and compared to the absorptivity of
* the cylindrical surfaces of the detector housing.  If the random
* number chosen is less than or equal to the value of absorptivity
* corresponding to this part of the detector housing, the incident
* ray is absorbed.  If not,...
*
      IF(R.LE.ABS) THEN

        IF(NTAG.EQ.1) THEN
          DF(II,32) = DF(II,32) + 1.D0
          FLAG = 0
          RETURN
```

```
        ENDIF

        IF(Z.LE.HDET1) THEN
          DF(II,22) = DF(II,22) + 1.D0
          FLAG = 0
          RETURN
        ELSE
          DF(II,25) = DF(II,25) + 1.D0
          FLAG = 0
          RETURN
        ENDIF
      ENDIF
*
* ...another random number is chosen to determine how the incident
* ray is reflected.  If the random number is less than or equal to
* the value of the reflectivity of the cylindrical surface of the
* detector housing, the ray is specularly reflected.  If not, a
* diffuse reflection occurs.
*
      CALL RANDOM (R)
*
* Compute unit vectors at point of incidence.
*
      NTEST = 1
      IF(R.LE.REFR) THEN
        CALL REFLECT (X1,Y1,Z1,X,Y,Z,UNX,UNY,UNZ,VOX,VOY,VOZ)
      ELSE
        X1 = X
        Y1 = Y
        Z1 = Z
        CALL VECTOR (X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
      ENDIF

      RETURN
      END




*********************************************************************
*                                                                   *
*                        SUBROUTINE CONE                            *
*                                                                   *
* This subprogram monitors the conical section of the ERBE scanning *
* radiometer's detector housing.  This reduces scattering of energy *
* close to the flake.  If an emitted energy bundle is intercepted   *
* by this part of the optics module, its absorption or reflection is *
* accounted for.                                                    *
*                                                                   *
*********************************************************************
*
      SUBROUTINE CONE (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Define storage variables and size.
*
      INTEGER*2 FLAG
      REAL*8 KAPPA1,KAPPA2,APEX(2),DF(40,40)
*
* Place all common variables in one, unique storage block.
*
      COMMON /GEOM/ H1,D1,D2,THETA1,C1,C2,C3,C4,C5,H2,D3,D4,D5
      COMMON /GEOM/ DEL1,DEL2,THETA2,PHI,H3,HL1,HL2,D6,THETA3
```

```
      COMMON /GEOM/ H4,HCURVE,D7,KAPPA1,H5,D8,KAPPA2,THETA4,H6,D9
      COMMON /GEOM/ THETA5,DIM1,DIM2,HREF,DREF

      COMMON /MATL/ ABS1,ABS2,ABS3,ABS4,ABS5,ABS6,ABS7,ABS8,ABS9
      COMMON /MATL/ ABS10,REFR1,REFR2,REFR3,REFR4,REFR5,REFR6
      COMMON /MATL/ REFR7,REFR8,REFR9,REFR10

      COMMON /FLAGS/ NFLAG,NTEST,NN
      COMMON /DISTR/ DF,II
*
* Initialize necessary variables.
*
      NTEST = 0
      IF(NFLAG.EQ.1) GOTO 1
      PI = DACOS(-1.0D0)
      OMEGA = (THETA3/2.0D0) * PI/180.0D0
      ALPHA = PI/2.0D0 - OMEGA
      DCONE = (HL1/2.0D0)/DTAN(OMEGA)
      RCONE = D6/2.0D0 + DCONE
      HA = HREF + H1 + H2 + H3 - HL1 - HL2
      HB = HA + HL1
      HCONE = HA + HL1/2.0D0
      VERTEX = RCONE / DTAN(ALPHA)
      APEX(1) = HCONE - VERTEX
      APEX(2) = HCONE + VERTEX
      V = DTAN(ALPHA) * DTAN(ALPHA)
*
* Determine if ray is incident on any part of the detector housing.
* The DO 10 and DO 20 loops test each conical surface of the detector
* housing to determine whether an energy bundle is incident
* or not.
*
      DO 10 I=1,2
        A = VOX*VOX + VOY*VOY - VOZ*VOZ*V
        B = 2.0D0*(X1*VOX + Y1*VOY) + 2.0D0*APEX(I)*VOZ*V
        B = B - 2.0D0*Z1*VOZ*V
        C = V*(Z1*Z1 + APEX(I)*APEX(I) - 2.0D0*APEX(I)*Z1)
        C = X1*X1 + Y1*Y1 - C
*
* If a reflected ray passes outside this surface,
* the discriminate will be negative.
* It is ignored and program execution continues.
*
        ARG = B*B - 4.0D0*A*C
        IF(ARG.LE.1.E-8) GO TO 10
*
* Determine the ray (vector) direction.
*
        T = -B + DSQRT(ARG)
*
* The DO 20 loop simply computes the x, y, and z coordinates of a
* possible intersection by looping through the plus/minus term of
* the quadratic equation.
*
      DO 20 J=1,2
        T = T/(2.0D0*A)

        X = T*VOX + X1
        Y = T*VOY + Y1
        Z = T*VOZ + Z1
*
* If the coordinates of this intersection coincide with the previous
* intersection, change the sign in the quadratic equation and
* continue program execution.
*
        IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     &    DABS(Z-Z1).LT.1.D-10) THEN
```

```
          IF(J.EQ.2) GO TO 20
          T = -B - DSQRT(ARG)
          GO TO 20
        ENDIF
*
* The nature of this program makes it necessary to be sure a ray
* doesn't actually penetrate a given surface to be incident on
* another.  This has been taken care of using the dot product.
*
        VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
        VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
        VMAG1 = DSQRT(VMAG1)
        VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
        VMAG2 = DSQRT(VMAG2)
*
* Determine if the dot product is 1 or -1.  If -1, the reflected
* ray passed through the plane tangent to the incident surface.
* If 1, the incident ray is reflected in the proper direction and
* program execution continues.
*
        IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) THEN
          IF(J.EQ.2) GO TO 20
          T = -B - DSQRT(ARG)
          GO TO 20
        ENDIF
*
* If an intersection occurs within the specified limits of the
* conical surfaces of the detector housing, record the point of
* incidence of emitted energy bundle and its starting point.
* Set flag and return to subprogram OPTICS.
*
        RADIUS = DSQRT(X*X + Y*Y)
        IF(Z.GE.HA.AND.Z.LE.HB) THEN
        IF(RADIUS.GE.D6/2.D0.AND.RADIUS.LE.RCONE) THEN
          NN = 8
          CALL COORD (X1,Y1,Z1,X,Y,Z)
*
* Compute normal unit vector to surface at point of incidence.
* This calculation is needed by subprograms REFLECT and VECTOR
* to determine the direction of the reflected ray.
*
          UN = DSQRT(X*X + Y*Y + V*V*(Z-APEX(I))*(Z-APEX(I)))
          UNX = X / UN
          UNY = Y / UN
          UNZ = -V*(Z-APEX(I)) / UN
          FLAG = 1
          RETURN
        ENDIF
        ENDIF
*
* If no intersections occur on this conic section, continue looping.
*
        IF(J.EQ.2) GO TO 20
*
* Change the sign on the root and continue looping.
*
        T = -B - DSQRT(ARG)
   20 CONTINUE
   10 CONTINUE

      FLAG = 1
      RETURN
*
* If subprogram COORD determines an intersection on the cones of the
* detector housing to be the shortest distance between the emission
* site and site of incidence, program execution is returned here
```

```
* to determine whether the ray is absorbed or reflected.
*
    1 CONTINUE
      CALL RANDOM (R)
*
* A random number is chosen and compared to the absorptivity of
* the point of incidence on the detector housing.  If the random
* number chosen is less than or equal to the value of absorptivity
* corresponding to the cones of the detector housing, the incident
* ray is absorbed.  If not,...
*
      IF(R.LE.ABS5) THEN
        IF(Z.LT.HCONE) THEN
          DF(II,23) = DF(II,23) + 1.D0
          FLAG = 0
          RETURN
        ELSE
          DF(II,24) = DF(II,24) + 1.D0
          FLAG = 0
          RETURN
        ENDIF
      ENDIF
      NTEST = 1
*
* ...another random number is chosen to determine how the incident
* ray is reflected.  If the random number is less than or equal to
* the value of the reflectivity of the cones of the detector housing,
* the ray is specularly reflected.  If not, a diffuse reflection
* occurs.
*
      CALL RANDOM (R)
      IF(R.LE.REFR5) THEN
        CALL REFLECT (X1,Y1,Z1,X,Y,Z,UNX,UNY,UNZ,VOX,VOY,VOZ)
      ELSE
        X1 = X
        Y1 = Y
        Z1 = Z
        CALL VECTOR (X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
      ENDIF

      RETURN
      END




*********************************************************************
*                                                                   *
*                         SUBROUTINE PRIMIR                         *
*                                                                   *
* This subprogram monitors the primary mirror of the ERBE scanning  *
* radiometer.  If an emitted energy bundle is intercepted by this   *
* part of the optics module, its absorption or reflection is        *
* accounted for.                                                    *
*                                                                   *
*********************************************************************
*
      SUBROUTINE PRIMIR (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Define storage variables, size and type.
*
      INTEGER*2 FLAG
```

```
      REAL*8 KAPPA1,KAPPA2,DF(40,40)
*
* Place all common variables in one, unique storage block.
*
      COMMON /GEOM/ H1,D1,D2,THETA1,C1,C2,C3,C4,C5,H2,D3,D4,D5
      COMMON /GEOM/ DEL1,DEL2,THETA2,PHI,H3,HL1,HL2,D6,THETA3
      COMMON /GEOM/ H4,HCURVE,D7,KAPPA1,H5,D8,KAPPA2,THETA4,H6,D9
      COMMON /GEOM/ THETA5,DIM1,DIM2,HREF,DREF

      COMMON /MATL/ ABS1,ABS2,ABS3,ABS4,ABS5,ABS6,ABS7,ABS8,ABS9
      COMMON /MATL/ ABS10,REFR1,REFR2,REFR3,REFR4,REFR5,REFR6
      COMMON /MATL/ REFR7,REFR8,REFR9,REFR10

      COMMON /FLAGS/ NFLAG,NTEST,NN
      COMMON /CHANNEL/ ALAMBDA,NCHANEL,NFLTRSW,NSPCTRM
      COMMON /SPEC/ NUMHITS,MIRTYPE
      COMMON /DISTR/ DF,II
*
* Initialize necessary variables.
*
      NTEST = 0
      IF(NFLAG.EQ.1) GO TO 1
      NTAG = 0
      RB = D7/2.0D0
      HA = HREF+H1+H2+H3
      VERTEX = HA + HCURVE
      RHO = KAPPA1 * 2.0D0
      HB = -RB*RB/RHO + VERTEX
      HC = HREF+H1+H2+H3+H4+HCURVE-H6
      RA = DSQRT(HCURVE * RHO)
*
* Determine if ray hits the primary mirror.
*---------------------------------------------------------------
* If a ray approaches the primary mirror parallel to the axis of
* the instrument, compute its intersection here.
*
      IF(DABS(VOX).LE.1.0E-8.AND.DABS(VOY).LE.1.0E-8) THEN
        X = X1
        Y = Y1
        Z = -(X*X + Y*Y)/RHO + VERTEX

        IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     &    DABS(Z-Z1).LT.1.D-10) THEN
          FLAG = 1
          RETURN
        ENDIF

        VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
        VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
        VMAG1 = DSQRT(VMAG1)
        VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
        VMAG2 = DSQRT(VMAG2)

        IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) THEN
          FLAG = 1
          RETURN
        ENDIF

        RADIUS = DSQRT(X*X + Y*Y)
*
* If ray falls within specified limits, compute the unit normal
* vector at the point of incidence for subroutines REFLECT and
* VECTOR and record that point in subprogram COORD.
*
        IF(RADIUS.LE.RA.AND.RADIUS.GE.RB) THEN
          NN = 9
          CALL COORD (X1,Y1,Z1,X,Y,Z)
```

```
          FLAG = 1
          RETURN
        ELSE
          GO TO 2
        ENDIF
      ENDIF
*------------------------------------------------------------------
      A = VOX*VOX+ VOY*VOY
      B = 2.0D0*X1*VOX + 2.0D0*Y1*VOY + RHO*VOZ
      C = X1*X1 + Y1*Y1 + RHO*(Z1-VERTEX)
*
* Compute the discriminate of the quadratic formula.
*
      ARG = B*B - 4.0D0*A*C
*
* If a reflected ray passes on the outside of a quadric surface,
* the discriminate will be negative.  If this happens, the surface
* is ignored and program execution continues.
*
      IF(ARG.LE.1.E-8) GO TO 2
*
* Determine the ray (vector) direction.
*
      T = -B + DSQRT(ARG)
*
* The DO 10 loop simply computes the x, y, and z coordinates of a
* possible intersection by looping through the plus/minus term of
* the quadratic equation.
*
      DO 10 I=1,2
        T = T/(2.0D0*A)

        X = T*VOX + X1
        Y = T*VOY + Y1
        Z = T*VOZ + Z1
*
* If the coordinates of this intersection coincide with the previous
* intersection, change the sign in the quadratic equation and
* continue program execution.
*
        IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     &    DABS(Z-Z1).LT.1.D-10) THEN
          IF(I.EQ.2) GO TO 2
          T = -B -DSQRT(ARG)
          GO TO 10
        ENDIF
*
* The nature of this program makes it necessary to be sure a ray
* doesn't actually penetrate a given surface to be incident on
* another.  This has been taken care of using the dot product.
*
        VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
        VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
        VMAG1 = DSQRT(VMAG1)
        VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
        VMAG2 = DSQRT(VMAG2)
*
* Determine if the dot product is 1 or -1.  If -1, the reflected
* ray passed through the plane tangent to the incident surface.
* If 1, the incident ray is reflected in the proper direction and
* program execution continues.
*
        IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) THEN
          IF(I.EQ.2) GO TO 2
          T = -B -DSQRT(ARG)
          GO TO 10
        ENDIF
```

```
      RADIUS = DSQRT(X*X + Y*Y)
*
* If an intersection occurs within the specified limits of the
* primary mirror, record point of incidence and starting point of
* energy bundle in subprogram COORD.
*
      IF(RADIUS.LE.RA.AND.RADIUS.GE.RB) THEN
        NN = 9
        CALL COORD (X1,Y1,Z1,X,Y,Z)
        GO TO 2
      ENDIF
*
* If no intersections occurs, continue label 2.
*
      IF(I.EQ.2) GO TO 2
*
* Change the sign on the root and continue looping.
*
      T = -B - DSQRT(ARG)

   10 CONTINUE
*
* Determine if ray hits inner primary mirror not covered by the
* primary insert.
*
    2 CONTINUE

     A = VOX*VOX + VOY*VOY
     B = 2.0D0*(X1*VOX + Y1*VOY)
     C = X1*X1 + Y1*Y1 - RB*RB
*
* Compute the discriminate of the quadratic formula.
*
     ARG = B*B - 4.0D0*A*C
*
* If the discriminate is negative, the ray passes outside the
* cylindrical surface.  Set flag and return to subprogram OPTICS.
*
     IF(ARG.LE.1.E-8) THEN
       FLAG = 1
       RETURN
     ENDIF
*
* Determine the ray(vector) direction.
*
     T = -B + DSQRT(ARG)
*
* The DO 20 loop determine a possible intersection with the
* cylindrical surface of the primary mirror.  Twice through
* the loop represents the plus/minus term of the quadratic
* formula.
*
     DO 20 J=1,2
       T = T/(2.0D0*A)

       X = T*VOX + X1
       Y = T*VOY + Y1
       Z = T*VOZ + Z1

       IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
    &    DABS(Z-Z1).LT.1.D-10) THEN
         IF(J.EQ.2) THEN
           FLAG = 1
           RETURN
         ENDIF
         T = -B - DSQRT(ARG)
```

```
          GO TO 20
        ENDIF

        VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
        VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
        VMAG1 = DSQRT(VMAG1)
        VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
        VMAG2 = DSQRT(VMAG2)

        IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) THEN
          IF(J.EQ.2) THEN
            FLAG = 1
            RETURN
          ENDIF
          T = -B - DSQRT(ARG)
          GO TO 20
        ENDIF
*
* If ray intersects cylindrical surface within specified limits,
* record the coordinates of the point of incidence and the ray's
* starting point.
*
        IF(Z.GE.HB.AND.Z.LE.HC) THEN
          NN = 13
          CALL COORD (X1,Y1,Z1,X,Y,Z)
          FLAG = 1
          RETURN
        ENDIF
*
* If no intersection occurs, set flag and return to subprogram
* OPTICS.
*
        IF(J.EQ.2) THEN
          FLAG = 1
          RETURN
        ENDIF
*
* Change the sign of the root and continue looping.
*
        T = -B - DSQRT(ARG)

   20 CONTINUE
*
* If subprogram COORD determines an intersection by the primary
* mirror to be the shortest distance between the emission site and
* site of incidence, program execution is returned here to determine
* whether the ray is absorbed or reflected.
*
*
* A random number is chosen and compared to the absorptivity of
* the primary mirror.  If the random number chosen is less than or
* equal to the value of absorptivity corresponding to the primary
* mirror, the incident ray is absorbed.  If not,...
*
    1 CONTINUE
*
* Monitor the times an energy bundle strikes the primary mirror.
* The curvature of the mirrors may cause an energy bundle to
* oscillate between the two.
*
      NUMHITS = NUMHITS + 1
      IF(NUMHITS.GT.10) THEN
        DF(II,26) = DF(II,26) + 1.D0
        FLAG = 0
        RETURN
      ENDIF
```

```
      IF(NN.EQ.9) THEN
        UN = DSQRT(4.0D0*X*X + 4.0D0*Y*Y + RHO*RHO)
        UNX = 2.0D0*X / UN
        UNY = 2.0D0*Y / UN
        UNZ = RHO / UN
      ELSE
        UNX = X/RB
        UNY = Y/RB
        UNZ = 0.0D0
      ENDIF

      CALL RANDOM (R)

      IF(R.LE.ABS6) THEN
      IF(NN.EQ.13) THEN
        DF(II,33) = DF(II,33) + 1.D0
        FLAG = 0
        RETURN
      ELSE
        DF(II,26) = DF(II,26) + 1.D0
        FLAG = 0
        RETURN
      ENDIF
      ENDIF

      NTEST = 1
*
* ...another random number is chosen to determine how the incident
* ray is reflected.  If the random number is less than or equal to
* the value of the reflectivity of the primary mirror, then
* the ray is specularly reflected.  If not, a diffuse reflection
* occurs.
*
*
* User set flag determines whether mirrors have a set specular-to-total
* reflectivity ratio or whether the ratio is wavelength dependent.
*
      IF(MIRTYPE.EQ.1) THEN
        CALL RHOSPEC (ALAMBDA,REFR)
        REFR6 = REFR
      ENDIF

      CALL RANDOM (R)

      IF(R.LE.REFR6) THEN
        CALL REFLECT (X1,Y1,Z1,X,Y,Z,UNX,UNY,UNZ,VOX,VOY,VOZ)
      ELSE
        X1 = X
        Y1 = Y
        Z1 = Z
        CALL VECTOR (X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
      ENDIF

      RETURN
      END




*********************************************************************
*                                                                   *
*                        SUBROUTINE INSERT                          *
*                                                                   *
* This subprogram monitors the copper insert of the primary mirror. *
* If an emitted energy bundle is intercepted by this part of the    *
* optics module, its absorption or reflection is accounted for.     *
```

```
*                                                                    *
**********************************************************************
*
      SUBROUTINE INSERT (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Define storage variables, size and type.
*
      INTEGER*2 FLAG
      REAL*8 KAPPA1,KAPPA2,DF(40,40)
*
* Place all common variables in one, unique storage block.
*
      COMMON /GEOM/ H1,D1,D2,THETA1,C1,C2,C3,C4,C5,H2,D3,D4,D5
      COMMON /GEOM/ DEL1,DEL2,THETA2,PHI,H3,HL1,HL2,D6,THETA3
      COMMON /GEOM/ H4,HCURVE,D7,KAPPA1,H5,D8,KAPPA2,THETA4,H6,D9
      COMMON /GEOM/ THETA5,DIM1,DIM2,HREF,DREF

      COMMON /MATL/ ABS1,ABS2,ABS3,ABS4,ABS5,ABS6,ABS7,ABS8,ABS9
      COMMON /MATL/ ABS10,REFR1,REFR2,REFR3,REFR4,REFR5,REFR6
      COMMON /MATL/ REFR7,REFR8,REFR9,REFR10

      COMMON /FLAGS/ NFLAG,NTEST,NN
      COMMON /DISTR/ DF,II
*
* Initialize necessary variables.
*
      NTEST = 0
      IF(NFLAG.EQ.1) GO TO 1
      HA = HREF+H1+H2+H3+H4+HCURVE-H6
      HC = HA + H6
      PI = DACOS(-1.0D0)
      PSI = THETA5 * PI/180.0D0
      ALPHA = PI/2.0D0 - PSI
      VERTEX = (D7/2.0D0)*DTAN(PSI)
      APEX = HA+VERTEX
      HB = (-D9/2.0D0)/DTAN(ALPHA) + APEX
      V = DTAN(ALPHA) * DTAN(ALPHA)
*
* Determine if ray is intercepted by the primary insert.
*
      A = VOX*VOX + VOY*VOY - VOZ*VOZ*V
      B = 2.0D0*X1*VOX + 2.0D0*Y1*VOY + 2.0D0*APEX*VOZ*V
      B = B - 2.0D0*Z1*VOZ*V
      C = X1*X1 + Y1*Y1 - V*(Z1*Z1 + APEX*APEX - 2.0D0*APEX*Z1)
*
* Compute the discriminate of the quadratic equation.
*
      ARG = B*B - 4.0D0*A*C
*
* If a reflected ray passes outside the insert, the discriminate will
* be negative.  Since this can not happen physically within the optics
* module. It is ignored and program execution continues.
*
      IF(ARG.LE.1.E-8) GO TO 2
*
* Determine the ray (vector) direction.
*
      T = -B + DSQRT(ARG)
*
* The DO 10 loop simply computes the x, y, and z coordinates of a
* possible intersection by looping through the plus/minus term of
* the quadratic equation.
*
```

```
      DO 10 I=1,2
        T = T/(2.0D0*A)

        X = T*VOX + X1
        Y = T*VOY + Y1
        Z = T*VOZ + Z1
*
* If the coordinates of this intersection coincide with the previous
* intersection, change the sign in the quadratic equation and
* continue program execution.
*
        IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     &    DABS(Z-Z1).LT.1.D-10) THEN
          IF(I.EQ.2) GO TO 2
          T = -B - DSQRT(ARG)
          GO TO 10
        ENDIF
*
* The nature of this program makes it necessary to be sure a ray
* doesn't actually penetrate a given surface to be incident on
* another.  This has been taken care of using the dot product.
*
        VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
        VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
        VMAG1 = DSQRT(VMAG1)
        VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
        VMAG2 = DSQRT(VMAG2)
*
* Determine if the dot product is 1 or -1.  If -1, the reflected
* ray passed through the plane tangent to the incident surface.
* If 1, the incident ray is reflected in the proper direction and
* program execution continues.
*
        IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) THEN
          IF(I.EQ.2) GO TO 2
          T = -B - DSQRT(ARG)
          GO TO 10
        ENDIF
*
* If an intersection occurs with the primary insert, record the
* point of incidence and starting point of energy bundle in
* subprogram COORD.  Return to subprogram OPTICS.
*
        IF(Z.GE.HA.AND.Z.LE.HB) THEN
          NN = 10
          CALL COORD (X1,Y1,Z1,X,Y,Z)
          GO TO 2
        ENDIF

        IF(I.EQ.2) GO TO 2
*
* Change the sign on the root and continue looping.
*
        T = -B - DSQRT(ARG)

   10 CONTINUE
*
* Determine if energy bundle hits the cylindrical part of the
* insert.
*
    2 CONTINUE
      A = VOX*VOX + VOY*VOY
      B = 2.0D0*(X1*VOX + Y1*VOY)
      C = X1*X1 + Y1*Y1 - D9*D9/4.0D0
*
* Compute the discriminate of the quadratic equation.
*
```

```
      ARG = B*B - 4.0D0*A*C
*
* If an energy bundle passes on the outside of the cylinder or
* parallel to its axis, the argument above is negative or zero,
* respectively.  Therefore, this part of the optics module is
* ignored and program execution is returned to subprogram OPTICS.
*
      IF(ARG.LE.1.E-8) THEN
        FLAG = 1
        RETURN
      ENDIF
*
* Determine the ray(vector) direction.
*
      T = -B + DSQRT(ARG)
*
* The DO 20 loop determines the coordinates of a possible intersection
* by looping through the plus/minus term of the quadratic equation.
*
      DO 20 J=1,2
        T = T/(2.0D0*A)

        X = T*VOX + X1
        Y = T*VOY + Y1
        Z = T*VOZ + Z1
*
* If the coordinates of this intersection coincide with the previous
* intersection, change the sign in the quadratic equation and
* continue program execution.
*
        IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     &    DABS(Z-Z1).LT.1.D-10) THEN
          IF(J.EQ.2) THEN
            FLAG = 1
            RETURN
          ENDIF
          T = -B - DSQRT(ARG)
          GO TO 20
        ENDIF
*
* The nature of this program makes it necessary to be sure a ray
* doesn't actually penetrate a given surface to be incident on
* another.  This has been taken care of using the dot product.
*
        VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
        VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
        VMAG1 = DSQRT(VMAG1)
        VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
        VMAG2 = DSQRT(VMAG2)
*
* Determine if the dot product is 1 or -1.  If -1, the reflected ray
* passed through the plane tangent to the incident surface.  Return
* to the subprogram OPTICS.  If 1, the incident ray is reflected in
* the proper direction and program execution continues.
*
        IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) THEN
          IF(J.EQ.2) THEN
            FLAG = 1
            RETURN
          ENDIF
          T = -B - DSQRT(ARG)
          GO TO 20
        ENDIF
*
* If an intersection occurs with the primary insert, record the
* point of incidence and starting point of energy bundle in
```

```
* subprogram COORD.  Return to subprogram OPTICS.
*
        IF(Z.GT.HB.AND.Z.LE.HC) THEN
          NN = 14
          CALL COORD(X1,Y1,Z1,X,Y,Z)
          FLAG = 1
          RETURN
        ENDIF

        IF(J.EQ.2) THEN
          FLAG = 1
          RETURN
        ENDIF
*
* Change the sign of the root and continue looping.
*
        T = -B - DSQRT(ARG)
   20 CONTINUE
*
* If subprogram COORD determines an intersection on the primary
* inseert to be the shortest distance between the emission site and
* site of incidence, program execution is returned here to determine
* whether the ray is absorbed or reflected.
*
*
* A random number is chosen and compared to the absorptivity of
* the primary insert.
* If the random number chosen is less than or equal to the value of
* absorptivity corresponding to that of the primary insert, the
* incident ray is absorbed.  If not,...
*
    1 CONTINUE
      CALL RANDOM (R)
      IF(NN.EQ.10) THEN
        UN = DSQRT(X*X + Y*Y + V*V*(Z-APEX)*(Z-APEX))
        UNX = X / UN
        UNY = Y / UN
        UNZ = -V*(Z-APEX) / UN
      ELSE
        UNX = 2.0D0 * X / D9
        UNY = 2.0D0 * Y / D9
        UNZ = 0.0D0
      ENDIF

      IF(R.LE.ABS8) THEN
      IF(NN.EQ.14) THEN
        DF(II,34) = DF(II,34) + 1.D0
        FLAG = 0
        RETURN
      ELSE
        DF(II,27) = DF(II,27) + 1.D0
        FLAG = 0
        RETURN
      ENDIF
      ENDIF
*
* ...another random number is chosen to determine how the incident
* ray is reflected.  If the random number is less than or equal to
* the value of the reflectivity of the insert of the primary mirror,
* the ray is specularly reflected.  If not, a diffuse reflection
* occurs.
*
      CALL RANDOM (R)
      NTEST = 1
      IF(R.LE.REFR8) THEN
        CALL REFLECT (X1,Y1,Z1,X,Y,Z,UNX,UNY,UNZ,VOX,VOY,VOZ)
      ELSE
```

```
        X1 = X
        Y1 = Y
        Z1 = Z
        CALL VECTOR (X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
      ENDIF

      RETURN
      END




*********************************************************************
*                                                                   *
*                          SUBROUTINE SECMIR                        *
*                                                                   *
* This subprogram monitors the reflective surface of the secondary  *
* mirror.  If an emitted energy bundle is intercepted by this       *
* part of the optics module, its absorption or reflection is        *
* accounted for.                                                    *
*                                                                   *
*********************************************************************
*
      SUBROUTINE SECMIR (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Define storage variables, size and type.
*
      INTEGER*2 FLAG
      REAL*8 KAPPA1,KAPPA2,DF(40,40)
*
* Place all common variables in one, unique storage block.
*
      COMMON /GEOM/ H1,D1,D2,THETA1,C1,C2,C3,C4,C5,H2,D3,D4,D5
      COMMON /GEOM/ DEL1,DEL2,THETA2,PHI,H3,HL1,HL2,D6,THETA3
      COMMON /GEOM/ H4,HCURVE,D7,KAPPA1,H5,D8,KAPPA2,THETA4,H6,D9
      COMMON /GEOM/ THETA5,DIM1,DIM2,HREF,DREF

      COMMON /MATL/ ABS1,ABS2,ABS3,ABS4,ABS5,ABS6,ABS7,ABS8,ABS9
      COMMON /MATL/ ABS10,REFR1,REFR2,REFR3,REFR4,REFR5,REFR6
      COMMON /MATL/ REFR7,REFR8,REFR9,REFR10

      COMMON /FLAGS/ NFLAG,NTEST,NN
      COMMON /SPEC/ NUMHITS,MIRTYPE
      COMMON /CHANNEL/ ALAMBDA,NCHANEL,NFLTRSH,NSPCTRM
      COMMON /DISTR/ DF,II
*
* Initialize necessary variables.
*
      NTEST = 0
      IF(NFLAG.EQ.1) GO TO 1
      RAD = D8/2.0D0
      RHO = 2.0D0*KAPPA2
      VERTEX = HREF + H1 + H2
*-------------------------------------------------------------------
* If a ray approaches the secondary mirror parallel to the axis of
* the instrument, compute its intersection here.
*
      IF(DABS(VOX).LE.1.0E-8.AND.DABS(VOY).LE.1.0E-8) THEN
        X = X1
        Y = Y1
        Z = -(X*X + Y*Y)/RHO + VERTEX
```

```
        IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     &    DABS(Z-Z1).LT.1.D-10) THEN
          FLAG = 1
          RETURN
        ENDIF

        VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
        VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
        VMAG1 = DSQRT(VMAG1)
        VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
        VMAG2 = DSQRT(VMAG2)
        IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) THEN
          FLAG = 1
          RETURN
        ENDIF

        RADIUS = DSQRT(X*X + Y*Y)
*
* If ray falls within specified limits, compute the unit normal
* vector at the point of incidence for subroutines REFLECT and
* VECTOR and record that point in subprogram COORD.
*
        IF(RADIUS.LE.RAD) THEN
          UN = DSQRT(4.0D0*X*X + 4.0D0*Y*Y + RHO*RHO)
          UNX = -2.0D0*X / UN
          UNY = -2.0D0*Y / UN
          UNZ = -RHO / UN
          NN = 11
          CALL COORD (X1,Y1,Z1,X,Y,Z)
          FLAG = 1
          RETURN
        ELSE
          FLAG = 1
          RETURN
        ENDIF
      ENDIF
*-----------------------------------------------------------------------
* Determine if ray hits the secondary mirror.
*
      A = VOX*VOX + VOY*VOY
      B = 2.0D0*(X1*VOX + Y1*VOY) + RHO*VOZ
      C = X1*X1 + Y1*Y1 + RHO*(Z1-VERTEX)
*
* Compute the discriminate of the quadratic equation.
*
      ARG = B*B - 4.0D0*A*C
*
* If a reflected ray does not hit the secondary mirror, the
* discriminate will be negative.  This surface is ignored and
* program execution continues.
*
      IF(ARG.LE.1.E-8) THEN
        FLAG = 1
        RETURN
      ELSE
*
* Determine the ray (vector) direction.
*
        T = -B + DSQRT(ARG)
      ENDIF
*
* The DO 10 loop simply computes the x, y, and z coordinates of a
* possible intersection by looping through the plus/minus term of
* the quadratic equation.
*
      DO 10 I=1,2
        T = T/(2.0D0*A)
```

```
      X = T*VOX + X1
      Y = T*VOY + Y1
      Z = T*VOZ + Z1
*
* If the coordinates of this intersection coincide with the previous
* intersection, change the sign on the root the quadratic equation
* and continue program execution.
*
      IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
   &    DABS(Z-Z1).LT.1.D-10) THEN
        IF(I.EQ.2) THEN
          FLAG = 1
          RETURN
        ENDIF
        T = -B - DSQRT(ARG)
        GO TO 10
      ENDIF
*
* The nature of this program makes it necessary to be sure a ray
* doesn't actually penetrate a given surface to be incident on
* another.  This has been taken care of using the dot product.
*
      VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
      VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
      VMAG1 = DSQRT(VMAG1)
      VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
      VMAG2 = DSQRT(VMAG2)
*
* Determine if the dot product is 1 or -1.  If -1, the reflected
* ray passed through the plane tangent to the incident surface.
* If 1, the incident ray is reflected in the proper direction and
* program execution continues.
*
      IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) THEN
        IF(I.EQ.2) THEN
          FLAG = 1
          RETURN
        ENDIF
        T = -B - DSQRT(ARG)
        GO TO 10
      ENDIF
*
* If an intersection occurs within the specified limits of the re-
* flective surface of the secondary mirror, record point of incidence
* and starting point of energy bundle in subprogram COORD.  Set flag
* and exit subroutine.
*
      RADIUS = DSQRT(X*X + Y*Y)
      IF(RADIUS.LE.RAD) THEN
        NN = 11
        CALL COORD (X1,Y1,Z1,X,Y,Z)
        FLAG = 1
        RETURN
      ENDIF
*
* If no intersections occurs, set flag and return to subroutine OPTICS.
*
      IF(I.EQ.2) THEN
        FLAG = 1
        RETURN
      ENDIF
*
* Change the sign of the root term and continue looping.
*
      T = -B - DSQRT(ARG)
```

```
   10 CONTINUE
*
* If subprogram COORD determines an intersection with the secondary
* mirror to be the shortest distance between the emission site and
* site of incidence, program execution is returned here to determine
* whether the ray is absorbed or reflected.
*
*
* A random number is chosen and compared to the absorptivity of
* the reflective surface of the secondary mirror. If the random
* number chosen is less than or equal to the value of absorptivity
* corresponding to this part of the mirror, the incident ray is
* absorbed.  If not,...
*
  1    CONTINUE
*
* Monitor the times an energy bundle strikes the secondary mirror.
* The curvature of the mirrors may cause and energy bundle to
* oscillate between the two.
*
       NUMHITS = NUMHITS + 1
       IF(NUMHITS.GT.10) THEN
         DF(II,21) = DF(II,21) + 1.D0
         FLAG = 0
         RETURN
       ENDIF

       CALL RANDOM (R)
       IF(R.LE.ABS7) THEN
         DF(II,21) = DF(II,21) + 1.D0
         FLAG = 0
         RETURN
       ELSE
         UN = DSQRT(4.0D0*X*X + 4.0D0*Y*Y + RHO*RHO)
         UNX = -2.0D0*X / UN
         UNY = -2.0D0*Y / UN
         UNZ = -RHO / UN
         FLAG = 1
       ENDIF
       NTEST = 1
*
* ...another random number is chosen to determine how the incident
* ray is reflected.  If the random number is less than or equal to
* the value of the reflectivity of this part of the secondary mirror,
* the ray is specularly reflected.  If not, a diffuse reflection
* occurs.
*
*
* User set flag determines whether mirrors have a set specular-to-total
* reflectivity ratio or whether the ratio is wavelength dependent.
*
       IF(MIRTYPE.EQ.1) THEN
         CALL RHOSPEC (ALAMBDA,REFR)
         REFR7 = REFR
       ENDIF

       CALL RANDOM (R)

       IF(R.LE.REFR7) THEN
         CALL REFLECT (X1,Y1,Z1,X,Y,Z,UNX,UNY,UNZ,VOX,VOY,VOZ)
       ELSE
         X1 = X
         Y1 = Y
         Z1 = Z
         CALL VECTOR (X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
       ENDIF
```

```
      RETURN
      END


*********************************************************************
*                                                                   *
*                         SUBROUTINE FOV                            *
*                                                                   *
* This subprogram monitors the field stop and the active flake of  *
* the radiometer.  If an emitted energy bundle is intercepted by   *
* this part of the optics module, its absorption or reflection is  *
* accounted for.                                                    *
*                                                                   *
*********************************************************************
*
      SUBROUTINE FOV (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Specify storage parameters, size and type.
*
      INTEGER*2 FLAG
      REAL*8 KAPPA1,KAPPA2,POINT(2,3000),DF(40,40)
*
* Place all common variables in one, unique storage block.
*
      COMMON /GEOM/ H1,D1,D2,THETA1,C1,C2,C3,C4,C5,H2,D3,D4,D5
      COMMON /GEOM/ DEL1,DEL2,THETA2,PHI,H3,HL1,HL2,D6,THETA3
      COMMON /GEOM/ H4,HCURVE,D7,KAPPA1,H5,D8,KAPPA2,THETA4,H6,D9
      COMMON /GEOM/ THETA5,DIM1,DIM2,HREF,DREF
      COMMON /GEOM1/ H7,D10,FLAKDIM,ABS11,REFR11

      COMMON /MATL/ ABS1,ABS2,ABS3,ABS4,ABS5,ABS6,ABS7,ABS8,ABS9
      COMMON /MATL/ ABS10,REFR1,REFR2,REFR3,REFR4,REFR5,REFR6
      COMMON /MATL/ REFR7,REFR8,REFR9,REFR10

      COMMON /PLOT/ POINT,NSPLOT(25,25)
      COMMON /FLAGS/ NFLAG,NTEST,NN

      COMMON /DISTR/ DF,II
*
* Initialize necessary data.
*
      PI = DACOS(-1.D0)
      ZSTOP = HREF+H1+H2+H3+H4+HCURVE
      ZFLAKE = HREF+H1+H2+H3+H4+HCURVE+H7

      IF(Z1.GE.ZSTOP) THEN
        CALL FOV2 (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
        RETURN
      ENDIF

      IF(NFLAG.EQ.1) GO TO 1
      NTAG = 0
*
* Compute where emitted energy bundle strikes the field stop.
*
      CONST = DIM1/2.0D0
      T = (ZSTOP-Z1)/VOZ
      X = T*VOX + X1
      Y = T*VOY + Y1
```

```
      Z = ZSTOP
*
* If the radius of this interception is not inside the inner diameter
* of the primary insert, set flag and return to subprogram OPTICS.
*
      RADIUS = DSQRT(X*X + Y*Y)
      IF(RADIUS.GE.D9/2.0D0.AND.Z1.LT.ZSTOP) THEN
        FLAG = 1
        RETURN
      ENDIF
*
* If the coordinates of this intersection coincide with the previous
* intersection, change the sign in the quadratic equation and
* continue program execution.
*
      IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     & DABS(Z-Z1).LT.1.D-10) THEN
        FLAG = 1
        RETURN
      ENDIF
*
* The nature of this program makes it necessary to be sure a ray
* doesn't actually penetrate a given surface to be incident on
* another.  This has been taken care of using the dot product.
*
      VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
      VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
      VMAG1 = DSQRT(VMAG1)
      VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
      VMAG2 = DSQRT(VMAG2)
*
* Determine if the dot product is 1 or -1.  If -1, the reflected ray
* passed through the plane tangent to the incident surface.  Return
* to the subprogram OPTICS.  If 1, the incident ray is reflected in
* the proper direction and program execution continues.
*
      IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) THEN
        FLAG = 1
        RETURN
      ENDIF
*
* Determine if emitted energy bundle strikes the field stop.  If so,
* record the coordinates in subprogram COORD.  Account for which
* side of the field stop receives the energy bundle and assign the
* appropriate absorptivity and reflectivity values.
*
      IF(Y.GE.DIM2/2.0D0.OR.Y.LE.-DIM2/2.0D0) THEN
        NN = 12
        Z = ZSTOP
        CALL COORD (X1,Y1,Z1,X,Y,Z)
        ABS = ABS9
        REFR = REFR9
        FLAG = 1
        RETURN
      ENDIF
*
* Determine if emitted energy bundle enters precision aperture of
* field stop.  If so, see if energy bundle strikes active flake and
* record point of incidence in subprogram COORD.
*
      CHK1 = -X + Y
      CHK2 =  X + Y

      IF(CHK1.LT.CONST.AND.CHK2.LT.CONST.AND.CHK1.GT.-CONST.AND.
     & CHK2.GT.-CONST) THEN
        GO TO 111
*
```

```
* If energy bundle does not fall within the precision aperture,
* energy bundle must hit the field stop at this point.  Record
* coordinates in subprogram COORD.
*
      ELSE
        ABS = ABS9
        REFR = REFR9
        MM = 1
        NN = 12
        CALL COORD (X1,Y1,Z1,X,Y,Z)
        FLAG = 1
        RETURN
      ENDIF

  111 CONTINUE
      T = (ZFLAKE-Z1)/VOZ
      X = T*VOX + X1
      Y = T*VOY + Y1
      Z = ZFLAKE

      IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     & DABS(Z-Z1).LT.1.D-10) THEN
        FLAG = 1
        RETURN
      ENDIF

      VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
      VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
      VMAG1 = DSQRT(VMAG1)
      VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
      VMAG2 = DSQRT(VMAG2)

      IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) THEN
        FLAG = 1
        RETURN
      ENDIF

      CONST = FLAKDIM/DSQRT(2.D0)
      CHK1 = -X + Y
      CHK2 =  X + Y
      IF(CHK1.LE.CONST.AND.CHK2.LE.CONST.AND.CHK1.GE.-CONST.AND.
     & CHK2.GE.-CONST) THEN
        NN = 12
        CALL COORD (X1,Y1,Z1,X,Y,Z)
        REFR = REFR11
        NTAG = 1
        FLAG = 1
        RETURN
      ELSE
        FLAG = 1
        RETURN
      ENDIF
*
* If subprogram COORD determines an intersection on the field stop
* to be the shortest distance between the emission site and
* site of incidence, program execution is returned here to determine
* whether the ray is absorbed or reflected.
*
*
* A random number is chosen and compared to the absorptivity of
* the point of incidence on the field stop. If the random
* number chosen is less than or equal to the value of absorptivity
* for the field stop or active flake, the incident ray
* is absorbed.  If not,...
*
* Check field stop.
*
```

```
    1 CONTINUE
      IF(NTAG.EQ.0) THEN
        CALL RANDOM(R)
        IF(R.LE.ABS) THEN
          DF(II,28) = DF(II,28) + 1.D0
          FLAG = 0
          RETURN
        ENDIF
      ENDIF
*
* Check active flake.
*
      IF(NTAG.EQ.1) THEN
        CALL RANDOM(R)

        IF(R.LE.ABS11) THEN
          DF(II,30) = DF(II,30) + 1.D0
*
* Record points on the active flake that energy bundles strike.
* May be used to plot image developed on flake.
*
          IF(DF(II,30).LE.3000.D0) THEN
            JJJ = INT(DF(II,30))
            POINT(1,JJJ) = X
            POINT(2,JJJ) = Y
          ENDIF
          FLAG = 0
          RETURN
        ENDIF
      ENDIF

    2 CONTINUE
      NTEST = 1
      UNX = 0.0D0
      UNY = 0.0D0
      UNZ = 1.0D0
*
* ...another random number is chosen to determine how the incident
* ray is reflected.  If the random number is less than or equal to
* the value of the reflectivity of that part of the active flake
* or field stop, the ray is specularly reflected.  If not, a
* diffuse reflection occurs.
*
      CALL RANDOM (R)
      IF(R.LE.REFR) THEN
        CALL REFLECT (X1,Y1,Z1,X,Y,Z,UNX,UNY,UNZ,VOX,VOY,VOZ)
      ELSE
        X1 = X
        Y1 = Y
        Z1 = Z
        CALL VECTOR (X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
      ENDIF

      RETURN
      END



*********************************************************************
*                                                                   *
*                         SUBROUTINE FOV2                           *
*                                                                   *
* This subprogram monitors the field stop and the active flake of   *
* the radiometer.  If an emitted energy bundle is intercepted by    *
* this part of the optics module, its absorption or reflection is   *
```

```
* accounted for.                                                      *
*                                                                     *
***********************************************************************
*
      SUBROUTINE FOV2 (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Specify storage parameters, size and type.
*
      INTEGER*2 FLAG
      REAL*8 KAPPA1,KAPPA2,POINT(2,3000),DF(40,40)
*
* Place all common variables in one, unique storage block.
*
      COMMON /GEOM/ H1,D1,D2,THETA1,C1,C2,C3,C4,C5,H2,D3,D4,D5
      COMMON /GEOM/ DEL1,DEL2,THETA2,PHI,H3,HL1,HL2,D6,THETA3
      COMMON /GEOM/ H4,HCURVE,D7,KAPPA1,H5,D8,KAPPA2,THETA4,H6,D9
      COMMON /GEOM/ THETA5,DIM1,DIM2,HREF,DREF
      COMMON /GEOM1/ H7,D10,FLAKDIM,ABS11,REFR11

      COMMON /MATL/ ABS1,ABS2,ABS3,ABS4,ABS5,ABS6,ABS7,ABS8,ABS9
      COMMON /MATL/ ABS10,REFR1,REFR2,REFR3,REFR4,REFR5,REFR6
      COMMON /MATL/ REFR7,REFR8,REFR9,REFR10

      COMMON /DISTR/ DF,II
      COMMON /PLOT/ POINT,NSPLOT(25,25)
      COMMON /FLAGS/ NFLAG,NTEST,NN
*
* Initialize necessary data.
*
      ZSTOP = HREF+H1+H2+H3+H4+HCURVE
      ZFLAKE = HREF+H1+H2+H3+H4+HCURVE+H7

      IF(NFLAG.EQ.1) GO TO 1
      NTAG = 0
*
* Compute where emitted energy bundle strikes the field stop.
*
      CONST = DIM1/2.0D0
      T = (ZSTOP-Z1)/VOZ
      X = T*VOX + X1
      Y = T*VOY + Y1
      Z = ZSTOP
*
* If the coordinates of this intersection coincide with the previous
* intersection, change the sign in the quadratic equation and
* continue program execution.
*
        IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     &    DABS(Z-Z1).LT.1.D-10) GO TO 111
*
* The nature of this program makes it necessary to be sure a ray
* doesn't actually penetrate a given surface to be incident on
* another.  This has been taken care of using the dot product.
*
        VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
        VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
        VMAG1 = DSQRT(VMAG1)
        VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
        VMAG2 = DSQRT(VMAG2)
*
* Determine if the dot product is 1 or -1.  If -1, the reflected ray
* passed through the plane tangent to the incident surface.  Return
* to the subprogram OPTICS.  If 1, the incident ray is reflected in
```

```
* the proper direction and program execution continues.
*
        IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) GO TO 111
*
* Determine if emitted energy bundle strikes the field stop.  If so,
* record the coordinates in subprogram COORD.  Account for which
* side of the field stop receives the energy bundle and assign the
* appropriate absorptivity and reflectivity values.
*
      IF(Y.GE.DIM2/2.0D0.OR.Y.LE.-DIM2/2.0D0) THEN
        NN = 12
        Z = ZSTOP
        CALL COORD (X1,Y1,Z1,X,Y,Z)
        ABS = ABS10
        REFR = REFR10
        FLAG = 1
        RETURN
      ENDIF
*
* Determine if emitted energy bundle enters precision aperture of
* field stop.  If so, see if energy bundle strikes active flake and
* record point of incidence in subprogram COORD.
*
      CHK1 = -X + Y
      CHK2 =  X + Y

      IF(CHK1.GE.CONST.OR.CHK2.GE.CONST.OR.CHK1.LE.-CONST.OR.
     & CHK2.LE.-CONST) THEN
        NN = 12
        CALL COORD (X1,Y1,Z1,X,Y,Z)
        ABS = ABS10
        REFR = REFR10
        FLAG = 1
        RETURN
      ELSE
        FLAG = 1
        RETURN
      ENDIF

  111 CONTINUE
      T = (ZFLAKE-Z1)/VOZ
      X = T*VOX + X1
      Y = T*VOY + Y1
      Z = ZFLAKE

      IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     & DABS(Z-Z1).LT.1.D-10) THEN
        FLAG = 1
        RETURN
      ENDIF

      VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
      VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
      VMAG1 = DSQRT(VMAG1)
      VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
      VMAG2 = DSQRT(VMAG2)

      IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) THEN
        FLAG = 1
        RETURN
      ENDIF

      CONST = FLAKDIM/DSQRT(2.D0)
      CHK1 = -X + Y
      CHK2 =  X + Y

      IF(CHK1.LE.CONST.AND.CHK2.LE.CONST.AND.CHK1.GE.-CONST.AND.
```

```
     &  CHK2.GE.-CONST) THEN
          NN = 12
          CALL COORD (X1,Y1,Z1,X,Y,Z)
          REFR = REFR11
          NTAG = 1
          FLAG = 1
          RETURN
        ENDIF

        FLAG = 1
        RETURN
*
* If subprogram COORD determines an intersection on the field stop
* to be the shortest distance between the emission site and
* site of incidence, program execution is returned here to determine
* whether the ray is absorbed or reflected.
*
*
* A random number is chosen and compared to the absorptivity of
* the point of incidence on the field stop. If the random
* number chosen is less than or equal to the value of absorptivity
* for the field stop or active flake, the incident ray
* is absorbed.  If not,...
*
* Check field stop.
*
      1 CONTINUE
        IF(NTAG.EQ.0) THEN
          CALL RANDOM(R)
          IF(R.LE.ABS) THEN
            DF(II,29) = DF(II,29) + 1.D0
            FLAG = 0
            RETURN
          ELSE
            UNX = 0.0D0
            UNY = 0.0D0
            UNZ = -1.0D0
          ENDIF
        ENDIF
*
* Check active flake.
*
        IF(NTAG.EQ.1) THEN
          CALL RANDOM(R)
          IF(R.LE.ABS11) THEN
            DF(II,30) = DF(II,30) + 1.D0
            FLAG = 0
            RETURN
          ELSE
            UNX = 0.0D0
            UNY = 0.0D0
            UNZ = 1.0D0
          ENDIF
        ENDIF

      2 CONTINUE
        NTEST = 1
*
* ...another random number is chosen to determine how the incident
* ray is reflected.  If the random number is less than or equal to
* the value of the reflectivity of that part of the reflector cap,
* the ray is specularly reflected.  If not, a diffuse reflection
* occurs.
*
        CALL RANDOM (R)
        IF(R.LE.REFR) THEN
          CALL REFLECT (X1,Y1,Z1,X,Y,Z,UNX,UNY,UNZ,VOX,VOY,VOZ)
```

```
      ELSE
        X1 = X
        Y1 = Y
        Z1 = Z
        CALL VECTOR (X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
      ENDIF

      RETURN
      END




************************************************************************
*                                                                      *
*                         SUBROUTINE HEADER                            *
*                                                                      *
* This subprogram monitors the detector header where the active        *
* flake is adhered.  If an emitted energy bundle is intercepted by     *
* this part of the optics module, its absorption or reflection is      *
* accounted for.                                                       *
*                                                                      *
************************************************************************
*
      SUBROUTINE HEADER (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Specify storage parameters, size and type.
*
      INTEGER*2 FLAG
      REAL*8 KAPPA1,KAPPA2,POINT(2,3000),DF(40,40)
*
* Place all common variables in one, unique storage block.
*
      COMMON /GEOM/ H1,D1,D2,THETA1,C1,C2,C3,C4,C5,H2,D3,D4,D5
      COMMON /GEOM/ DEL1,DEL2,THETA2,PHI,H3,HL1,HL2,D6,THETA3
      COMMON /GEOM/ H4,HCURVE,D7,KAPPA1,H5,D8,KAPPA2,THETA4,H6,D9
      COMMON /GEOM/ THETA5,DIM1,DIM2,HREF,DREF
      COMMON /GEOM1/ H7,D10,FLAKDIM,ABS11,REFR11

      COMMON /MATL/ ABS1,ABS2,ABS3,ABS4,ABS5,ABS6,ABS7,ABS8,ABS9
      COMMON /MATL/ ABS10,REFR1,REFR2,REFR3,REFR4,REFR5,REFR6
      COMMON /MATL/ REFR7,REFR8,REFR9,REFR10

      COMMON /PLOT/ POINT,NSPLOT(25,25)
      COMMON /FLAGS/ NFLAG,NTEST,NN

      COMMON /DISTR/ DF,II
*
* Initialize necessary data.
*
      PI = DACOS(-1.D0)
      ZSTOP = HREF+H1+H2+H3+H4+HCURVE
      ZFLAKE = HREF+H1+H2+H3+H4+HCURVE+H7

      IF(NFLAG.EQ.1) GO TO 1
      NTAG = 0
*
* Compute where emitted energy bundle strikes the substrate.
*
      T = (ZFLAKE-Z1)/VOZ
      X = T*VOX + X1
      Y = T*VOY + Y1
```

```
      Z = ZFLAKE
*
* If the coordinates of this intersection coincide with the previous
* intersection, change the sign in the quadratic equation and
* continue program execution.
*
      IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     & DABS(Z-Z1).LT.1.D-10) GO TO 2
*
* The nature of this program makes it necessary to be sure a ray
* doesn't actually penetrate a given surface to be incident on
* another.  This has been taken care of using the dot product.
*
      VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
      VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
      VMAG1 = DSQRT(VMAG1)
      VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
      VMAG2 = DSQRT(VMAG2)
*
* Determine if the dot product is 1 or -1.  If -1, the reflected ray
* passed through the plane tangent to the incident surface.  Return
* to the subprogram OPTICS.  If 1, the incident ray is reflected in
* the proper direction and program execution continues.
*
      IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) GO TO 2
*
* Determine if emitted energy bundle strikes the substrate.  If so,
* record the coordinates in subprogram COORD.
*
      RADIUS = DSQRT(X*X + Y*Y)
      IF(RADIUS.LE.D10/2.D0) THEN
      CONST = FLAKDIM/DSQRT(2.D0)
      CHK1 = -X + Y
      CHK2 =  X + Y
      IF(CHK1.GT.CONST.OR.CHK2.GT.CONST.OR.CHK1.LT.-CONST.OR.
     & CHK2.LT.-CONST) THEN
        NN = 17
        CALL COORD (X1,Y1,Z1,X,Y,Z)
        FLAG = 1
        RETURN
      ENDIF
      ENDIF

    2 CONTINUE
*
* Determine if ray is incident on the detector header cylinder.
*
      A = VOX*VOX + VOY*VOY
      B = 2.0D0*X1*VOX + 2.0D0*Y1*VOY
      C = X1*X1 + Y1*Y1 - D10*D10/4.D0
*
* If radiation is incident perpendicular to the aperture of the
* optics module it cannot possibly hit the cylinder.
*
      IF(DABS(A).LE.1.E-8) THEN
        FLAG = 1
        RETURN
      ENDIF
*
* Compute the discriminate of the quadratic equation.
*
      ARG = B*B - 4.0D0*A*C
*
* If a reflected ray passes on the outside of a cylinder,
* the discriminate will be negative.
* The argument is ignored and program execution continues.
*
```

```
      IF(ARG.LT.0.0D0) THEN
        FLAG = 1
        RETURN
      ENDIF
*
* Determine the ray (vector) direction.
*
      T = -B + DSQRT(ARG)
*
      DO 10 I=1,2
        T = T/(2.D0*A)
        X = T*VOX + X1
        Y = T*VOY + Y1
        Z = T*VOZ + Z1
*
* If the coordinates of this intersection coincide with the
* previous intersection, return to subprogram OPTICS and
* continue program execution.
*
      IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     & DABS(Z-Z1).LT.1.D-10) THEN
        T = -B - DSQRT(ARG)
        IF(I.EQ.2) THEN
          FLAG = 1
          RETURN
        ENDIF
        GO TO 10
      ENDIF
*
* The nature of this program makes it necessary to be sure a ray
* doesn't actually penetrate a given surface to be incident on
* another.  This has been taken care of using the dot product.
*
      VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
      VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
      VMAG1 = DSQRT(VMAG1)
      VMAG2 = VOX*VOX + VOY*VOY+ VOZ*VOZ
      VMAG2 = DSQRT(VMAG2)
*
* Determine if the dot product is 1 or -1.  If -1, the reflected ray
* passed through the plane tangent to the incident surface.  Return
* to the subprogram OPTICS.  If 1, the incident ray is reflected in
* the proper direction and program execution continues.
*
      IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) THEN
        T = -B - DSQRT(ARG)
        IF(I.EQ.2) THEN
          FLAG = 1
          RETURN
        ENDIF
        GO TO 10
      ENDIF
*
* If an intersection occurs on the header cylinder,
* record point of incidence and starting point of
* energy bundle in subprogram COORD.  Set flag and
* return to subprogram OPTICS.
*
      IF(Z.GE.ZSTOP.AND.Z.LE.ZFLAKE) THEN
        NN = 17
        NTAG = 1
        CALL COORD(X1,Y1,Z1,X,Y,Z)
        FLAG = 1
        RETURN
      ENDIF

      T = -B - DSQRT(ARG)
```

```
   10 CONTINUE

      FLAG = 1
      RETURN
*
* If subprogram COORD determines an intersection on the header
* to be the shortest distance between the emission site and
* site of incidence, program execution is returned here to determine
* whether the ray is absorbed or reflected.
*
*
* A random number is chosen and compared to the absorptivity of
* the point of incidence on the header.  If the random
* number chosen is less than or equal to the value of
* absorptivity for the detector header, the incident ray
* is absorbed.  If not,...
*
* Check cylindrical part of detector header.
*
    1 CONTINUE
      IF(NTAG.EQ.1) THEN
        CALL RANDOM(R)
        IF(R.LE.ABS10) THEN
          DF(II,36) = DF(II,36) + 1.D0
          FLAG = 0
          RETURN
        ELSE
          UNX = 2.D0*X/D10
          UNY = 2.D0*Y/D10
          UNZ = 0.D0
        ENDIF
      ENDIF
*
* Check flake substrate.
*
      IF(NTAG.EQ.0) THEN
        CALL RANDOM(R)
        IF(R.LE.ABS10) THEN
          DF(II,37) = DF(II,37) + 1.D0
          FLAG = 0
          RETURN
        ELSE
          UNX = 0.0D0
          UNY = 0.0D0
          UNZ = 1.0D0
        ENDIF
      ENDIF

    3 CONTINUE
      NTEST = 1
*
* ...another random number is chosen to determine how the incident
* ray is reflected.  If the random number is less than or equal to
* the value of the reflectivity of that part of the detector header,
* the ray is specularly reflected.  If not, a diffuse reflection
* occurs.
*
      CALL RANDOM (R)
      IF(R.LE.REFR10) THEN
        CALL REFLECT (X1,Y1,Z1,X,Y,Z,UNX,UNY,UNZ,VOX,VOY,VOZ)
      ELSE
        X1 = X
        Y1 = Y
        Z1 = Z
        CALL VECTOR (X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
      ENDIF
```

```
      RETURN
      END




************************************************************
*                                                          *
*                    SUBROUTINE COORD                      *
*                                                          *
* This subroutine records the coordinates of any emitted energy  *
* bundles incident on any part of the optics module.  Since an   *
* emitted energy bundle may hit several surfaces mathematically, *
* it may only hit one surface physically.  Thus, the distance    *
* formula is used to determine the shortest distance to one of   *
* the recorded coordinates of a given surface, given the         *
* coordinates of the energy bundle's emission site.              *
*                                                          *
************************************************************
*
      SUBROUTINE COORD (X1,Y1,Z1,X,Y,Z)
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Define storage variables, size and type.
*
      REAL*8 XN(20),YN(20),ZN(20),X1N(20),Y1N(20),Z1N(20),DIST(20)
*
* Place all common variables in one, unique storage block.
*
      COMMON /FLAGS/ NFLAG,NTEST,NN
      COMMON /COORDS/ XN,YN,ZN,X1N,Y1N,Z1N
*
* This is a flag to compute the distance from the emission site to the
* coordinates of surfaces that have intercepted the emitted energy
* bundle.
*
      IF(NN.EQ.20) GO TO 1
*
* Record coordinates where energy bundle is intercepted and which
* surface intercepts it (variable NN is a number designated to a
* particular part of the optics module).
*
      XN(NN) = X
      YN(NN) = Y
      ZN(NN) = Z
      X1N(NN) = X1
      Y1N(NN) = Y1
      Z1N(NN) = Z1
      RETURN
*
* Execution continues to determine shortest distance from emission
* site to intercepting surface.
*
    1 CONTINUE
      JJ = 0
      DO 10 I=1,20
        DISTNCE = (XN(I)-X1N(I))*(XN(I)-X1N(I))
        DISTNCE = DISTNCE + (YN(I)-Y1N(I))*(YN(I)-Y1N(I))
        DISTNCE = DISTNCE + (ZN(I)-Z1N(I))*(ZN(I)-Z1N(I))
        DISTNCE = DSQRT(DISTNCE)
*
* If no coordinates are recorded, skip that particular part of the
```

```
* optics module.
*
        IF(DISTNCE.LE.1.E-8) THEN
          GO TO 10
        ELSE
          JJ = JJ + 1
          IF(JJ.EQ.1) DISTOLD = DISTNCE
*
* Record coordinates of surface with shortest distance from emission
* site.  Return to subroutine OPTICS to execute the subprogram labeled
* with the integer corresponding to the number recorded in this
* loop to determine whether energy bundle is absorbed or reflected.
*
          IF(DISTNCE.LE.DISTOLD) THEN
            NN = I
            X = XN(I)
            Y = YN(I)
            Z = ZN(I)
            X1 = X1N(I)
            Y1 = Y1N(I)
            Z1 = Z1N(I)
            DISTOLD = DISTNCE
          ENDIF
        ENDIF
   10 CONTINUE
*
* Reset array of coordinates to zero.
*
      DO 20 J=1,20
        XN(J) = 0.0D0
        YN(J) = 0.0D0
        ZN(J) = 0.0D0
        X1N(J) = 0.0D0
        Y1N(J) = 0.0D0
        Z1N(J) = 0.0D0
   20 CONTINUE

      RETURN
      END




*********************************************************************
*                                                                   *
*                       SUBROUTINE FILTER                           *
*                                                                   *
* This subprogram monitors the filters of the longwave and shortwave *
* radiometric channels.  The wavelength of the emitted energy bundle *
* is compared to the transmissivity of the filters at that wave-    *
* length to determine whether the energy bundle is reflected or     *
* transmitted.                                                      *
*                                                                   *
*********************************************************************
*
      SUBROUTINE FILTER (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Specify storage parameters, size and type.
*
      INTEGER*2 FLAG
      REAL*8 KAPPA1,KAPPA2
*
```

```
* Place all common variables in one, unique storage block.
*
      COMMON /GEOM/ H1,D1,D2,THETA1,C1,C2,C3,C4,C5,H2,D3,D4,D5
      COMMON /GEOM/ DEL1,DEL2,THETA2,PHI,H3,HL1,HL2,D6,THETA3
      COMMON /GEOM/ H4,HCURVE,D7,KAPPA1,H5,D8,KAPPA2,THETA4,H6,D9
      COMMON /GEOM/ THETA5,DIM1,DIM2,HREF,DREF
      COMMON /GEOM1/ H7,D10,FLAKDIM,ABS11,REFR11

      COMMON /MATL/ ABS1,ABS2,ABS3,ABS4,ABS5,ABS6,ABS7,ABS8,ABS9
      COMMON /MATL/ ABS10,REFR1,REFR2,REFR3,REFR4,REFR5,REFR6
      COMMON /MATL/ REFR7,REFR8,REFR9,REFR10

      COMMON /FLAGS/ NFLAG,NTEST,NN
      COMMON /CHANNEL/ ALAMBDA,NCHANEL,NFLTRSW,NSPCTRM
*
* Initialize necessary data.
*
      NTEST = 0
      IF(NFLAG.EQ.1) GO TO 1
      IF(NFLTRSW.EQ.1) THEN
        ZFILTER = HREF+C5+(H1-C5)/2.D0
      ELSE
        ZFILTER = HREF+H1+H2+H3+H4+HCURVE+(H7/2.D0)
      ENDIF
*
* Compute where emitted energy bundle strikes a filter.
*
      T = (ZFILTER-Z1)/VOZ
      X = T*VOX + X1
      Y = T*VOY + Y1
      Z = ZFILTER
*
* If the coordinates of this intersection coincide with the previous
* intersection, change the sign in the quadratic equation and
* continue program execution.
*
      IF(DABS(X-X1).LT.1.D-10.AND.DABS(Y-Y1).LT.1.D-10.AND.
     &  DABS(Z-Z1).LT.1.D-10) GO TO 2
*
* The nature of this program makes it necessary to be sure a ray
* doesn't actually penetrate a given surface to be incident on
* another.  This has been taken care of using the dot product.
*
      VDOT = (X-X1)*VOX + (Y-Y1)*VOY + (Z-Z1)*VOZ
      VMAG1 = (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) + (Z-Z1)*(Z-Z1)
      VMAG1 = DSQRT(VMAG1)
      VMAG2 = VOX*VOX + VOY*VOY + VOZ*VOZ
      VMAG2 = DSQRT(VMAG2)
*
* Determine if the dot product is 1 or -1.  If -1, the reflected ray
* passed through the plane tangent to the incident surface.  Return
* to the subprogram OPTICS.  If 1, the incident ray is reflected in
* the proper direction and program execution continues.
*
      IF(VDOT/(VMAG1*VMAG2).LT.0.0D0) GO TO 2
*
* If energy bundle hits the filter, record inpingement location in
* subprogram COORD and continue program execution.
*
      RADIUS = DSQRT(X*X + Y*Y)
      IF(RADIUS.LT.D4/2.0D0) THEN

        IF(NFLTRSW.EQ.1) THEN
          NN = 15
        ELSE
          NN = 16
        ENDIF
```

```
        CALL COORD (X1,Y1,Z1,X,Y,Z)
        FLAG = 1
        RETURN
      ENDIF

    2 CONTINUE
      FLAG = 1
      RETURN
*
* If subprogram COORD determines an intersection with the filter
* to be the shortest distance between the emission site and
* site of incidence, program execution is returned here to determine
* whether the ray is transmitted or reflected.
*
*
* A random number is chosen and compared to the transmissivity of
* the filter at that wavelength. If the random number chosen
* is less than or equal to the value of the transmissivity
* for the filter,  the energy bundle is transmitted.  If not, the
* bundle is reflected.
*
* Check the filter.
*
    1 CONTINUE
      NTEST = 1
*
* Choose appropriate unit normal depending on which side of the
* filter the energy bundle strikes.
*
      IF(Z1.LE.Z) THEN
        UNX = 0.0D0
        UNY = 0.0D0
        UNZ = 1.0D0
      ELSE
        UNX = 0.0D0
        UNY = 0.0D0
        UNZ = -1.0D0
      ENDIF
*
* Find the transmissivity of the filter according to the wavelength
* of the energy bundle.
*
      CALL XMIT (TAU)
      CALL RANDOM (R)
*
* If the bundle is transmitted, its direction is the same as the
* incident direction.
*
      IF(R.LE.TAU) THEN
        VOX = X-X1
        VOY = Y-Y1
        VOZ = Z-Z1
        VMAG = VOX*VOX + VOY*VOY + VOZ*VOZ
        VMAG = DSQRT(VMAG)
        VOX = VOX/VMAG
        VOY = VOY/VMAG
        VOZ = VOZ/VMAG
        X1 = X
        Y1 = Y
        Z1 = Z
        FLAG = 1
        RETURN
      ENDIF
*
* If the bundle is not transmitted, it is specularly reflected.
*
```

```
      CALL REFLECT (X1,Y1,Z1,X,Y,Z,UNX,UNY,UNZ,VOX,VOY,VOZ)

      RETURN
      END



*****************************************************************
*                                                               *
*                      SUBROUTINE REFLECT                       *
*                                                               *
* This subroutine computes the direction of a specularly        *
* reflected ray within the optics module.                       *
*                                                               *
*****************************************************************
*
      SUBROUTINE REFLECT (X1,Y1,Z1,X,Y,Z,UNX,UNY,UNZ,VOX,VOY,VOZ)
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Compute dot product of incident ray and unit normal to incident
* surface.
*
      VDOT = (X-X1)*UNX + (Y-Y1)*UNY + (Z-Z1)*UNZ

*
* Compute direction of reflected ray based on Snell's law.
*
      VOX = X - X1 - 2.0D0*VDOT*UNX
      VOY = Y - Y1 - 2.0D0*VDOT*UNY
      VOZ = Z - Z1 - 2.0D0*VDOT*UNZ
*
* Normalize this vector.
*
      VMAG = DSQRT(VOX*VOX + VOY*VOY + VOZ*VOZ)

      VOX = VOX/VMAG
      VOY = VOY/VMAG
      VOZ = VOZ/VMAG
*
* Make current point of interception new emission site.
*
      X1 = X
      Y1 = Y
      Z1 = Z

      RETURN
      END



*****************************************************************
*                                                               *
*                      SUBROUTINE VECTOR                        *
*                                                               *
* This subroutine finds the local normal vector to the surface  *
* and finds the diffusely reflected shot in terms of the global *
* coordinate system.                                            *
*                                                               *
*****************************************************************
*
```

```
      SUBROUTINE VECTOR (X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)

      PI = DACOS(-1.0D0)
*
* Define the center of the unit sphere tangent to the cone surface.
*
      IF(UNZ.EQ.1.0D0.OR.UNZ.EQ.-1.0D0) GO TO 10

      UTX = -UNX*DSQRT(UNZ*UNZ/(1.D0 - UNZ*UNZ))
      UTY = -UNY*DSQRT(UNZ*UNZ/(1.D0 - UNZ*UNZ))
      UTZ = DSQRT(1.D0 - UNZ*UNZ)
      IF(UNZ.LT.1.E-8) UTZ = -UTZ
      GO TO 11

   10 CONTINUE
      UTX = 0.0D0
      UTY = 1.0D0
      UTZ = 0.0D0

   11 CONTINUE
      USX = -UNY*UTZ + UNZ*UTY
      USY = -UNZ*UTX + UNX*UTZ
      USZ = 0.D0
*
* In terms of the local coordinate system associated with the
* unit sphere, x, y, and z are defined as follows:
*
      CALL RANDOM (R)
      THETA = DASIN(DSQRT(R))

      CALL RANDOM (R)
      PHI = 2.0D0 * PI * R

      XS = DCOS(PHI) * DSIN(THETA)
      YS = DSIN(PHI) * DSIN(THETA)
      ZS = DCOS(THETA)
*
* In the global coordinate system, x, y, and z are defined:
*
      VOX = XS*UTX + YS*USX - ZS*UNX
      VOY = XS*UTY + YS*USY - ZS*UNY
      VOZ = XS*UTZ + YS*USZ - ZS*UNZ

      RETURN
      END




*****************************************************************
*                                                               *
*                      SUBROUTINE VECTOR1                       *
*                                                               *
* This subroutine finds the local normal vector to the surface  *
* and finds the diffusely reflected shot in terms of the global *
* coordinate system.                                            *
*                                                               *
*****************************************************************
*
      SUBROUTINE VECTOR1 (X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
*
```

```
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)

      PI = DACOS(-1.0D0)
      ZETA = 14.314D0*PI/180.D0
      ZETA = DCOS(ZETA)
*
* Define the center of the unit sphere tangent to the cone surface.
*
      XCTR = X1 - UNX
      YCTR = Y1 - UNY
      ZCTR = Z1 - UNZ
*
* In terms of the local coordinate system associated with the
* unit sphere, x, y, and z are defined as follows:
*
      CALL RANDOM (R)
      ZS = R * (1.0D0-ZETA) + ZETA
      THETA = DACOS(ZS)

      CALL RANDOM (R)
      PHI = 2.0D0 * PI * R

      XS = DCOS(PHI) * DSIN(THETA)
      YS = DSIN(PHI) * DSIN(THETA)
*
* In the global coordinate system, x, y, and z are defined:
*
      X2 = XCTR + XS
      Y2 = YCTR + YS
      Z2 = ZCTR + ZS

      VOX = X2 - X1
      VOY = Y2 - Y1
      VOZ = Z2 - Z1

      RETURN
      END



*****************************************************************
*                                                               *
*                       SUBROUTINE RANDOM                       *
*                                                               *
* This subroutine uses the ISML subroutine RNUN to generate     *
* uniform pseudo-random numbers in an efficient way.  Courtesy  *
* of N. E. Tira.                                                *
*                                                               *
*****************************************************************
      SUBROUTINE RANDOM(R)
      COMMON /RAND/ IN
      REAL*4 R,Z(200)
      IF(IN.EQ.201) GO TO 10
      IN = IN+1
      IF(IN.LE.200) GO TO 20
   10 CONTINUE
      IN = 1
      NR = 200
      CALL RNUN (NR,Z)
   20 CONTINUE
      R = Z(IN)
      RETURN
      END
```

```
*****************************************************************
*                                                               *
*                      SUBROUTINE BBFCN                         *
*                                                               *
* This subroutine is used by the SUBROUTINE NEWTON to integrate *
* Planck's blackbody distribution function to find the wavelength *
* of an emitted energy bundle.                                  *
*                                                               *
*****************************************************************
*
      SUBROUTINE BBFCN (ALAMBDA,TEMP)
*
* Declare all variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Declare function subprogram.
*
      EXTERNAL FCN
*
* Initialize necessary variables.
*
      N = 1
      MAXFCN = 10000
      TOL = 0.1D-4
*
* Define limits of integration and call SUBROUTINE ROMB.
*
      A = 0.1D-8
      B = 1897.8D0
      CALL ROMB(FCN,A,B,TOL,RESULT)
*
* Determine the wavelength-temperature product corresponding to the
* random value of the blackbody fraction.
*
      CALL NEWTON (B,F,RESULT,NFLAG,OLDERR,ITER,OMEGA)
*
* If the product is greater than 100000 µm-K, set the product at that
* value.
*
      IF(B.GE.100000.D0) THEN
         B = 100000.D0
         NFLAG = 0
      ENDIF
*
* If the product is negative, try finding the correct value by taking
* the initial guess at 50000 µm-K instead of 1897.8 µm-K.
*
      IF(B.LT.0.0D0) THEN
        OMEGA = 0.1D0
        B  = 50000.D0
        ITER = 0
        GO TO 1
      ENDIF
*
* Iterate until root (wavelength-temperature product) is found.
*
      ITER = ITER + 1
      IF(NFLAG.EQ.1) GO TO 1
*
* Calculate the wavelength.
*
```

```
      ALAMBDA = B/TEMP
      IF(ALAMBDA.GT.50.D0) ALAMBDA = 50.D0

      RETURN
      END
C
C -----------------------------------------------------------------
C
*
* This is the function subprogram for SUBROUTINE BBFCN containing
* Planck's blackbody distribution function.
*
      FUNCTION FCN(X)
*
* Declare all variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Initialize all necessary variables.
*
      C1 = 0.59544D8
      C2 = 14388.D0
      C3 = 2897.8
      C4 = 4.095D-12
      SIGMA = 5.6696D-8
      PI = DACOS(-1.D0)
*
* Call the no floating point underflow subroutine to avoid errors in
* dividing a number on the order of 1.E-8 by another number of the
* same order.
*
      CALL XUFLOW(0)
*
* The function.
*
      FCN = (2.D0*PI*C1) * DEXP(-C2/X)
      FCN = FCN/(X**5 * (1.D0 - DEXP(-C2/X)))
      FCN = FCN/SIGMA
      RETURN
      END
C
C   -------------------------------------------------------------
C
      SUBROUTINE ROMB(FCN,A,B,TOL,RESULT)
C
C   -------------------------------------------------------------
C
C     SUBROUTINE ROMB:
c                        Subroutine for Romberg integration. Program
c     begins with trapezoidal integration with 10 subintervals.
c     Intervals are then halved and results are extapolated up to
c     eight order.  Maximum number of subintervals used in program is
C     2560.  Courtesy of Dr. Farshad Kowsary.
C
C   -------------------------------------------------------------
C
C     parameters are :
C     FCN    - function that computes f(x), declared external in main
C     A,B    - integration limits
C     TOL    - tolerance value used to terminate the iteration
C     RESULT - returns value to the integral caller
C     TRAP   - doubly subscripted array that holds intermediate values
C              for comparisons and extrapolation
C     KFLAG  - flag used internally to signal non-convergence. when
C              kflag = 0, means non-convergent, =1 means all ok.
C
```

```
C    -----------------------------------------------------------------
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION TRAP(9,9)
      INTEGER I,L,K,KFLAG
*
*  Set flag at 1 initially
*
      KFLAG = 1
      IPRINT = 0
*
*  Compute first integral with 10 subintervals and using trap rule
*
      H = (B-A)/10.0D0
      SUM = FCN(A)+FCN(B)
      X = A
      DO 10 I = 2,10
        X = X+H
        SUM = SUM+FCN(X)*2.0D0
 10   CONTINUE
      TRAP(1,1) = H/2.0D0*SUM
*
*   Recompute integral with H halved, extrapolate and test. Repeat
*   up to eight times.
*
      DO 20 I=1,8
       H = H/2.0D0
       X = A+H
       K = 10*2**I
       DO 30 J=2,K,2
         SUM = SUM + FCN(X)*2.0D0
         X = X+H+H
 30    CONTINUE
       TRAP(1,I+1) = H/2.0D0*SUM
       DO 40 L = 1,I
        TRAP(L+1,I+1) = TRAP(L,I+1)+1.0D0/(4.0D0**L-1.0D0)*
     *                 (TRAP(L,I+1)-TRAP(L,I))
 40    CONTINUE
       IF(DABS(TRAP(I+1,I+1)-TRAP(I,I+1))-TOL) 50,50,20
 20    CONTINUE
*
*    -----------------------------------------------------------------
*
*      If tolerance not met after 8 extrapolations, print note and set
*      KFLAG = 0.
*
       KFLAG = 0
       WRITE 200
*
*      Print intermediate results
*
 50    I = I+1
       IF(IPRINT.EQ.0) GOTO 80
       DO 70 L = 1,I
        WRITE 203, (TRAP(J,L),J=1,L)
 70     CONTINUE
 80     IF(KFLAG.EQ.0) STOP
        RESULT = TRAP(I,I)
 200    FORMAT(/' TOLERANCE NOT MET. CALCULATED VALUES HERE ')
 203    FORMAT(1X,8F12.6)
        RETURN
        END
```

```
*****************************************************************
*                                                               *
*                      SUBROUTINE NEWTON                        *
*                                                               *
* This subroutine picks the wavelength of an emitted energy     *
* bundle for gray surfaces.  A random number is chosen which    *
* corresponds to the blackbody fraction, F.  The Newton-Raphson *
* method is used to find the wavelength-temperature product     *
* corresponding to that fraction.                               *
*                                                               *
*****************************************************************
*
      SUBROUTINE NEWTON (B,F,RESULT,NFLAG,OLDERR,ITER,OMEGA)
*
* Declare all variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Initialize variables.
*
      NFLAG = 0
      TOL = 0.1D-3
      C1 = 0.59544D8
      C2 = 14388.D0
      SIGMA = 5.6696D-8
      PI = DACOS(-1.D0)
*
* Keep old error calculation for error analysis.
*
      BOLD = B
      RESULT = RESULT - F
*
* Underflow results from nature of program.  Call resident subroutine
* to fix underflow calculations.
*
      CALL XUFLOW (0)
*
* Compute derivative of blackbod;y fraction integral.
*
      DF = (2.D0*PI*C1) * DEXP(-C2/B)
      DF = DF/(B**5 * (1.D0 - DEXP(-C2/B)))
      DF = DF/SIGMA
*
* Compute final result for the wavelength-temperature product.
*
      B = BOLD - OMEGA * RESULT/DF
*
* Compute error estimate.
*
      ERROR = B - BOLD
      RELERR = DABS(ERROR) - DABS(OLDERR)
*
* If error between two successive calculations is small, compute
* average value and return.
*
      IF(DABS(RELERR).LE.0.1D0) THEN
        B = (B + BOLD)/2.D0
        NFLAG = 0
        RETURN
      ENDIF
*
* If solution is not converging, then cusion the derivative used in
* Newton's method and search for root again.
*
      IF(ITER.GT.20) THEN
        OMEGA = OMEGA/10.D0
        IF(OMEGA.LT.0.001D0) THEN
```

```
          B = (B + BOLD)/2.D0
          NFLAG = 0
          RETURN
        ENDIF
        ITER = 0
      ENDIF
*
* If error tolerance is not met, continue searching for the root.
*
      IF(DABS(ERROR).E.TOL) THEN
        OLDERR = ERROR
        NFLAG = 1
      ENDIF

      RETURN
      END




*******************************************************************
*                                                                 *
*                        SUBROUTINE RHOSPEC                       *
*                                                                 *
* This subprogram finds the specular-to-total reflectivity ratio  *
* of the mirrors of the ERBE scanning radiometer.  This ratio is  *
* based on the wavelength of the incoming radiation (energy       *
* bundle).                                                        *
*                                                                 *
*******************************************************************
*
      SUBROUTINE RHOSPEC (ALAMBDA,REFR)
*
* Declare all variables double precision.
*
      IMPLICIT REAL*8 (A-H,L,O-Z)
*
* Interpolate the value of REFR for the wavlelength of the energy bundle
*
      LLAMBDA = 0.0D0
      LREFR = 0.0D0

    1 CONTINUE
      READ(13,*) ULAMBDA,UREFR
*
* Interpolate to find value for REFR.
*
      IF(ALAMBDA.GE.LLAMBDA.AND.ALAMBDA.LE.ULAMBDA) THEN
        REFR = (UREFR - LREFR) / (ULAMBDA - LLAMBDA)
        REFR = REFR * (ALAMBDA - LLAMBDA) + LREFR
      ELSE
        LLAMBDA = ULAMBDA
        LREFR = UREFR
        GO TO 1
      ENDIF

      REWIND 13

      RETURN
      END
```

```
*****************************************************************
*                                                               *
*                     SUBROUTINE XMIT                           *
*                                                               *
* This subprogram finds the transmissivity of the filters of the *
* longwave and shortwave radiometric channels of the ERBE scanning *
* thermistor bolometer radiometer.  This transmissivity, TAU, is *
* based on the wavelength of the incoming radiation (energy     *
* bundle).                                                      *
*                                                               *
*****************************************************************
*
      SUBROUTINE XMIT (TAU)
*
* Declare all variables double precision.
*
      IMPLICIT REAL*8 (A-H,L,O-Z)
*
* Place frequently used variables in a unique storage location.
*
      COMMON /CHANNEL/ ALAMBDA,NCHANEL,NFLTRSH,NSPCTRM
*
* Depending on the radiometric channel, read values of wavelength
* and transmissivity for that channel and interpolate the value
* of TAU for the wavelength of the energy bundle.
*
      NUNIT = NCHANEL + 10

      LLAMBDA = 0.0D0
      LTAU = 0.0D0

    1 CONTINUE
      READ(NUNIT,*) ULAMBDA,UTAU
*
* Interpolate to find value for TAU.
*
      IF(ALAMBDA.GE.LLAMBDA.AND.ALAMBDA.LE.ULAMBDA) THEN
        TAU = (UTAU - LTAU) / (ULAMBDA - LLAMBDA)
        TAU = TAU * (ALAMBDA - LLAMBDA) + LTAU
      ELSE
        LLAMBDA = ULAMBDA
        LTAU = UTAU
        GO TO 1
      ENDIF

      REWIND NUNIT

      RETURN
      END




*****************************************************************
*                                                               *
*                     SUBROUTINE PRINT                          *
*                                                               *
* This subprogram prints the number of energy bundles absorbed  *
* by each part of the optics module of the ERBE scanning        *
* radiometer.                                                   *
*                                                               *
*****************************************************************
*
      SUBROUTINE PRINT (ABSSHT,NSHOTS)
*
```

```
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
*
* Specify size and type of storage perimeters.
*
      REAL*8 DF(40,40)
*
* Place all common variables in one, unique storage block.
*
      COMMON /DISTR/ DF,II
*
* Title the file.
*
      WRITE(9,100)
*
* Account for all absorbed energy bundles.
*
      DO 1 I=1,37
   1    ABSSHT = ABSSHT + DF(II,I)

      DO 5 I=1,10
        IF(I.EQ.10) THEN
          WRITE(9,900) DF(II,35)
        ELSE
          WRITE(9,200) I,DF(II,I)
        ENDIF
    5 CONTINUE

      COUNT1 = DF(II,25) + DF(II,32)
      COUNT2 = DF(II,27) + DF(II,34)
      WRITE(9,300) DF(II,17),DF(II,18),DF(II,20),DF(II,19),DF(II,31),
     &DF(II,22),DF(II,23),DF(II,24),COUNT1,COUNT2,DF(II,10)

      WRITE(9,400) (I,DF(II,10+I),I=1,3)
      WRITE(9,500) (J,DF(II,13+J),J=1,3)

      COUNT13 = DF(II,26) + DF(II,33)
      WRITE(9,600) COUNT13,DF(II,21)
      WRITE(9,700) DF(II,28),DF(II,29),DF(II,30)
      NSUM = DF(II,36) + DF(II,37)
      WRITE(9,999) NSUM

      ESCSHT = NSHOTS - ABSSHT
      WRITE(9,800) ABSSHT,ESCSHT
*
************ Format Statements *****************
*
  100 FORMAT(10X,'NUMBER OF ENERGY BUNDLES ABSORBED',/)
  200 FORMAT(5X,'Reflector cap, part',I3,':',F7.1)
  300 FORMAT(5X,'Perimeter of middle of secondary mirror mount :',
     &F7.1,/,5X,'Perimeter of secondary mirror :',F7.1,/,5X,
     &'Conical portion of secondary mirror mount :',F7.1,/,
     &5X,'Cylindrical portion of secondary mirror mount :',F7.1,/,
     &5X,'Rim of secondary mirror mount towards flake :',
     &F7.1,/,5X,'Detector housing, cylinder 1 :',F7.1,/,5X,
     &'Detector housing, cone 1 :',F7.1,/,5X,
     &'Detector housing, cone 2 :',F7.1,/,
     &5X,'Detector housing, cylinder 2 :',F7.1,/,5X,
     &'Primary insert :',F7.1,/,5X,
     &'Top of secondary mirror mount :',F7.1)
  400 FORMAT(5X,'Top of spider leg number',I3,':',F7.1)
  500 FORMAT(5X,'Bottom of spider leg number',I3,':',F7.1)
  600 FORMAT(5X,'Primary mirror :',F7.1,/,5X,'Secondary mirror :',F7.1)
  700 FORMAT(5X,'Field of view limiter, top :',F7.1,/,5X,
     &'Field of view limiter, bottom :',F7.1,/,5X,
     &'Active Flake :',F7.1,/)
```

```
800 FORMAT(5X,'Number of energy bundles absorbed =',F8.1,/,5X,
   &         'Number of energy bundles escaped =',F8.1,/)
900 FORMAT(5X,'Aperture :',F7.1)
999 FORMAT(5X,'Number of shots missing flake :',F7.1)

    RETURN
    END
```

Vita

Jeffrey L. Meekins is a native of Lanexa, Virginia. He moved to Richmond, Virginia in December, 1979, and graduated from Highland Springs High School in June, 1983. In September, 1983, Jeffrey matriculated at Virginia Polytechnic Institute and State University where he received his Bachelor of Science in Electrical Engineering on July 1, 1988. During his undergraduate study, he participated in the Cooperative Education program as a plant engineer for Philip Morris, U.S.A., Richmond, Virginia. He received scholarships from corporations like ALCOA and DuPont for outstanding achievement in electrical engineering.

In August, 1988, Jeffrey started his graduate studies at Virginia Polytechnic Institute and State University as a Minority Graduate Research Fellow for the National Aeronautics and Space Administration (NASA). He served in this capacity until receiving his Master of Science in Mechanical Engineering in August, 1990.

Jeffrey L. Meekins

Jeffrey L. Meekins