A STUDY OF THE THERMAL AND OPTICAL CHARACTERISTICS OF RADIOMETRIC CHANNELS FOR EARTH RADIATION BUDGET APPLICATIONS

FINAL REPORT

64877
P-306

Grants NAG-1-1016 and NAG-1-1167

Prof. J. R. Mahan, Principal Investigator
Dr. Nour E. Tira, Project Assistant

December 4, 1991

Notice

This document is essentially Dr. Nour E. Tira's doctoral dissertation. It documents the effort done by Dr. Tira and Prof. J. R. Mahan under NASA Grants NAG-1-1016 and NAG-1-1167 between June 8, 1989, and October 21, 1991. The authors are indebted to the Radiation Sciences Branch at NASA's Langley Research Center for their support of this work, and especially to Mr. Robert B. Lee, III, of the Radiation Sciences Branch.
Questions about the work documented in this report should be directed to:

Professor J. R. Mahan
Mechanical Engineering Department
Virginia Polytechnic Institute & State University
Blacksburg, VA 24061
Phone: (703) 231-7467

A Study of the Thermal and Optical Characteristics of Radiometric Channels for Earth Radiation Budget Applications

by

Nour Eddine Tira

Dr. J. R. Mahan, Chairman

(ABSTRACT)

An improved dynamic electrothermal model for the Earth Radiation Budget Experiment (ERBE) total, nonscanning channels is formulated. This model is then used to accurately simulate two types of dynamic solar observation: the solar calibration and the so-called pitchover maneuver. Using a second model, the nonscanner active cavity radiometer (ACR) thermal noise is studied. This study reveals that radiative emission and scattering by the surrounding parts of the nonscanner cavity are acceptably small.

The dynamic electrothermal model is also used to compute the ACR instrument transfer function. Accurate in-flight measurement of this transfer function is shown to depend on the energy distribution over the frequency spectrum of the radiation input function. A new array-type field-of-view (FOV) limiter, whose geometry controls the input function, is proposed for in-flight calibration of an ACR and other types of radiometers.

Finally, the point spread function (PSF) of the ERBE and the Clouds and Earth's Radiant Energy System (CERES) scanning radiometers is computed. This PSF is useful in characterizing the channel optics. It also has potential use for recovering the distribution of the radiative flux coming from the Earth by its deconvolution with satellite measurements.

Acknowledgements

Many thanks are due to my advisor, J.R. Mahan, for his guidance throughout this research. I also thank my committee members, especially Dr. A. L. Wicks, for serving on my committee. A particular debt of gratitude is owed to the Radiation Sciences Branch of NASA's Langley Research Center for their continuous financial support of this research through grants NAG-1-1016 and NAG-1-1167. I am especially grateful to Mr. Robert B. Lee, III, for his help in more than one aspect of my work. I thank my wife, Alice, for the support and tolerance she has shown during my long years of study. Finally, for those who made my life joyful; my mother Zohra, my father Hamda, my son Joseph and all my friends; I salute you all.

PRECEDING PAGE BLANK NOT FILMED

Table of Contents

0.0 Preface 1

0.1 Background 1

0.1.1 The Earth Radiation Budget Experiment (ERBE) 3

0.1.2 The Clouds and Earth's Radiant Energy System (CERES) . . . 3

0.2 Dissertation Organization 4

PART I

1.0 Introduction 8

2.0 The Dynamic Electrothermal Model 12

2.1 Radiative Interchange 12

2.2 Heat Conduction 15

3.0 Determination of the Illuminated Area in the Cavity 17

3.1 First Case: Intersection Above the x-Axis 21

3.2 Second Case: Intersection Below the x-Axis 21

4.0 Model Simulation Results 23

4.1 The Distribution Factors 23

4.2 Solar Calibration 24

4.3 The Pitchover Maneuver 29

5.0 Evaluation of Thermal Noise 31

5.2 Scattering Thermal Noise 33

6.0 Thermal Noise Results 36

7.0 Conclusion 43

PART II

1.0 Introduction 45

1.1 Objective 45

1.2 Data Classification 46

1.2.1 Deterministic Data 47

1.2.2 Random Data 47

2.0 Review of Fourier Analysis 49

2.1 Fourier Series 49

2.2 Fourier Transform 52

2.4 Sampling Theorem 53

2.5 Sampling Errors 54

2.5.1 Aliasing 54

2.5.2 Leakage 55

2.5.3 Round-Off 56

2.6 The Discrete Fourier Transform (DFT) and the Fast Fourier Transform (FFT) 60

3.0 The Transfer Function 62

3.1 Introduction 62

4.0 The Transfer Function of the ERBE Nonscanner 65

4.1 The Direct Method 65

4.2 Method of Least Squares 67

4.3 The Newton-Raphson Method 69

4.4 Transfer Function Results 71

4.5 Use of Square-Wave Input Function 75

4.6 Study of Simulated and Real Solar Calibration Data for the ACR 81

5.0 Investigation of the Role of Field-of-View Limiters in Dynamic Calibration 89

5.1 A Single Circular Aperture 90

5.2 An Array of Equal-Radius Circular Apertures 99

5.3 An Array of Varying-Radius Circular Apertures 103

5.4 Study of the Transfer Functions Obtained Using Array Apertures 116

6.0 Concluding Remarks 124

PART III

1.0 Introduction 127

1.1 Scanner Optical Design 127

1.2 Objective 133

2.0 The Point Spread Function 134

2.1 Definition 134

2.2 General Distribution 141

2.2.1 Linearity 142

2.2.2 Isoplanatism 142

2.2.3 Coherence 143

3.0 Aberrations 145

3.1 Spherical Aberration 147

3.2 Astigmatism 149

3.3 Coma 149

3.4 Field Curvature 152

3.5 Distortion 152

3.6 Chromatic Aberration 153

4.0 Scanner Optical Model Formulation 155

5.0 Results and Discussion 157

5.1 Diffuse Source Radiation 158

5.2 Collimated Radiation 160

5.2.1 PSF for a Given Angle of Incidence 160

5.2.2 PSF for Random Angles of Incidence 167

6.0 Conclusions and Recommendations 175

References 177

NOISE PROGRAM 181

FOV PROGRAMS 193

PSF PROGRAMS 198

List of Illustrations

Fig. I.1 ERBE Nonscanning ACR Instrument Cavity. 10

Fig. I.2 The ERBE ACR Electrical Feedback Temperature Control: (a) Block Diagram, (b) Electrical Schematic. 11

Fig. I.3 Collimated Radiation Entering the Cavity During (a) Solar Calibration and (b) the Pitchover Maneuver. 14

Fig. I.4 Illuminated Area in the Cavity 20

Fig. I.5 Distribution Factors for Solar Calibration at Incident Angles (a) 3.3, (b) 6.6 and (c) 7.7 deg. 25

Fig. I.6 Distribution Factors for the Pitchover Maneuver at Incident Angles (a) 19.2, (b) 48.0 and (c) 60.0 deg. 26

Fig. I.7 Comparison of the Simulated and Observed Solar Calibration Event for the ERBS Total, WFOV Channel (12/28/1984) (a) Present Model and (b) Previous Model [16]. 27

Fig. I.8 Comparison of the Simulated and Observed Solar Calibration Event for the ERBS Total, WFOV Channel (10/31/1985) (a) Present Model and (b) Previous Model [16]. 28

Fig. I.9 Comparison of the Simulated and Observed Pitchover Maneuver for the ERBS Total, WFOV Channel on (a) November 21, 1984 (b)

October 19, 1985. 30

Fig. I.10 The Radiometer Enclosure Geometry. 32

Fig. I.11 Variation of Cavity Absorption with the Number of Energy Bundles Used in the Monte-Carlo Ray Trace During (a) Solar Calibration and (b) the Pitchover Maneuver. 39

Fig. I.12 Total and Direct Cavity Absorption of Radiation During (a) Solar Calibration and (b) the Pitchover Maneuver. 40

Fig. II.1 Illustration of Leakage errors. 58

Fig. II.2 Illustration of Round-off errors. 59

Fig. II.3 Input and Output Functions for a Sinusoidal Input Signal, f = 0.04 Hz. 68

Fig. II.4 ACR Bode Diagram: (a) Gain, (b) Phase Angle. 74

Fig. II.5 Square-Wave Input Function (First 10 Harmonics). 78

Fig. II.6 Bode Diagram for Square Wave (First 10 Harmonics): (a) Gain, (b) Phase Angle. 79

Fig. II.7 Bode Diagram for Square Wave (Second 10 Harmonics): (a) Gain, (b) Phase Angle. 80

Fig. II.8 Simulated Solar Calibration Data of 12/28/1984: (a) Input and Output Functions, (b) Corresponding Frequency Spectra. 83

Fig. II.9 Satellite Solar Calibration Data of 12/28/1984: (a) Input and Output Functions, (b) Corresponding Frequency Spectra. 84

Fig. II.10 Bode Diagram for the Simulated Solar Calibration Data of

12/28/1984: (a) Gain, (b) Phase Angle. 85

Fig. II.11 Bode Diagram for the Actual Solar Calibration Data of 12/28/1984: (a) Gain, (b) Phase Angle. 86

Fig. II.12 Bode Diagram for Simulated Solar Calibration Data of 12/28/1984 (with Hanning Window): (a) Gain, (b) Phase Angle. 87

Fig. II.13 Bode Diagram for the Actual Solar Calibration Data of 12/28/1984 (with Hanning Window): (a) Gain, (b) Phase Angle. 88

Fig. II.14 A Simple Circular Aperture FOV Limiter. 92

Fig. II.15 Case 1, Table II.2: (a) Input Function, (b) Corresponding Frequency Spectrum. 94

Fig. II.16 Case 2, Table II.2: (a) Input Function, (b) Corresponding Frequency Spectrum. 95

Fig. II.17 Case 3, Table II.2: (a) Input Function, (b) Corresponding Frequency Spectrum. 96

Fig. II.18 Case 4, Table II.2: (a) Input Function, (b) Corresponding Frequency Spectrum. 97

Fig. II.19 Case 5, Table II.2: (a) Input Function, (b) Corresponding Frequency Spectrum. 98

Fig. II.20 A FOV Limiter Consisting of an Array of Circular Secondary Apertures. 101

Fig. II.21 An Array of Equal-Radius Secondary Apertures: (a) Input Function, (b) Corresponding Frequency Spectrum. 102

Fig. II.22 Case 1, Table II.3: (a) Input Function, (b) Corresponding Frequency

Spectrum. 108

Fig. II.23 Case 2, Table II.3: (a) Input Function, (b) Corresponding Frequency Spectrum. 109

Fig. II.24 Case 2(a), Table II.3: (a) Input Function, (b) Corresponding Frequency Spectrum. 110

Fig. II.25 Case 2(b), Table II.3: (b) Input Function, (b) Corresponding Frequency Spectrum. 111

Fig. II.26 Case 2(c), Table II.3: (a) Input Function, (b) Corresponding Frequency Spectrum. 112

Fig. II.27 Case 3, Table II.3: (a) Input Function, (b) Frequency Spectrum. . . . 113

Fig. II.28 Case 4, Table II.3: (a) Input Function, (b) Frequency Spectrum. . . . 114

Fig. II.29 Case 5, Table II.3: (a) Input Function, (b) Frequency Spectrum. . . . 115

Fig. II.30 Case 2(a), Table II.3: (a) Input and output Functions, (b) Corresponding Frequency Spectra. 118

Fig. II.31 Case 2(c), Table II.3: (a) Input and output Functions, (b) Corresponding Frequency Spectra. 119

Fig. II.32 Bode Diagram for Case 2(a), Table II.3: (a) Gain, (b) Phase Angle. 120

Fig. II.33 Bode Diagram for Case 2(c), Table II.3: (a) Gain, (b) Phase Angle. 121

Fig. II.34 Case 5, Table II.3: (a) Input and output Functions, (b) Corresponding Frequency Spectra. 122

Fig. II.35 Bode Diagram for Case 5, Table II.3: (a) Gain, (b) Phase Angle. . . 123

Fig. III.1 ERBE Scanner Optical Assembly. 129
Fig. III.2 ERBE Scanner Optical Module. 130
Fig. III.3 Relative Positions of the Field Stop and the Active Flake Along the Optical Axis. 131
Fig. III.4 Projection of the Field Stop on the Active Flake. 132
Fig. III.5 Illustration of a Point Source. 135
Fig. III.6 Radiation Mound: The Point Spread Function (Adopted from Williams and Becklund [40]). 136
Fig. III.7 The Line Spread Function (Adopted from Williams and Becklund [40]). 139
Fig. III.8 Optical System Response as a Function of Spatial Frequency. . . . 140
Fig. III.9 Schematic of a simple optical system. 146
Fig. III.10 Illustration of Focus for Minimum Spot Size and Paraxial Focus. . . 148
Fig. III.11 Off-Axis Aberration: Astigmatism. 150
Fig. III.12 Off-Axis Aberration: Coma. 151
Fig. III.13 Illustration of Distortion. 154
Fig. III.14 Distribution of Radiation Power on the Active Flake Due to Diffuse Radiation Incident to the Instrument Aperture. 159
Fig. III.15 PSF at Zenith and Azimuth Angles of 0 deg. 162
Fig. III.16 PSF at a Zenith Angle of 1 deg and an Azimuth Angle of 0 deg. . . 163
Fig. III.17 PSF at a Zenith Angle of 2.0 deg and an Azimuth Angle of 0 deg. 164
Fig. III.18 PSF at a Zenith Angle of 2.5 deg and an Azimuth Angle of 0

deg. 165

Fig. III.19 PSF at a Zenith Angle of 3.0 deg and an Azimuth Angle of 0 deg. 166

Fig. III.20 Cumulative PSF at Source Angle Limit of 2.0 deg. 169

Fig. III.21 Cumulative PSF at Source Angle Limit of 3.0 deg. 170

Fig. III.22 Cumulative PSF at Source Angle Limit of 2.25 deg. 171

Fig. III.23 The OTF Corresponding to the PSF at Source Angle Limit of 2.25 deg. 172

Fig. III.24 Cumulative PSF at Source Angle Limit of 2.0 deg (CERES Design). 173

Fig. III.25 The OTF Corresponding to the PSF at Source Angle Limit of 2.0 deg (CERES Design). 174

List of Tables

Table I.1 Thermal Noise Due to Scattered Solar Radiation 41

Table I.2 Thermal Noise Due to Radiation Emission from the FOV Limiter, the Aperture Plate and the Active Cavity 42

Table II.1 ACR Transfer Function Results. 73

Table II.2 Single Circular Aperture Parameters 93

Table II.3 Array of Varying-Radius Circular Apertures. 107

Nomenclature

c	Specific heat (J/kg-K)
D	Distribution factor (-)
E_b	Bias voltage (V)
E_0	Bridge supply voltage (V)
E_1	Bridge output voltage (V)
E_2	Voltage across the heater wire (V)
F	Radiative source strength for a uniform diffuse source (W)
f_n	Corner frequency (Hz)
G	Volumetric heat generation (W/m^3)
H	Collimated beam flux (W/m^2)
k	Thermal conductivity (W/m-K)
N	Number of data samples (-)
Q	Heat input (W)
R	Electrical resistance (Ω)
R_r	Specular reflectivity ratio (-)
T, ΔT	Temperature (K) or sample period (s), temperature drop between the resistance temperature detector (RTD) and the heat sink (K)

t, Δt	Time, time increment (s)
W	Flux distribution (-)
x,y	Cartesian coordinates (m)

Greek

α	Absorptivity (-), linear temperature coefficient of resistance (1/K)
δ	Material thickness (m)
δ	Dirac delta function (-)
ε	Emissivity (-)
ζ	Damping ratio (-)
Θ	Time approximation weighting parameter (-)
θ	Angle of incidence of a collimated beam (deg)
ν	Test function (-)
ρ	Mass density (kg/m^3)
ρ^d	Diffuse component of reflectivity (-)
ρ^s	Specular component of reflectivity (-)
σ	Stefan-Boltzmann constant (5.667×10^{-8} W/m^2-K^4)
τ	Electrical or thermal time constant (s)
ψ	Interpolation function (-)
ω	Time, space or angular frequency (t^{-1}, m^{-1} or deg^{-1})

0.0 Preface

0.1 Background

It is well-known that the climate is controlled by the amount and distribution of solar radiation the Earth absorbs and reflects and the amount of radiation it emits. If the radiation balance is monitored for a sufficiently long period of time, it is possible that long-term weather patterns and perhaps even climate changes can be understood and predicted. These predictions can only be in a broad sense such as a possible drought or a heavy rainy season in a particular region of the Earth. This idea has been the driving force behind more than a century of research on the Earth radiation budget.

Earth radiation budget research started as early as 1837 when Pouillet [1] first measured the solar constant and obtained a value of 1211 W/m^2. Only moderate further progress had been made prior to 1960 [2]. The decade of the 1960's brought a promising era in Earth radiation budget studies as a consequence of international competition in space exploration. An array of Television Infrared Observational Satellites (TIROS) was launched in the late sixties. During the 1970's, another series of the National Oceanic and Atmospheric Administration (NOAA) weather satellites was put into orbit.

In the middle eighties, the National Aeronautics and Space Administration (NASA) launched three satellites carrying the highly successful Earth Radiation Budget Experiment (ERBE). A detailed survey of this experiment and those dating back to the beginning of the space age is given in Reference 3. The decade of the nineties will see the implementation of an extensive array of satellites under the Earth Observing System (EOS) project [4,5]. These satellites will include:

- NASA's Earth Observing System (EOS-A), which will have a 705 km, sun-synchronous polar orbit.

- NASA's Tropical Rainfall Measuring Mission (TRMM), which will have a 350 km, 35-deg inclined orbit.

- The European Space Agency (ESA) Polar Orbiting Earth-Observing Mission (POEM), which will have a 799 km, sun-synchronous polar orbit.

- The German DARA's Atmosphere and Oceanic Surface Exploration Satellite (ATMOS), which will have 775 km, sun-synchronous polar orbit.

On board each of these satellites is a new suite of radiation detectors called the Clouds and Earth's Radiant Energy System (CERES), among other instruments. CERES is based on the ERBE scanning instrument, with modifications to improve its performance.

0.1.1 The Earth Radiation Budget Experiment (ERBE)

The Earth Radiation Budget Experiment (ERBE) is an operational system of radiometric instruments in Earth orbit on three separate satellites: NASA's Earth Radiation Budget Satellite (ERBS) and two National Oceanic and Atmospheric Administration satellites, NOAA-9 and NOAA-10 [6]. The purpose of ERBE is to monitor the long-wave and visible components of the Earth's radiation budget with a view towards establishing a data base that will contribute to improved long-term weather and climate prediction.

Each of the ERBE satellites has four Earth-viewing channels which operate in the nonscanning, or staring, mode. A set of two of these channels has a wide field-of-view (WFOV), and the other set of two has a medium-field-of-view (MFOV). Within each set, one channel is sensitive only to visible radiation and the other is sensitive to all wavelengths. Furthermore, on board these satellites there are three narrow field-of-view (NFOV) scanning channels: a shortwave (0.2 - 5.0 μm), a longwave (5.0 - 50.0 μm) and a total (0.2 - ≥ 50.0 μm) channel. The satellites also have a solar monitor. Details of the design of the nonscanning and the scanning channels are given in References 7 and 8, respectively, and elsewhere in this dissertation.

0.1.2 The Clouds and Earth's Radiant Energy System (CERES)

The Clouds and Earth's Radiant Energy System (CERES) instrument, based on the ERBE scanning radiometer, is currently under design review [9]. At the core of the instrument lies a thermistor bolometer and the same Cassegrain-type telescope used in

the ERBE scanner instruments [8]. CERES will have three channels:

1) A total channel (0.3 to 50 μm),
2) A longwave channel (8 to 12 μm), and
3) A shortwave channel (0.3 to 5 μm).

Such channels are considered to be obsolete when their radiometric accuracy falls below 0.5 and 1.0 percent, respectively, for the total and shortwave channels. The spatial resolution of these instruments is 35 km, except for EOS-A whose resolution is 21 km.

Solar "blip" is the name given to the phenomenon which occurs when the sun comes into the field-of-view of the instrument while it is scanning the Earth. This phenomenon causes saturation of the instrument, and hence leads to unrecoverable data. Based on experience with ERBE solar blip, efforts are being made to conceive a solar avoidance system in CERES.

The telescope field stop size has been changed from 3 x 4.5 deg (ERBE) to 1.12 x 2.8 deg for CERES. This required that the bolometer response time be halved from 12 to 6 *ms* because of the decreased amount of energy that enters the system.

0.2 Dissertation Organization

This dissertation documents the continuation of the author's Master of Science research. The new developments may be classified into three parts.

Prelude

PART I

In Part I, some improvements are brought to the dynamic electrothermal model of the ERBE nonscanning instruments, first developed by the author during his Master of Science research. The effects of thermal noise on the performance of the instrument are also investigated in this part. This required the development of a new radiation heat transfer numerical model.

Much of the research in this part has already been presented at an SPIE symposium held in March, 1989 [10], and published in the **Journal of Optical Engineering** [11].

PART II

Based on the excellent agreement between the measured and simulated dynamic electrothermal response of the wide field-of-view (WFOV) channel demonstrated in Part I, we feel that the channel dynamics are well represented by the model. Therefore, the dynamic electrothermal model of Part I is used in Part II to study proposed in-flight dynamic calibration scenarios. Different field-of-view (FOV) limiting geometries are investigated to determine their usefulness in performing these in-flight dynamic calibrations. It is shown in Part II that the best FOV limiter geometry is the one that spreads the input signal energy as uniformly as possible within its Fourier spectrum.

PART III

The research of Part III takes advantage of the numerical optical model conceived by J. L. Meekins for his Master of Science thesis [12]. This optical model has been extensively modified as follows:

1) Modules have been added to allow the study of the scanner point spread function.
2) Array dimensions and DO-loop modifications lead to increased accuracy and to improved numerical efficiency.
3) The model has been modified to allow it to run on a personal computer.

The modified optical model is used to compute the point spread function of the Earth Radiation Budget Experiment (ERBE) scanner instruments for different angles of incidence. The point spread function is also computed for the new but similar instrument to be used in the Clouds and Earth's Radiant Energy System (CERES).

PART I

IMPROVEMENT OF THE ERBE ACR DYNAMIC ELECTROTHERMAL MODEL AND STUDY OF CHANNEL THERMAL NOISE

1.0 Introduction

Operation of the Earth Radiation Budget Experiment (ERBE) nonscanning channels is based on the active cavity radiometer (ACR) concept. An ACR consists of an electrically heated cavity that is thermally connected to a constant-temperature heat sink as shown in Fig. I.1. The cavity is actively controlled to a constant temperature difference above that of the heat sink by the feedback circuit shown schematically in Fig. I.2. During operation, the electrical power input to the cavity automatically increases or decreases as required to maintain this set-point temperature difference when the radiative power input changes.

Validation of the satellite data was performed by calibrating the ACR instruments two to three times weekly during the first two months of operation, and every 14 days thereafter [13]. In one calibration mode the radiometer views the sun through a secondary aperture called the solar port. The instrument data are then compared to the well-known solar flux, whose value is confirmed by the on-board solar monitor.

A dynamic electrothermal model of the ACR instrument has been developed and is summarized in detail in Reference 14. This numerical model, which is briefly described in Chapter 2 of Part I, has been used to predict the response of the total, WFOV Earth-viewing channels during solar calibration as reported in earlier papers [15,16]. As a concession to simplicity, the precision aperture labeled in Fig. I.3, in this former model was

assigned the same diameter as the cavity opening, and the solar constant was scaled down by the ratio of the precision aperture area to the cavity opening area so that the correct total amount of solar power entered the cavity. This simplification provoked a slight variation between the simulated and measured radiometer responses corresponding to the initial and final stages of the solar calibration event as described in References 15 and 16 and demonstrated in Sections 4.2 and 4.3 of Part I. Then one of the main objectives of this part of the dissertation is to modify the aperture geometry so that the model can correctly represent the optical path followed by the radiation incident to the instrument. A further objective of the current work is to evaluate contamination of the signal due to thermal noise sources. These include the effect of scattering of the incident solar beam, the effect of emission from the field-of-view (FOV) limiter and the primary aperture plate, and the effect of cavity self-contamination.

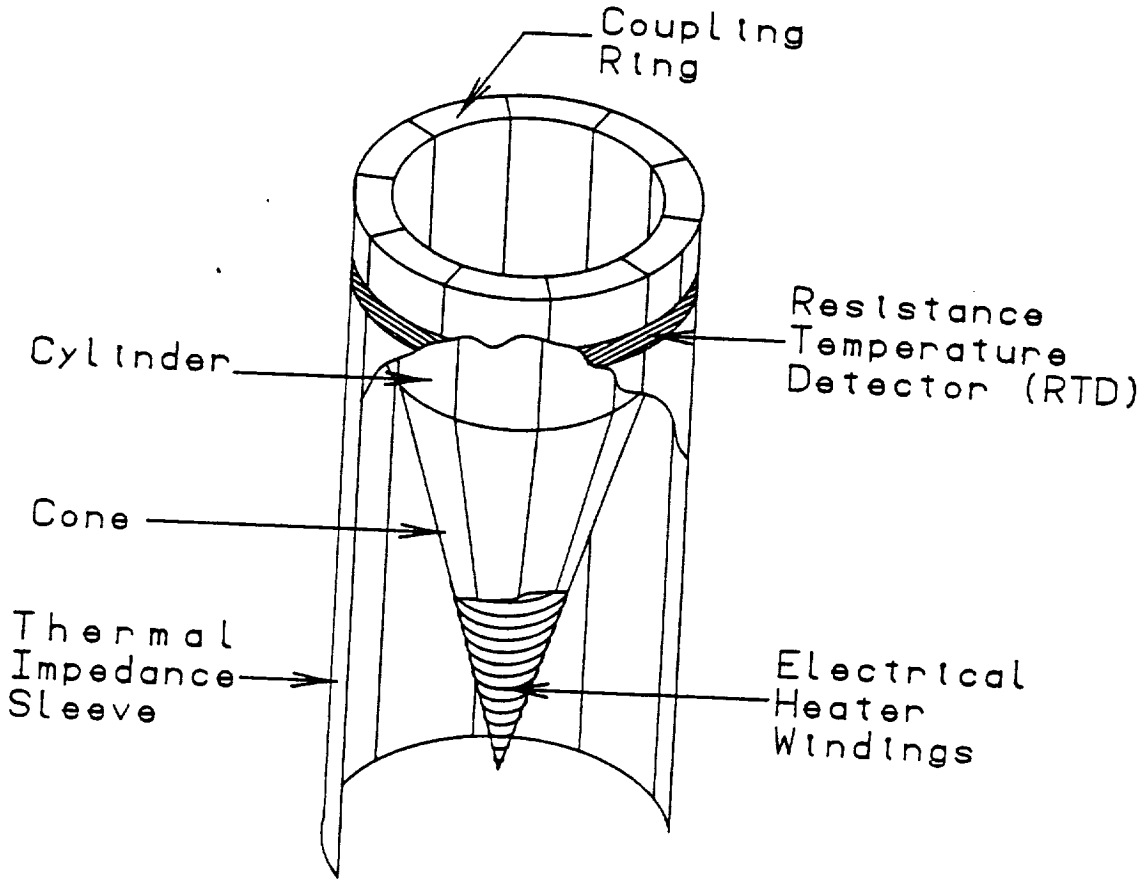
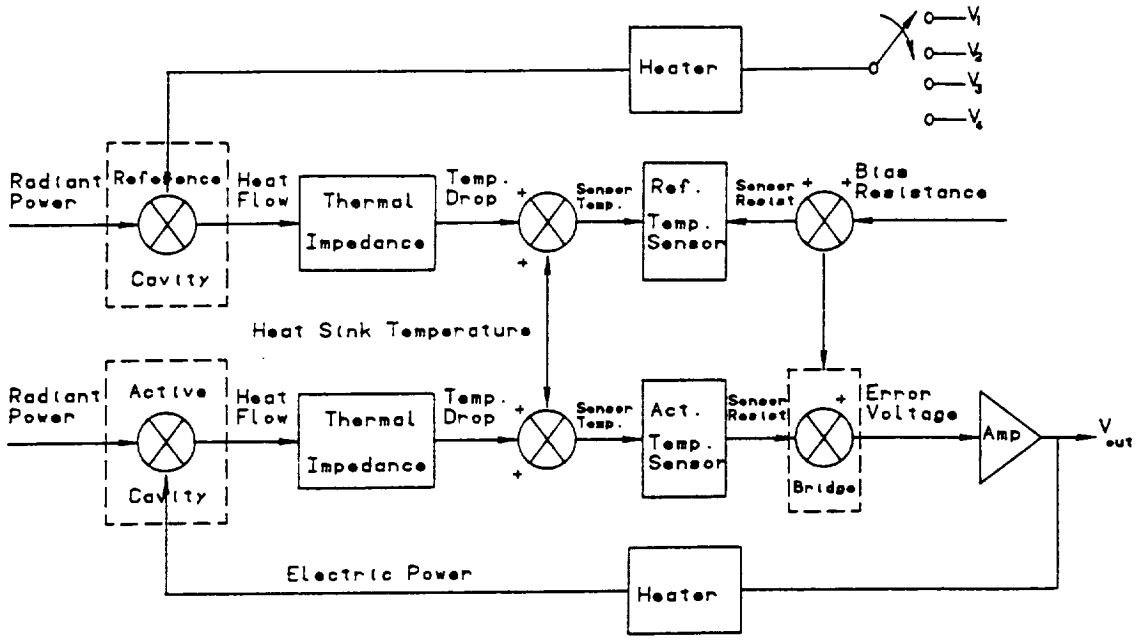
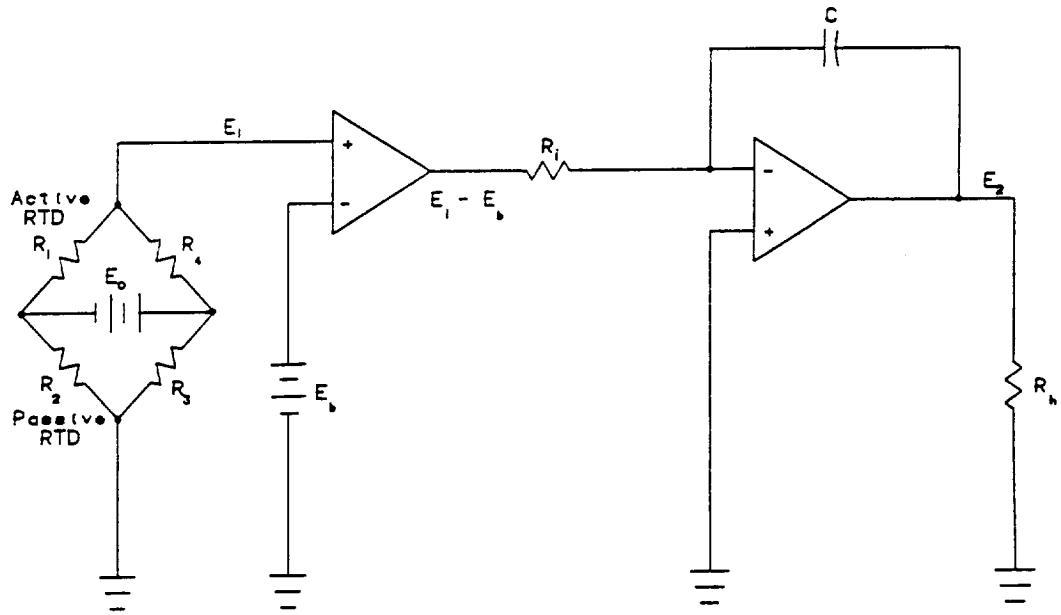


Fig. I.1 ERBE Nonscanning ACR Instrument Cavity.



(a)



(b)

Fig. I.2 The ERBE ACR Electrical Feedback Temperature Control: (a) Block Diagram, (b) Electrical Schematic.

ORIGINAL PAGE IS
OF POOR QUALITY

2.0 The Dynamic Electrothermal Model

2.1 Radiative Interchange

The first step in simulating the total, WFOV ERBE nonscanning channels is to model the radiative exchange inside the active cavity. This is accomplished by first calculating an array of *distribution factors*, which represent the manner in which energy is distributed among the various surfaces of the cavity [15-19]. These distribution factors are analogous to the more familiar configuration factors, or angle factors, which account only for diffuse radiation. The distribution factors, on the other hand, account for both diffuse and specular components of reflection. Mahan and Eskin [17,18] introduced and exploited the concept of distribution factors in an early effort to model radiative exchange within an ACR cavity. They used the Monte-Carlo method to compute these factors for the cylinder-cone cavity shown in Fig. I.1. The Monte-Carlo method is a statistically-based approach in which individual energy bundles, or "rays", are traced through multiple reflections, some of which are diffuse and the rest specular, until their energy is absorbed or escapes the enclosure. The use of the Monte-Carlo method to compute distribution factors among diffuse-specular surfaces is described briefly in Section 3.1; however, the interested reader is referred to References 15 through 19, and especially to Reference 20, for the details of this procedure and its application to cylinder-cone cavities. For a more

general treatment of the Monte-Carlo method, the reader is referred to Siegel and Howell [21].

Two types of distribution factor are used: those which describe the distribution of radiation emitted from the cavity walls, and those which describe the distribution on the cavity walls of solar radiation incident to the instrument. The second situation requires one set of distribution factors during solar calibration, and another set during the pitchover maneuver. The difference between these two kinds of distribution factor is dictated by the difference in the way solar radiation enters into the cavity. During solar calibration the access of solar radiation to the instrument is limited by the solar port, as shown in Fig. I.3(a), and during the pitchover maneuver the solar radiation input is controlled by the field-of-view limiter, as shown in Fig. I.3(b). The pitchover maneuver is a 180-deg rotation of the satellite primarily intended to point the radiometric channels away from the Earth and toward cold space and then toward the sun. This maneuver, which occurs when the sun is in the satellite orbital plane, permits measurement of the zero-input offset of the channels. A secondary benefit is that the nonscanning channels are able to observe the sun once per orbit while they are in their Earth-viewing configuration.

As already mentioned, the precision aperture in an earlier model [14,15,16] was assumed to have the same diameter as the cavity opening, and the solar constant was adjusted accordingly. In reality, however, the area of the cavity opening is 40 percent greater than that of the precision aperture. However, the precision aperture not only limits the amount of energy entering the cavity, but it also determines its spatial distribution on the cavity surface as discussed and illustrated in Section 4.1. Therefore, its geometry must be precisely accounted for in the model.

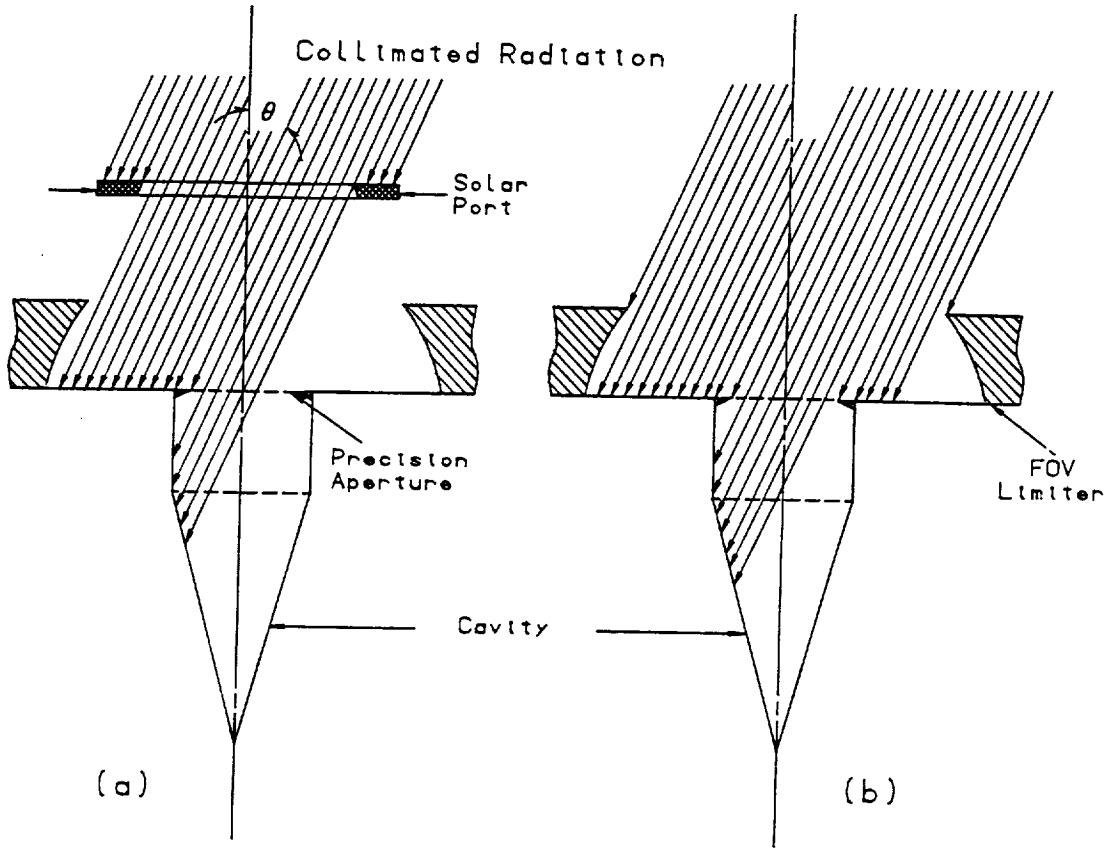


Fig. I.3 Collimated Radiation Entering the Cavity During (a) Solar Calibration and (b) the Pitchover Maneuver.

Electrothermal Model

2.2 Heat Conduction

The second part of the simulation involves solution of the equation expressing conservation of energy within the cavity walls and the thermal impedance connecting the cavity to the heat sink,

$$\frac{\partial}{\partial x}(k\frac{\partial T}{\partial x}) + \frac{\partial}{\partial y}(k\frac{T}{\partial y}) + G = \rho c\frac{\partial T}{\partial t} \ . \qquad \text{(I.1)}$$

In Eq. (I.1) k is the thermal conductivity, ρ is the mass density, c is the specific heat and G is a volumetric heat source term. This latter includes both radiative and electrical local heating of the cavity walls.

Reddy [22] and Zienkiewcz [23] both present developments of the finite element form for both one- and two-dimensional heat conduction problems. The numerical solution of Eq. (I.1) is based on these developments. The finite element method is a good alternative to other numerical methods, such as the finite difference method, especially when the geometry is complex as in the case at hand. Since the cavity material is very thin and is made of silver, which has a very high thermal conductivity, the conduction problem is two-dimensional; that is, the temperature gradient through the wall thickness is assumed negligible.

The cavity wall is divided into a number of quadrilateral segments, or elements.

Since the temperature on the element domain is dependent on time and space, it can be approximated according to the Ritz variational method [22,23] by

$$T(x,y) = \sum_{j=1}^{n} T_j(t)\psi_j(x,y) \ , \tag{I.2}$$

where n is the number of nodes in the element, $T_j(t)$ is the factor which accounts for the time dependence of the temperature at node j, and ψ_j is the contribution of node j to the spatial variation of temperature at point (x,y). When Eq. (I.2) is substituted into Eq. (I.1) and appropriate mathematical operations are invoked, there results a linear system of algebraic equations for the temperatures of all the element nodes of the form

$$[A^e]\{\dot{T}\} + [B^e]\{T\} = \{F^e\} \ , \tag{I.3}$$

where A^e and B^e are the thermal capacity and thermal conduction matrices of a typical element e and F^e is the heat input vector. Equation (I.3) is written for all elements and then assembled to form a global system of algebraic equations representing all nodes in the domain. This system contains 150 elements and 620 nodes. Because the formulation and solution of this problem is the main thrust of author's 1987 Master of Science thesis [14], the details are not repeated here.

Electrothermal Model

3.0 Determination of the Illuminated Area in the Cavity

Given the geometry of the FOV limiter for the nonscanning radiometer instrument, shown in Fig. I.4(a), incident collimated radiation passes first through the secondary aperture and then through the primary aperture and on into the cavity. If θ is the angle the z-axis makes with the line passing through the centers of the two apertures, then both apertures project as ellipses on the x-y plane, as shown in Figs. I.4(b,c). The equations of these ellipses in the x-y plane are given by

$$\frac{x^2}{R_1^2} + \frac{(y - H\sin\theta)^2}{(R_1\cos\theta)^2} = 1 \tag{I.4}$$

and

$$\frac{x^2}{R_2^2} + \frac{y^2}{(R_2\cos\theta)^2} = 1, \tag{I.5}$$

where R_1 and R_2 are the radii of the upper and lower apertures, respectively, and H is the vertical distance between them. To find the intersections of the two ellipses we combine Eqs. (I.4) and (I.5) to eliminate x^2, obtaining

$$R_1^2 - R_2^2 - \frac{(y - H\sin\theta)^2}{\cos^2\theta} + \frac{y^2}{\cos^2\theta} = 0. \tag{I.6}$$

Solving for y, we obtain

$$y = (R_2^2 - R_1^2)\frac{\cos^2\theta}{2\,H\tan\theta} + \frac{H}{2}\sin\theta. \tag{I.7}$$

For an incident angle between 0 and θ_c (see Fig. I.4), the illuminated area is simply the area of the lower ellipse. Beyond an incident angle of θ_c, the ellipses intersect at two symmetrically opposite points with respect to the x-axis. So for each value of y in Eq. (I.7) there results two values of x; i. e., from Eq. (I.5)

$$x = \pm\sqrt{R_2^2 - (\frac{y^2}{\cos^2\theta})}. \tag{I.8}$$

The next step is to determine the illuminated area which is shown cross-hatched in Figs. I.4(b,c). A finite strip of differential width dx is taken across the area and then integrated from the left intersection to the right one. To carry out the integration, the equations of the ellipses are needed. These are

$$y_1(x) = H \sin\theta \pm \cos\theta\sqrt{R_1^2 - x^2} \tag{I.9}$$

and

$$y_2(x) = \pm \cos\theta\sqrt{R_2^2 - x^2}\,. \tag{I.10}$$

The illuminated area is the integral of the difference between the upper and lower curves, which are the elliptical arcs that meet at the two points of intersection. This area is given by

$$A_{III} = \int_{-x_0}^{x_0} [y_2(x) - y_1(x)]dx\,. \tag{I.11}$$

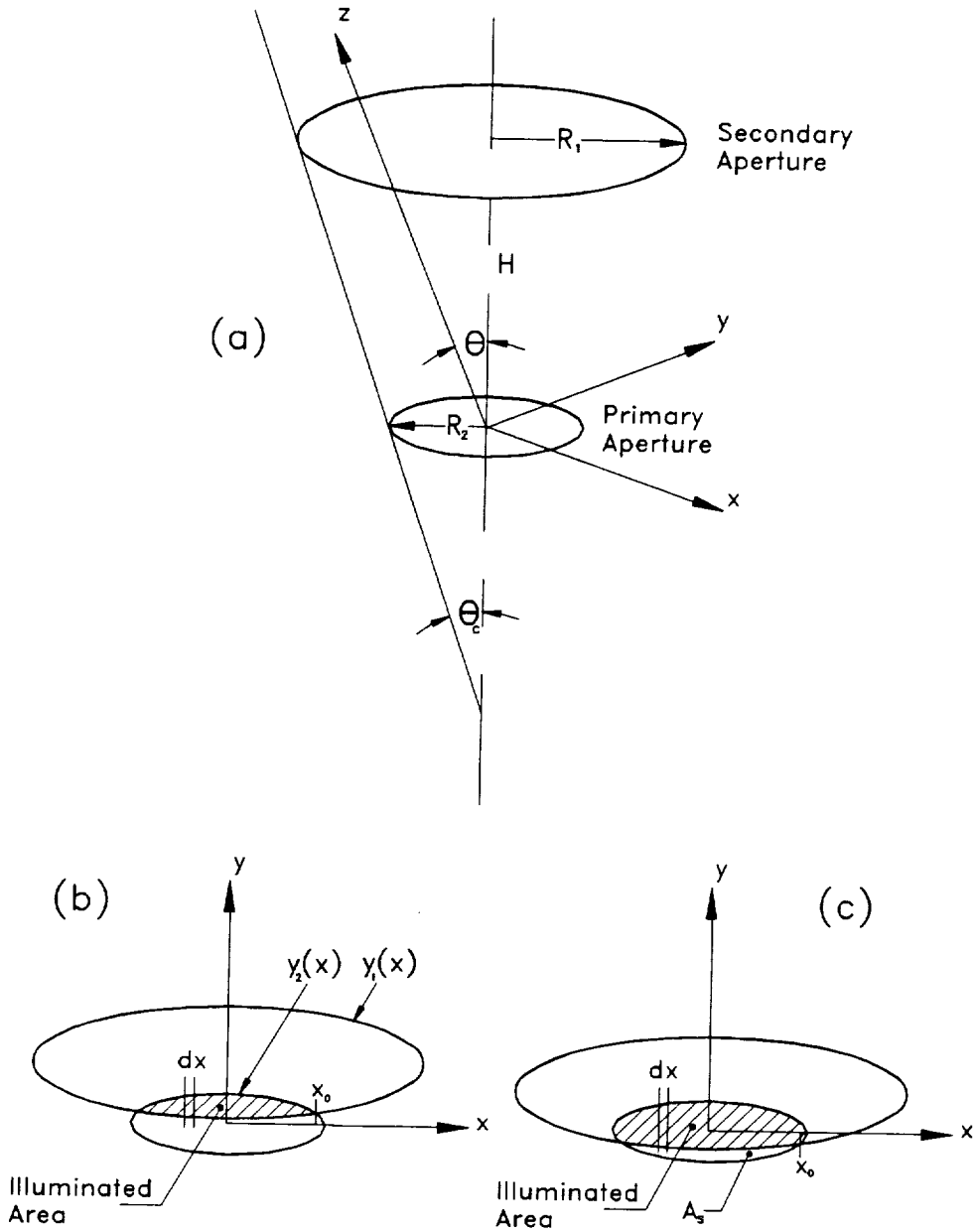


Fig. I.4 Illuminated Area in the Cavity

3.1 First Case: Intersection Above the x-Axis

In this case, shown in Fig. I.4(b), the illuminated area is bounded by the lower arc of the first ellipse and the upper arc of the bottom ellipses. Hence, substituting Eqs. (I.9) and (I.10), with the appropriate signs, into Eq. (I.11) and taking symmetry into consideration, the expression for the illuminated area becomes

$$A_{III} = 2\int_0^{x_0}[\cos\theta\sqrt{R_2^2-x^2} - H\sin\theta + \cos\theta\sqrt{R_1^2-x^2}\,]dx. \tag{I.12}$$

Integration of Eq. (I.12) leads to the final expression for the illuminated area

$$\begin{aligned} A_{III} &= -2x_0 H\sin\theta + \\ &\cos\theta[x_0(\sqrt{R_2^2-x_0^2} + \sqrt{R_1^2-x_0^2}) + R_2^2\sin^{-1}\frac{x_0}{R_2} + R_1^2\sin^{-1}\frac{x_0}{R_1}]. \end{aligned} \tag{I.13}$$

3.2 Second Case: Intersection Below the x-Axis

In this case, shown in Fig. I.4(c), the surface area A_s is found and then subtracted from the total area of the ellipse. The area A_s is given by

$$A_s = 2\int_0^{x_0}[H\sin\theta - \cos\theta\sqrt{R_1^2-x^2} + \cos\theta\sqrt{R_2^2-x^2}\,]dx. \tag{I.14}$$

Integration of Eq. (I.14) leads to

$$A_s = 2x_0 H\sin\theta +$$
$$\cos\theta[x_0(\sqrt{R_2^2-x_0^2} + \sqrt{R_1^2-x_0^2}) + R_2^2 \sin^{-1}\frac{x_0}{R_2} + R_1^2 \sin^{-1}\frac{x_0}{R_1}]. \tag{I.15}$$

So the final expression of the illuminated area in this case is

$$A_{III} = \pi R_2^2\cos\theta - A_s. \tag{I.16}$$

4.0 Model Simulation Results

4.1 The Distribution Factors

The simulation of solar calibration is based on a transient event in which a collimated beam enters the cavity at angles ranging from 8.8 to -8.8 deg with respect to the instrument optical axis. This range of angles is determined by the diameters of the solar port and the precision aperture and the distance between them. The distribution factors are calculated corresponding to a series of angles within one-half of the angle span. Figure I.5 shows these distribution factors at angles of 3.3, 6.6 and 7.7 deg. The vertical axis represents the relative amount of radiation absorbed at a given location on the cavity. In Fig. I.5 the cavity has been split vertically, unrolled and mapped onto a rectangle. The circular opening of the cavity is to the left and the cone vertex is to the right. Note how little radiation is absorbed on the cylindrical wall of the cavity. This is because the precision aperture, which is smaller than the cavity opening, creates a shadow on the cylinder, thereby concentrating the absorbed radiation on the cone. During the pitchover maneuver, the transient solar irradiation event is very similar to that for solar calibration except that it extends over a wider range of angles and takes longer. In this

case, the angle span is from 71.4 to -71.4 deg. This difference in span occurs because the solar port has been removed and it is now the channel FOV limiter which determines the angle span. Figure I.6 shows a selection (corresponding to incidence angles of 19.2, 48.0 and 60.0 deg) of the distribution factor maps for the pitchover maneuver. In this case, unlike in the case of solar calibration, significant radiation absorption takes place on the cylindrical surface as well as on the conical surface of the cavity. This is because the angle span is sufficiently large to allow collimated radiation to reach the upper part of the cavity.

4.2 Solar Calibration

The results for two solar calibration events are shown in Figs. I.7 and I.8. Figure I.7(a) shows a comparison between the simulated and the measured ERBS total, WFOV active cavity heater power corresponding to the solar calibration event of December 28, 1984; and Fig. I.8(a) shows the same comparison for the solar calibration event of October 31, 1985. The durations of the events are approximately 310 and 360 *s*, respectively. Comparison of the results of the new simulations with those given in References 15 and 16 for the same solar calibration events (Figs. I.7(b) and I.8(b)), reveals an improved agreement between the simulated and the observed results, especially at the lower corners of the cup-shaped curves. The reason for this improvement is the improvement in the way the precision aperture has been treated, as described at the end of Section 2.1. Now absorption of radiation occurs deeper in the cavity due to the shadow created by the precision aperture.

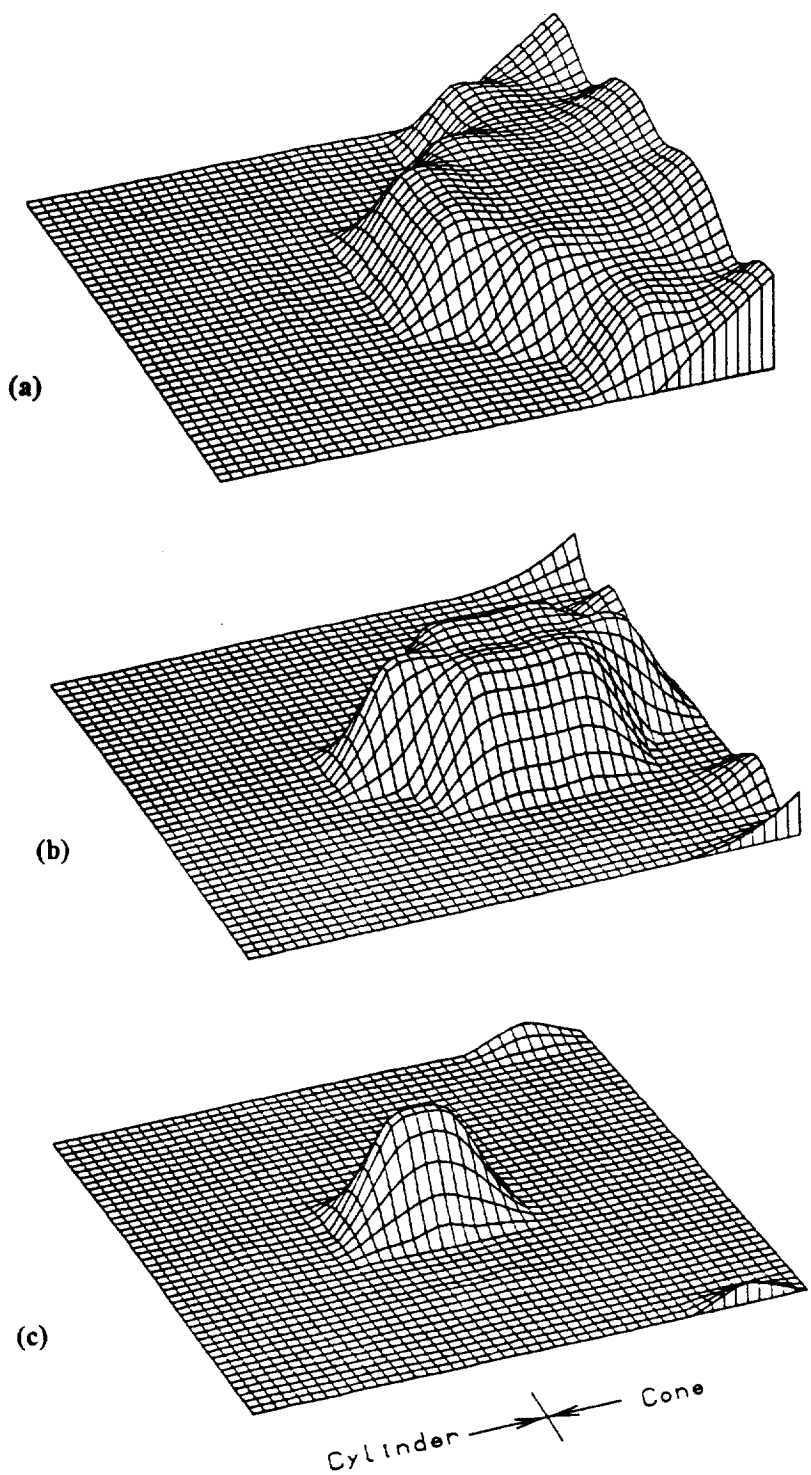


Fig. I.5 Distribution Factors for Solar Calibration at Incident Angles (a) 3.3, (b) 6.6 and (c) 7.7 deg.

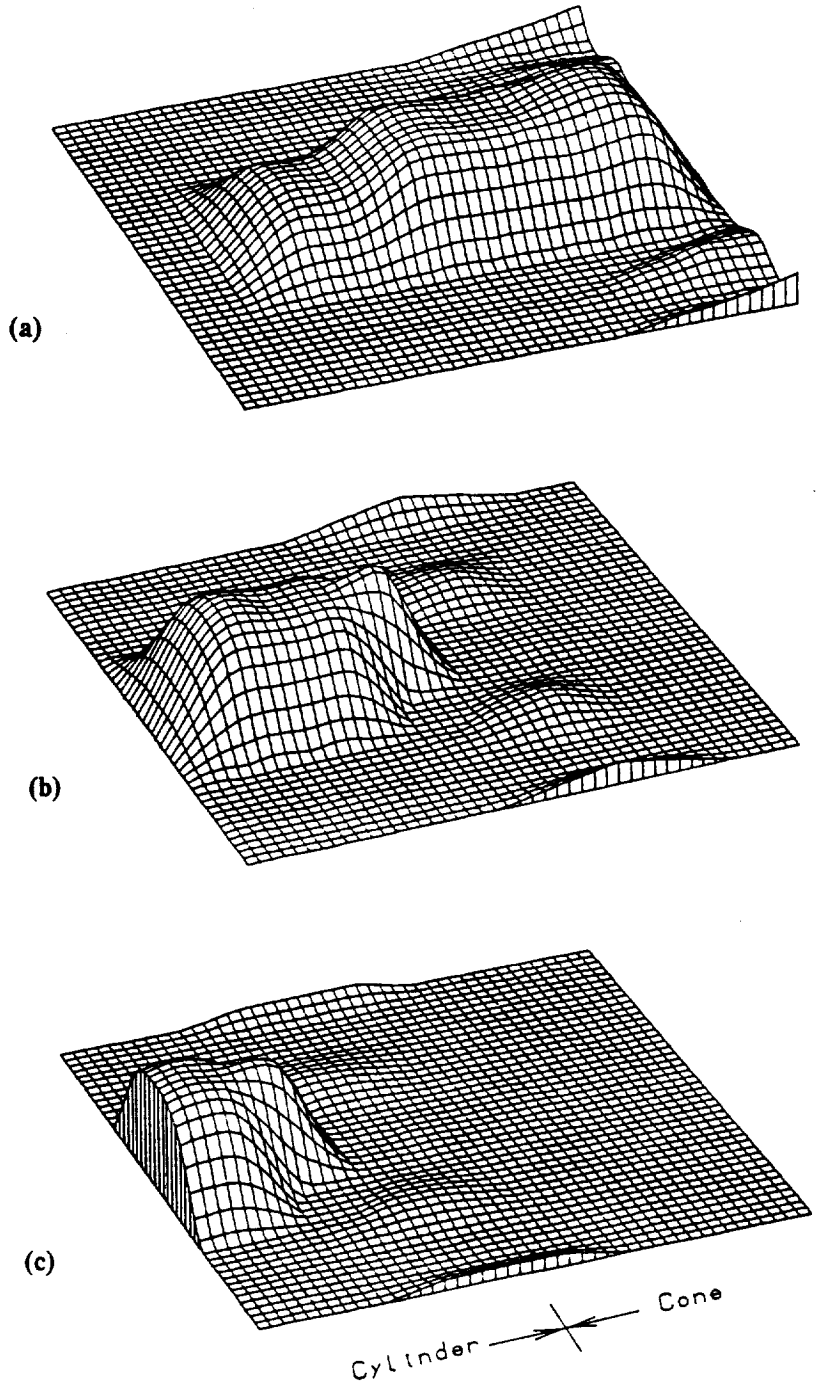
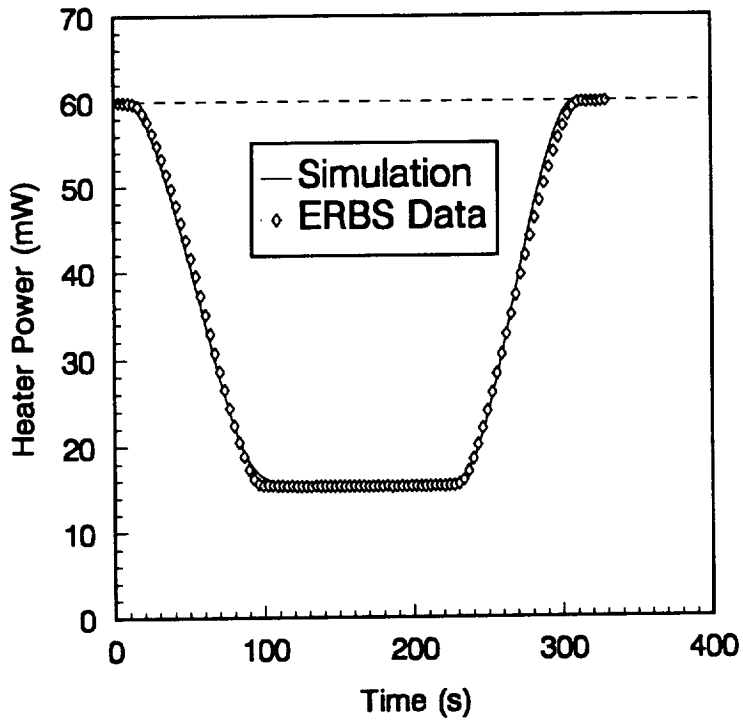
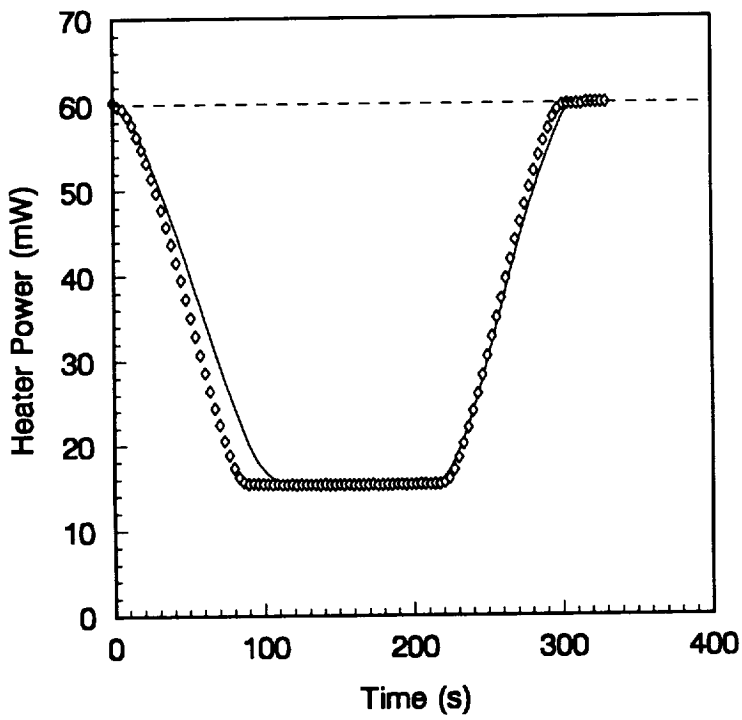


Fig. I.6 Distribution Factors for the Pitchover Maneuver at Incident Angles (a) 19.2, (b) 48.0 and (c) 60.0 deg.

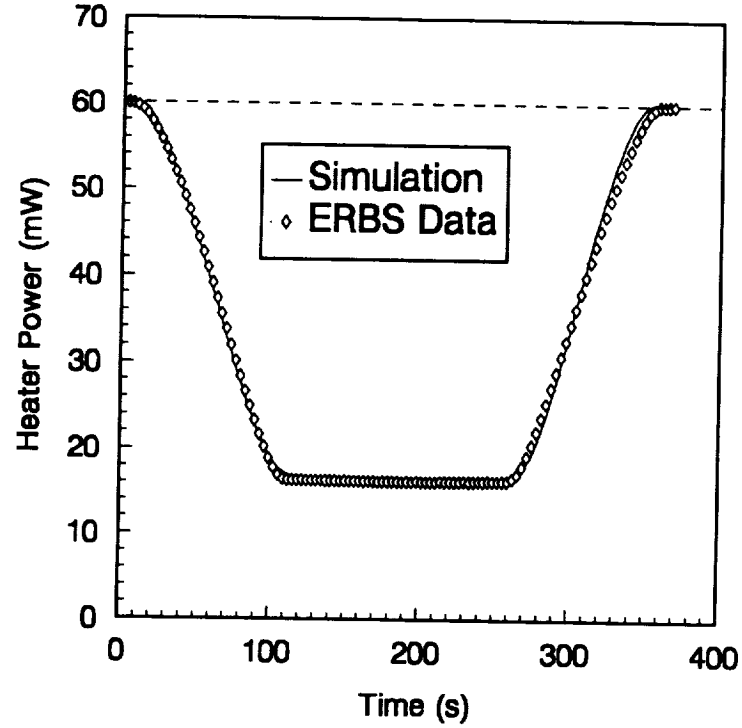


(a)

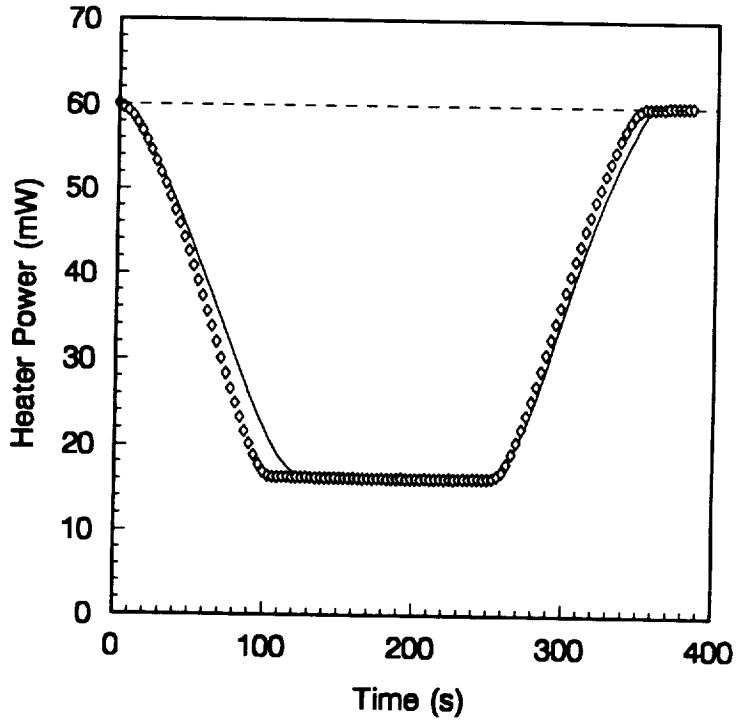


(b)

Fig. I.7 Comparison of the Simulated and Observed Solar Calibration Event for the ERBS Total, WFOV Channel (12/28/1984) (a) Present Model and (b) Previous Model [16].



(a)



(b)

Fig. I.8 Comparison of the Simulated and Observed Solar Calibration Event for the ERBS Total, WFOV Channel (10/31/1985) (a) Present Model and (b) Previous Model [16].

4.3 The Pitchover Maneuver

The pitchover maneuver event, described in Section 2.1, lasts approximately 2340 *s* from the moment solar radiation first enters the cavity until the cavity is once again completely shadowed by the FOV limiter. Figure I.9(a) and (b) shows the simulation of the pitchover maneuver events of November 21, 1984, and of October 19, 1985, respectively, and compares them with the ERBS data. The curve is shaped like a cosine function over much of its range, as expected. At both extremities (68 and -68 deg), however, the cosinusoidal shape is truncated because the precision aperture is eclipsed by the FOV limiter.

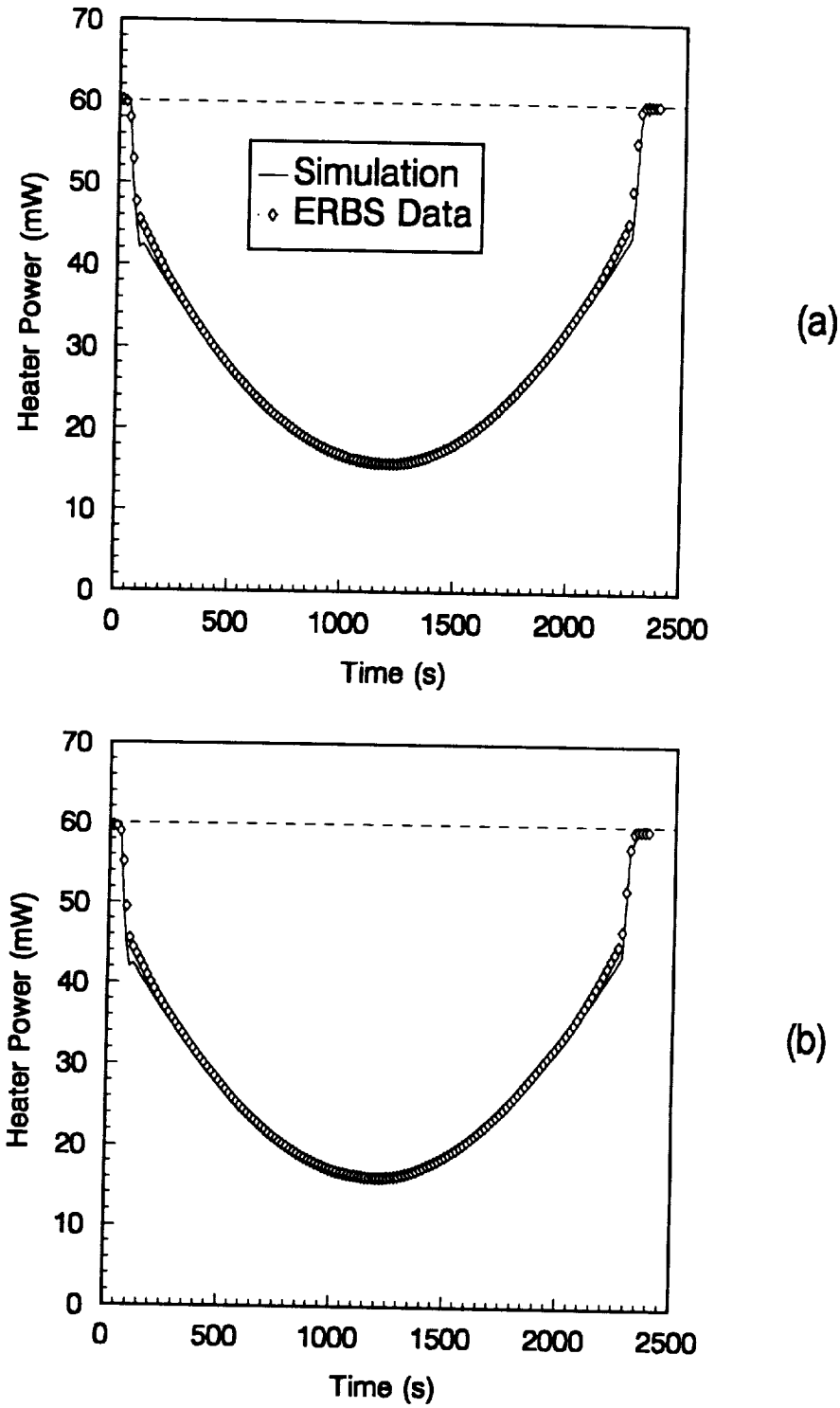


Fig. I.9 Comparison of the Simulated and Observed Pitchover Maneuver for the ERBS Total, WFOV Channel on (a) November 21, 1984 (b) October 19, 1985.

5.0 Evaluation of Thermal Noise

5.1 Cumulative Thermal Noise

In Eq. (I.1), the term G includes only electrical and *direct* radiative energy. By "direct" it is meant that the radiation part of G does not include the various noise sources; that is, it lacks spurious radiation contributions from the cavity surroundings. Figure I.10 shows the active cavity radiometer (ACR) with its surrounding parts: the field-of-view limiter and the aperture plate. While the radiometer has been designed to minimize contamination of the signal by radiation from these components, they do nevertheless emit and reflect some radiation into the cavity. Then the total thermal noise entering the cavity is given by

$$P_{noise} = P_{scat} + P_{FOV} + P_{ap} + P_{sc}, \tag{17}$$

where:

P_{scat} is the solar radiation scattered by the FOV limiter and the aperture plate which is eventually absorbed by the cavity,

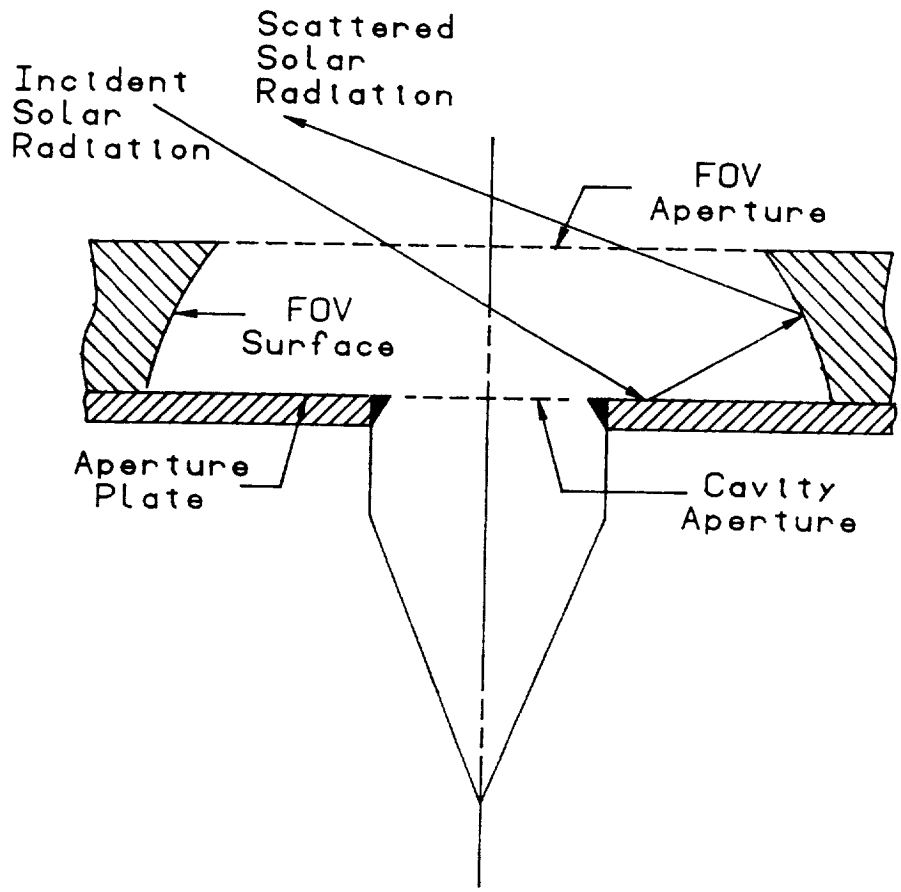


Fig. I.10 The Radiometer Enclosure Geometry.

P_{FOV} is the radiation emitted by the FOV limiter which is eventually absorbed by the cavity,

P_{ap} is the radiation emitted by the aperture plate which is eventually absorbed by the cavity, and

P_{sc} is cavity self-contamination; that is, the portion of the energy emitted by the cavity that returns, because of reflection from the FOV limiter and the aperture plate, to be absorbed by the cavity.

The various thermal noise sources in Eq. (I.17) are evaluated using a radiative exchange model independent of the dynamic electrothermal model already described in Chapter 2 of Part I. The possible influence of thermal noise associated with the impact of gas molecules on the instrument surfaces is not considered in the present study. This is justified because at the satellite altitude (approximately 800 *km*), there are virtually no gas molecules.

5.2 Scattering Thermal Noise

The scattering noise is calculated using the Monte-Carlo method as follows. A large number of parallel rays is introduced into the enclosure shown in Fig. I.10 either through the solar port or the FOV limiter (see also Fig. I.3), depending on whether the solar calibration or the pitchover maneuver is being considered. In either case the incident radiation passes through the FOV limiter opening, so the difference between the

two cases could be visualized by thinking of either the solar port or the FOV opening as the source of that radiation. These energy bundles are then traced within the enclosure system. All radiation incident to the precision aperture and the secondary (FOV limiter) aperture from within this enclosure is assumed to leave the enclosure. Both the FOV limiter and the aperture plate are assumed to have an absorptivity of 0.1 and a specular-to-total reflectivity ratio of 0.9 (i.e., ninety percent mirror-like) for the total, WFOV instrument. The accuracy with which the surface properties of the instrument can be known is limited. Therefore, these values for the absorptivity and the reflectivity ratio have been varied by ± 0.01, and the impact of these variations assessed.

The number of the solar energy bundles incident to the secondary aperture that subsequently enters the cavity through the precision aperture represents the total energy absorbed by the cavity P_{total} due to the solar beam. This quantity includes the energy that strikes the cavity directly, P_{direct}, and the energy that gets scattered by the FOV limiter and the aperture plate but eventually enters to the cavity, P_{scat}. These quantities are related by

$$P_{scat} = P_{total} - P_{direct}. \tag{18}$$

The direct solar radiation is given simply by

$$P_{direct} = H\,A^{\theta}, \tag{19}$$

where H is the solar constant corrected for the Earth-sun distance (W/m^2) and A^{θ} is the projected cavity area (m^2) seen by the collimated beam incident at the angle θ.

6.0 Thermal Noise Results

The computer code developed to study the thermal noise input to the ERBE total, WFOV channel is listed in Appendix A. It is based on an enclosure consisting of four different surfaces, two real and two imaginary: the aperture plate, the field-of-view limiter surface, the primary aperture (cavity opening), and the secondary aperture (FOV opening), as shown in Fig. I.10. The code has the capability of treating collimated radiation entering through either the solar port or the FOV limiter. It can also treat any of the four surfaces, mentioned above, as diffuse sources.

In the determination of the radiative exchange between these surfaces, the Monte-Carlo method is used to trace the energy bundles within the enclosure. A sufficiently large number of these bundles will exactly simulate the actual behavior of radiative exchange within the enclosure. Figure I.11 shows the variation of relative energy that is absorbed by the cavity as a function of the number of energy bundles used in the ray trace. By relative energy it is meant the fraction of the solar energy incident to the secondary aperture which is absorbed by the cavity. Note that the curves become smoother as this number is increased. The curves of Fig. I.11(b) are less smooth than those in Fig. I.11(a) because the FOV limiter aperture is much larger than the solar port (37.6 and 2.38 cm^2,

respectively). Therefore, if say 300,000 energy bundles adequately represent the solar beam entering through the solar port, the beam will be under-described if the same number of energy bundles passes through the much larger FOV limiter. To obtain the same smoothness in both curves, the number of entering energy bundles must be proportional to the respective surface areas.

First, consider thermal noise due to scattering of the solar radiation by the FOV limiter and the aperture plate during solar calibration and the pitchover maneuver. Table I.1 shows that, in the case of solar calibration, the energy that finds its way into the cavity after being reflected by either the FOV limiter or the aperture plate ranges between -0.324 and 0.281 *mW*. Of course, the negative values of the noise are physically meaningless. In terms of the numerical model, however, they indicate that the thermal noise is of the same order of magnitude as the uncertainty in the analysis itself. But this latter is random and so can be negative. The average of all the values in Table I.1 is approximately 0.071 *mW*, which amounts to a thermal noise of 0.156 percent of the maximum power input to the cavity. In the case of the pitchover maneuver, this noise ranges between -0.832 and 1.368 *mW*. The average of this thermal noise is 0.207 *mW* (0.460 percent), a little higher than in the case of solar calibration.

Figure I.12 shows both the total and direct relative energy absorbed by the cavity. It is important to note that the curves representing the total energy are slightly higher (overall) than those representing the direct energy. The small difference between the direct and the total power input is indicative of the small magnitude of the scattered energy.

Next, since the field-of-view limiter and the aperture plate have varying

temperatures, it is desired to evaluate their noise contributions to the active cavity. The results of this study are given in Table I.2. The diffuse energy that enters the cavity due to emission from the FOV limiter (P_{FOV}) ranges from 0.094 to 0.120 *mW* as the FOV limiter temperature ranges from 290 to 308 *K*. This corresponds to a noise contribution ranging from 0.212 to 0.270 percent. However, it is important to note that during most ERBE operations, the FOV limiter temperature does not exceed 295 *K*. Hence, the actual noise contribution is at the lower end of this range. The primary aperture plate, on the other hand, contributes far less noise, less than 0.011 *mW* (0.024 percent), even though it operates at a temperature of around 306 *K*. The fact that the aperture plate contributes less than the FOV limiter is expected since the former faces away from the cavity opening while the latter generally faces the cavity opening.

Table I.2 also gives thermal noise due to self-contamination. The amount of energy that leaves the cavity and returns (P_{sc}) is about 0.034 *mW*. This amounts to a thermal noise of approximately 0.071 percent of the maximum power input to the cavity (≈ 45 *mW*). This value corresponds to a mean cavity temperature of 308 *K*, the temperature at which the active cavity is maintained.

When the absorptivity and the reflectivity ratio are varied by ± 0.01 about their nominal values the corresponding variations in the values given in Tables I.1 and I.2 are within the uncertainty of the analysis itself.

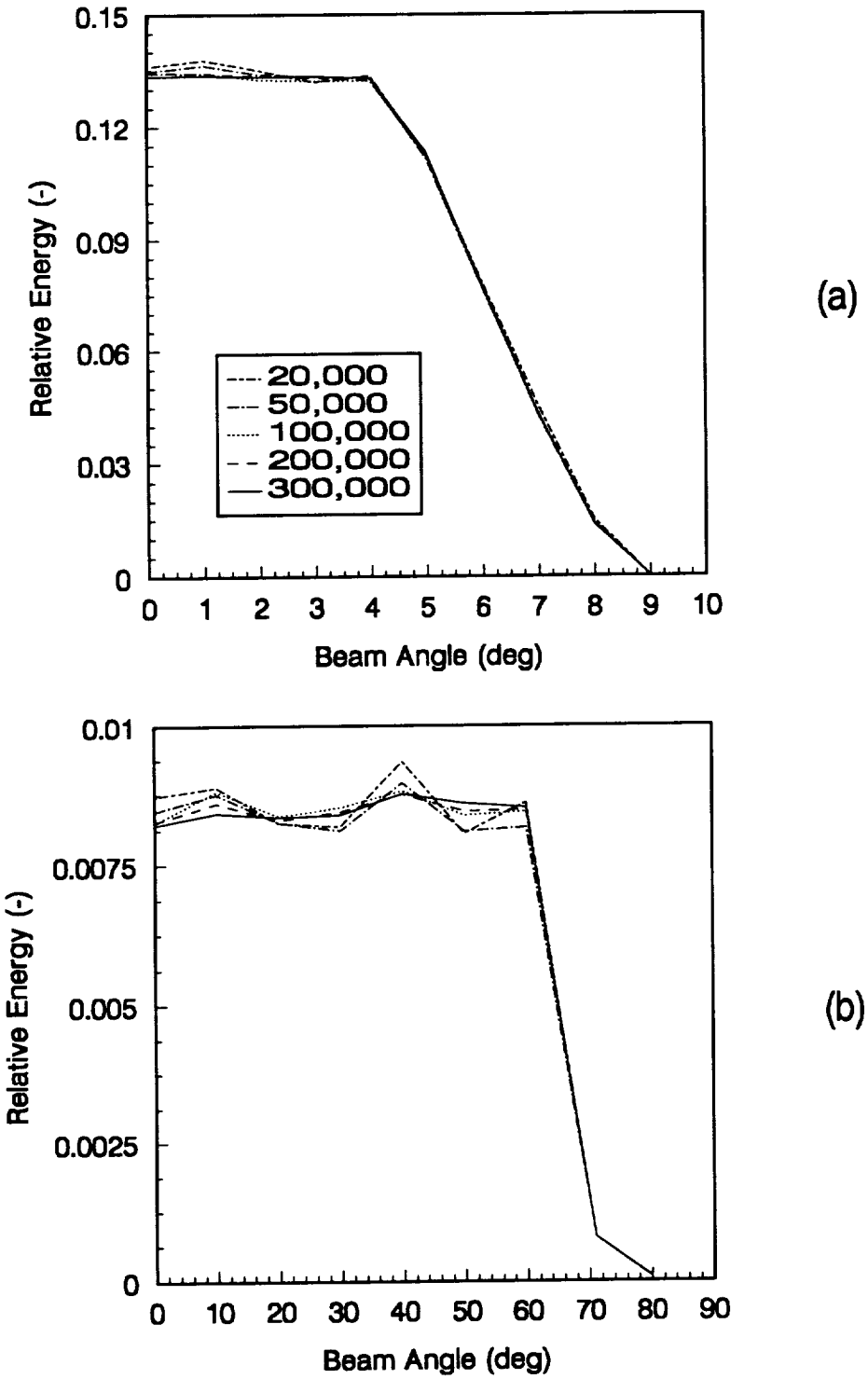


Fig. I.11 Variation of Cavity Absorption with the Number of Energy Bundles Used in the Monte-Carlo Ray Trace During (a) Solar Calibration and (b) the Pitchover Maneuver.

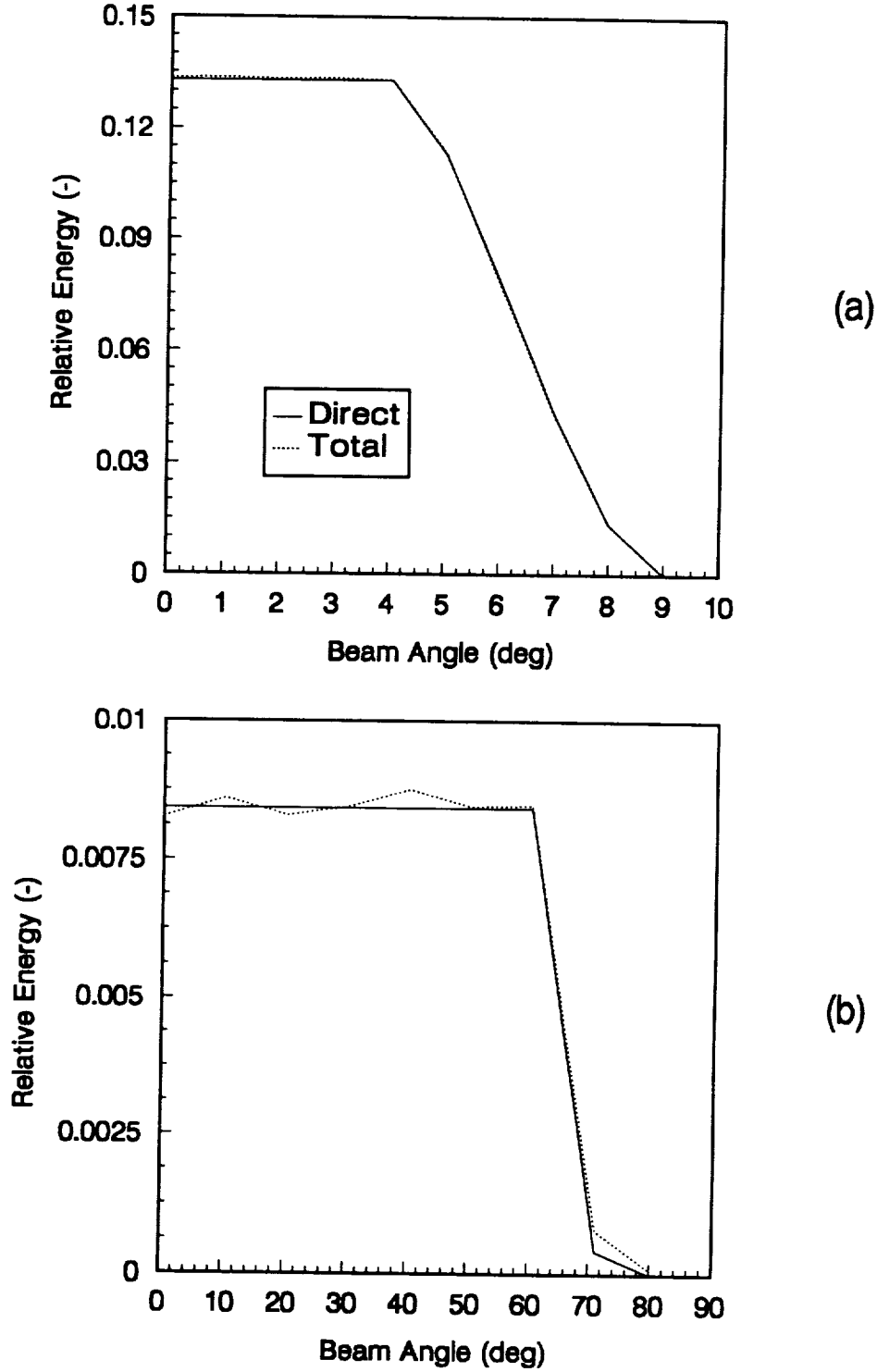


Fig. I.12 Total and Direct Cavity Absorption of Radiation During (a) Solar Calibration and (b) the Pitchover Maneuver.

Table I.1 Thermal Noise Due to Scattered Solar Radiation

Solar Calibration			Pitchover Maneuver		
θ (deg)	P_{scat} (mW)	Noise* (%)	θ (deg)	P_{scat} (mW)	Noise* (%)
0.0	0.213	0.478	0.0	-0.832	-1.912
1.0	0.281	0.630	10.0	0.908	2.038
2.0	0.176	0.396	20.0	-0.635	-1.547
3.0	0.200	0.449	30.0	0.100	0.260
4.0	0.049	0.112	40.0	1.368	3.875
5.0	0.135	0.358	50.0	0.107	0.373
6.0	-0.324	-1.261	60.0	0.136	0.608
7.0	-0.100	-0.705	71.0	0.636	48.0
8.0	0.036	0.800	80.0	0.079	100.0
9.0	0.044	100.0	-	-	-
Average: 0.071			Average: 0.207		

$$* \; Noise = \frac{P_{total} - P_{direct}}{P_{total}} x100\%$$

Table I.2 Thermal Noise Due to Radiation Emission from the FOV Limiter, the Aperture Plate and the Active Cavity

	FOV Limiter		Aperture Plate		Self-Contamination	
Temp. (K)	P_{FOV} (mW)	Noise* (%)	P_{ap} (mW)	Noise* (%)	P_{sc} (mW)	Noise* (%)
290.0	0.094	0.212	0.008	0.019	-	-
295.0	0.101	0.227	0.009	0.020	-	-
300.0	0.108	0.243	0.010	0.022	-	-
307.0	0.118	0.266	0.011	0.024	0.033	0.071
307.5	0.119	0.268	0.011	0.024	0.034	0.071
308.0	0.120	0.270	0.011	0.024	0.034	0.071

$$* \; Noise = \frac{P}{P_{max}} x100\%$$

7.0 Conclusion

The solar calibration data presented in Reference 10 agree with pre-flight ground measurements to within ±0.6 percent. These data were processed with limited screening for thermal noise, which at the time of publication of Reference 10 was assumed to be capable of influencing the measurements. When all of the thermal noise sources computed in the present paper are combined, as indicated by Eq. (I.17), the total noise is 0.508 percent for solar calibration and 0.812 percent for the pitchover maneuver; i.e., both are near the ±0.6 percent pre-flight

measurement uncertainty. This finding suggests that extensive screening of the flight data is unnecessary, at least in the case of the solar calibration event. As long as the total, WFOV nonscanning radiometer channels are regularly calibrated in flight, their degradation, which is the main remaining obstacle to radiometer accuracy, probably will not impact the data.

The good agreement between the simulation and the ERBS flight data allows the model developed to be utilized as a standard tool to validate the consistency of the radiometer data throughout its operating life. The establishment of a data base and knowledge of the thermal noise and other error sources is essential to a comprehensive interpretation of the instrument data.

PART II

A STUDY OF A CERTAIN CLASS OF SECONDARY APERTURES FOR USE IN DYNAMIC SOLAR CALIBRATION OF RADIOMETRIC CHANNELS

1.0 Introduction

1.1 Objective

The physical characteristics of a space-borne radiometer, such as the ERBE nonscanner channel, will inevitably change with time. This change can be due to deterioration of the material properties of the radiometer surfaces because of ultraviolet radiation, micrometeor impact, condensation of outgassed products and charged-particle flux from the sun. As a consequence, the transfer function of such an instrument may change. We then would like to be able to measure the transfer function *in situ;* that is, we would like to develop a method for in-flight dynamic calibration. In order to evaluate candidate in-flight calibration methods using numerical simulation, it is desirable to have available an instrument dynamic model with known transfer function. It is even better if the dynamic model has a transfer function which is the same as that of the instrument channel for which the dynamic in-flight calibration method is intended. Encouraged by the excellent agreement between the measured and simulated dynamic electrothermal response of the wide field-of-view (WFOV) channel demonstrated in Part I, we now undertake to use this model to obtain the transfer function of this channel. We then

propose and evaluate a series of in-flight calibration scenarios which permit this transfer function to be obtained during channel solar calibration.

This part of the dissertation draws heavily on frequency-domain signal analysis. Therefore, the basic principles and concepts are presented in this chapter and in the subsequent two chapters. Data types are discussed in this chapter. The Fourier transform, sampling errors and the transfer function are treated in the next two chapters. These first three chapters are presented for the benefit of the reader who may feel the need to review certain concepts before proceeding. The reader who is already familiar with frequency-domain signal analysis will find these chapters to be a convenient reference and nomenclature.

1.2 Data Classification

Temperature, velocity, and displacement are examples of parameters which characterize a wide range of engineering phenomena. These parameters are usually given as a function of independent variables such as time and location. If these parameters can be predicted with reasonable accuracy, based on mathematical analysis or on empirical correlations obtained from experimental observations, then the physical process is said to be deterministic. Many if not most physical phenomena, however, do not behave in a deterministic way, nor do they follow the same pattern over time. Such nondeterministic processes can only be described using probabilistic methods and are said to be random.

1.2.1 Deterministic Data

The simplest engineering processes are those that produce deterministic data. There are two types of deterministic data: periodic and nonperiodic. Periodic data may be sinusoidal or complex. The cosine function

$$Y(t) = A\cos(\omega t), \tag{II.1}$$

is an example of sinusoidal data, and square wave or sawtooth functions are examples of complex periodic data.

Nonperiodic deterministic data include quasi-periodic and transient processes. The sum of nonharmonic frequencies is an example of quasi-periodic data. The temperature decay in a lumped heat capacity system, given by the general expression

$$T(t) = T_i e^{(-t/\tau)}, \tag{II.2}$$

is a familiar example of a nonperiodic transient process.

1.2.2 Random Data

Most practical physical processes are represented by random data, also referred to as stochastic data. Such processes are said to be *stationary* if the harmonic content

of the time series which describes them is invariant with time. In this case all of the statistical properties of the signal, such as the *mean value* and the *mean square value*, are also invariant with time. In other words, if two sample records are taken at two different periods of time, the mean value of the first will be equal to that of the second, and so forth. All other types of stochastic data are considered nonstationary. Stochastic stationary processes are said to be *ergodic* if the statistical properties for a particular sample record are equivalent to those computed for all sample records of the ensemble. Ergodic data are very important in statistical signal analysis because computation of the statistical properties is then required only over one sample record; hence, the potentially overwhelming amount of statistical computations can be reduced considerably. The remaining types of stationary stochastic data are nonergodic.

2.0 Review of Fourier Analysis

Consider the example of a radiometer on a space-borne satellite. This radiometer instrument detects the radiative flux coming from the Earth and transforms it into a digital signal. This signal, whose units may be W/m^2, may contain more information than simply an instantaneous quantity of heat flux arriving at the instrument. It could also provide information on the spatial distribution of radiation in the radiometer footprint, or field-of-view (FOV), and even on the dynamic response of the instrument itself. *Signal processing* is a means of transforming time-series data from one form to another that permits broader understanding of both the signal and the instrument which produces it.

In order to exploit the domain of signal processing, certain mathematical tools and rules must be mastered. These include the Fourier series, the Fourier transform and Shannon's sampling theorem.

2.1 Fourier Series

In this section and hereafter the variable t is used to refer not only to time but also to space (distance and angle). Hence, when the term (angular) frequency ω is used in

general, it can refer to temporal, spatial or angular frequency. Now let us assume that *f(t)* is a periodic function with period *T*. Then *f(t)* can be represented by the Fourier series

$$f(t) = a_0 + \sum_{n=1}^{\infty} (a_n \cos\omega_n t + b_n \sin\omega_n t), \qquad \textbf{(II.3)}$$

where $\omega_n = 2\pi n/T$. The Fourier coefficients of Eq. (II.3) are

$$a_n = \frac{2}{T} \int_{-T/2}^{T/2} f(t)\cos(\omega_n t)\, dt, \qquad n = 1,2,3,\ldots, \qquad \textbf{(II.4)}$$

and

$$b_n = \frac{2}{T} \int_{-T/2}^{T/2} f(t)\sin(\omega_n t)\, dt, \qquad n = 1,2,3,\ldots . \qquad \textbf{(II.5)}$$

The first term of Eq. (II.3) represents the mean value of f(t) and is given by

$$\mu_f = a_0 = \frac{1}{T} \int_{-T/2}^{T/2} f(t)\, dt. \qquad \textbf{(II.6)}$$

Based on trigonometric relations, Eq. (II.3) can alternatively be represented by

$$f(t) = A_0 + \sum_{n=1}^{\infty} A_n \cos(\omega_n t - \theta_n), \qquad \omega_n = 2\pi n/T, \tag{II.7}$$

where

$$A_n = \sqrt{a_n^2 + b_n^2}, \qquad n = 1,2,3,\ldots, \tag{II.8}$$

and

$$\theta_n = \tan^{-1}[b_n/a_n]. \tag{II.9}$$

The expression for the Fourier series can further be extended by using the Euler relation

$$e^{-i\alpha} = \cos\alpha - i\sin\alpha, \tag{II.10}$$

where $i = \sqrt{-1}$ is the imaginary operator. Equation (II.7) then becomes

$$f(t) = \sum_{n=-\infty}^{\infty} A_n e^{i\omega_n t}, \tag{II.11}$$

where

$$A_n = \frac{1}{2}(a_n - ib_n) = \frac{1}{T}\int_0^T f(t)\, e^{-i\omega_n t}\, dt, \qquad n = \pm 1, \pm 2, \pm 3, \ldots \quad . \qquad \textbf{(II.12)}$$

2.2 Fourier Transform

In the previous section it was assumed that *f(t)* is periodic. In many if not most cases, however, the signal is nonperiodic. For stationary, nonperiodic functions the Fourier series can be extended by assuming the period T to be infinite. This leads to the Fourier Integral,

$$F(\omega) = \int_{-\infty}^{\infty} f(t)\, e^{-i\omega t}\, dt, \qquad -\infty < \omega < \infty. \qquad \textbf{(II.13)}$$

Equation (II.13) represents the frequency spectrum of *f(t)*. The complex form of $F(\omega)$ contains information about the amplitude and the phase of the signal. If the spectrum $F(\omega)$ is known, the time-series records can be generated using the inverse Fourier transform

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega)\, e^{i\omega t}\, d\omega\,, \qquad -\infty < t < \infty. \tag{II.14}$$

Equations (II.13) and (II.14) are referred to as the *Fourier transform pairs*, and they are continuous over their domain of application.

2.4 Sampling Theorem

One of the most important features of signal processing is that a continuous (or analog) signal can be represented in a digital form without any loss of information if the signal is properly sampled. In other words, the amplitude of a continuous function can be sampled at a number of equally spaced discrete points. These points can then be used to reconstruct the original (continuous) function. This can be accomplished if the sampling rate conforms to *Shannon's theorem*, also referred to as the *sampling, or Nyquist, theorem.* Blackman and Tukey [24] give a detailed treatment of the sampling theorem, which states that the sampling rate must be at least twice the period of the highest frequency component of the time series. Suppose we take N samples from a continuous signal during a period of time of T_s. Then the highest frequency that can be properly sampled is

$$f_N = \frac{N}{2T_s} = \frac{1}{2\Delta t}\,, \tag{II.15}$$

where $\Delta t = \frac{T_s}{N}$ is the sampling interval. The corresponding sampling frequency is twice f_N, or

$$f_m = \frac{N}{T_s} . \qquad \text{(II.16)}$$

For example, a signal containing frequencies of up to 10 Hz should be sampled at least 20 times per second. The frequency f_N is called the *Nyquist, or the folding,* frequency.

2.5 Sampling Errors

In the previous section it is claimed that proper use of the sampling theorem permits the recovery of the original signal. In practice, however, chances are not good for full recovery of the signal. This is due to certain errors that could be present in the system. These are predominately *aliasing, leakage* and *round-off* errors.

2.5.1 Aliasing

Aliasing, which is the most important error in signal processing, is caused by undersampling. That is, not enough samples are taken, as required by Eq. (II.16), to recover all frequency components. As a consequence, frequencies above the Nyquist

frequency are folded over, or aliased, onto the lower frequencies. These aliased (or disguised) frequencies will appear below the Nyquist frequency according to the following rule: If f_u is the undersampled frequency, then because of violation of the sampling theorem, f_u will disguise itself with the lower frequencies f_k such that [25,26]

$$f_k = 2kf_N - f_u, \qquad k = 1, 2, 3, \ldots \quad . \tag{II.17}$$

The effect of aliasing due to f_u is strongest at f_1, and diminishes with increasing k.

Aliasing can be avoided by sampling the data correctly; i.e, by obeying Shannon's theorem, or by filtering out those frequencies above the Nyquist frequency.

2.5.2 Leakage

Fourier analysis assumes that the time series being analyzed is periodic of period T in order to reconstruct the original signal. This reconstructed signal is repeated over the infinite domain $-\infty \leq t \leq +\infty$. This requires a smooth transition from one sample record, of period T, to the next. If the sample records do not begin and end with the same ordinate value (see Fig. II.1), then discontinuities will occur between sample periods as they are repeated over the infinite domain. The discontinuities between adjacent records generate relatively high amplitude high-frequency components. This phenomenon leads to what is referred to as *leakage* which manifests itself by distorting the resulting frequency spectra. Contributions from the pseudo-frequency components created by

leakage combine with and so distort the actual frequency components of the signal.

Leakage can be avoided by choosing overlapping sample records if the time series is long enough to permit such partitions. Another way of avoiding leakage is by utilizing *time-domain windowing*. In this approach the sample record is multiplied (weighted) by a windowing function. One such windowing function is the Hanning function,

$$w_h(t) = \frac{1}{2}[1 - \cos\frac{2\pi t}{T}]. \qquad \textbf{(II.18)}$$

Since $w_h(t)$ is zero at the beginning and end ($t=0$ and $t=T$) of the time record, the latter's beginning and end will have the same ordinate value (zero) and hence continuity is attained.

The only disadvantage to windowing is a decrease of the spectral resolution [27]. Other types of windows are the Hamming, the Nuttall, the rectangle, the triangle, the Kieser-Bessel, and the Blackman, to name a few. The reader is referred to References 27 and 28 for further details on windowing.

2.5.3 Round-Off

When a continuous function $f(t)$ is sampled, its magnitude is taken at discrete points. The magnitude for each sample is expressed by a digital number that is an approximation to $f(t)$ at that point. The difference between this number and the actual instantaneous magnitude of $f(t)$ is *called round-off*, or *quantization* error. Round-off error,

which is illustrated in Fig. II.2, is basically hardware related (analog-to-digital converters or computers). Any machine used to record digital data has a limited resolution. Consequently, the reconstructed signal will be slightly different from the original signal it represents. In most engineering applications the round-off error is small compared to the other types of errors. In some cases, however, the round-off error can be significant, as is demonstrated in Section 4.5. Round-off errors can be minimized by oversampling and by using high-resolution hardware. In communication systems, there exists a superior method for the elimination of quantization errors called *coding* [29,30]. This method, which cannot be used in the present, is beyond the scope of this research.

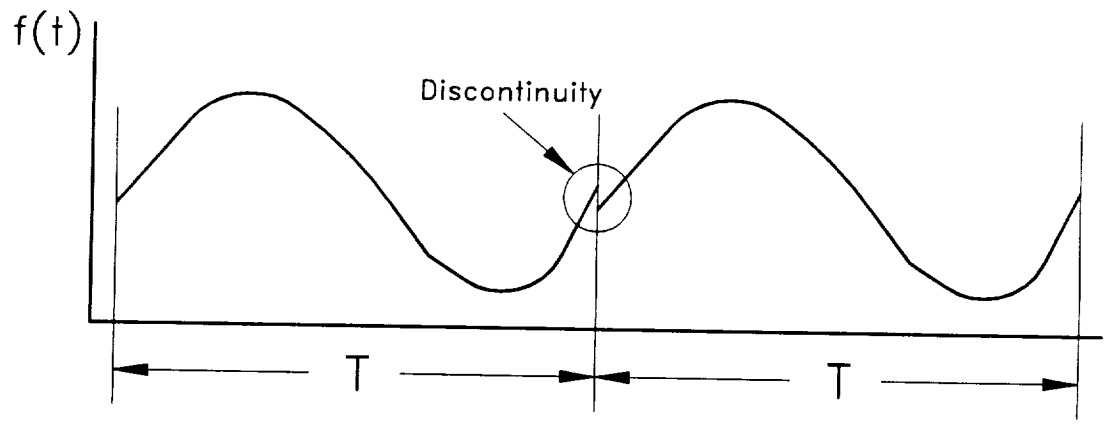


Fig. II.1 Illustration of Leakage errors.

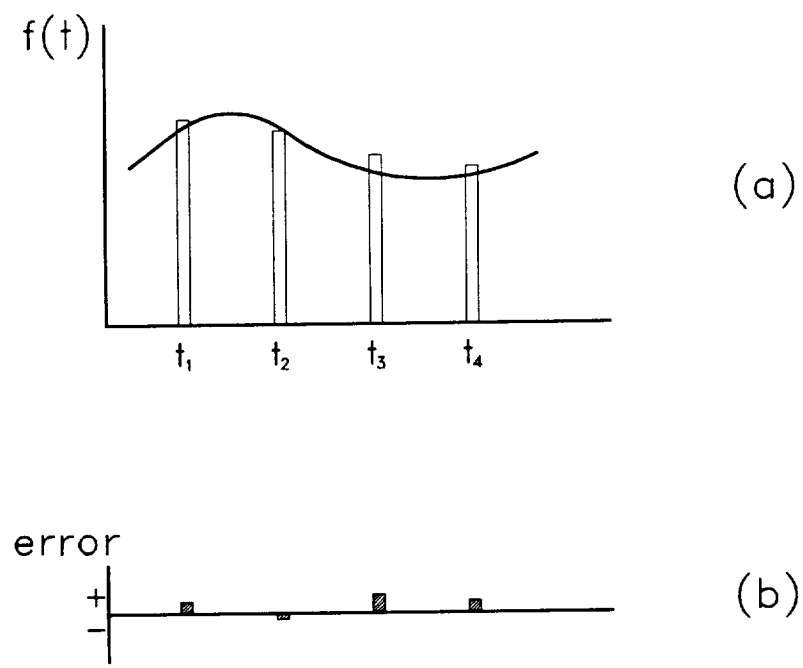


Fig. II.2 Illustration of Round-off errors.

2.6 The Discrete Fourier Transform (DFT) and the Fast Fourier Transform (FFT)

The continuous Fourier transform pair, represented by Eqs. (II.13) and (II.14), exists only in theory. This is because an infinite domain is not possible in practice, and because the function $f(t)$ can be represented only at discrete points. Hence, the integral limits of these equations are confined to the domain of the sample record.

Suppose the Fourier transform is to be computed for a hypothetical case where the amplitude of $f(t)$ is sampled at N evenly spaced discrete points. The Fourier integral of Eq. (II.13) can then be approximated as

$$F(\omega) = \sum_{k=1}^{N} f(k\Delta t)\, e^{-i\omega k\Delta t}\, \Delta t, \qquad \textbf{(II.19)}$$

where N is the number of samples taken in period $T = N\Delta t$.

The finite-range Fourier transform exists only at discrete frequencies which are multiples of $\omega_0 = 2\pi/T$, and $F(\omega)$ has the dimensions of $f(t)\Delta t$. Thus, ω is represented by $m\omega_0$ in Eq. (II.19) and the equation is divided through by the scaling factor $N\Delta t$, giving

$$\mathscr{F}(m\omega_0) = \frac{1}{N}\sum_{k=1}^{N} f(k\Delta t)\, e^{-im\omega_0 k\Delta t}, \qquad 0 \le m \le N-1\ . \qquad \textbf{(II.20)}$$

where ω_0 is the fundamental frequency given by

$$\omega_0 = \frac{2\pi}{T}. \tag{II.21}$$

Substituting Eq. (II.21) into Eq. (II.20), we obtain

$$\mathscr{F}(m\omega_0) = \frac{1}{N}\sum_{k=1}^{N} f(k\Delta t)\, e^{-2\pi i m k/N}, \qquad 0 \le m \le N-1 \ . \tag{II.22}$$

Equation (II.22) is the final form of the Discrete Fourier Transform (DFT). Note that this transform is periodic in m with period N.

It is obvious from Eq. (II.22) that to find one frequency component N computations are needed; thus, N^2 computations are required to obtain the entire spectrum since the time series contains N samples. This number of computations can be excessive for any computer because of the exponential term, especially for a large number N of data samples. Fortunately, Cooley and Tukey [31] devised an algorithm that takes advantage of repetitive computations, and efficiently reduces the number of computations to $N \log_{10} N$. This algorithm is appropriately called the *Fast Fourier Transform* (FFT). There is, however, a restriction on the use of the FFT; the number of data samples must be an integer multiple of two. Other types of transforms available for frequency-domain analysis include the z-Transform and Laplace Transform [29,32].

3.0 The Transfer Function

3.1 Introduction

Mathematical modeling is the first and most important task in the analysis and design of a control system. Such modeling is carried out through the application of physical laws governing the response of the system to some forcing function. In this task, a relationship is sought between the input, or the cause, and the output, or the effect. Often the resulting differential equations are first- and second-order, although they could be of higher order. For small deviations from equilibrium, it is usually assumed that nonlinear effects are negligible and so these equations are generally linear.

Suppose, for a particular system model, the procedure outlined above yields a linear differential equation of the general from

$$a_2\frac{d^2y}{dt^2} + a_1\frac{dy}{dt} + a_0y = b_1\frac{dx}{dt} + b_0x, \qquad \text{(II.23)}$$

with the output vector on the left-hand side and the input vector on the right-hand side. The a's and b's are constant parameters.

If the differential operator d/dt is denoted by D, then Eq. (II.23) becomes

$$(a_2 D^2 + a_1 D + a_0)y = (b_1 D + b_0)x. \tag{II.24}$$

One can apply the Laplace transform to Eq. (II.24), assuming that all initial conditions are zero, and obtain

$$Y(s) = G(s)X(s), \tag{II.25}$$

where $Y(s)$ and $X(s)$ are the Laplace transforms of $y(t)$ and $x(t)$, respectively, and s is the complex frequency variable of the Laplace transform. The term $G(s)$, which relates the output to the input in the frequency domain, is the *transfer function* and is given in this example by

$$G(s) = \frac{b_1 s + b_0}{a_2 s^2 + a_1 s + a_0}. \tag{II.26}$$

The complex form of the transfer function *G(s)* contains two parts: (1) the gain, which is the magnitude of *G(s)*, and (2) the phase angle, which is the inverse tangent of the ratio of the imaginary part over the real part of *G(s)*.

4.0 The Transfer Function of the ERBE Nonscanner

The dynamic electrothermal model of the ERBE nonscanner ACR instrument developed in Reference 14 and described briefly in Part I has been proven to give excellent agreement between simulated and actual solar calibration data. This agreement establishes the validity of the model. Now, the instrument and its model are actually nonlinear because of the radiation term. Furthermore, the model's governing equation, Eq. (I.1), is a partial differential equation. However, the variation of temperature within the instrument cavity as well as with time during solar calibration is less than one kelvin degree [14]. This small variation leads to a behavior very similar to that of a lumped-capacity model. As a consequence, it is valid to compute the transfer function of the ACR system.

To obtain this transfer function, a series of simulated signals of different waveforms are introduced into the model. The procedure for the computation of the transfer function involves the successive application of three methods: the direct method, the method of least squares, and the Newton-Raphson method.

4.1 The Direct Method

As mentioned earlier, any signal can be represented by an infinite sine series. Because the ACR model behaves in a quasi-linear manner, the principle of superposition applies. Then instead of using a composite signal, the individual sinusoidal components of this signal can be introduced to the system model one at a time. Not only is this superposition easy to implement in a simulated input to the dynamic electrothermal model, but also it is the best way to eliminate signal analysis errors such as aliasing.

The quasi linearity of the ACR model is exploited in the direct method. Because the input function *f(t)* can be written as an infinite sine series and system nonlinearities are insignificant, the system's response to any one of the terms of the series is not affected by the presence or absence of the other terms.

A series of harmonic signals is chosen whose frequencies are within the range of the model's frequency response, or bandwidth. The magnitude ratio of, and the phase difference between, the input and output for each signal are computed and plotted on a Bode diagram. Each signal is of sufficient duration to assure that steady-state conditions are attained.

In the direct method the equation of the input function is assumed to be of the form

$$y_i(t) = A_i + B_i \sin(f_n t) \ , \qquad \textbf{(II.27)}$$

where A_i is a predetermined constant (mean value), and B_i is the magnitude of the input

function whose frequency is f_n. The phase angle in Eq. (II.27) is arbitrarily set to zero. This is acceptable because we are interested only in the phase difference between the input and the output. The equation of the output should be similar in form to that of the input, or

$$y_o(t) = A_o + B_o \sin(f_n t + \theta_o) . \tag{II.28}$$

Figure II.3 shows the input and the output waveforms for one input function whose frequency is 0.04 Hz. Because of the linearity of the system, the output frequency must be equal to that of the input [33]. In practice, however, there will be a slight difference in frequency due to round-off errors associated with the computation. At this point, the main task is to determine the output parameters A_o, B_o and θ_o. To do this, a curve must be fit to the output signal according to Eq. (II.28) using the method of least-squares and the Newton-Raphson method. These topics are treated in the next two sections. Once these parameters are obtained, the gain at frequency f_n is given by B_o/B_i and the phase angle is θ_o.

4.2 Method of Least Squares

The object here is to obtain the analytical expression of the output based on data of the type shown in Fig. II.3. The form of this analytical expression is given by Eq. (II.28). We now seek to minimize the expression

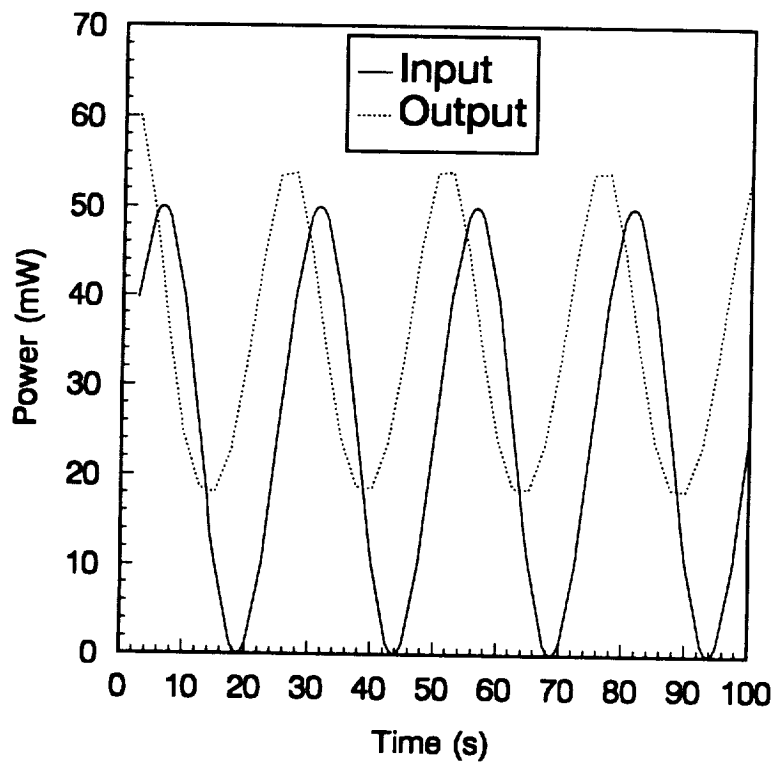


Fig. II.3 Input and Output Functions for a Sinusoidal Input Signal, f = 0.04 Hz.

$$S = \sum_{k=1}^{N} [A_o + B_o \sin(f_n t_k + \theta_o) - Y_k]^2 \ , \tag{II.29}$$

where N is the number of discrete data points of the output signal taken at proper sampling intervals, and the Y_k's are the values of the discrete data; i.e., the corresponding ordinate values of Fig. II.3. Equation (II.29) is differentiated with respect to each parameter (A_o, B_o, θ_o and f_n) whose value is sought and the results set equal to zero,

$$\begin{aligned} \frac{\partial S}{\partial A_o} &= g_1 = 0 \, , \\ \frac{\partial S}{\partial B_o} &= g_2 = 0 \, , \\ \frac{\partial S}{\partial f_n} &= g_3 = 0 \, , \\ \frac{\partial S}{\partial \theta_o} &= g_4 = 0 \, . \end{aligned} \tag{II.30}$$

The parameter f_n is included in the list of partial derivatives because it facilitates the use of the Newton-Raphson method. The use of the method of least squares for general curve fits is described in Reference 34.

4.3 The Newton-Raphson Method

Since Eqs. (II.30) are not linear, an iterative approach such as the Newton-Raphson Method is needed to solve them. Differentiation of Eqs. (II.30) with respect to each parameter, and their multiplication by the variations in these parameters; ΔA_o, ΔB_o,

$\Delta\theta_o$ and Δf_n; leads to the linear system

$$
\begin{aligned}
\frac{\partial g_1}{\partial A_o}\Delta A_o + \frac{\partial g_1}{\partial B_o}\Delta B_o + \frac{\partial g_1}{\partial f_n}\Delta f_n + \frac{\partial g_1}{\partial \theta_o}\Delta\theta_o &= g_1 , \\
\frac{\partial g_2}{\partial A_o}\Delta A_o + \frac{\partial g_2}{\partial B_o}\Delta B_o + \frac{\partial g_2}{\partial f_n}\Delta f_n + \frac{\partial g_2}{\partial \theta_o}\Delta\theta_o &= g_2 , \\
\frac{\partial g_3}{\partial A_o}\Delta A_o + \frac{\partial g_3}{\partial B_o}\Delta B_o + \frac{\partial g_3}{\partial f_n}\Delta f_n + \frac{\partial g_3}{\partial \theta_o}\Delta\theta_o &= g_3 , \\
\frac{\partial g_4}{\partial A_o}\Delta A_o + \frac{\partial g_4}{\partial B_o}\Delta B_o + \frac{\partial g_4}{\partial f_n}\Delta f_n + \frac{\partial g_4}{\partial \theta_o}\Delta\theta_o &= g_4 .
\end{aligned}
\tag{II.31}
$$

The Newton-Raphson iterative procedure involves the following steps:

1. Guess the values of the four parameters A_i, B_i, θ_o and f_n.
2. Compute g_1, g_2, g_3 and g_4, and their derivatives.
3. Solve the linear system of equations Eqs. (II.31), for ΔA_o, ΔB_o, Δf_n and $\Delta\theta_o$.
4. Increment the parameters A_o, B_o, θ_o and f_n by their variations in Step 3.
5. Repeat Steps 2 through 4 each time utilizing the updated parameters.
6. Stop when g_1, g_2, g_3 and g_4, are within an acceptable error or if convergence has not occurred after a specified number of iterations.

A more general approach as well as extensive details of the Newton-Raphson method can be found in Reference 35. A computer code based on the method of least-squares and the Newton-Raphson method, and adapted to the problem at hand, is given in Appendix B.

4.4 Transfer Function Results

In all cases of the sinusoidal signals used in this analysis, the number of iterations needed for convergence, when attained, never exceeds twenty. Sometimes convergence is never attained, however, and in this case a new initial guess of the parameters must be chosen. This is inherent in the Newton-Raphson method [35] and for that reason it is very important to make guesses as close to the actual values as possible. In the present analysis this is facilitated by the fact that one of the parameters, the frequency f_n, is known. Table II.1 gives the transfer function for the various frequencies used. Figure II.4 shows the gain, its corresponding value in decibels, expressed as $20\log_{10}(B_o/B_i)$, and the phase angle (θ_o) of the ACR model transfer function. Note that there is a nearly constant gain of essentially zero decibel in the frequency range $0 \leq f \leq 0.03$ *Hz*. In most of this frequency range, the gain is actually above zero (see the gain values in Table II.1) suggesting a slightly underdamped second-order system. Beyond a frequency of about 0.03 *Hz*, which may be assumed to be the natural frequency f_n, or corner frequency, the curve starts decreasing at a rate of -40 *dB/decade*. The phase angle ranges from 0 to -180 *deg*, with the phase angle at the naturel frequency being -90 *deg*. These are the characteristics of a second-order system [33,36,37], whose analytical transfer function is given by

$$G(\omega) = \frac{1}{1 - (\omega/\omega_n)^2 + 2i\zeta(\omega/\omega_n)}, \tag{II.32}$$

where ζ is the damping ratio, and ω_n is the natural angular frequency. In the remainder of this part of the dissertation, the curves of Fig. II.4 are referred to as the *theoretical transfer function.*

According to D'Souza [33], a pseudo-thermal time constant τ_t for such a second-order linear system can be defined as

$$\tau_t = \frac{1}{2\zeta\omega_n}. \tag{II.33}$$

A curve fit of Fig. II.4 utilizing Eq. II.32 reveals that $f_n = \omega_n/2\pi = 0.038$ Hz and $\zeta = 0.6$. Using these values the thermal time constant is approximately 3.5 s.

In radiometry, on the other hand, the cut-off, or corner frequency is defined as the frequency at which the square of the gain drops to 50 percent of its highest value [38]; i.e., the gain drops by -3 dB. This gain corresponds to a frequency of approximately f_n = 0.04 Hz as shown in Table II.1. The corresponding thermal time constant is then four seconds. It is important to note that the frequency response of the ACR in the frequency range of the present analysis is known to be dominated by the thermal time constant. The electrical time constant of the circuit in Fig. I.2 is 0.01 s. This time constant is obviously well beyond the range of the thermal model; and hence, it does not show up in the transfer function plot of Fig. II.4.

The Transfer Function of the ERBE Nonscanner

Table II.1 ACR Transfer Function Results.

frequency (Hz)	Gain, B_o/B_i (-)	$20Log_{10}$[Gain] (dB)	Phase (deg)
0.001563	1.00532	0.04609	-4.80
0.003125	1.01752	0.15086	-9.15
0.012500	1.05700	0.48150	-35.35
0.025000	1.03264	0.27898	-68.36
0.033300	0.92940	-0.63595	-97.83
0.040000	0.75660	-2.42267	-113.04
0.050000	0.51980	-5.68327	-124.45
0.066700	0.31185	-10.12109	-140.12
0.100000	0.14134	-16.99470	-158.67
0.200000	0.03643	-28.77082	-170.70
0.400000	0.00957	-40.38176	-177.71

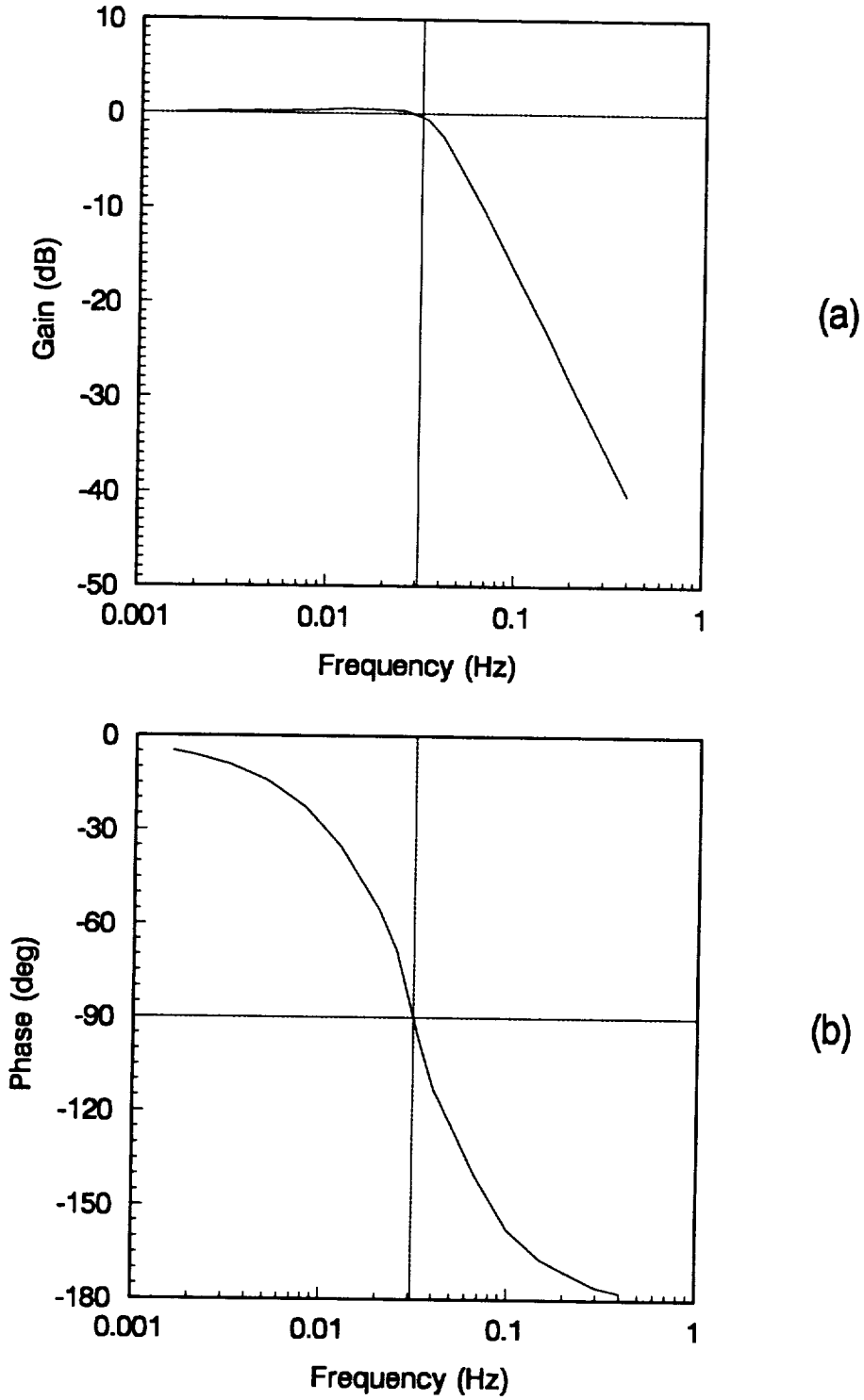


Fig. II.4 ACR Bode Diagram: (a) Gain, (b) Phase Angle.

4.5 Use of Square-Wave Input Function

As mentioned in Chapter 1, the main objective of Part II is to define a technique for measuring the ACR transfer function *in situ*. It is common practice in engineering to evaluate the dynamic characteristics of a system by introducing a square-wave function and studying the output waveform. A square-wave function can be represented by

$$f(t) = M\frac{4}{\pi}\sum_{\substack{k=1\\odd}}^{\infty}\frac{1}{k}\sin\left(\frac{2\pi k t}{T}\right), \qquad \text{(II.34)}$$

where M and T are the square-wave magnitude and period, respectively. Because most of the signal energy is concentrated at low frequencies, we first use only the ten lowest frequencies in the square-wave input function. Figure II.5 verifies that the signal still retains the essential features of a square wave. The ordinate scale of Fig II.5 is based on the ACR model's input power limit.

If at least twenty samples are taken over the period of the waveform, any aliasing problems will be eliminated. The fast Fourier transform (FFT) of the input and output time signals are obtained. This permits the amplitude and phase to be computed for each frequency using Eqs. (II.8) and (II.9). The gain and phase shift of the signal are then given by

$$Gain = \frac{(A_n)_{output}}{(A_n)_{input}} = \frac{(\sqrt{a_n^2 + b_n^2})_{output}}{(\sqrt{a_n^2 + b_n^2})_{input}}, \qquad \textbf{(II.35)}$$

and

$$Phase = (\theta_n)_{output} - (\theta_n)_{input} = (\tan^{-1}[b_n/a_n])_{output} - (\tan^{-1}[b_n/a_n])_{input}. \qquad \textbf{(II.36)}$$

The result is shown in Fig. II.6. Note that the gain and phase shift for low frequencies conform to the theoretical curve of Fig. II.4 (solid line). The gains corresponding to the last few harmonics, however, diverge from the theoretical curve in a tail-like pattern. Aliasing is not present because the time series is sampled according to Shannon's theorem. Leakage is not present either because the time-series period is chosen so that the ordinates at its beginning and end are equal, as verified in Fig. II.5. Therefore, the tail must be due to round-off error. In fact, there are two additional reasons for this deviation. First, when the input function of Eq. (II.34) is digitally generated, the number of decimal points is chosen to be six. A small number is deliberately chosen to emphasize the effect of round-off error. Second, when the even harmonics and those beyond the first ten odd harmonics are observed, they are found to be small but not zero. This means that there are pseudo-frequency components present where none are expected. These pseudo frequencies combine with the actual ones.

Now the reason that only the last few frequencies are affected is because their component magnitudes are low and hence can be overpowered by neighboring false frequencies of comparable magnitude. This phenomenon is further clarified in the next case. The outcome is actually manifested in the form of aliasing, for the pseudo

frequencies above the Nyquist frequency are folded back onto those below it. The effect is regressive so that the harmonic closest to the Nyquist frequency is affected more than its neighbor at a lower frequency, and so forth.

There may seem to be a contradiction in the above statement, since it has already been claimed that aliasing errors are eliminated. In fact, this is true because sampling is correctly performed based on the Nyquist theorem. Thus, the tail in Fig. II.6 is not due to the actual frequency content of the signal, but rather it is due to computer-generated pseudo frequencies.

To further clarify this phenomenon, a second signal containing only the next ten harmonics of the square wave is introduced into the ACR dynamic model. In this case, a maximum number of sixteen digits is allowed by the computer for the input signal. The result is shown in Fig. II.7, where it is clear that the transfer function is predicted correctly for the frequency range where the tail appears in Fig. II.6 (see data points between the vertical bars in the Figs. II.5 and II.6). There is still a deviation, however, at the higher frequencies, but it is not as pronounced as in the case of Fig. II.6. The lesson to be learned here is that the dynamic range of the computer limits the usefulness of the relatively low-amplitude frequency components. If the ratio of the high-amplitude components to the low-amplitude components approaches or exceeds the dynamic range of the computer, the low-amplitude components will be pushed down into the noise level. The consequence is a degradation of the recovery of the transfer function at the higher frequencies. This observation is the guiding principle of the array-type secondary aperture, or solar port array, investigated in Sections 5.2 through 5.4.

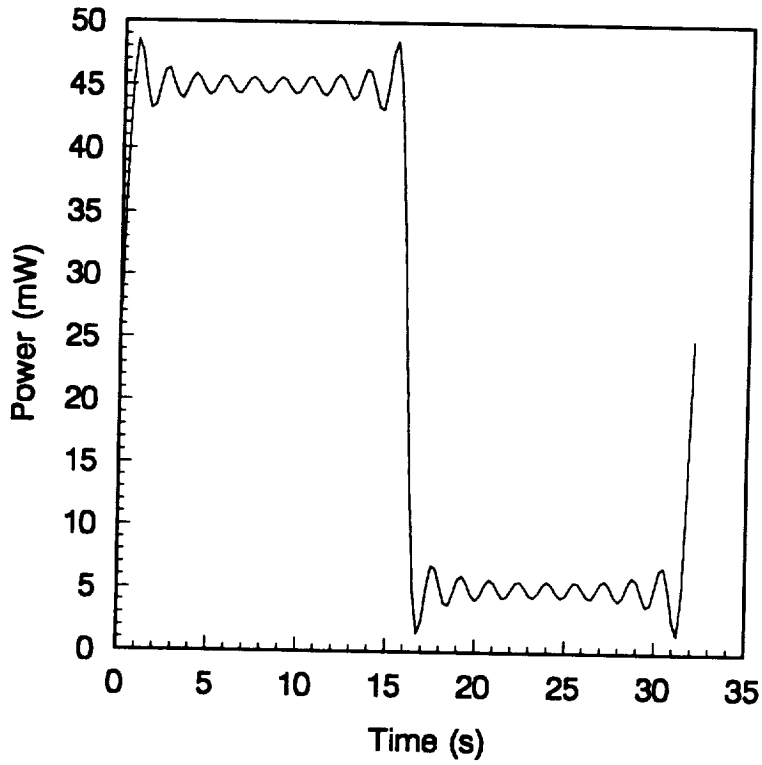


Fig. II.5 Square-Wave Input Function (First 10 Harmonics).

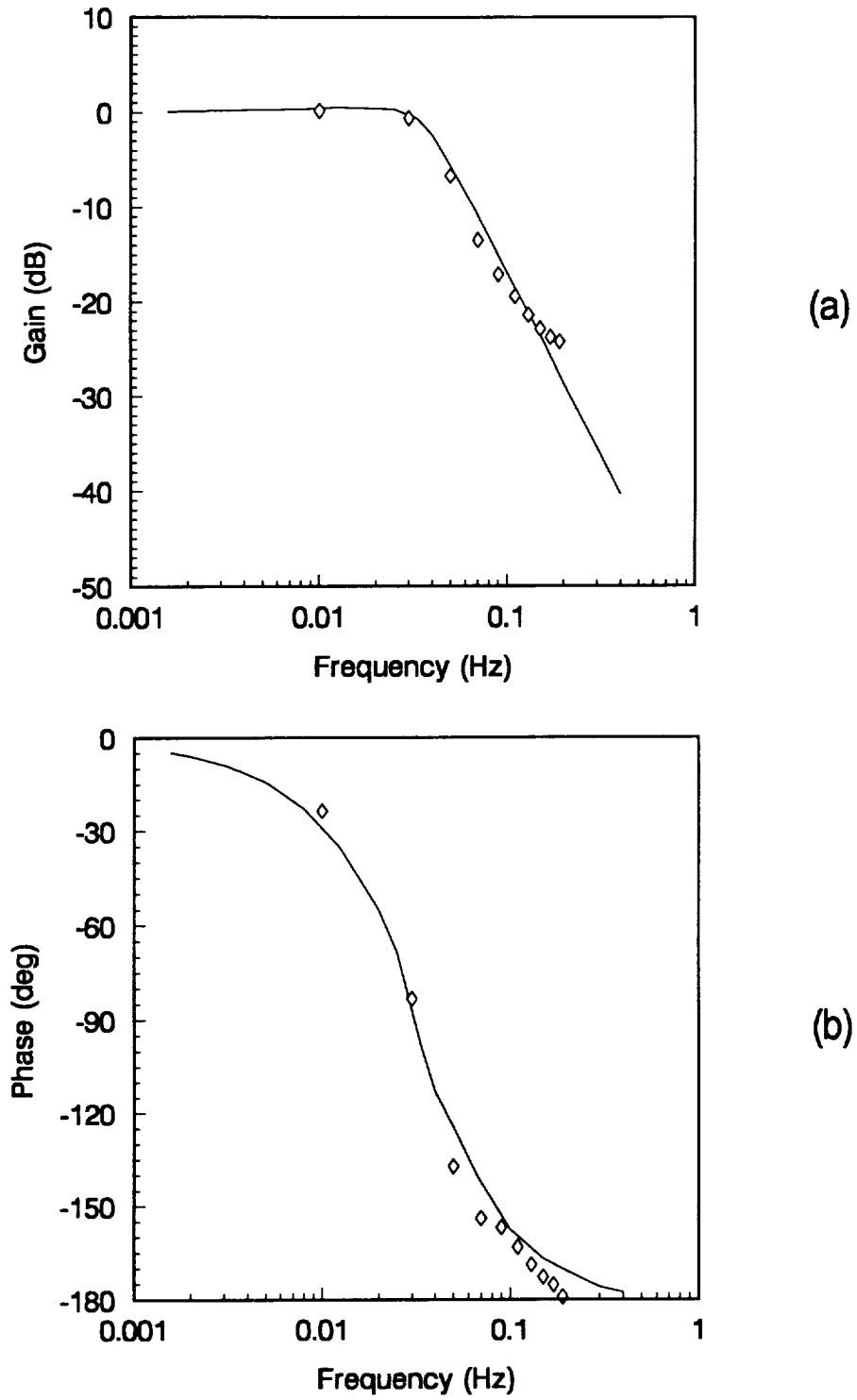


Fig. II.6 Bode Diagram for Square Wave (First 10 Harmonics): (a) Gain, (b) Phase Angle.

C-2

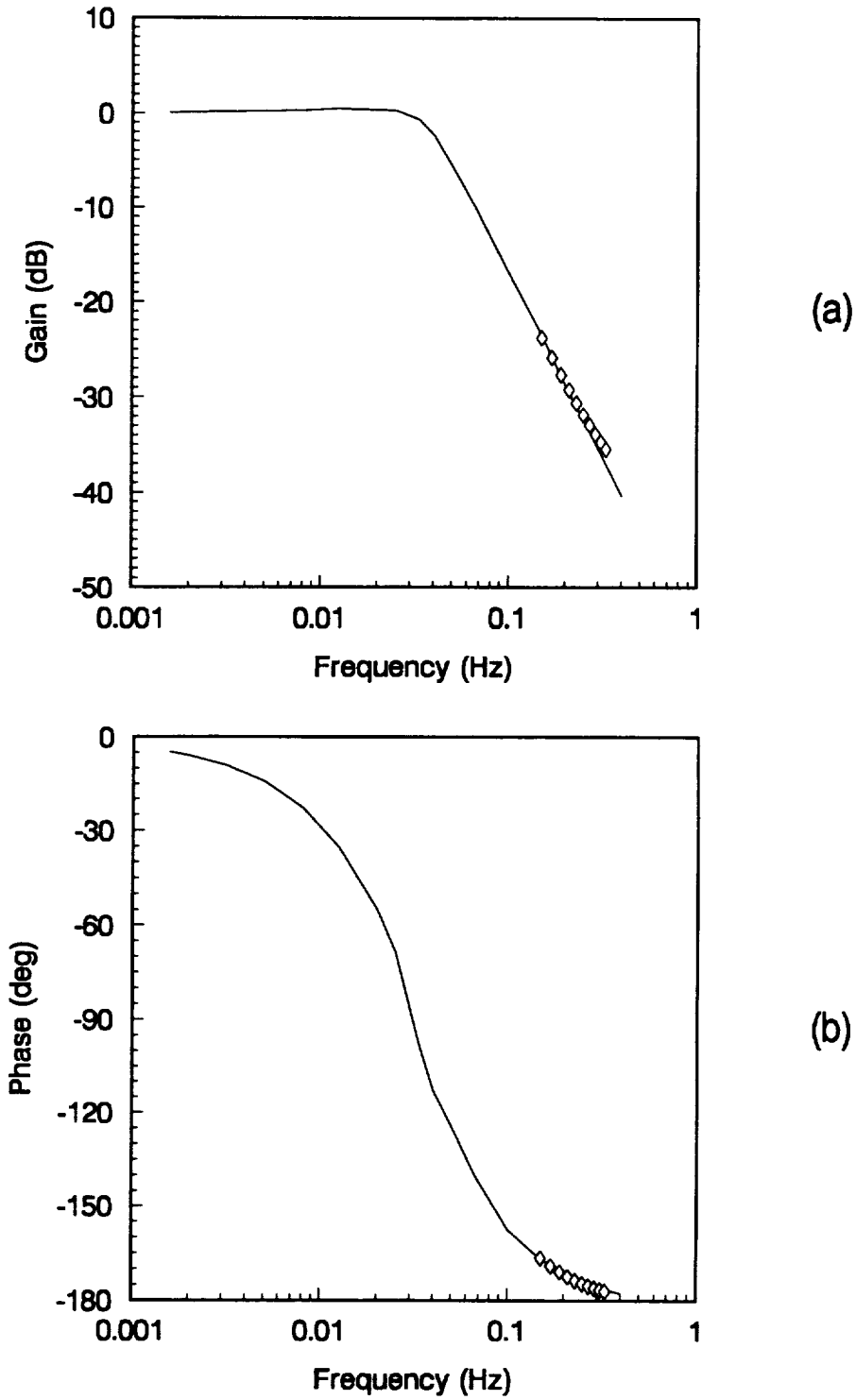


Fig. II.7 Bode Diagram for Square Wave (Second 10 Harmonics): (a) Gain, (b) Phase Angle.

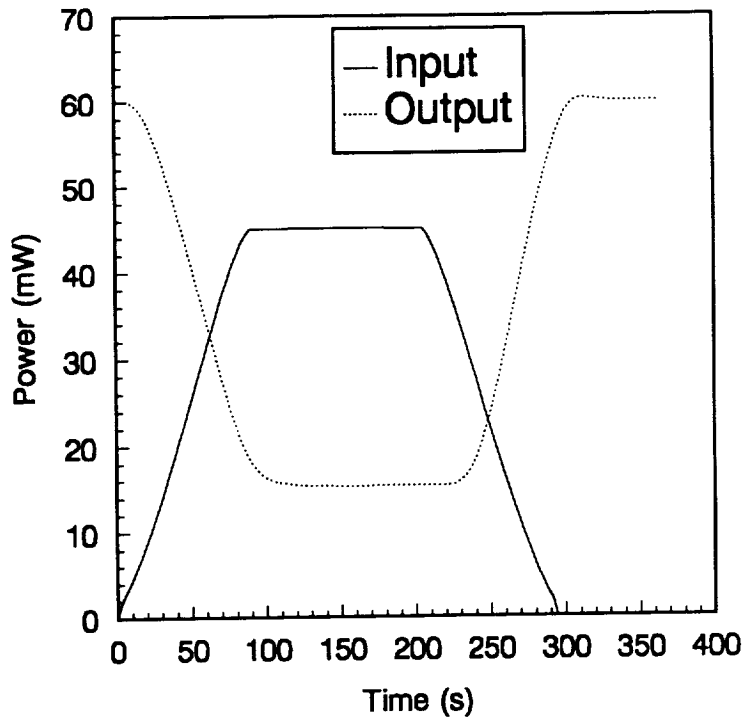
4.6 Study of Simulated and Real Solar Calibration Data for the ACR

Now we are ready to compute the transfer function based on the simulated solar calibration data of December 28, 1984, as well as on the actual satellite data for that same solar calibration event (see Fig. I.7). The frequency spectra for the input and output functions for both cases are shown in Figs. II.8 and II.9, respectively. The input function associated with satellite data during the solar calibration event is not measured but may only be assumed (but nonetheless to a high degree of confidence) on the basis of the known solar constant, the known evolution of satellite-sun orientation, and the known instrument aperture geometry. Therefore the same theoretical input function is used for both the simulated time series and the actual satellite data. It is significant that the spectra of Figs. II.8 and II.9 reveal a great concentration of energy at low frequencies. Figures II.10 and II.11 show the transfer function results plotted on the same axes as the theoretical transfer function for both cases. There is extensive scatter in the result, perhaps extensive enough to give up hope and abandon the task at hand. Apparently, all kinds of signal processing errors are present. Aliasing cannot be remedied since the sampling rate is fixed by ERBE operating considerations of overriding importance. It is possible in principle to minimize the other noise errors by performing averages on several sets of data, but unfortunately each solar calibration event is unique and consists of only a single data set.

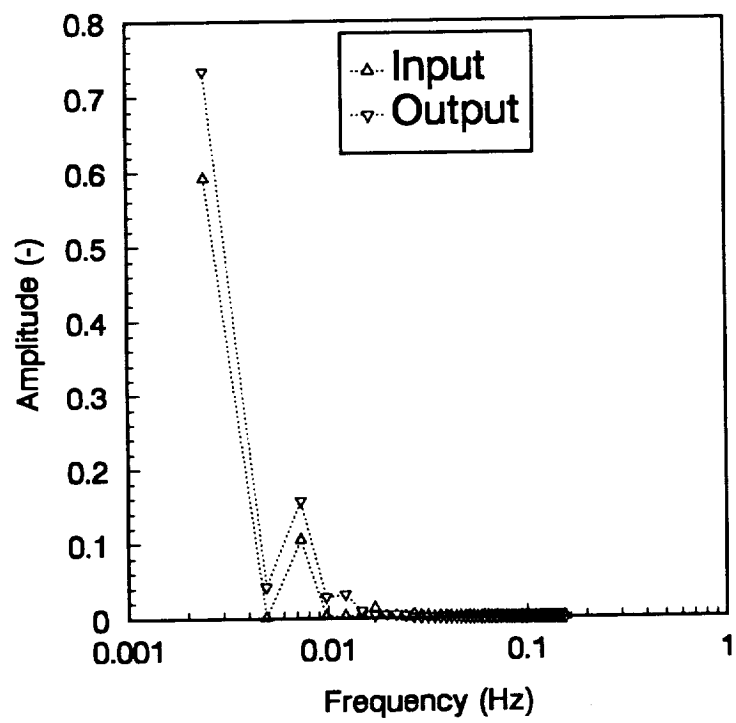
Leakage, on the other hand, can be remedied by the use of time-domain windowing as described in Section 1.2.5. The Hanning window (sometimes called the

squared-cosine window) is used for the analysis reported here. This choice of window is based on the speculation that only moderate to small signal interference may be present during solar calibration scenarios. This supposition is based on the smallness of the thermal noise components as established in Part I. Marple [27] gives a detailed analysis of the choice of windowing methods for a few types of interference. A further advantage of the Hanning window is its ease of implementation in the time domain.

Figures II.12 and II.13 show the transfer function using the simulated and actual data but with the enhancement of a Hanning window. There seems to be a slight improvement of the result in Fig. II.12 when compared to that in Fig II.10, but the same cannot be said for the actual data. It is evident that use of the solar calibration scenario for the input function is not a promising way of accurately measuring the ACR transfer function *in situ*. It may be speculated that this discouraging result is due to the extensive concentration of the signal energy at low frequencies. This encourages us to investigate other types of input functions that will allow the transfer function of the ACR and perhaps other types of radiometric instruments to be measured *in situ*.

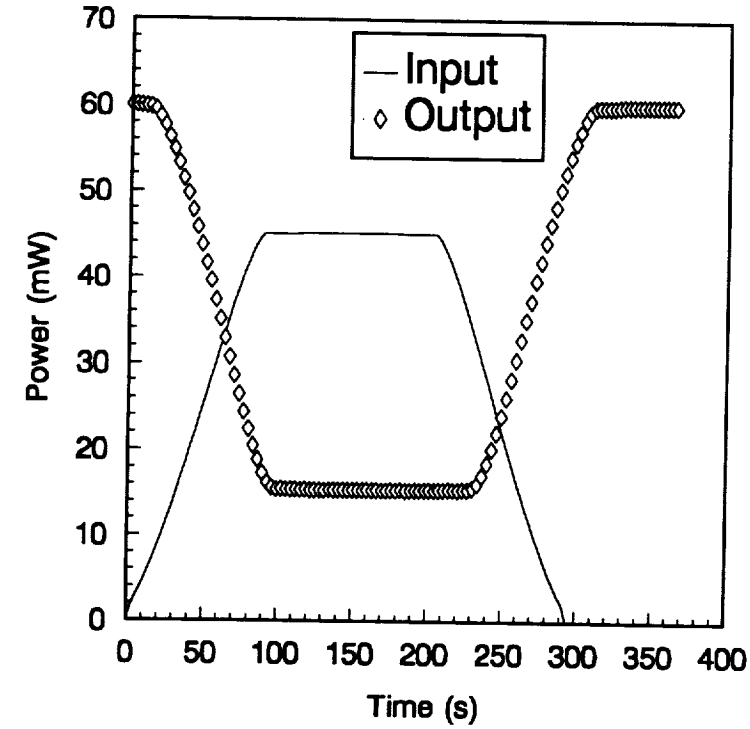


(a)

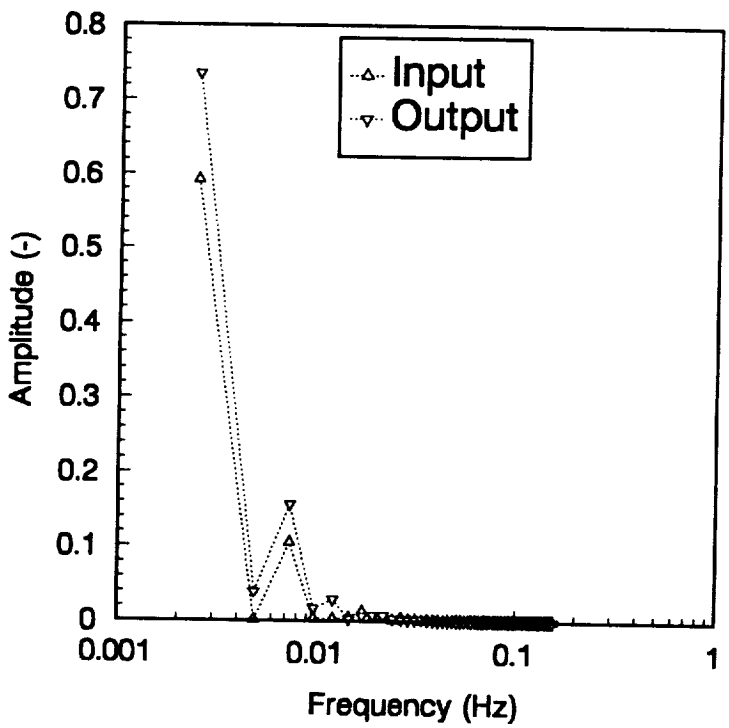


(b)

Fig. II.8 Simulated Solar Calibration Data of 12/28/1984: (a) Input and Output Functions, (b) Corresponding Frequency Spectra.



(a)



(b)

Fig. II.9 Satellite Solar Calibration Data of 12/28/1984: (a) Input and Output Functions, (b) Corresponding Frequency Spectra.

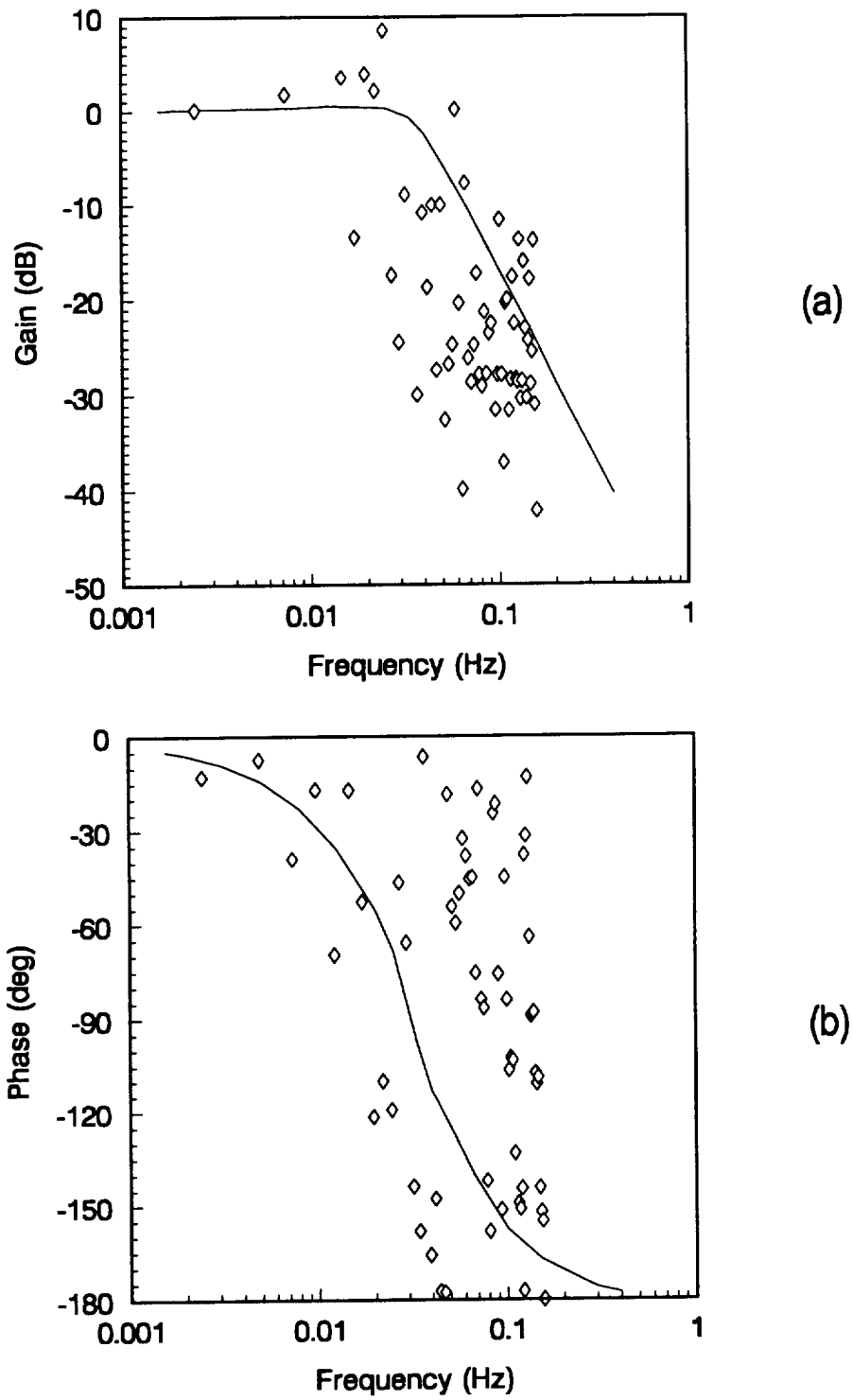


Fig. II.10 Bode Diagram for the Simulated Solar Calibration Data of 12/28/1984: (a) Gain, (b) Phase Angle.

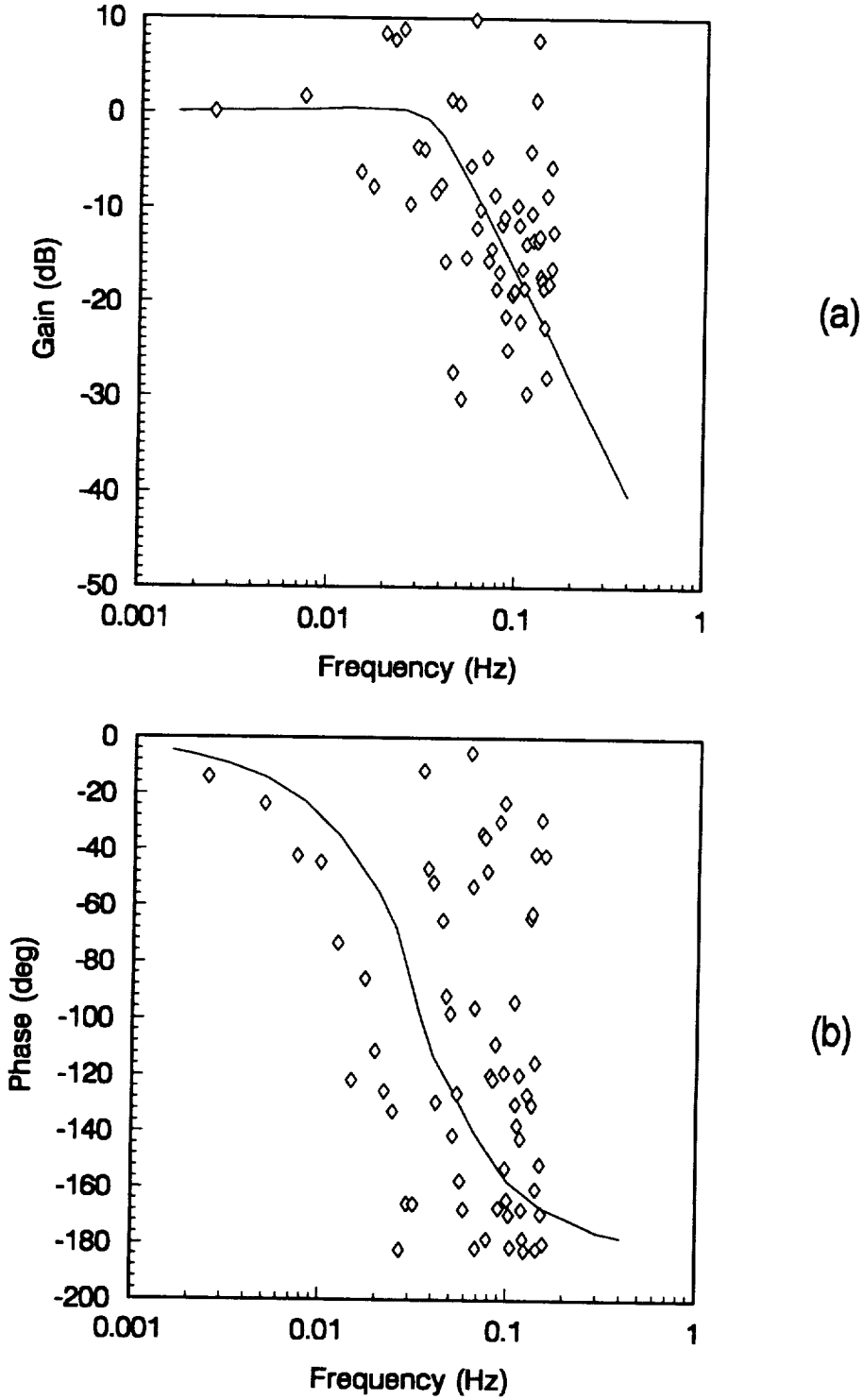


Fig. II.11 Bode Diagram for the Actual Solar Calibration Data of 12/28/1984: (a) Gain, (b) Phase Angle.

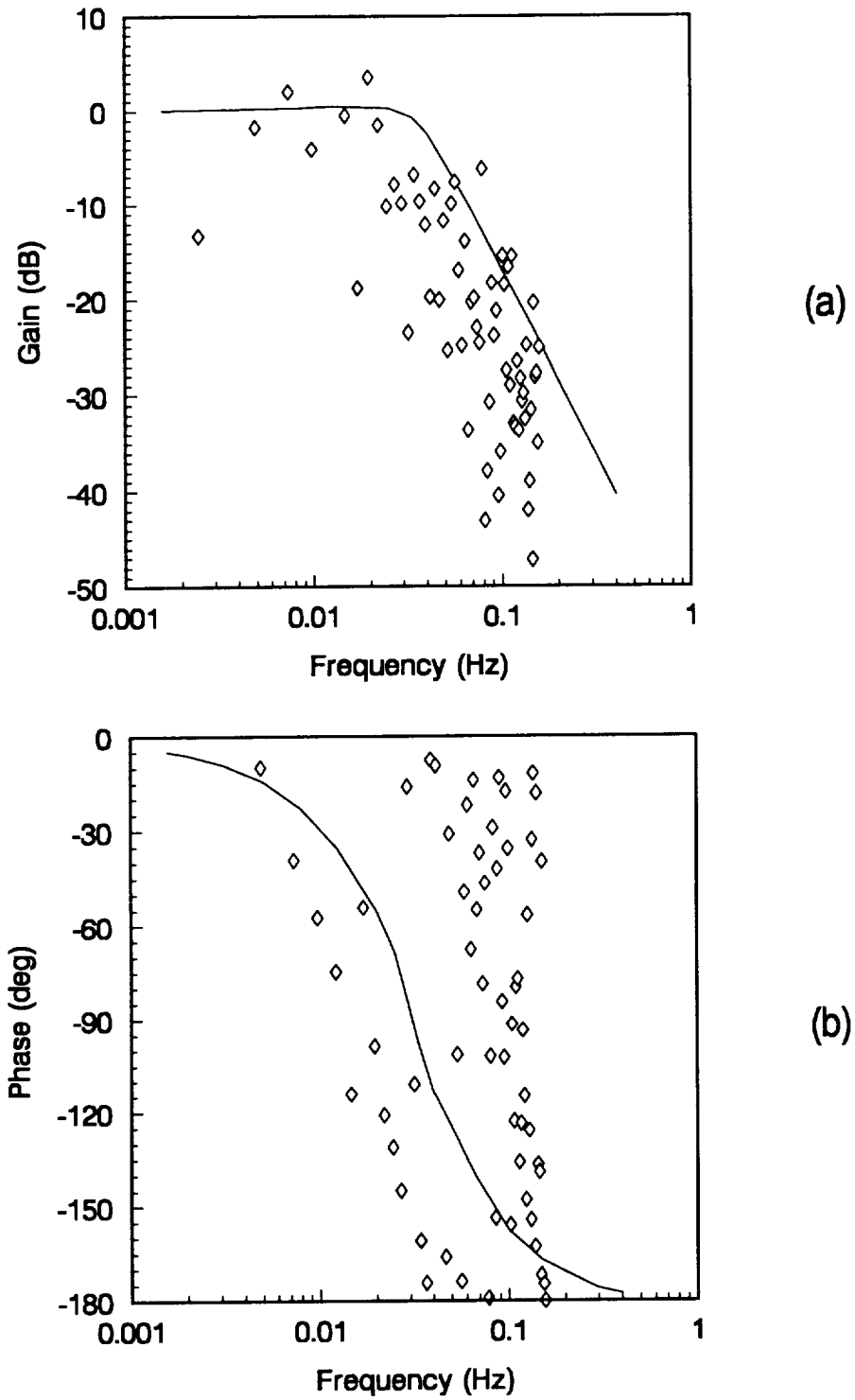


Fig. II.12 Bode Diagram for Simulated Solar Calibration Data of 12/28/1984 (with Hanning Window): (a) Gain, (b) Phase Angle.

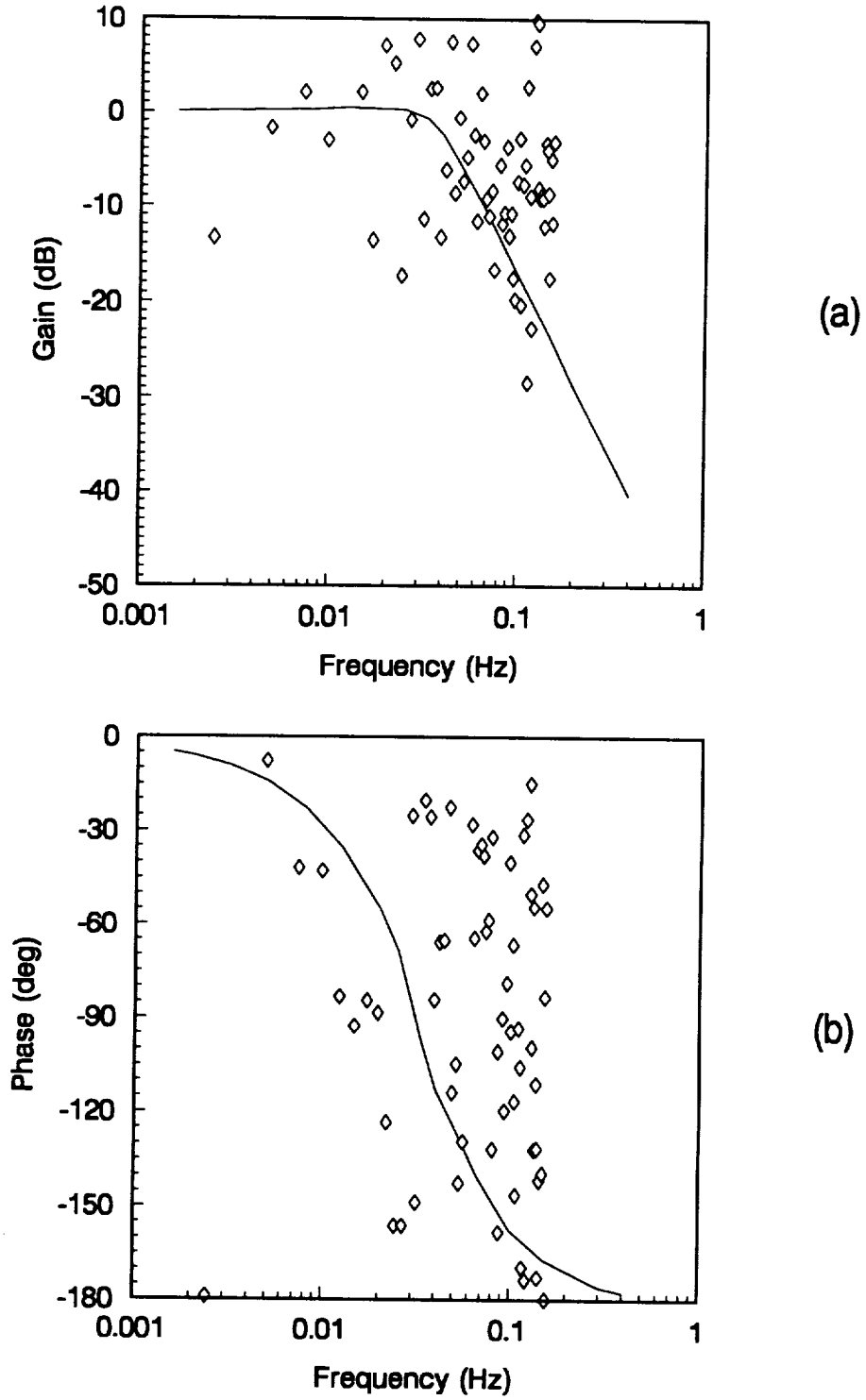


Fig. II.13 Bode Diagram for the Actual Solar Calibration Data of 12/28/1984 (with Hanning Window): (a) Gain, (b) Phase Angle.

5.0 Investigation of the Role of Field-of-View Limiters in Dynamic Calibration

As mentioned in Chapter 1, Part II, the objective here is to formulate a strategy for measuring the transfer function in space; that is, to develop an aperture system which permits in-flight dynamic calibration. It is demonstrated in Section 4.5 that accurate measurement of the transfer function requires an input signal whose total energy is more or less uniformly distributed over the frequency range of the instrument transfer function. This is further demonstrated in Section 4.6 where in-flight measurement of the ACR transfer function is shown to be very poor due to most of the signal energy being limited to a few low-frequency components. So it is concluded that there is a need to shift more of the energy of the signal power spectrum to higher frequencies or, better still, to the frequency band of interest.

The geometry of the field-of-view (FOV) limiter has a very significant effect on the form of the input signal used to measure the instrument frequency response. Then it is desired to design a FOV limiter having limited bandwidth, or a specified passband. Such a design can give an appreciable control of the measurement of the instrument transfer function. In this chapter the effect of the type of field-of-view limiter geometry on its

passband is studied. In particular, three classes of FOV limiter geometry are investigated: one consisting of a simple circular aperture, a second consisting of an array of equal-radius multiple apertures, and a third consisting of an array of varying-radius multiple apertures.

5.1 A Single Circular Aperture

The first case to be studied is a simple field-of-view (FOV) system where the secondary aperture is circular, as is the case with the ERBE nonscanning instruments. This geometry is shown in Fig. II.14. The radius of the primary aperture (cavity opening R_1) is fixed at 3.125 *mm*, and the secondary aperture radius, R_2, and the vertical spacing between apertures, H, are varied. Five cases are considered for five different values of R_2 and H, as shown in Table II.2. The first three cases correspond to actual ERBE channels configurations: the wide and medium field-of-view configured for Earth viewing and the wide field-of-view channel configured for solar calibration. The other two cases are fictitious.

For each case, the fast Fourier transform (FFT) of the input function, corresponding to the sun moving past the aperture system as in a solar calibration event, is computed. The illuminated area (the occlusion of the two ellipses shown in the Fig I.4) as a function of incident angle is computed. The results are shown in Figs. II.15 through II.19 for the five cases. One observation to be made here is that the highest-amplitude frequency components shift to higher frequencies as the FOV gets narrower; that is, as the radius

of the secondary aperture, R_2, decreases (going from Case 1 to Case 3 and from Case 4 to Case 5). This means that the instrument response has to be faster for narrower field-of-view instruments. However, the most important observation is that the energy of the input signal is concentrated in the first one or two frequency components instead of being spread more evenly across the spectrum.

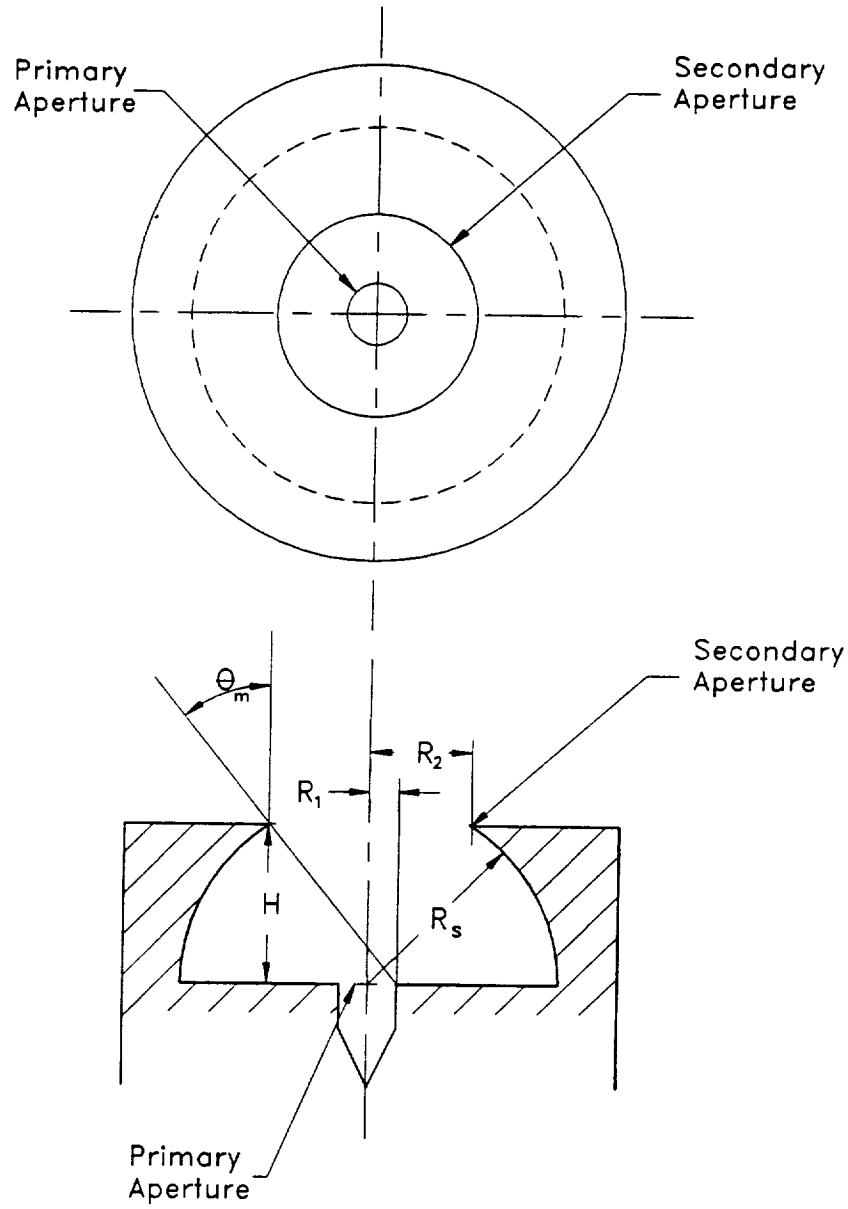


Fig. II.14 A Simple Circular Aperture FOV Limiter.

Table II.2 Single Circular Aperture Parameters

Case No.	R_2 (mm)	H (mm)	R_s (mm)	θ_{max} (deg)	ERBE Channel
1	34.595	12.700	36.855	71.40	WFOV
2	24.892	28.829	38.100	44.23	MFOV
3	8.712	76.760	-	8.80	WFOVsc
4	15.000	35.023	38.100	27.43	-
5	5.000	37.770	38.100	12.21	-

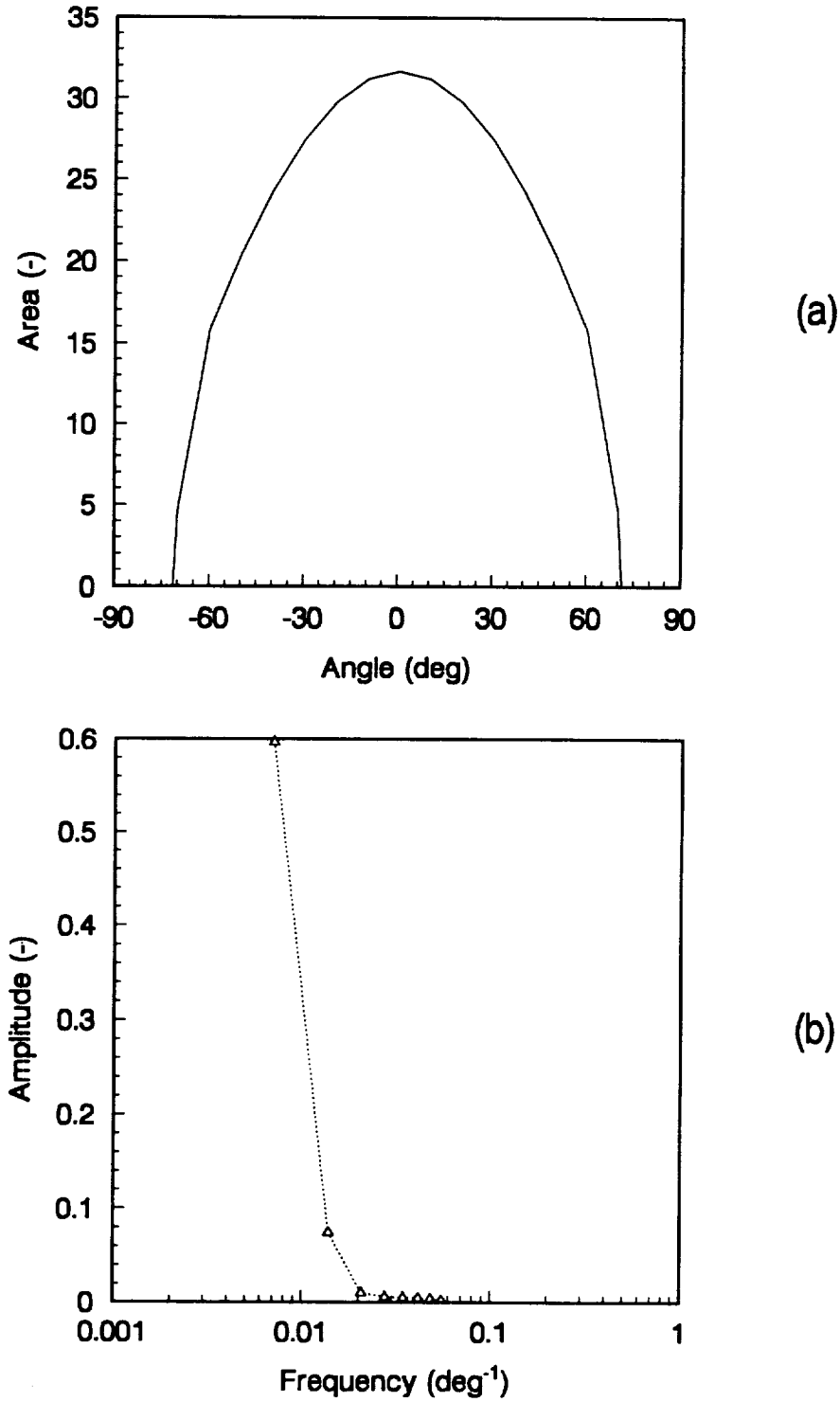
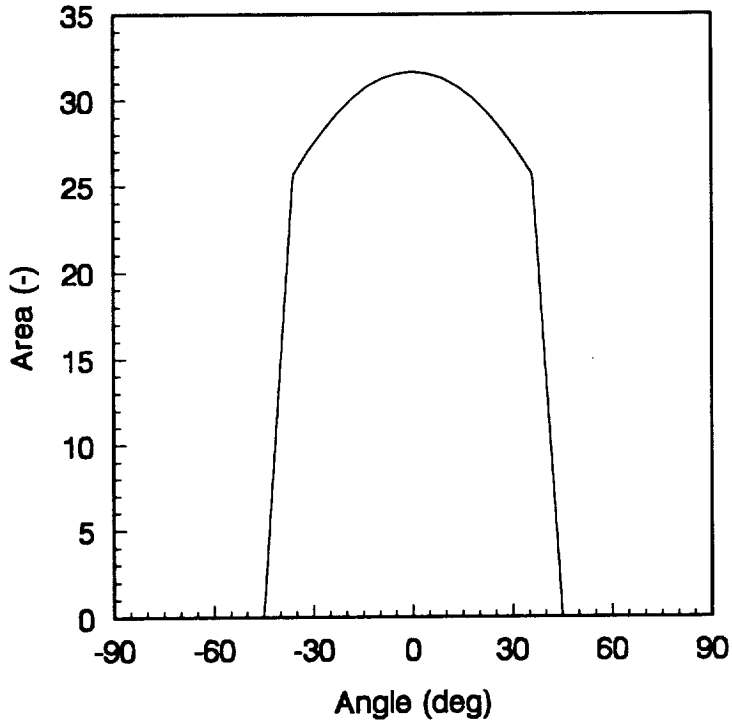
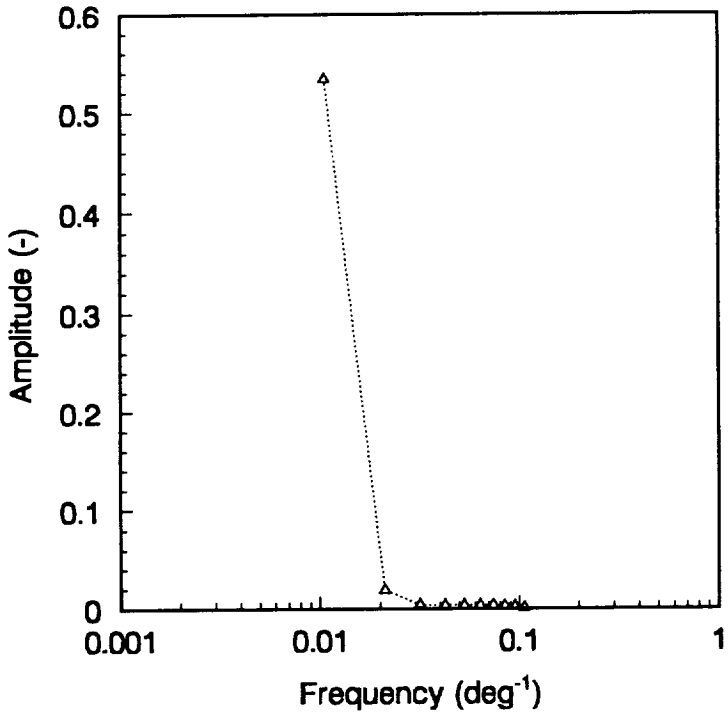


Fig. II.15 Case 1, Table II.2: (a) Input Function, (b) Corresponding Frequency Spectrum.



(a)



(b)

Fig. II.16 Case 2, Table II.2: (a) Input Function, (b) Corresponding Frequency Spectrum.

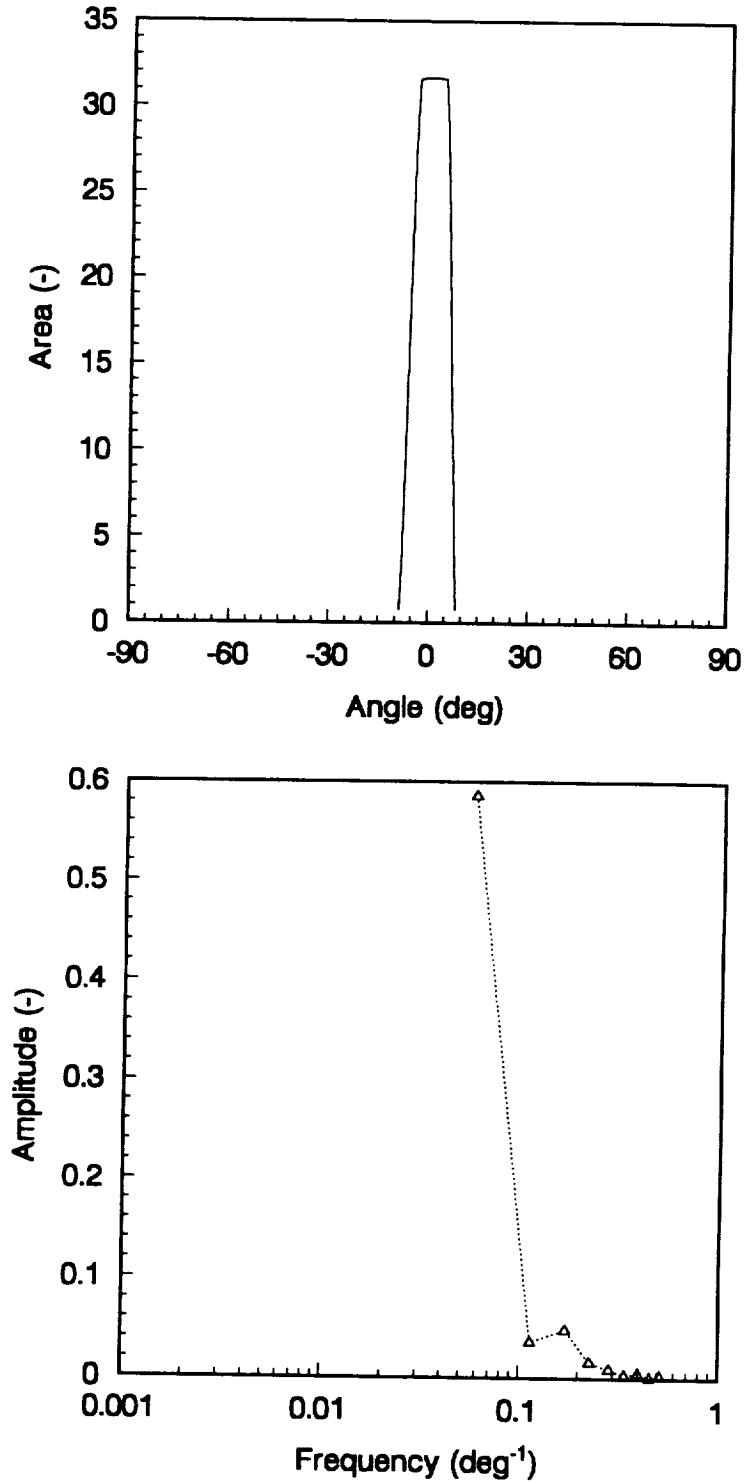
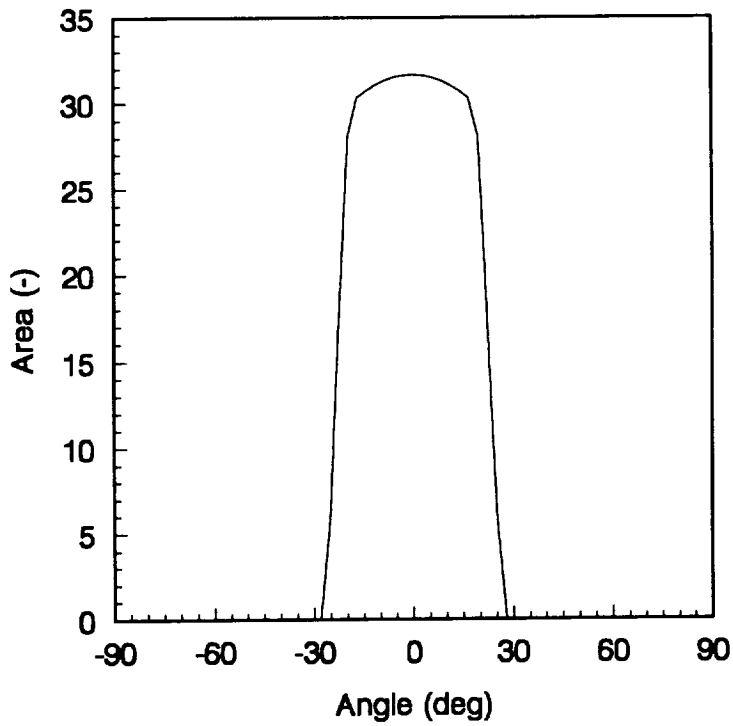
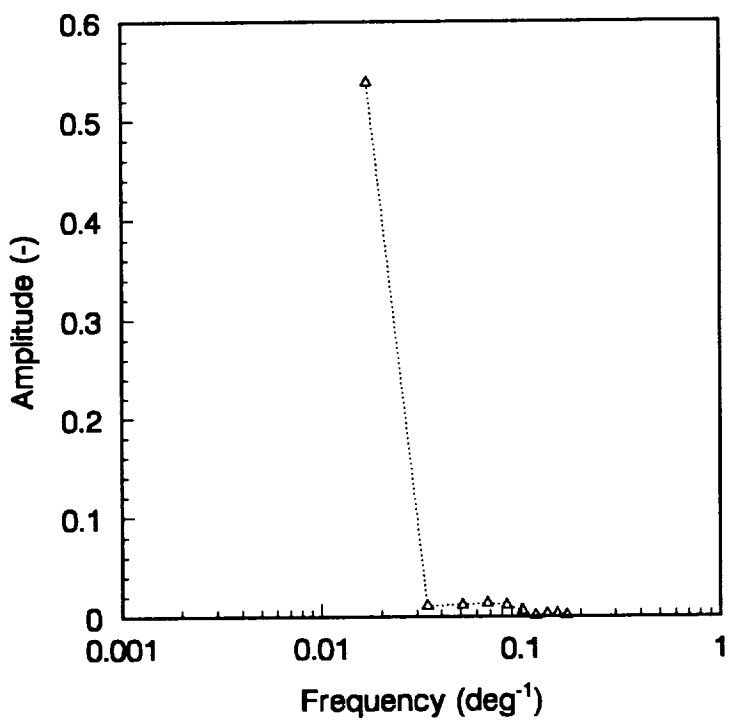


Fig. II.17 Case 3, Table II.2: (a) Input Function, (b) Corresponding Frequency Spectrum.

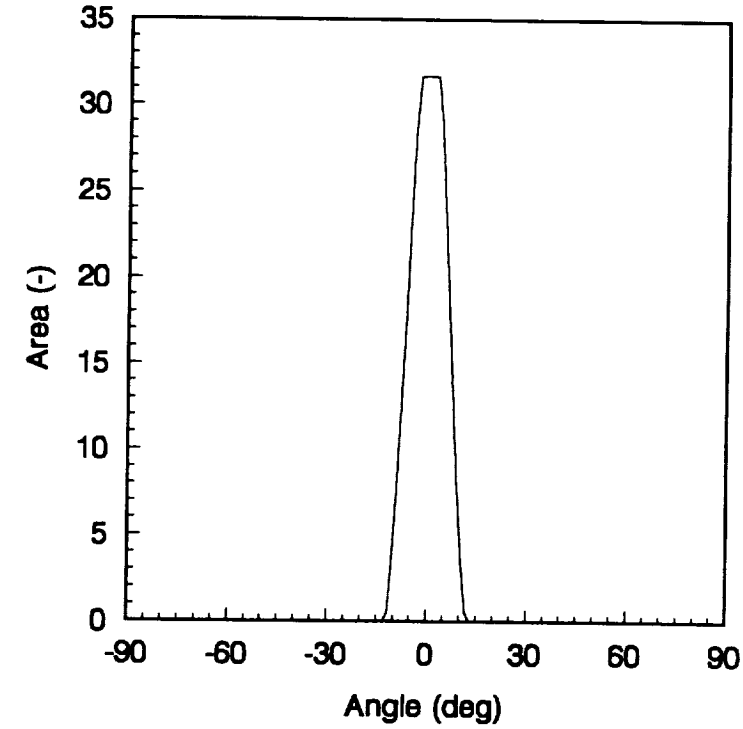


(a)

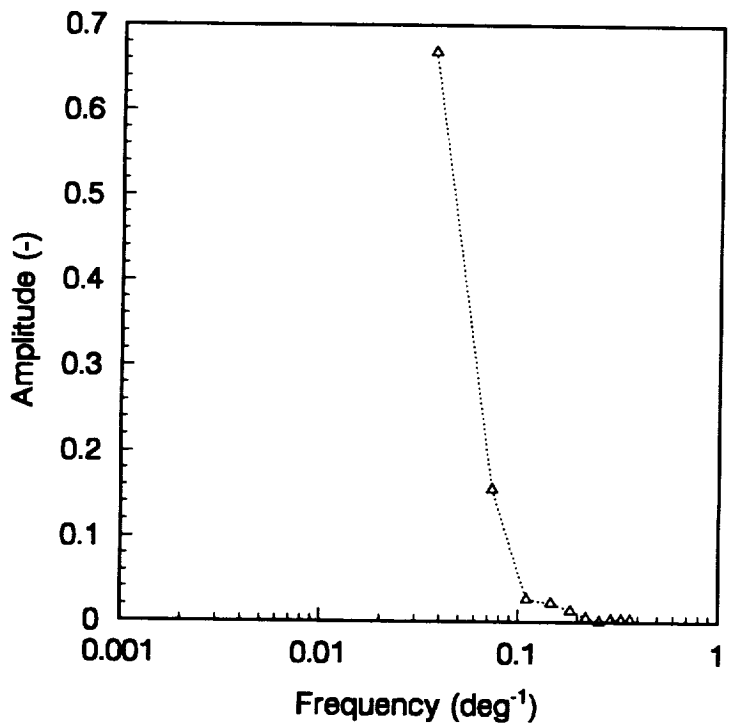


(b)

Fig. II.18 Case 4, Table II.2: (a) Input Function, (b) Corresponding Frequency Spectrum.



(a)



(b)

Fig. II.19 Case 5, Table II.2: (a) Input Function, (b) Corresponding Frequency Spectrum.

5.2 An Array of Equal-Radius Circular Apertures

The next class of secondary apertures studied consists of an array of circular apertures, as shown in Fig. II.20. In the remainder of this part, we speak of circular apertures even though in practice these apertures are truncated as indicated by the cross-hatched areas in the figure. The cross-hatched areas of the apertures, if they were cut out, would not alter the amount of radiation entering the primary aperture. If not cut out they actually reduce stray light, or thermal noise; hence, the truncation is preferred.

First, consider the case where all apertures are of equal radius R and spacing S. The radii of the primary (sensor) and secondary field-of-view limiting apertures are one and two units, respectively, and the height H between them is ten units. The fixed spacing S between centers of the adjacent secondary apertures is six units. The input function, which is the variation of projected area on the sensor as a function of angle of incidence, is shown in Fig. II.21(a). The number of secondary apertures is set according to the maximum angle θ_{max} beyond which no more direct radiation can enter the primary aperture. For example, for the current case with the parameters given above, this maximum angle θ_{max} is 80 deg and the number of secondary apertures is fifteen. A FORTRAN code that performs the computations of θ_{max} and the number of secondary apertures for given values of R_0 and H is given in Appendix B.

Note that the magnitude and the width of each lobe in Fig. II.21 decrease with increasing incident angle with respect to the normal, following the cosine law for projected area. The effect of this array of apertures is to distribute more of the energy in the input

signal to higher frequencies as shown in Fig. II.21(b). When this frequency spectrum is compared to those of Figs. II.15 through II.19 the differences are obvious. Secondary peaks (0.03 to 0.08 deg^{-1}) are now present whose amplitudes are between 50 and 70 percent of the amplitude of the fundamental frequency component (0.006 deg^{-1}). It is also important to note that the frequency of the first peak corresponds to the overall sample period, the frequency of the second peak corresponds to the period of the center lobe (centered at 0 deg), and so forth.

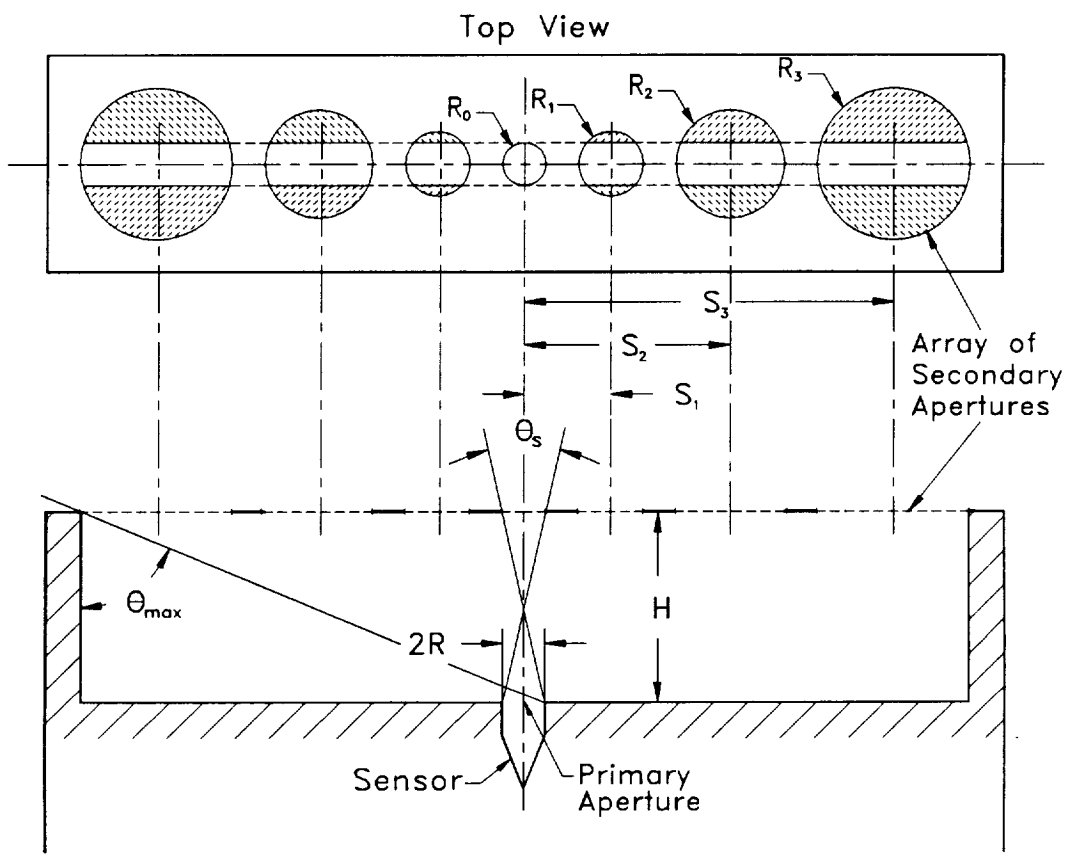


Fig. II.20 A FOV Limiter Consisting of an Array of Circular Secondary Apertures.

5.3 An Array of Varying-Radius Circular Apertures

The secondary aperture array investigated in this section is similar to that of the previous section, except that now we allow the secondary aperture radii R and the array spacing S to increase in order to keep the width of each lobe in the time series constant. Figure II.20 shows the geometry for this case, where θ_s is now maintained constant. A FORTRAN code that calculates the radii and spacing for a particular geometry is given in Appendix B.

We now analyze several cases where the primary aperture R_s is fixed at one unit and the parameters R_0, S and H are varied. The five cases studied are shown in Table II.3.

Figure II.22 shows the input function and the corresponding frequency spectrum for Case 1. There are nine lobes in the input function. Inspection of the frequency spectrum in Fig. II.22(b) reveals that the first peak (0.006 deg^{-1}), which corresponds to the overall period of the calibration event, is not dominant in this case. The highest peak is rather the second one (0.06 deg^{-1}), which corresponds to the period of the central lobe (16.6 *deg*) of Fig. II.22(a). The third peak is at a frequency (0.12 deg^{-1}), which is twice that of the second, and which corresponds to the width of two lobes. The fourth peak is at three times the frequency of the second, and so forth. This pattern continues for integer multiples of the frequency corresponding to the period of the central lobe (0.06

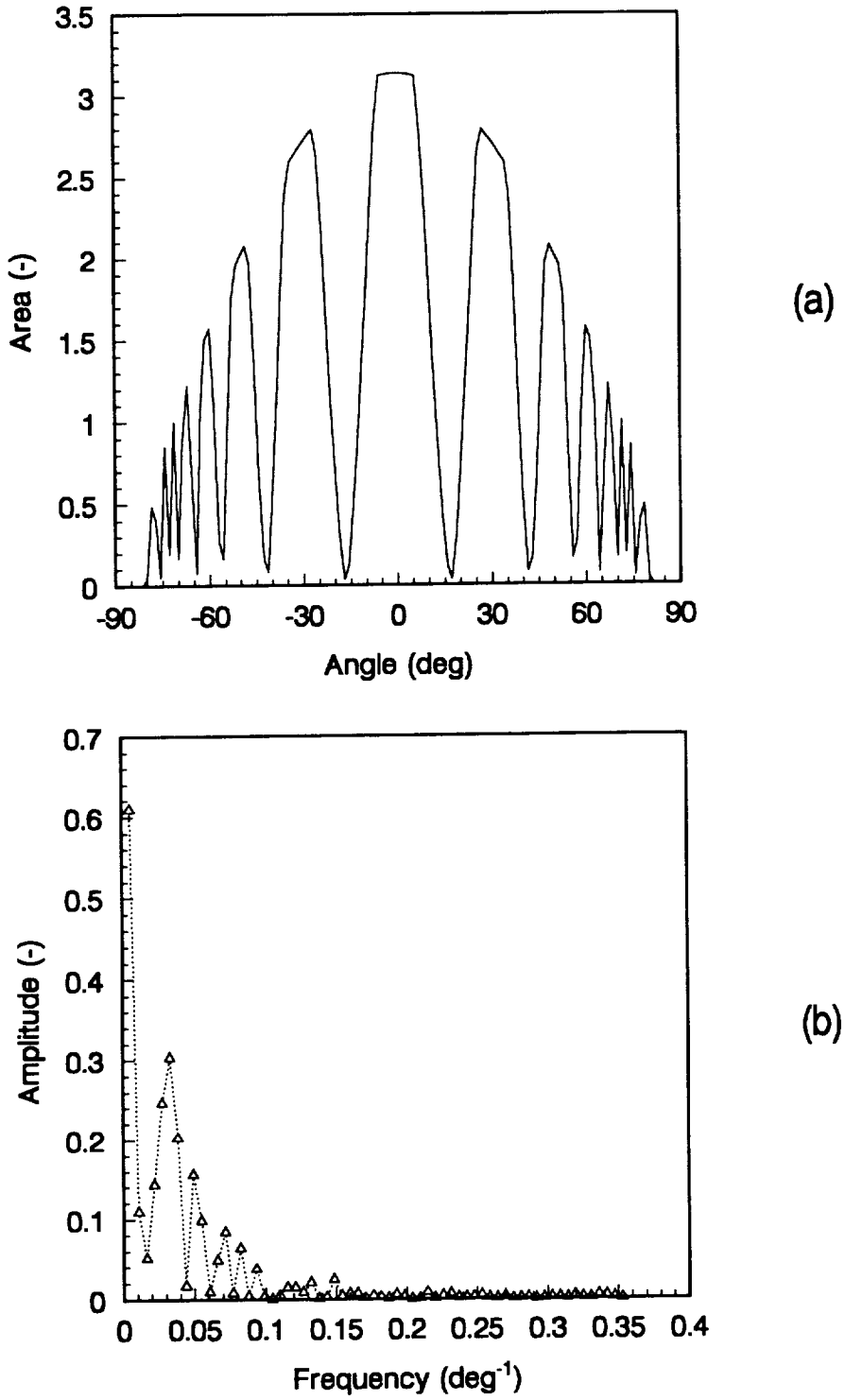


Fig. II.21 An Array of Equal-Radius Secondary Apertures: (a) Input Function, (b) Corresponding Frequency Spectrum.

deg^{-1}).

In Case 2, Fig. II.23, reducing the radius of the central aperture of the array, R_s, by half causes the lobes of the input function to be sharper and narrower than those of Case 1. Furthermore, the total angular sweep has increased slightly, and the number of lobes is now fifteen. The frequency spectrum of Fig. II.23(b) is very similar to that of Fig. II.22(b), except that the third peak is not quite as high. Also, the entire spectrum has shifted to higher frequencies so that the peak corresponding to the central lobe is now at 0.08 deg^{-1}.

It is appropriate to ask if such a field-of-view limiting system is practical. The secondary aperture was allowed to increase in radius, without limit, but up to the maximum incident angle θ_{max}. In Case 2, the largest secondary aperture has a radius R_7 of 95 units and the array has a total width of 524 units. This means that if the primary aperture is 1 cm in radius, then the total width of the array will be 5.24 meters! Such a FOV limiter is obviously impractical. This encourages us to study the effect on the frequency spectrum of reducing the number of apertures in the array. Figure II.24 shows the input function and the corresponding frequency spectrum of Case 2, but without the largest secondary apertures (on each end of the array). That is, the input function now (Case 2(a)) has only 13 lobes, as shown in Fig. II.24(a). The only significant difference between this spectrum and that of Fig. II.23(b) is the attenuation of the first peak; otherwise, the two spectra are essentially the same.

When an additional lobe is eliminated from each end of the input function, leaving a total of 11 lobes (Case 2(b)), a similar spectrum is obtain, as shown in Fig. II.25(b). The first peak, however, has decreased even further in amplitude. Finally, a further decrease

of the number of input function lobes to nine (Case 2(c)), leads to a deterioration in the spectral resolution at higher frequencies, as seen in Fig. II.26(b). Here spectral resolution is taken to mean the measure of how clearly one can distinguish between different peaks in the frequency spectrum [27]. For example, the highest-amplitude peak (at $f \approx 0.08$ deg^{-1}) of Fig. II.23(b) is somewhat more clearly defined than the corresponding peak of Fig. II.26(b).

The study involving elimination of some of the apertures on the ends of the array leads to the conclusion that reducing the number of secondary apertures, up to a point, does not seriously degrade the favorable characteristics of the spectrum, which are well defined high-amplitude, higher-order peaks. In fact, this actually enhances the resolution and reduces the contribution of the lowest frequency component to the signal up to a point. Furthermore, it is worth noting that for the 9-lobe input function, the total width of the array is only 48 units, which makes this a practical design.

Now consider Case 3 of Table II.3, where the height H between the primary and the secondary apertures is increased but where the secondary aperture radius remains the same as in Case 2. This leads to more lobes (23) as shown in Fig. II.27(a). Consequently, there are higher frequency components of significant amplitude in Fig. II.27(b) compared with previous spectra. The highest frequency components which rise above the noise level correspond to a frequency of about 0.65 deg^{-1}.

In Case 4, the radius of the central secondary aperture R_0 is increased to 5 units. This FOV geometry gives only seven lobes, as shown in Fig. II.28(a). Its corresponding frequency spectrum (Fig. II.28(b)) reveals a regular pattern of decreasing amplitude peaks spaced at integer multiples of 0.04 deg^{-1} up to a frequency of about 0.22 deg^{-1}.

Case 5 is similar to Case 3, except that the height H is doubled to 60 units. Fig. II.29(a) shows the corresponding input function having 15 lobes. The corresponding spectrum, Fig. II.29(b), shows nine distinct peaks extending to a frequency of 0.7 deg^{-1}. This frequency spectrum has a similar pattern as that of Case 4, but with better spectral resolution; i.e, more sharply defined peaks.

Table II.3 Array of Varying-Radius Circular Apertures.

Case No.	R_0 (-)	H (-)	Number of secondary apertures	θ_{max} (deg)	R_{max} (-)	Overall width (-)
1	2	20	9	76.78	24.43	168
2	1	20	15	85.66	95.30	524
2(a)	1	20	13	74.24	14.96	140
2(b)	1	20	11	62.82	5.95	76
2(c)	1	20	9	51.40	3.14	48
3	1	30	23	87.72	290.46	1507
4	5	30	7	79.17	54.70	311
5	5	60	15	85.66	287.91	1578

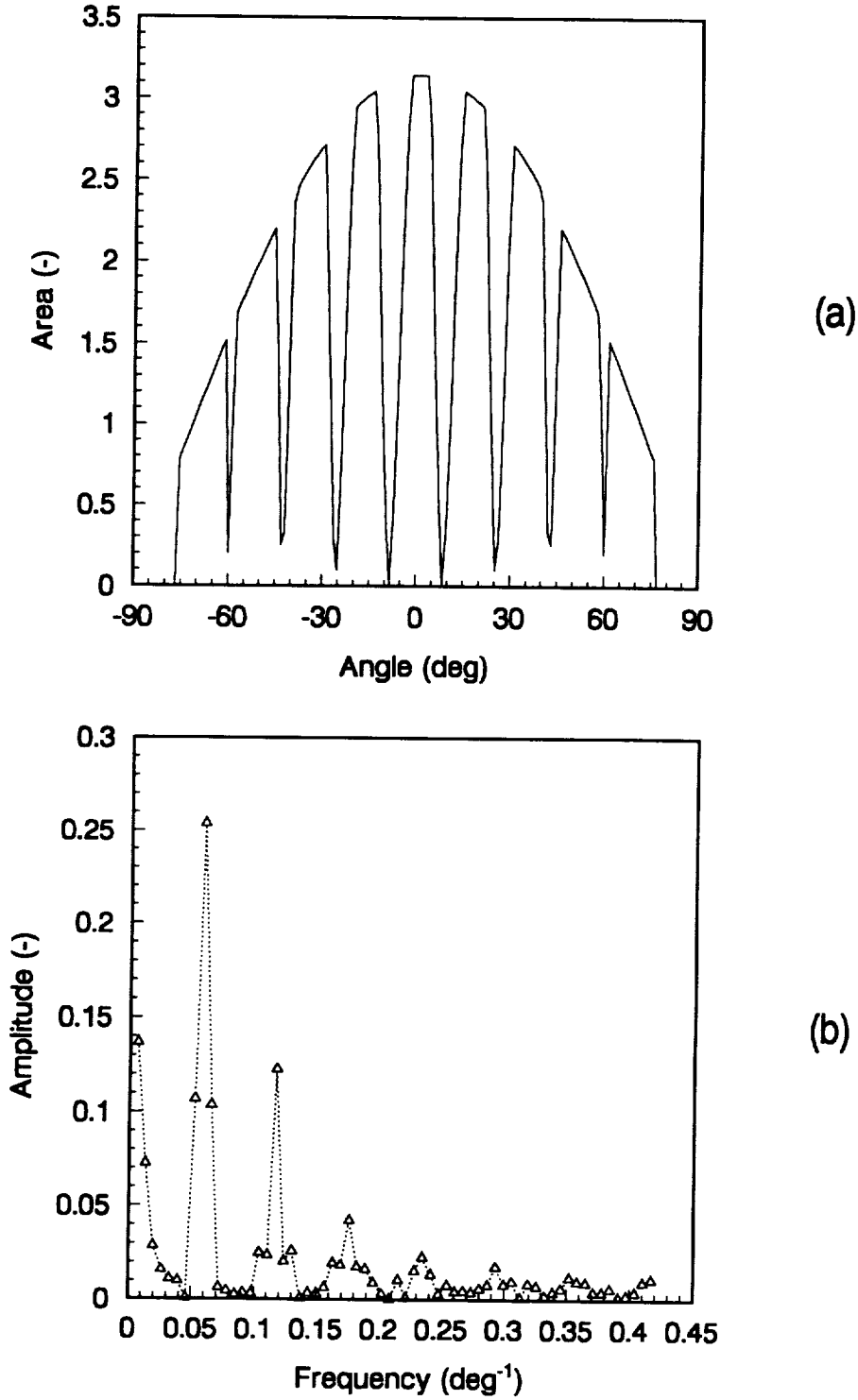


Fig. II.22 Case 1, Table II.3: (a) Input Function, (b) Corresponding Frequency Spectrum.

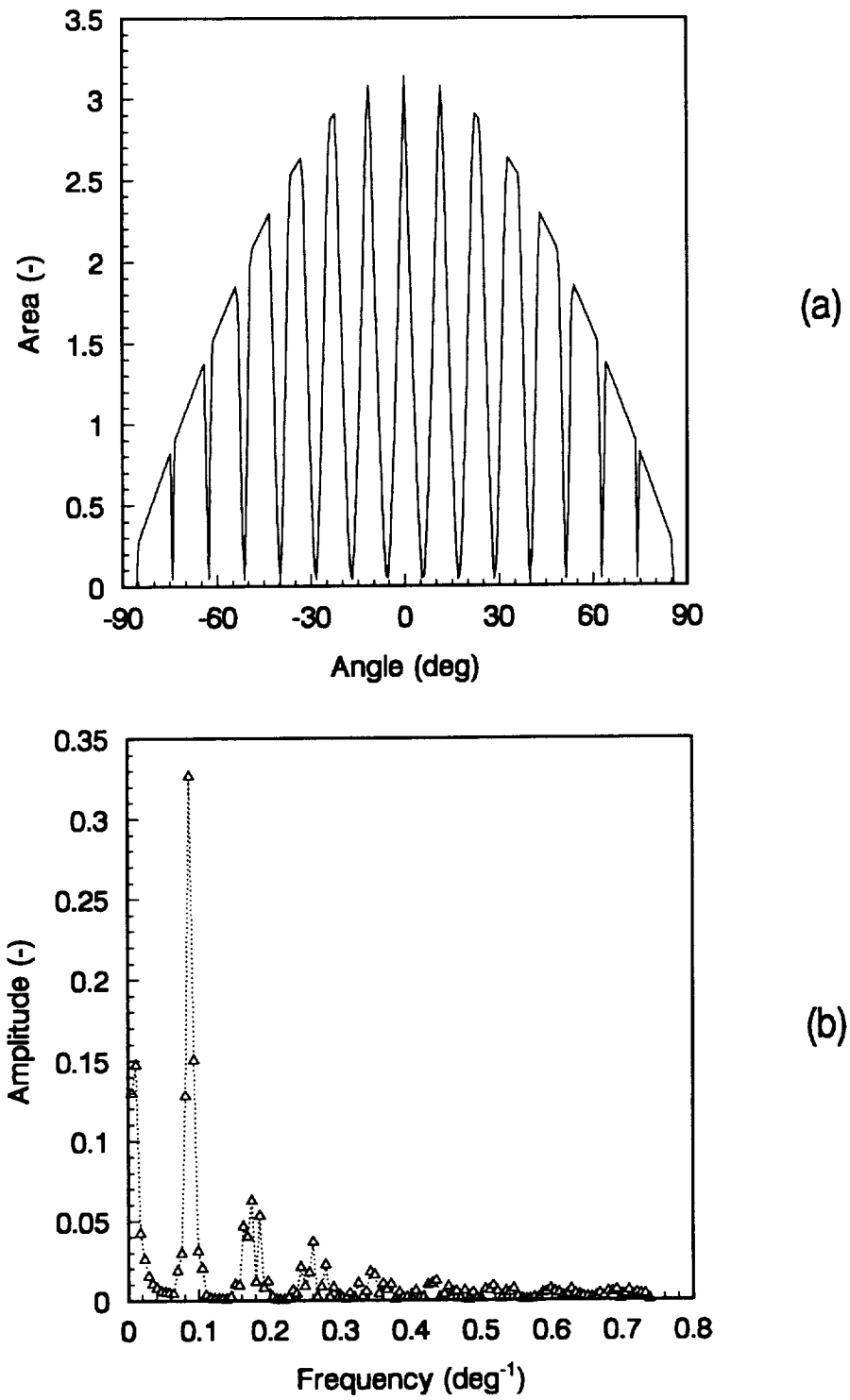


Fig. II.23 Case 2, Table II.3: (a) Input Function, (b) Corresponding Frequency Spectrum.

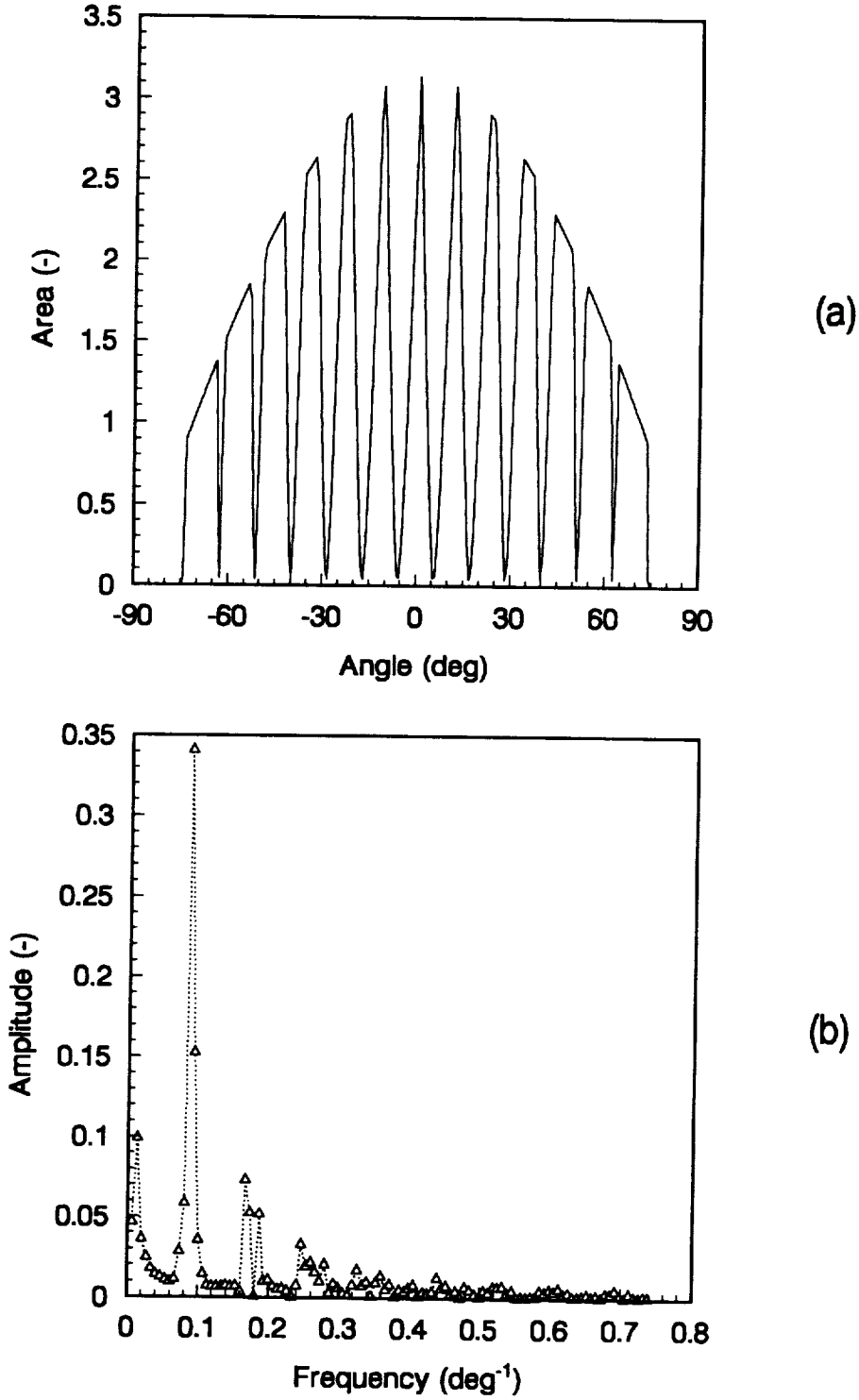


Fig. II.24 Case 2(a), Table II.3: (a) Input Function, (b) Corresponding Frequency Spectrum.

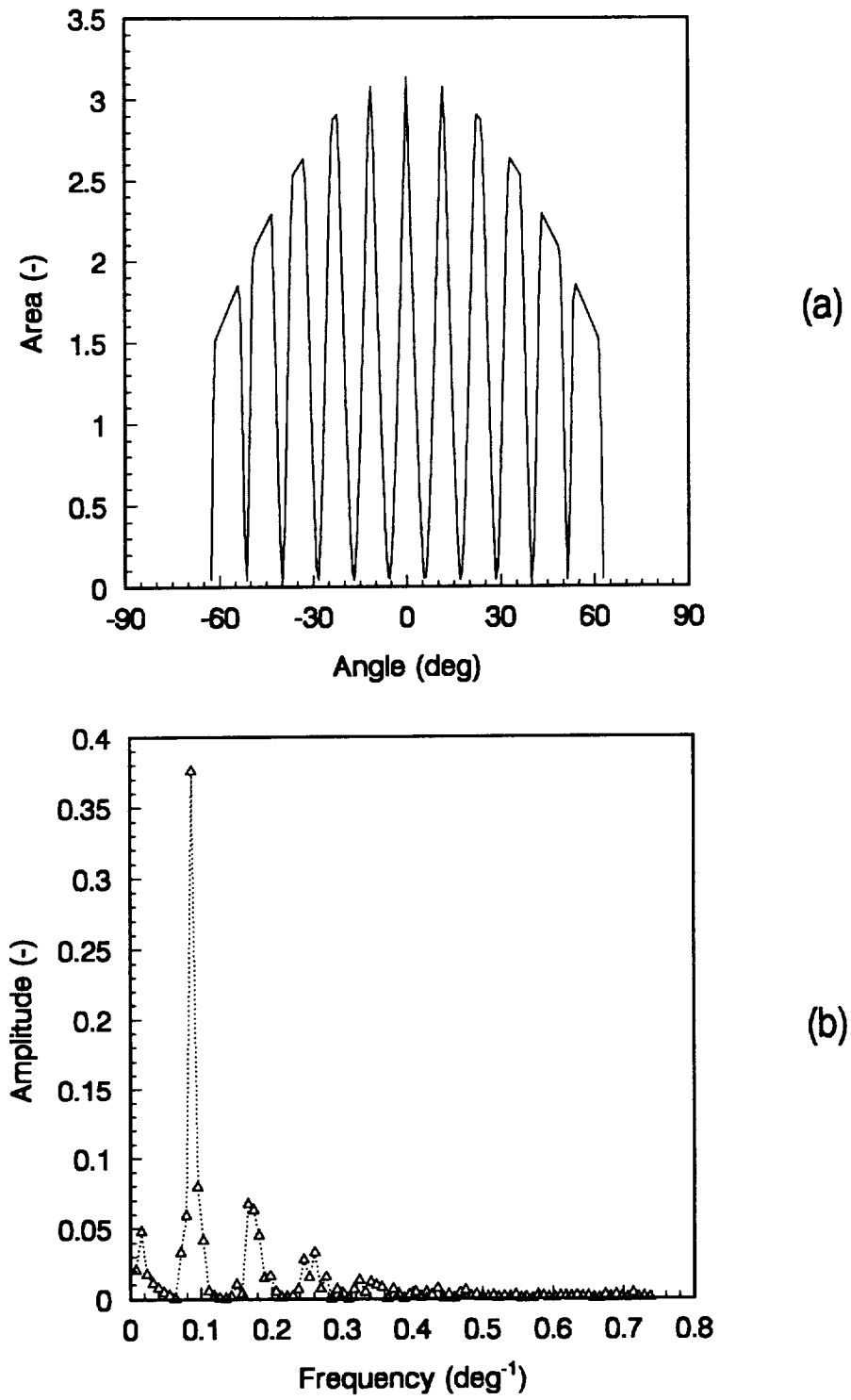
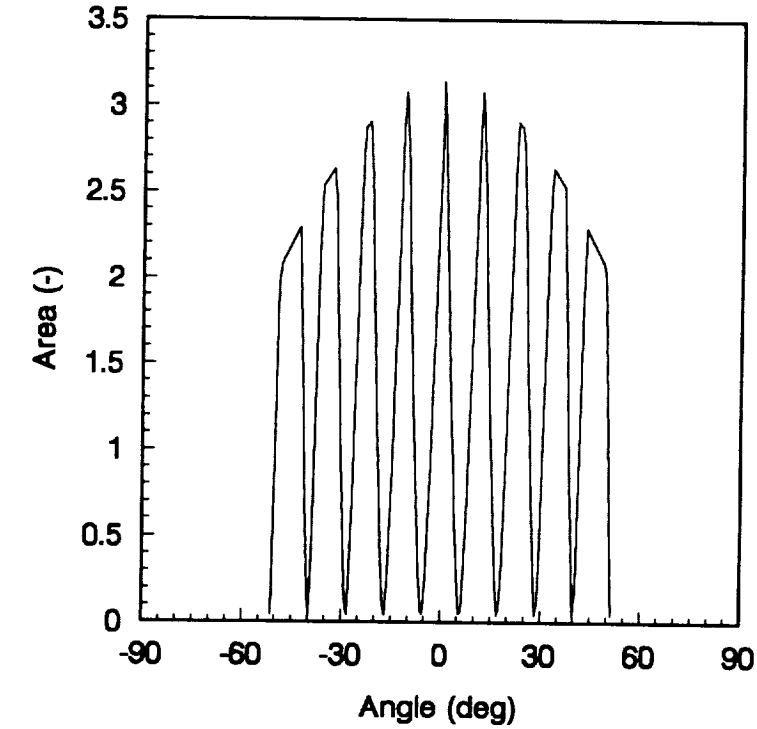
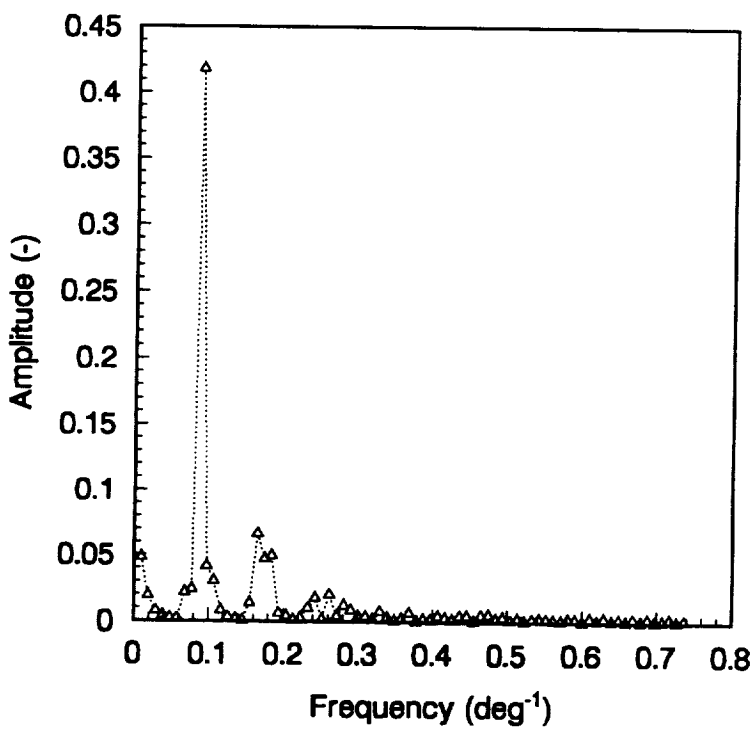


Fig. II.25 Case 2(b), Table II.3: (b) Input Function, (b) Corresponding Frequency Spectrum.



(a)



(b)

Fig. II.26 Case 2(c), Table II.3: (a) Input Function, (b) Corresponding Frequency Spectrum.

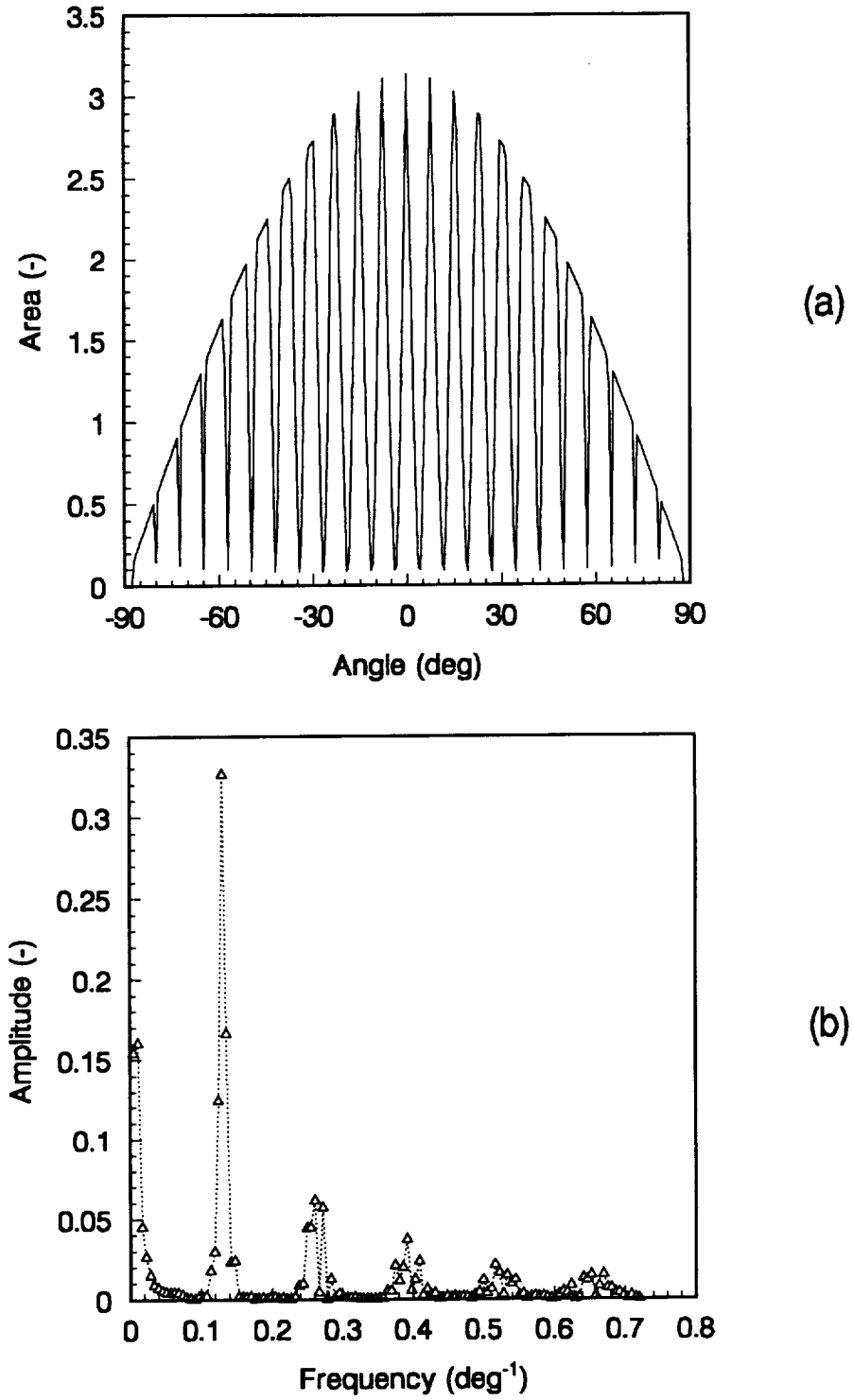


Fig. II.27 Case 3, Table II.3: (a) Input Function, (b) Frequency Spectrum.

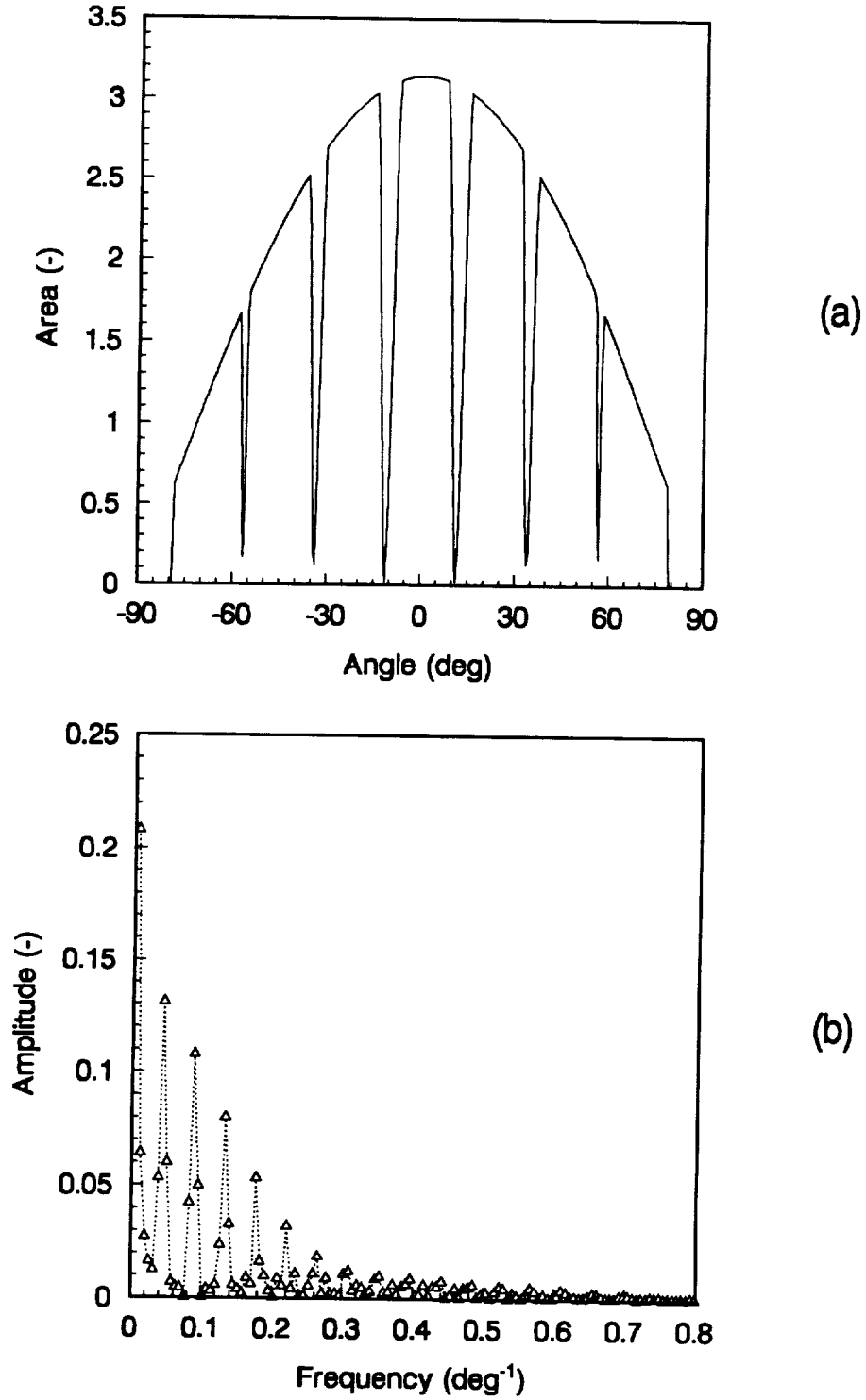


Fig. II.28 Case 4, Table II.3: (a) Input Function, (b) Frequency Spectrum.

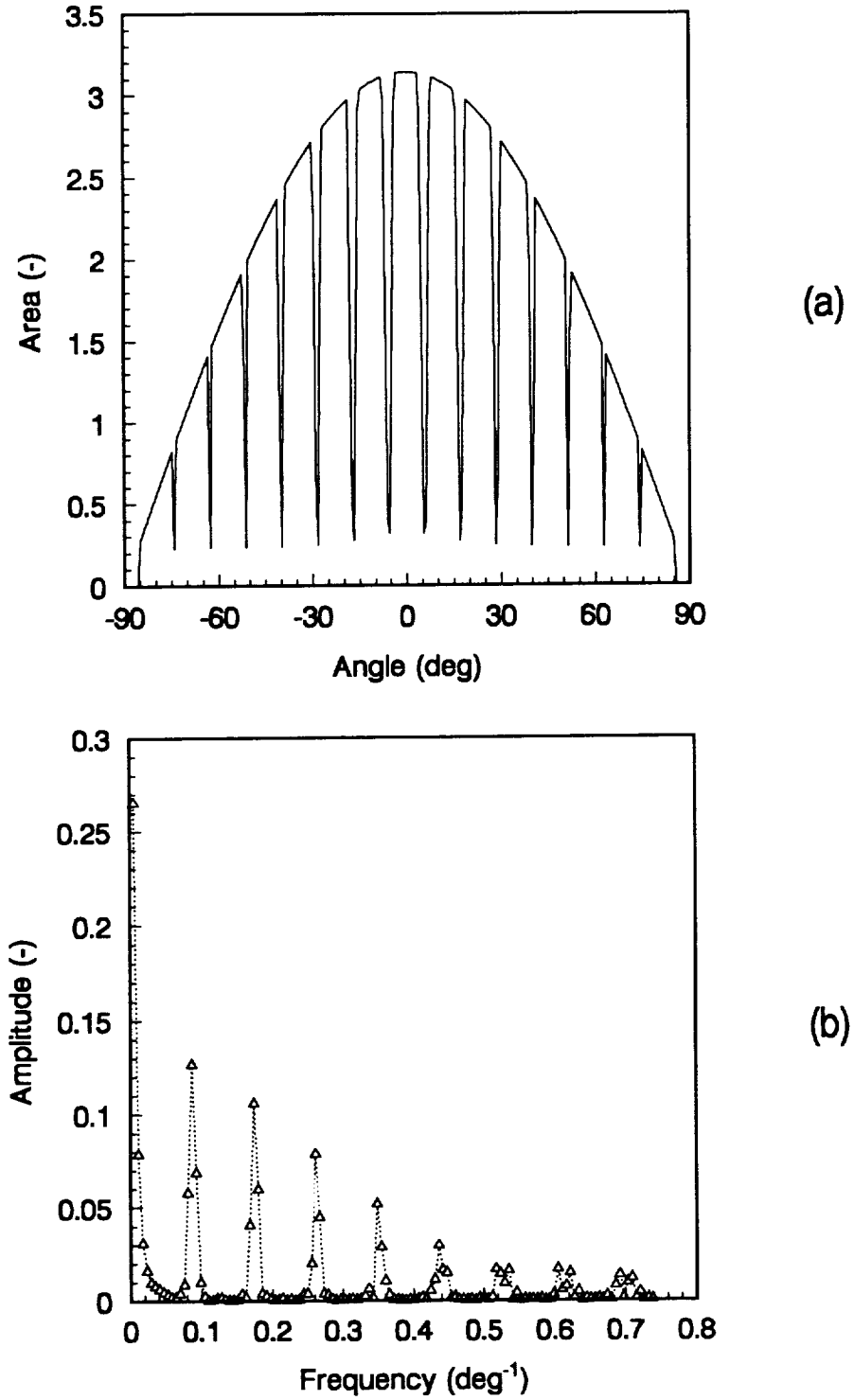


Fig. II.29 Case 5, Table II.3: (a) Input Function, (b) Frequency Spectrum.

5.4 Study of the Transfer Functions Obtained Using Array Apertures

Cases 2(a), 2(c) and 5 of Section 5.3 (Table II.3) are chosen for further analysis of their possible use in the in-flight measurement of the ACR transfer function. Case 2(a) is chosen as the baseline case and Case 2(c) is chosen because it is a practical design. Case 5 is chosen because it produces the most promising power spectrum. The input functions are transformed from angular domains ranging from -75 to 75 *deg*, from -63 to 63 *deg* and from 86 to -86 *deg* to time domains of 223 *s*, 193 *s* and 256 *s*, respectively, for these three cases. The number of samples is set equal to the length of the time domain in seconds, giving one sample per second. Furthermore, the area is transformed into heat input, scaled appropriately for the ERBE nonscanner wide field-of-view (WFOV) channel. The resulting heat input (W/m^2) as a function of time is shown in Figs. II.30(a) and II.31(a) for the first two cases. These functions, consisting of 13 and 9 lobes (Case 2(a) and Case 2(c)), are then introduced into the dynamic electrothermal model described in Part I. The input and output functions, as well as the corresponding frequency spectra, obtained using a fast Fourier transform (FFT) analysis, are shown in Figs. II.30 and II.31 for the Cases 2(a) and 2(c). A horizontal line is drawn to separate frequencies with high energy amplitudes. The corresponding transfer functions (gain and phase angle) are shown in Figs. II.32 and II.33, respectively. The darkened symbols correspond to frequencies with amplitudes above the noise level. Also shown in these figures are the theoretical channel gain and phase shift (solid lines) obtained in Chapter 4 of Part II. The

agreement between the theoretical and the "measured" transfer functions is quite good, especially the gains, showing a tremendous improvement over the cases involving simulated and actual solar calibration events reported in Chapter 4 (Figs. II.10 through II.13). It is important to recall that Hanning windowing has been used in this analysis also.

For Case 5 of Table II.3, it is observed that several peaks are present in the frequency spectrum, as shown in Fig. II.34(b). The corresponding transfer function is shown in Fig. II.35. When this figure is compared to those in Figs. II.32 and II.33, it is clear that an even better agreement with the theoretical transfer function, especially at high frequencies, is obtained. Unfortunately, the dimensions of this aperture array make it impractical for the current application.

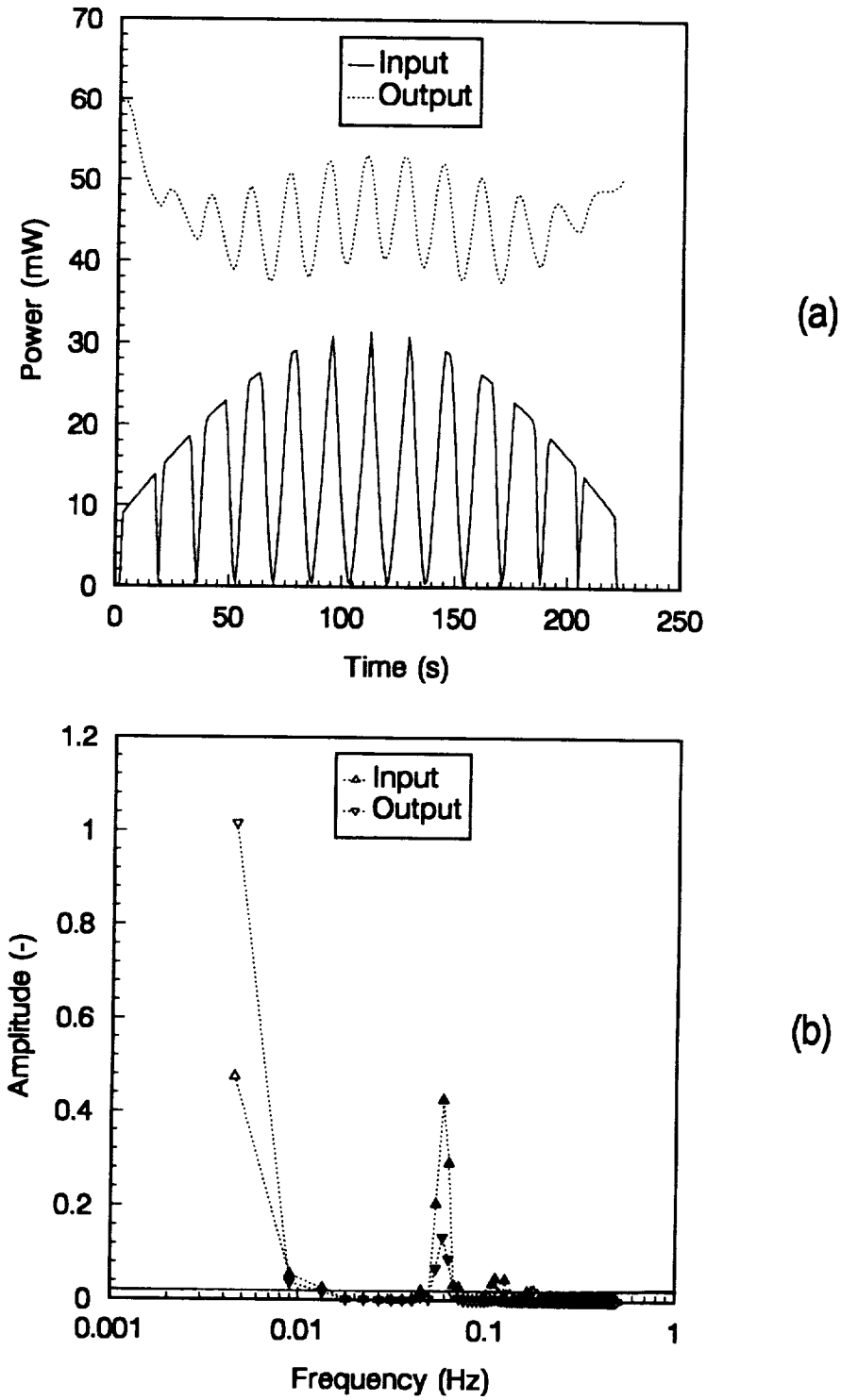


Fig. II.30 Case 2(a), Table II.3: (a) Input and output Functions, (b) Corresponding Frequency Spectra.

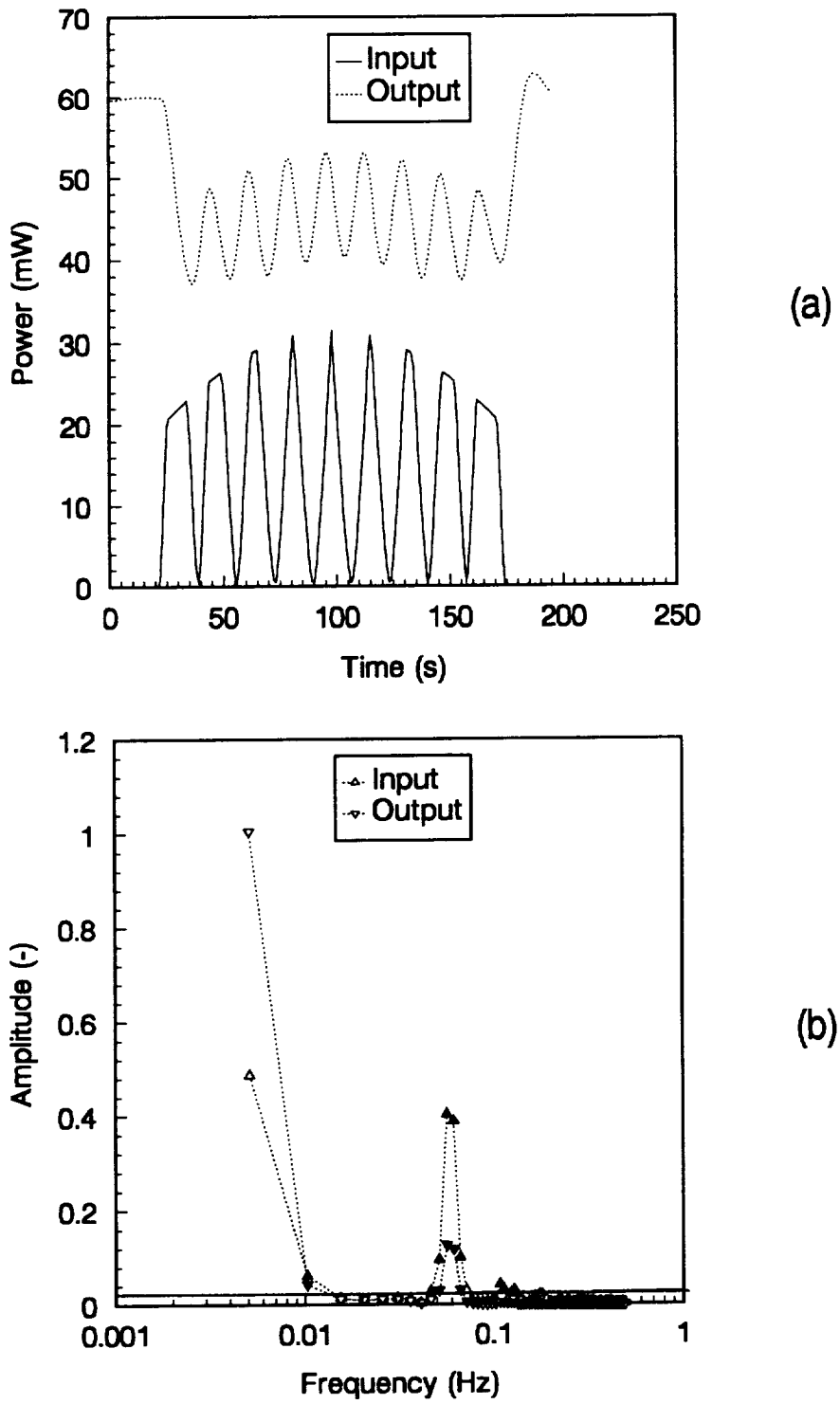


Fig. II.31 Case 2(c), Table II.3: (a) Input and output Functions, (b) Corresponding Frequency Spectra.

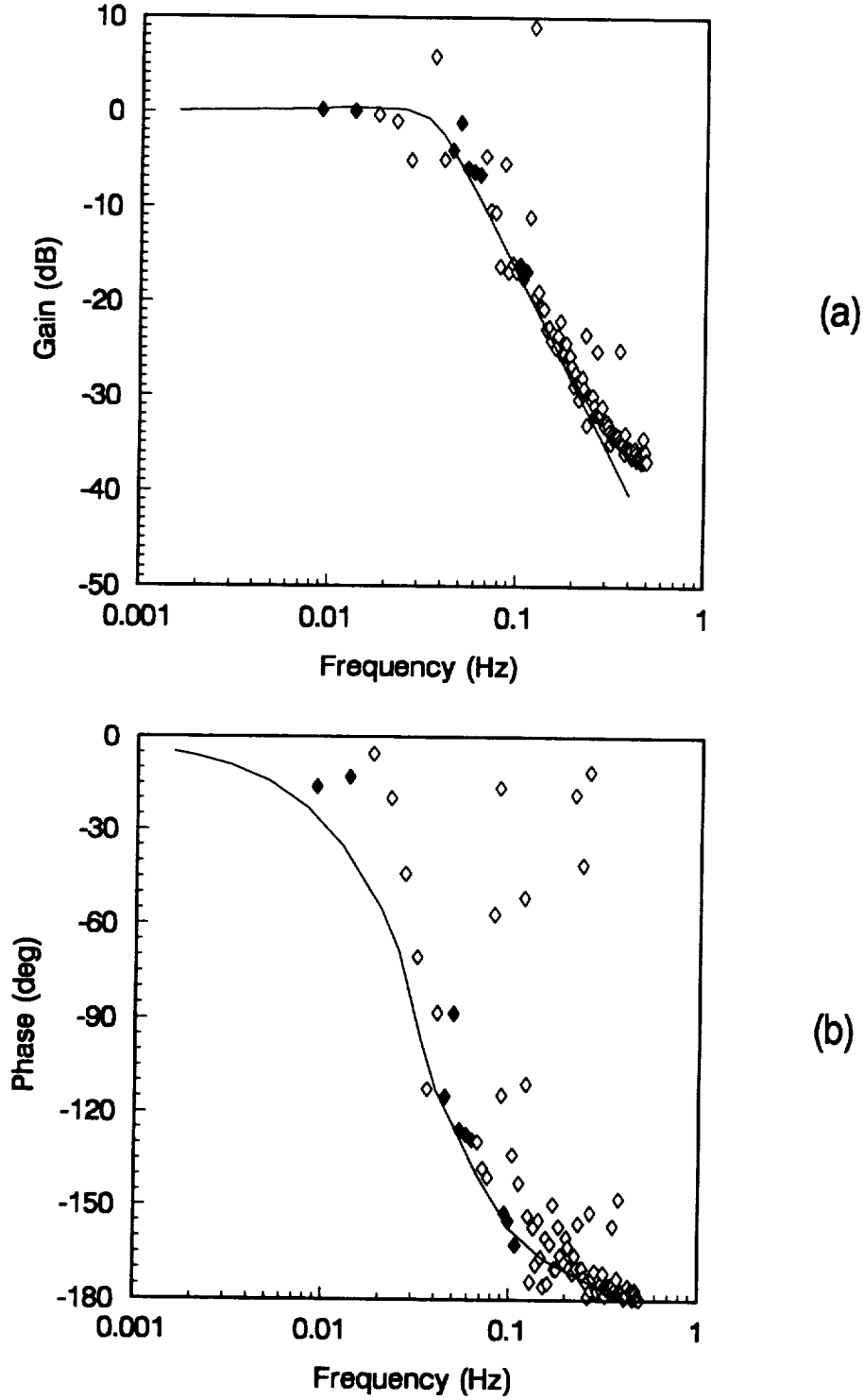


Fig. II.32 Bode Diagram for Case 2(a), Table II.3: (a) Gain, (b) Phase Angle.

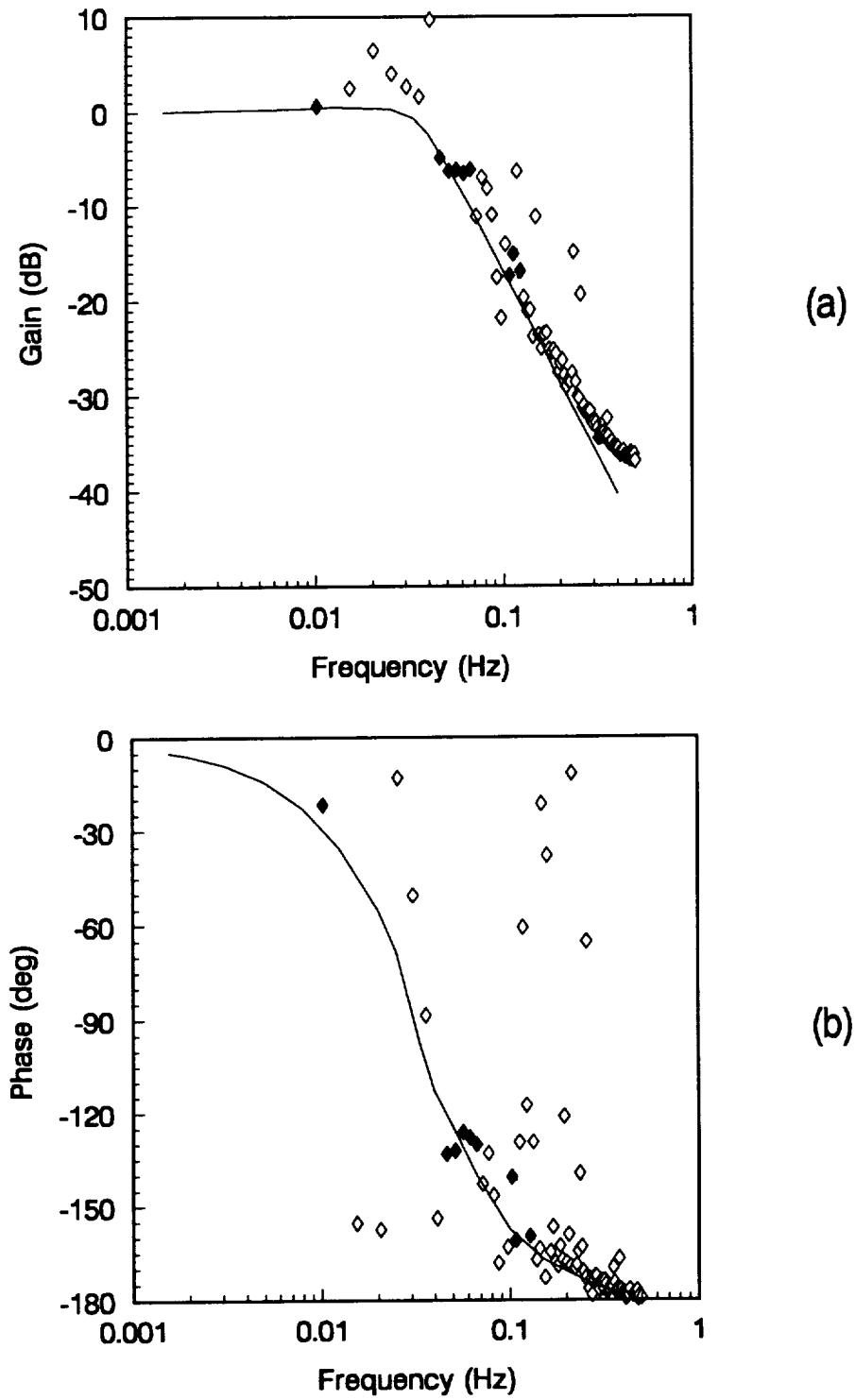


Fig. II.33 Bode Diagram for Case 2(c), Table II.3: (a) Gain, (b) Phase Angle.

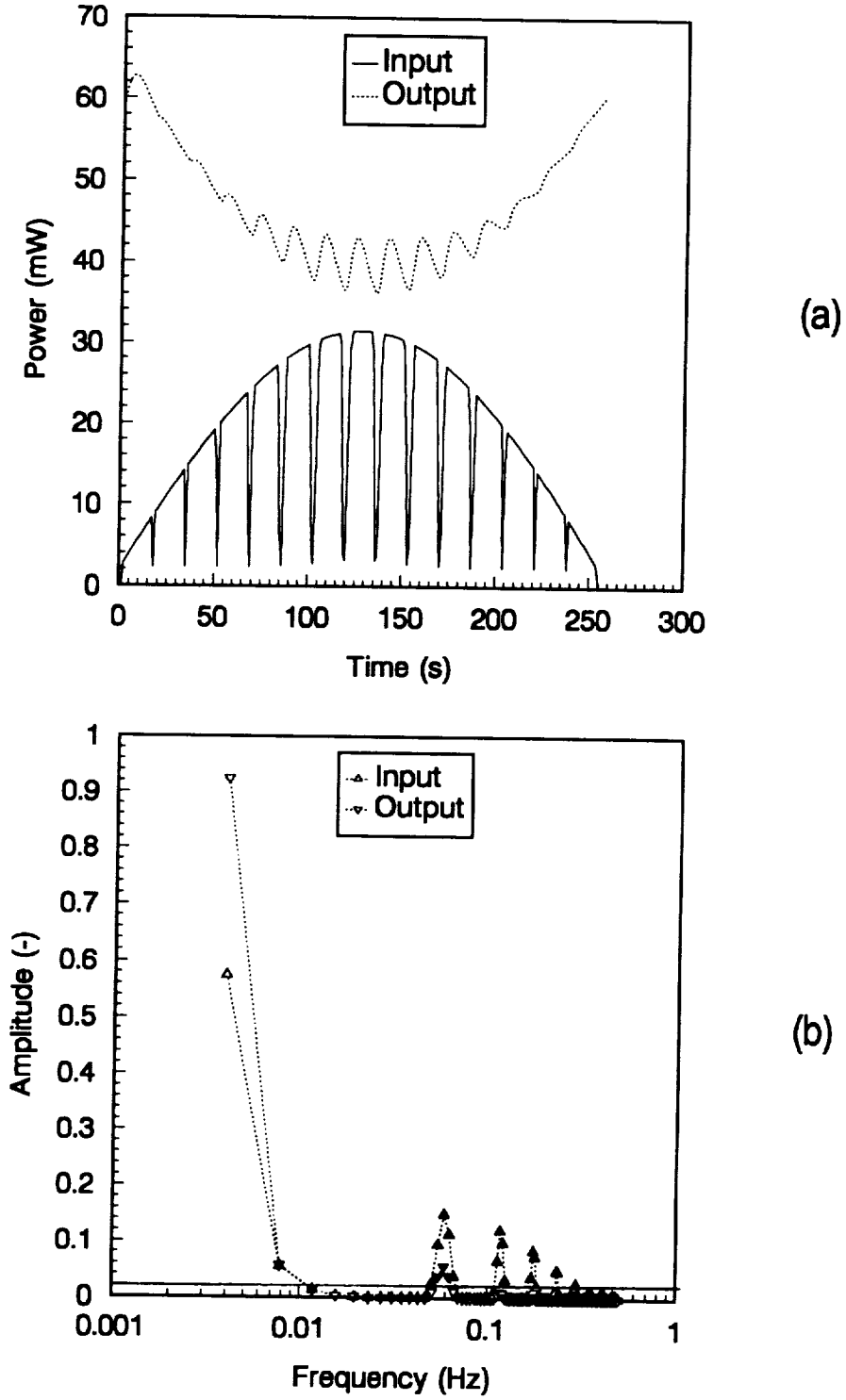


Fig. II.34 Case 5, Table II.3: (a) Input and output Functions, (b) Corresponding Frequency Spectra.

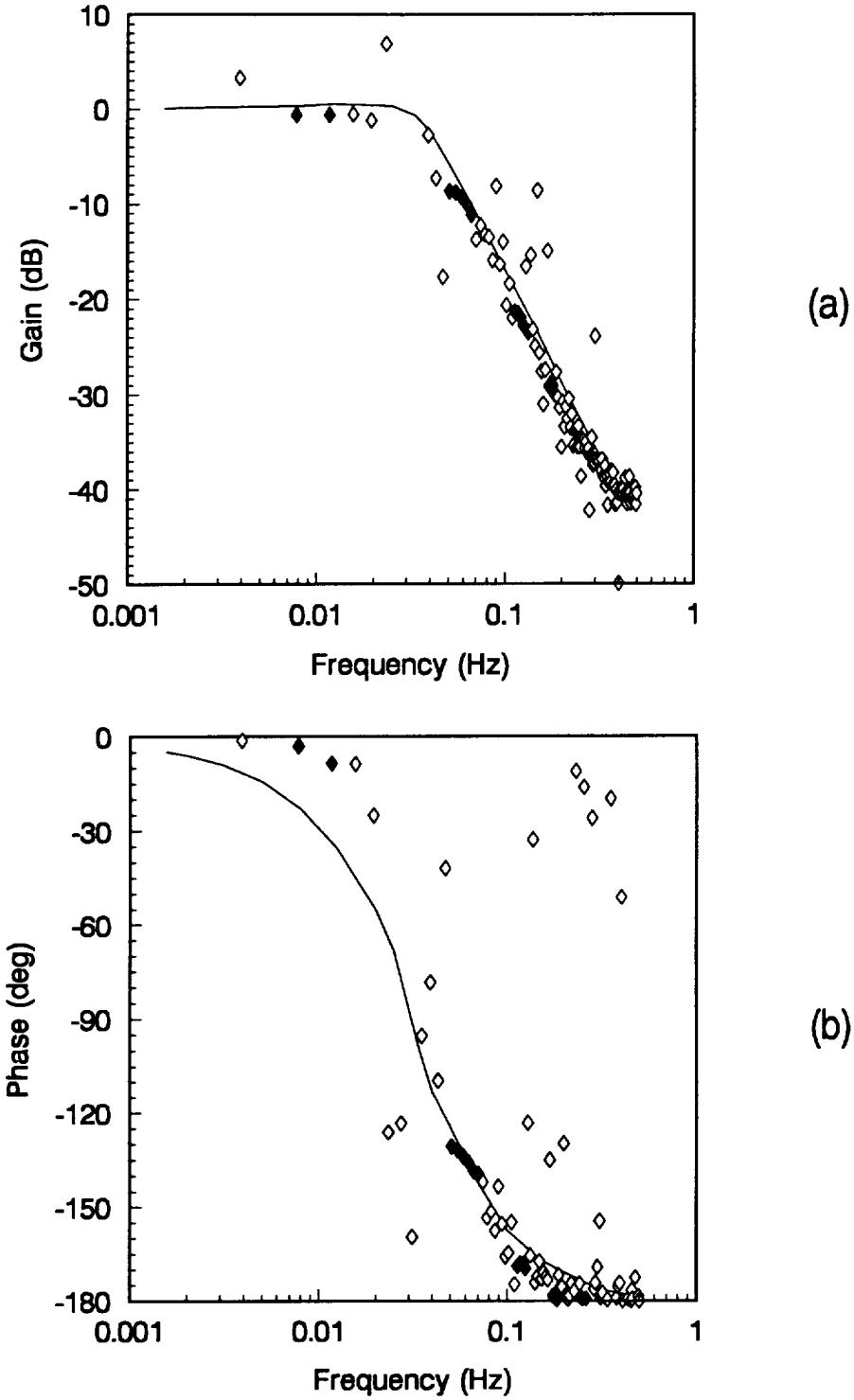


Fig. II.35 Bode Diagram for Case 5, Table II.3: (a) Gain, (b) Phase Angle.

6.0 Concluding Remarks

The followings conclusions follow from the present research:

1. The transfer function of the ERBE total WFOV nonscanning instrument has been accurately computed based on the channel dynamic electrothermal model.

2. In flight Measurement of the transfer function of a radiometric channel based on solar calibration input functions improves when the spectral energy of the corresponding frequency spectra is more evenly distributed across the wavelength interval of interest.

3. An array of secondary apertures replacing the solar port leads to a more favorable spectral distribution of energy for the purposes of dynamic calibrations.

4. The thermal time constant of the ERBE total WFOV nonscanning instrument is between 3.5 and 4 s, depending on the definition adopted for the cut-off frequency.

5. Hanning windowing has been verified to be an effective means for reducing the errors in the spectral analyses leading to definition of the transfer function.

6. Truncated circular holes suffice in the design of the secondary aperture (mentioned in Conclusion 3) consisting of an array of increasing diameter circles. Not only will this geometry reduce thermal noise, but it will also make the aperture array smaller and thus more practical.

7. For the case considered in Conclusion 6, there is no need to include a full set of apertures up to the maximum angle of sweep. Essentially the same spectrum can be obtained by eliminating a few of the larger outlying apertures. This may be necessary to render the FOV limiter practical.

The following recommendations may be drawn from the present research:

1. Use the model with specified variations in the radiometer thermo-physical properties to study the sensitivity of one or more candidate array apertures to changes in these properties.

2. A filter dome should be added to the ACR model so that its effect on thermal noise and the transfer function can be evaluated.

PART III

A STUDY OF THE SCANNER POINT SPREAD FUNCTION

1.0 Introduction

1.1 Scanner Optical Design

As mentioned in Section 0.1.2, each of the Earth Radiation Budget Experiment (ERBE) satellites has three narrow field-of-view (NFOV) scanning channels: a shortwave (0.2 - 5.0 μm) channel, a longwave (5.0 - 50.0 μm) channel and a total (0.2 - ≥50.0 μm) channel. The ERBE scanner optics, shown in Figs. III.1 and III.2, are nearly identical for each channel. The only difference is that the shortwave channel has two Superasil-W1 filters and the longwave channel has a multicoated-diamond substrate filter as shown in Fig. III.1 [8].

The scanner optics consist of a Cassegrain-type telescope [39]. This telescope employs a hyperbolic primary mirror and a parabolic secondary mirror to focus radiation onto a sensor. The mirrors are positioned so that the virtual hyperbolic focus coincides with the real parabolic focus. The sensor, referred to as the "active flake" in Figs. III.1 and III.2, is a thin film thermistor bolometer.

It is placed at or close to the real focal point of the hyperbolic primary mirror.

The thermistor bolometer is made of a semiconductor material composed of a

mixture of manganese, cobalt and nickel. The operating principle of this type of detector is based on the change of its resistance with temperature when it is exposed to thermal radiation. This change of resistance is converted to an electrical signal by a bridge circuit. The main advantage of a thin film thermistor bolometer is its fast time response compared to other types of detectors such as active cavity radiometers discussed in Parts I and II [38,39]. Because these channels scan back and forth across the orbital path they must respond much faster than the nonscanner instrument investigated in Parts I and II.

In the ERBE design the active flake is placed at 1.22 mm behind the field stop for the shortwave and longwave channels and 1.07 mm behind the field stop for the total channel (See Fig. III.3). The active flake whose area is 4.0 mm^2, and the field stop are projected on the same plane in Fig. III.4. The field stop is a 1.84 x 1.23 mm-hexagon, which translates to a 4.5 x 3.0 deg field-of-view (FOV). For the Clouds and Earth's Radiant Energy System (CERES) scanner instrument the preliminary dimensions are 1.5 x 0.75 mm (2.8 x 1.12 deg). CERES instrument dimensions are preliminary.

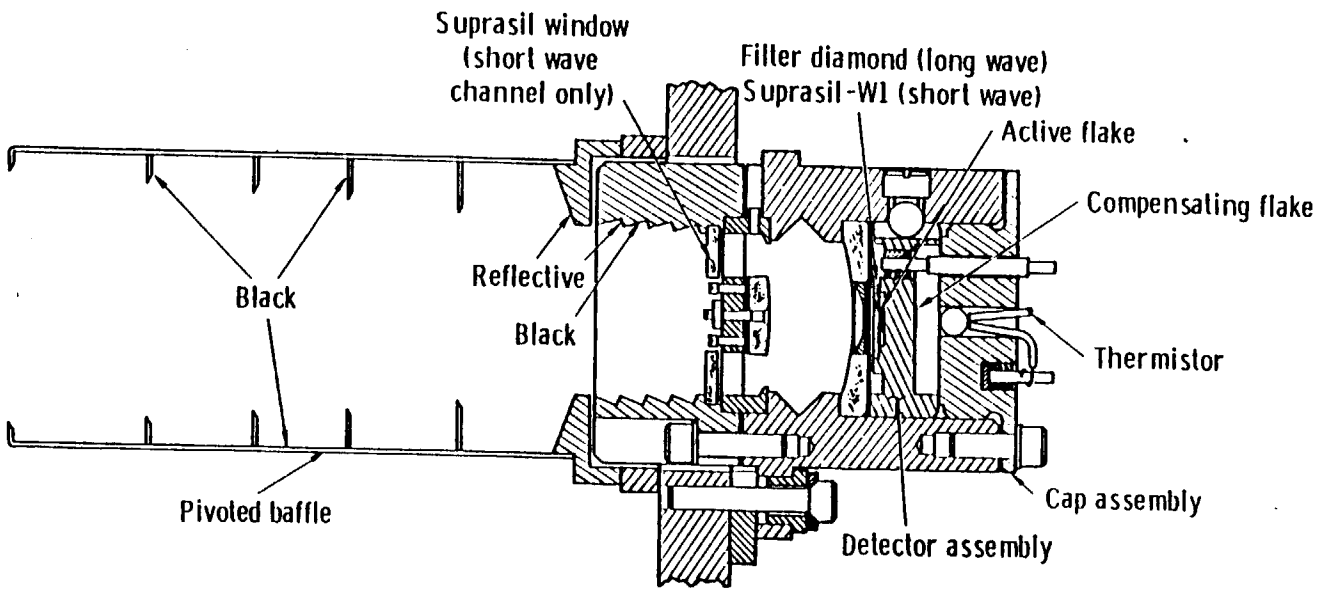
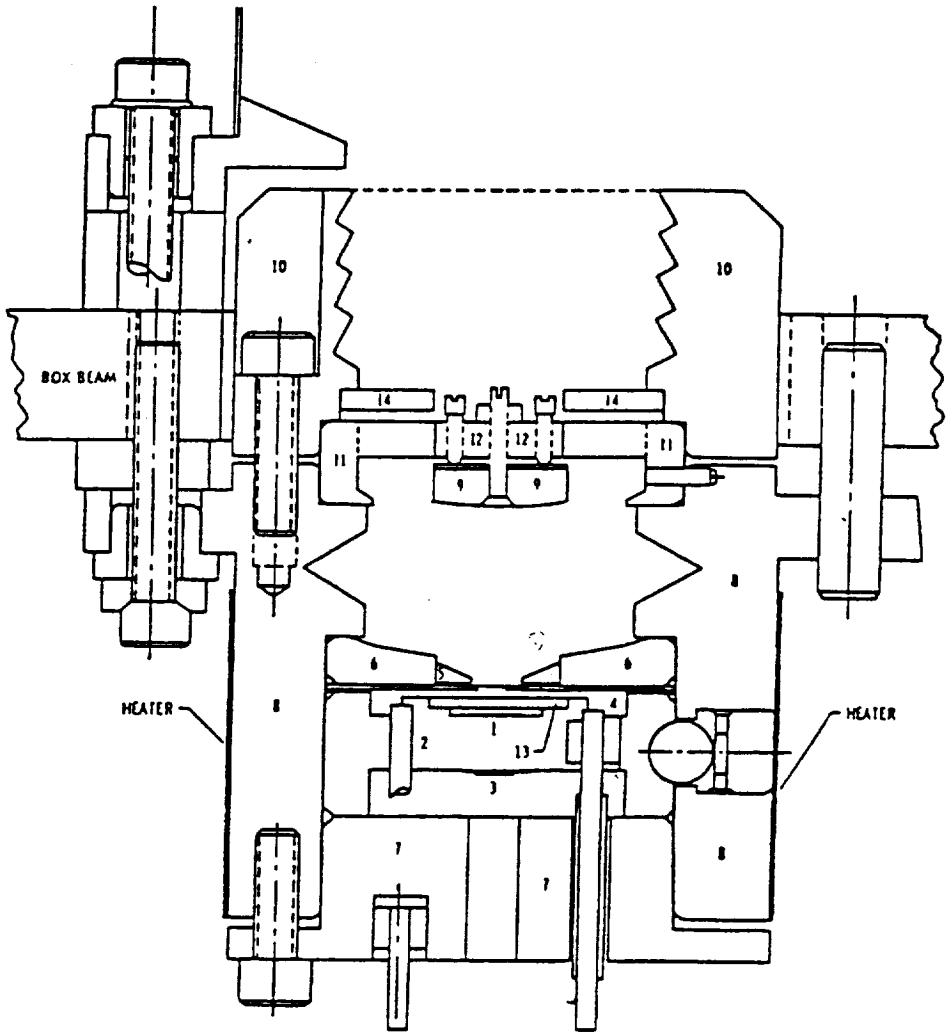


Fig. III.1 ERBE Scanner Optical Assembly.



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. III.2 ERBE Scanner Optical Module.

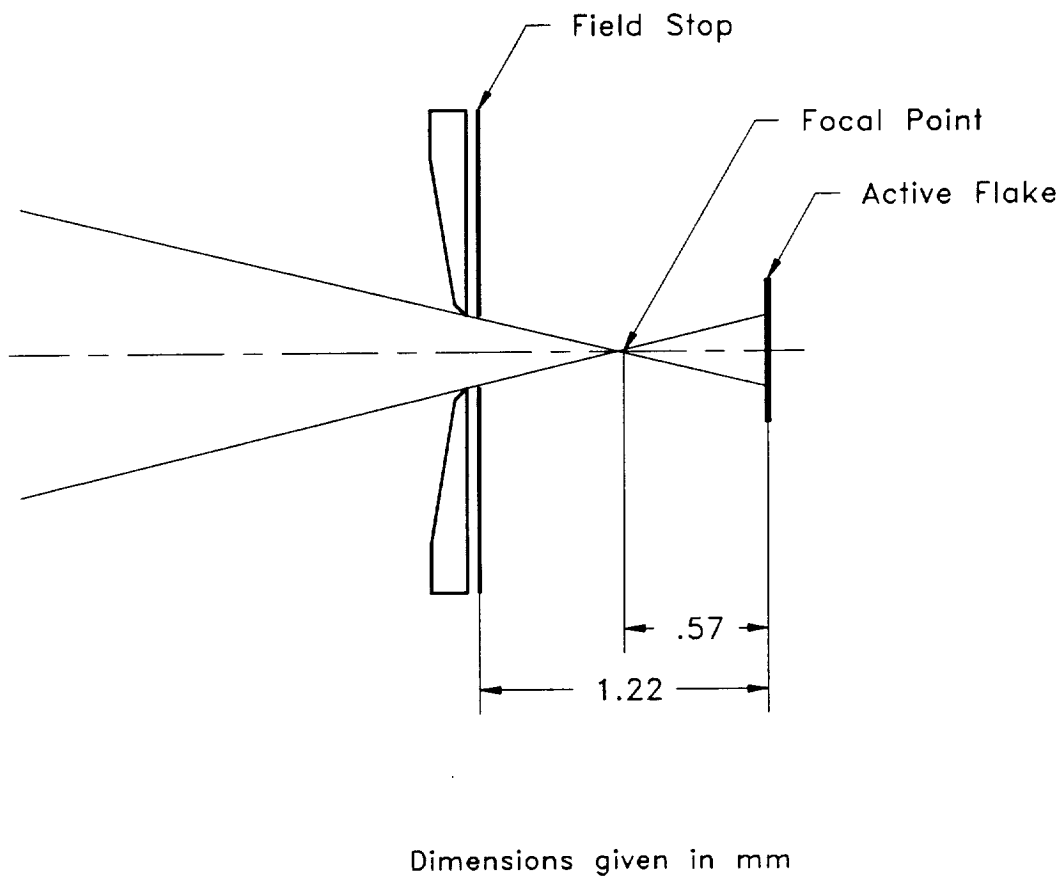


Fig. III.3 Relative Positions of the Field Stop and the Active Flake Along the Optical Axis.

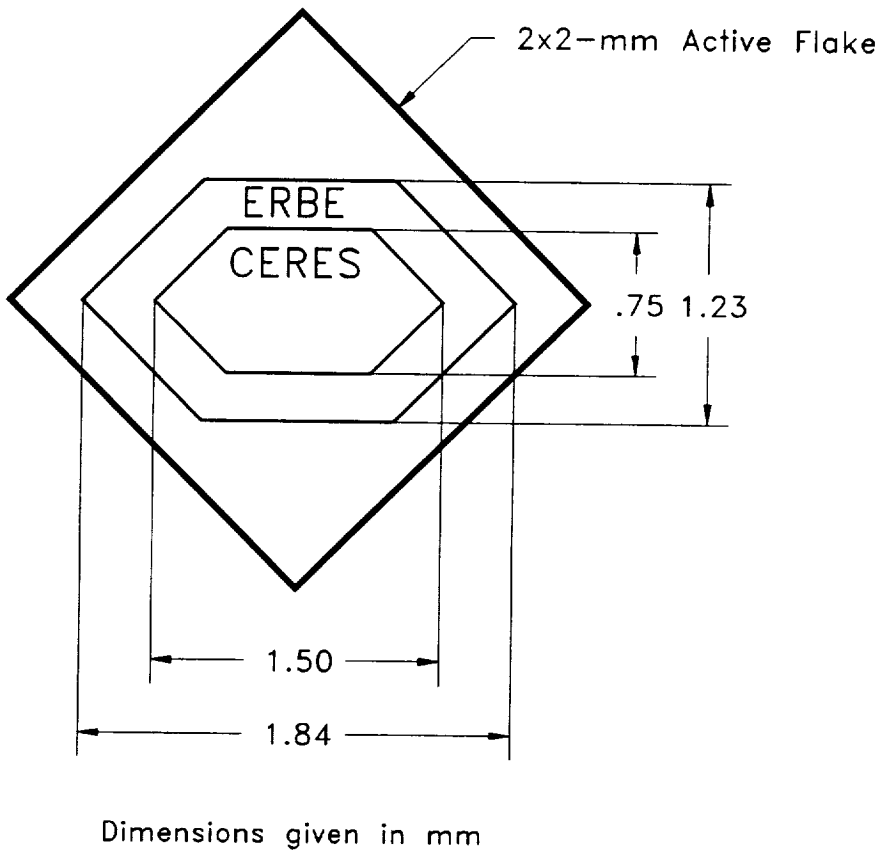


Fig. III.4 Projection of the Field Stop on the Active Flake.

1.2 Objective

As mentioned earlier, the main objective of both the Earth Radiation Budget Experiment (ERBE) and the Clouds and Earth's Radiant Energy System (CERES) is to monitor the Earth's radiative energy budget. Specifically, it is desired to measure the net radiative flux from finite regions on the Earth's surface. In the following analysis the plane containing the surface to be measured is referred to as the *object plane*, for example a region on the Earth. The plane containing the active flake of the scanner optical system is referred to as the *image plane*.

The spatial distribution of the object flux is given by its convolution with the *point spread function* (PSF) of the optical system. The PSF is defined as the image distribution of a radiative flux originating from a distant point source. Now if the point spread function is known and the image flux is measured, then in principle, it is possible to extract the object flux distribution by performing a deconvolution of the measured image flux and the point spread function.

The objective of this part of the research is to compute the point spread function of the scanner instrument. The tool used to obtain this goal is a Monte-Carlo based optical ray trace model developed by Meekins [12] in his Master of Science thesis and modified by the present author. Before undertaking the task of computing the PSF, the concept of the point spread function needs be understood. The PSF is defined and some optical considerations concerning its use are explored in Chapter 2. Then in Chapter 3 *aberrations*, which are errors associated with the optical system, are considered.

2.0 The Point Spread Function

2.1 Definition

Before the concept of the point spread function (PSF) can be defined, it is necessary to define what is meant by a point source. Even though it is quite easy to visualize, a point source of light cannot be realized in practice. However, use of point sources of light can lead to very convenient mathematical analysis and modeling of an optical system. For practical purposes, a point source can be thought of as a spherical light source of very small but finite dimensions. In practice, its size is negligible compared to the dimensions of the optical system which views it. A point source emits radiation in the form of spherical waves such that an observer placed very far away from the source will see collimated radiation (see Fig. III.5). A distant star being observed by a telescope is an example of a point source of light.

In most practical cases, a distributed source can be represented by an array of point sources, provided certain requirements and/or tacit assumptions are made regarding the flux coming from the source distribution or object [40]. In there are no loss mechanisms within an optical system, this flux will be transmitted through the system to

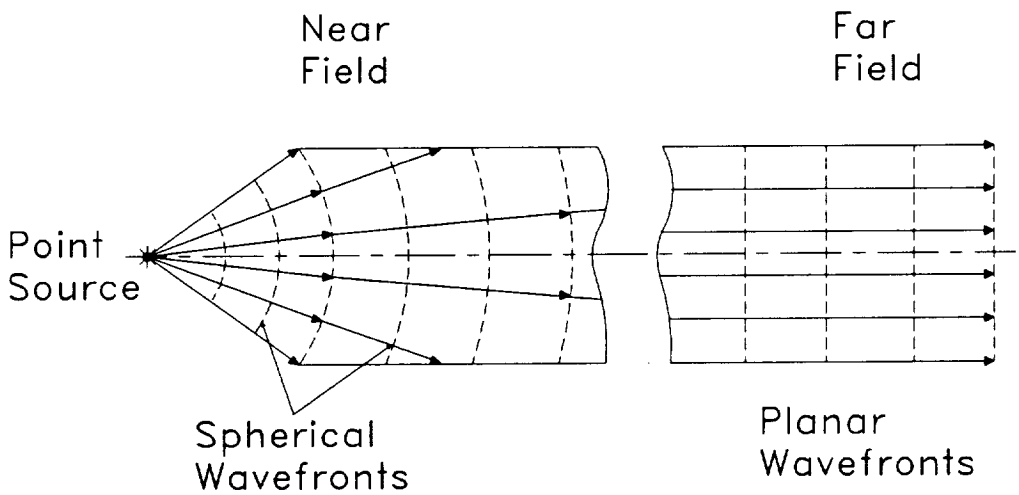


Fig. II.5 Illustration of a Point Source.

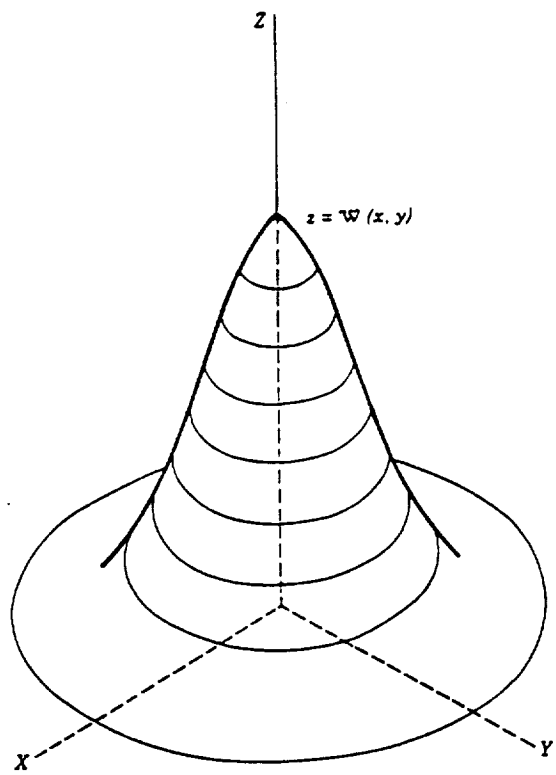


Fig. II.6 Radiation Mound: The Point Spread Function (Adopted from Williams and Becklund [40]).

form an image with a 100 percent efficiency. However, in reality the flux will be diminished along the optical path due to absorption, scattering and, for off-axis sources, vignetting [39,40].

The image of a point source can never be a point itself. This is due to several factors which cause the radiant energy that enters the system to be dispersed. Some of these factors are diffraction of the beam by the field stop, scattering from dust particles and imperfections on the surfaces of optical system components, geometrical irregularities, and aberrations. The image of a point source has a mound shape as shown in Fig. III.6. The mathematical function of this image, $W(x',y')$, gives the flux density distribution on the image plane and is called the *point spread function.* Assuming that the total radiative energy leaving the point source and reaching the optical system is unity, and that there is no loss of that energy as it passes through the optical system, then the integral over the image plane of the point spread function will be unity [40]; i.e,

$$\iint_{-\infty}^{+\infty} W(x',y')\,dx'dy' = 1. \qquad \textbf{(III.1)}$$

The primed coordinates are associated with the image plane. The mathematical expression relating the image flux distribution to that of the object is given by

$$W_i(x',y') = \iint_{-\infty}^{+\infty} W_0(x,y)\,\delta(x'-x,y'-y)\,dxdy, \qquad \textbf{(III.2)}$$

where $\delta(x'-x,y'-y)$ is the Dirac delta function and $W_0(x,y)$ is the point spread function.

If instead of a point source, the object plane contains a line source of radiation, which can be thought of as a linear distribution of point sources, then the corresponding image distribution is called, appropriately, the *line spread function*. The latter, whose shape is shown in Fig. III.7, is basically a series of point spread functions distributed along a straight line. The line spread function is important in the evaluation of the spatial frequency response of an optical system. A common laboratory experiment consists of placing several black and white bars on the object plane [41]. The spacing of these bars can then be changed to create different spatial frequencies, and their images are examined. This idea is presented schematically in Fig. III.8. *The optical transfer function*, which is the Fourier transform of the point spread function, is then used to describe the frequency response of the optical system.

Manalo and Smith [42] used the optical transfer function to evaluate aliasing and blur errors. In the cited paper, the authors mistakenly refer to the optical transfer function as the *modulation transfer function*. The modulation transfer function is actually the real part of the optical transfer function. The imaginary part is called the *phase transfer function*. To minimize aliasing and blur errors, the authors of Reference 42 propose that an optimum scanner aperture size of 2.8 x 1.12 deg be used in the Clouds and Earth's Radiant Energy System (CERES) mission.

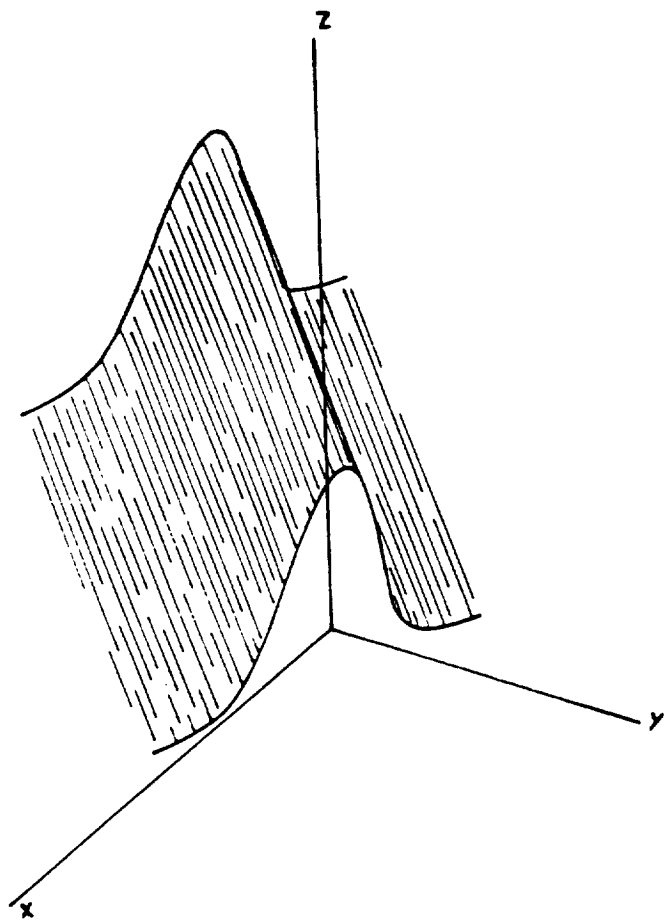


Fig. III.7 The Line Spread Function (Adopted from Williams and Becklund [40]).

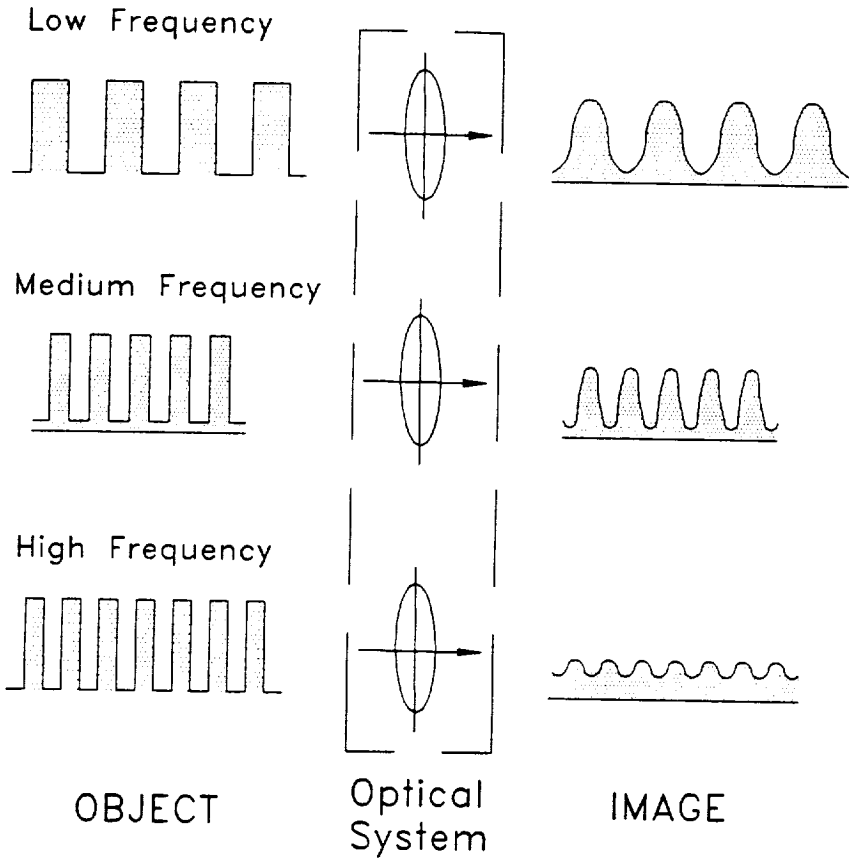


Fig. III.8 Optical System Response as a Function of Spatial Frequency.

2.2 General Distribution

In all practical cases an object source has a finite dimension. Therefore, Eq. (III.2) needs be modified before it can be of any practical use. This is accomplished by modeling the object source as an array of point sources; i. e., the object is thought of as a superposition of many point sources. Hence Eq. (III.2) can be written as

$$W_i(x',y') = \iint_{-\infty}^{+\infty} W_1(x,y)\, W_o(x'-x, y'-y)\, dxdy, \qquad \textbf{(III.3)}$$

where $W_1(x,y)$ is the function describing the general distribution of the source flux density. The Green's function, Eq. (III.3) may be interpreted to mean that the distribution of flux density in the image plane is produced by convolution of the point spread function and the distribution of flux density in the object plane. This idea is the driving force behind the research in Part III.

The juxtaposition of many point sources to form a practical object source requires that three conditions be satisfied. These conditions; linearity, isoplanatism and coherence; are discussed bellow.

2.2.1 Linearity

Linear superposition means that the effect in the image plane is directly proportional to flux coming from the object source. The effect of two different source fluxes, when applied separately, should be the same as when they are applied simultaneously. Mathematically, if $I_1(x,y)$ and $I_2(x+\Delta x, y+\Delta y)$, two different sources, produce respectively $W_1(x',y')$ and $W_2(x'+\Delta x', y'+\Delta y')$ in the image plane when applied separately, then when applied together the combined source produces $W_3(x',y')$ in the image plane such that

$$W_3(x',y') = W_1(x',y') + W_2(x'+\Delta x', y'+\Delta y') . \qquad \textbf{(III.4)}$$

Furthermore, if the source $I_1(x,y)$ is amplified by a factor A, so that it becomes $AI_1(x,y)$, then the distribution in the image plane is $AW_1(x',y')$.

2.2.2 Isoplanatism

The condition of isoplanatism requires that the point spread function of an optical system be invariant when the position of the point source is changed. In other words, if an object source at point (α,β) produces the point spread function $W(\alpha',\beta')$, a translation of the object point to $(\alpha+\Delta\alpha,\beta+\Delta\beta)$ produces the point spread function $W(\alpha'+\Delta\alpha',\beta'+\Delta\beta')$, where $\Delta\alpha' = m\Delta\alpha$ and $\Delta\beta' = m\Delta\beta$, m being a constant. Both functions are identical in

shape and magnitude; i. e., the translation of the point source in the object plane causes no change to the point spread function except its location in the image plane. The region where this is true is called an *isoplanatic patch.* In practice, however, isoplanatism in its strict definition is never present. Therefore, it is up to the optical designer or the analyst to choose certain regions in the object plane where the corresponding point spread function in the image plane does not change significantly. The result is the subdivision of the object source into isoplanatic patches where the point spread function is known for each patch, and that it may be assumed with acceptable error that it does not change significantly within a given patch.

2.2.3 Coherence

The radiant energy emitted by two point sources on the image plane, S_1 and S_2, is in the form of wave-trains oscillating with a given amplitude and phase. If the phase difference between the two sources, at a given point along the radiation path, does not change with time, then they are said to be *coherent.* Coherent light is obtained when two beams are created by splitting light from the same source, using pinholes whose size is comparable to the beams wavelength. When the two beams are then recombined, they form coherent light if the difference in path length followed by the two beams is an integer multiple of the wavelength. Most sources of visible radiation emit incoherent light. Light coming from the sun and from an incandescent lamp is incoherent; a laser beam is coherent. The resultant intensity at any point along the path of incoherent radiation is equal to the sum of the intensities that each source S_1 and S_2 produces separately at the

point. The intensity for each source is proportional to the square of its corresponding electrical field wave amplitude. For coherent light, on the other hand, the amplitudes of the two sources are combined vectorially, with the constant phase difference properly taken into account. The intensity of the combined source is proportional to the square of the resultant amplitude [43]. According to Young's experiment bright and dark fringes are observed for coherent light (visible radiation) because of the in-phase and out-of-phase interference [43]. This is not the case for incoherent light since the phase difference between sources is random.

3.0 Aberrations

Radiation emitted from a point source is in the form of concentric spheres centered around it as shown in Fig. III.5. The formation of the image is obtained by the opposite process, so that the wavefront converges on point *P'*, shown in Fig. III.9. In an ideal optical system, the wavefront should maintain a spherical shape centered at *P'*. In reality, however, it becomes aspheric, as shown in Fig. III.9. The difference, along a particular ray, between the aspheric wavefront and its reference sphere is the wavefront aberration. The numerical value of this difference is the product of the distance between *B'* and *C'* and the index of refraction of the medium.

There are two different classifications of aberrations that can be present in an optical system: Seidel aberrations and chromatic aberrations. The Seidel aberrations include spherical aberration, astigmatism, coma, field curvature and distortion aberration. All of these types of aberration are briefly discussed below. Further details can be found in most texts on optics such References 40, 41, and 44.

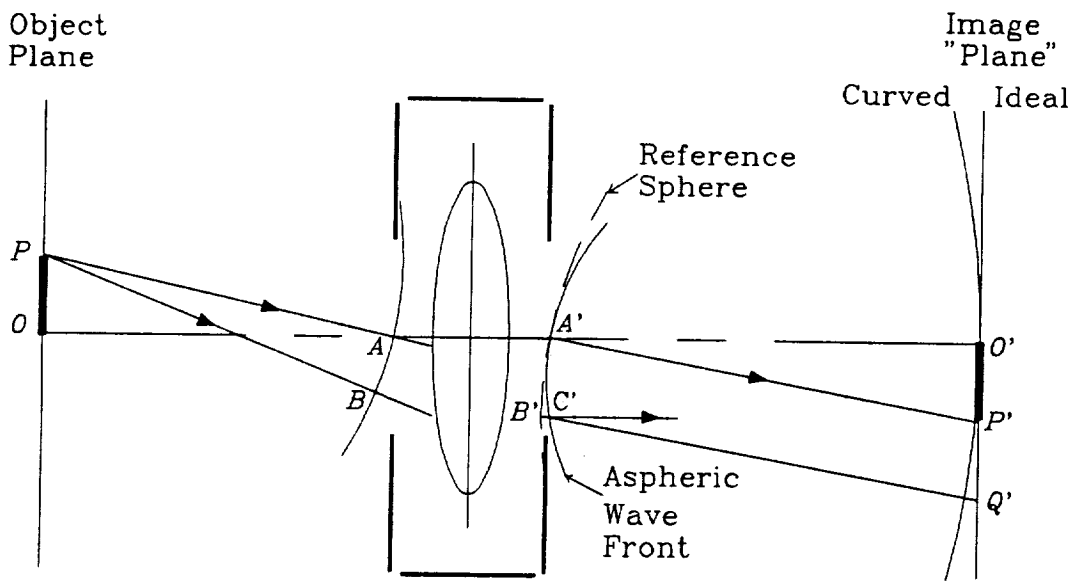


Fig. III.9 Schematic of a simple optical system.

3.1 Spherical Aberration

Paraxial rays; that, is rays in the vicinity of the optical axis, focus on one point. However, rays farther away from the optical axis tend to intersect the focal plane at ever-increasing radial distance from the optical axis. The further away the rays are from the axis the more pronounced the error in focus (see Fig. III.10). This type of imaging error is called *spherical aberration.* Being the most significant monochromatic aberration, spherical aberration is entirely due to the imperfection of optical components such as lenses and mirrors. Figure III.10 shows two focal planes: (1) a plane for paraxial focus, where only on-axis rays are focused, and (2) a plane of focus for minimum spot size. Depending on the application, each has its own merits. For example, in photography, the paraxial focal plane is preferred because it produces the sharpest image, even though some of the incoming energy is dispersed causing a flare around the bright spot at the focal point. In nonimaging applications, however, the focus for minimum spot size is preferred since flare is eliminated, and hence most of the incoming energy is captured on a small a patch. All of the energy is confined to a spot that is about 25 percent larger in diameter than that for paraxial focus [45]. The Cassegrain optics used in both the Earth Radiation Budget Experiment (ERBE) and in the Clouds and Earth's Radiant Energy System (CERES) scanner channels are nonimaging and so in both applications the thermistor bolometer is located at the focal plane for minimum spot size.

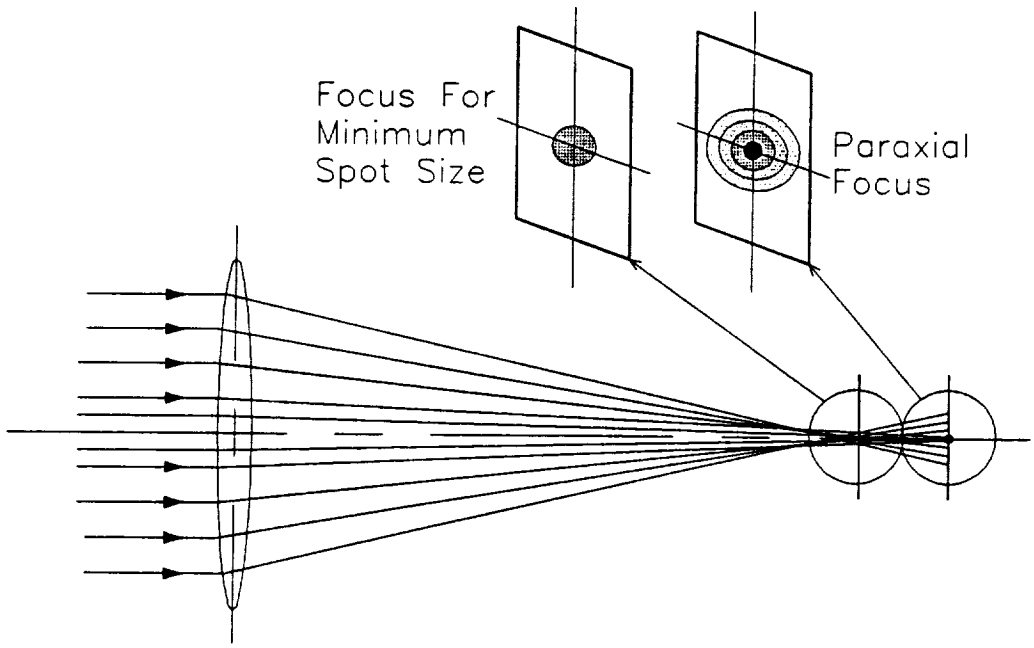


Fig. III.10 Illustration of Focus for Minimum Spot Size and Paraxial Focus.

3.2 Astigmatism

Figure III.11 shows how fans of rays having different orientations tend to focus at different focal points, and form distinct curved surfaces. If both sagittal and tangential fans were to focus at the same focal point, the image would be an ideal cylindrical spot (Airy profile). The plane on which this ideal spot is formed is called the *Petzval surface.* However, due to astigmatism the image spot tends to blur into an elliptical patch because the sagittal fan and the tangential fan focus at two different focal points. Astigmatism is defined as the difference between the sagittal and the tangential field curves.

3.3 Coma

Figure III.11 shows how each annular ring of incident rays tends to focus at a different height and with a different spot size on the image plane. The overall image spot has a conical shape with a bright core and flares with diminishing brightness extending toward the optical axis. This phenomenon is referred to as *coma.* Coma can be minimized by aligning the optical axis with the center of the object.

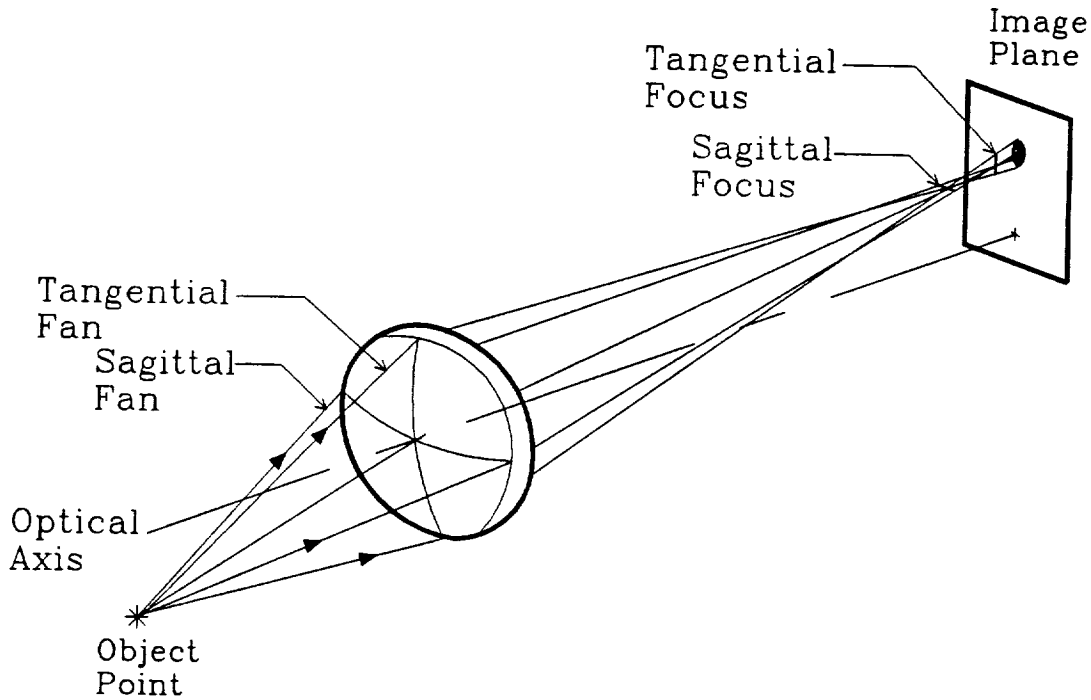


Fig. III.11 Off-Axis Aberration: Astigmatism.

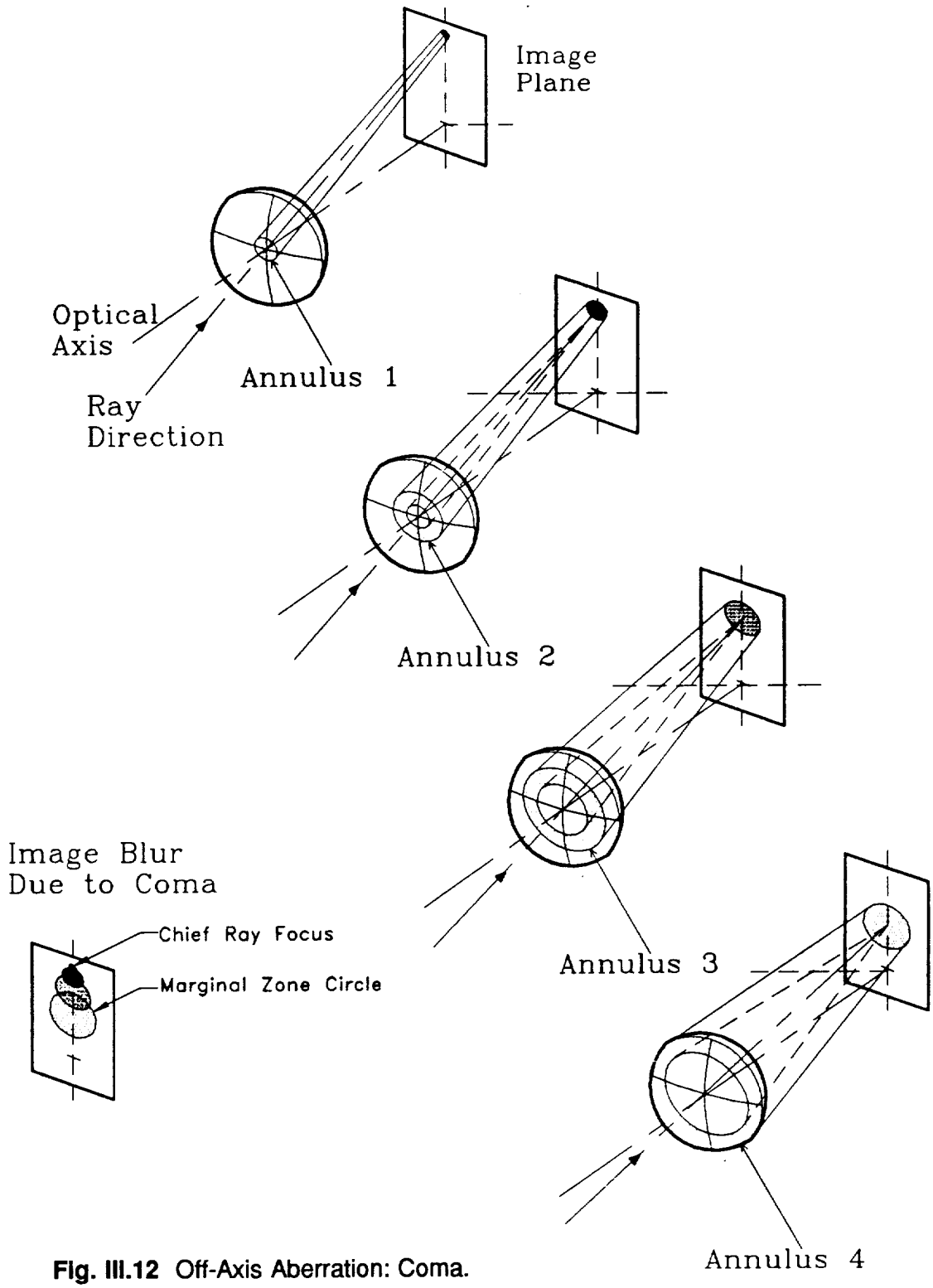


Fig. III.12 Off-Axis Aberration: Coma.

3.4 Field Curvature

When astigmatism is not present in an optical system, the sagittal and tangential fans focus on the same plane and form a curved image. The curvature associated with this image is called the *Petzval curvature*. In many applications such as cameras and projectors, it is desirable to form an image on a plane surface. So field curvature is especially objectionable in these imaging applications. To correct for field curvature, the image "plane", or screen must be curved.

3.5 Distortion

When the optical system produces an image that differs only in shape from the object, this is simply referred to as *distortion*. There are two types of distortion aberration, as illustrated in Fig. III.13:

1) Barrel distortion: This aberration operates to move the image points incrementally toward the optical axis from their true location with increasing radial distance, as shown in Fig. III.13(b).
2) Pincushion distortion: This aberration operates to move the image points incrementally away from the optical axis from their true location with increasing radial distance, as shown Fig. III.13(c).

3.6 Chromatic Aberration

This type of aberration is important only in the case of a broad spectral bandwidth. Since the index of refraction of optical material varies as a function of wavelength, the focal point for rays having different wavelengths will vary in location along the optical axis. This chromatic dispersion, or spreading, of radiation is referred to as *chromatic aberration.* Chromatic aberration can be minimized by setting focal point for the middle of the spectral band, and by shrinking this band as much as possible.

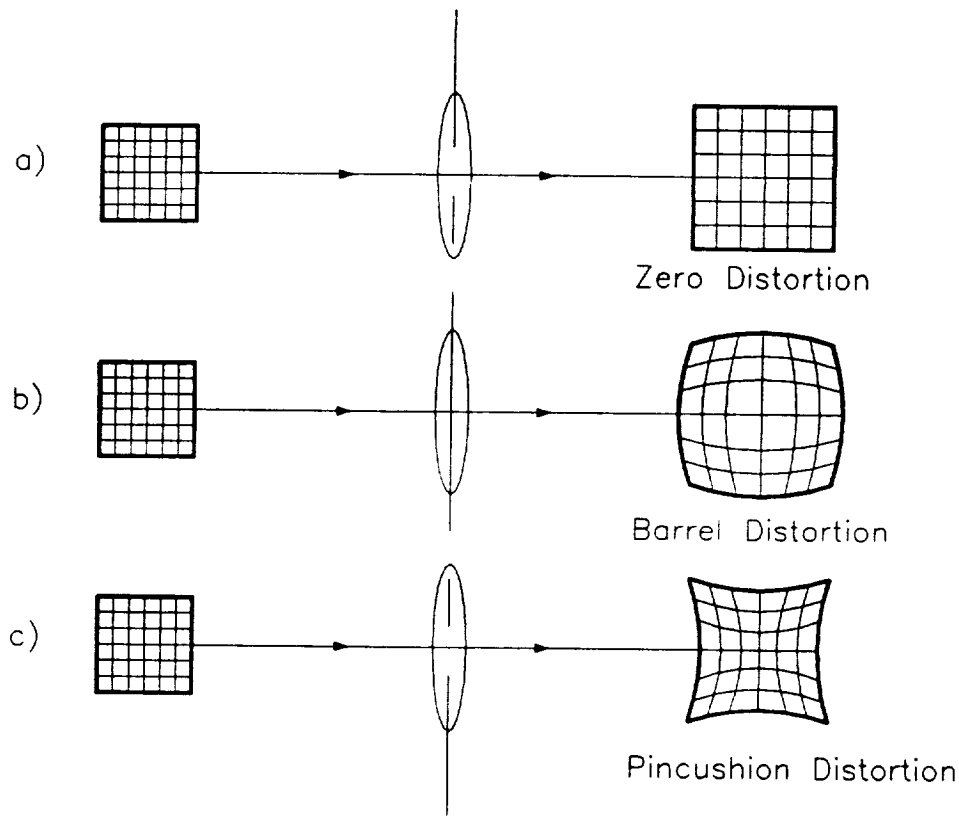


Fig. III.13 Illustration of Distortion.

4.0 Scanner Optical Model Formulation

The ERBE scanner optical module, shown in Fig. III.2 is based on a Cassegrain-type telescope design. The module is divided into 37 different parts, as described by Meekins in his Master of Science thesis [12]. Each part, or element, constitutes a surface possibly in the radiation path, and so is modeled according to its geometry and surface properties. To generate the point spread function, a ray trace method is used. Ray tracing involves two basic optical ideas: refraction and reflection. Refraction occurs when a ray passes from one transparent medium whose index of refraction is n_i into another whose index of refraction is n_r. The angle of refraction α_r is determined using Snell's Law,

$$n_i \sin\alpha_i = n_r \sin\alpha_r, \tag{III.5}$$

where α_i is the angle of incidence with respect to the surface normal. For the transparent components; i.e., the longwave and shortwave channel filters, Meekins assumed that refraction is negligible. That is, the transmitted radiation does not change direction.

For the case of specular reflection off an opaque surface, however, Snell's law still applies. In Eq. (III.5) the indices of refraction are canceled out since they are the same; and hence the angle of refraction, which becomes the angle of reflection, is equal to that

of incidence. Note that this is actually the familiar law governing reflection from ideal mirrors. An extensive and detailed description of the ray trace procedure for general optical systems is given by Stavroudis [46].

The Monte-Carlo method is used to generate a large number of rays passing into the scanner optical module from the desired direction. These rays are traced throughout the scanner optical module, as described above, and their ultimate distribution on the flake is recorded. The Monte-Carlo procedure, as applied to the scanner optical module and the nonscanner ACR instrument, is described in Reference 12 and 14, respectively. As mentioned earlier, the scanner optical model exploited in this part of the dissertation was formulated by Meekins in his Master of Science thesis [12]. The present author has modified this model as mentioned in the Preface, for use in the prediction of the point spread function.

5.0 Results and Discussion

Conventional wisdom has it that the Monte-Carlo method becomes more accurate as the number of rays is increased. Because computer execution speed is continually on the rise, Meekins' original model was modified to ensure that a limitless number of rays can be used with the present code listed in Appendix C. The current code is user-friendly in the sense that it prompts the user with very straightforward questions regarding the geometry and surface properties of the optical scanner module components, as well as the type of channel: total, shortwave or longwave channel. This flexibility of the program is very important in the analysis of the optical module. Using this optical model, one can obtain answers to a wide variety of design and performance questions before the radiometer is actually built. For example, to change from the ERBE foot-print resolution of 45 *km* at nadir to less than 35 *km* for CERES, it has been proposed to reduce the scanner FOV [9,42] by reducing the size of the field stop from 3.0 x 4.5 deg to 1.12 x 2.8 deg. The effect of this proposed modification is investigated in the present work and the results are discussed in Section 5.2.2.

The present code offers three different options for introducing radiation into the optical system: (1) diffuse radiation, (2) collimated radiation at a given angle of incidence with respect to the optical axis, and (3) collimated radiation at random angles around the

optical axis with a specified angle limit. It is important for the reader to realize that Options (1) and (3) are not the same. Option (1) obeys Lambert's cosine law, while in Option (3) incident radiative energy is equally probable at all angles within a specified range of angles.

5.1 Diffuse Source Radiation

In this case, radiation is emitted in the form of random rays from an imaginary disk coinciding with the module aperture. This disk is a Lambertian source; that is, it emits radiation according to the cosine law. In other words, the radiative flux varies as the cosine of the angle with respect to the surface normal. Figure III.14 shows how diffuse radiation is eventually distributed on the active flake of the ERBE total channel radiometer, which has a field stop of 3.0 x 4.5 deg. The 2 x 2-mm active flake is divided into 50 x 50 elements. A total of 200,000 rays have been used. Only 6392 rays, or 3.2 percent of the total number, reached the active flake. This is due to the directivity of the optical system, which has been conceived to prevent stray light from reaching the active flake [8,47]; i.e., radiation incident to the instrument beyond 2.25 deg from the normal. The figure shows a relatively small variation of radiation distribution throughout the flake.

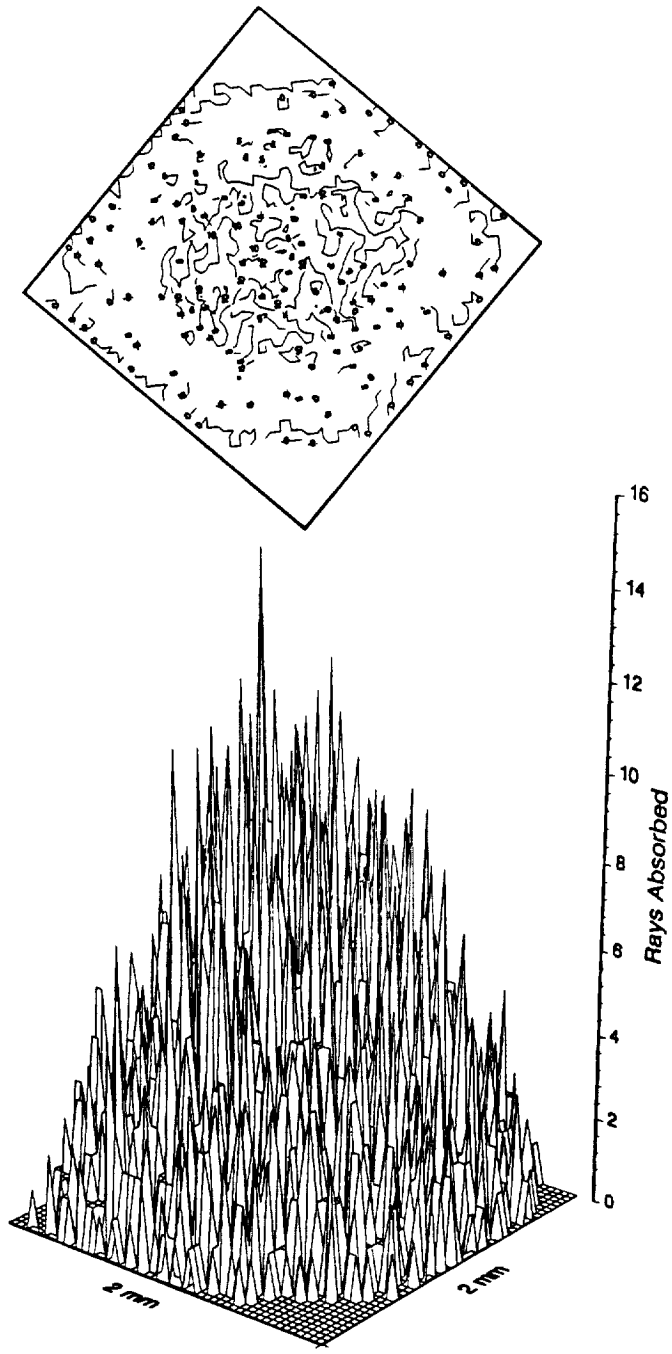


Fig. III.14 Distribution of Radiation Power on the Active Flake Due to Diffuse Radiation Incident to the Instrument Aperture.

5.2 Collimated Radiation

As noted earlier, radiation coming from a distant point arrives at the instrument in the form of parallel rays. In this case the image formed on the thermistor flake constitutes the point spread function (PSF). The 2 x 2-mm active flake is divided into 50 x 50 elements. The number of rays chosen to describe the instrument PSF is 50,000, which gives a relatively smooth surface of the three-dimensional Airy profile of the PSF. Each numerical experiment takes approximately 20 min on a 486-25 MHz based personal computers.

5.2.1 PSF for a Given Angle of Incidencee

Figures III.15 through III.19 give the point spread function at angles of incidence of 0.0, 1.0, 2.0, 2.5 and 3.0 deg, respectively. The base grid of these figures is the 2 x 2 mm-square active flake, and the z-axis represents the number of rays absorbed by the flake. In all of these cases the field stop configuration is that for ERBE scanning radiometer (3.0 x 4.5); i.e., and the distance between the detector flake and the field stop is 1.22 *mm* and the flake is 0.57 *mm* to the right the focal point, as shown in Fig. III.3. This places the image plane at the focus for minimum spot size as described in Section 3.1 and shown in Fig. III.9.

For an angle of incidence of 0.0 deg (Fig. III.15), the PSF is perfectly centered in the middle of the active flake and occupies a base area of about 12 percent that of the

flake. The amount of energy absorbed is about 43 percent of the total energy entering the system. At angles of 1.0 and 2.0 deg (Fig.s III.16 and III.17), the PSF maintains its general shape but as the angle of incidence increases, the base area becomes slightly larger and noticeably more distorted and the image moves toward the corner of the flake. This distortion is due to field curvature aberration as described in Section 3.4; that is, because the image tends to form on a curved surface.

At an angle of 2.5 deg, where radiative absorption on the thermistor bolometer flake is only 7.5 percent of the power incident to the instrument aperture, secondary absorption may be observed near the left corner of the flake (see Fig. III.18). This phenomenon has been observed experimentally by Kopia [48], and was suspected to be an unexplained characteristic of the scanners' Cassegrain telescope design. Observation of this phenomenon in the present research confirms this suspicion to a fact. Finally at an angle of 3.0 deg (Fig. III.19), all radiative absorption is secondary; however, this amount is a relatively insignificant one-third of one percent of the radiation incident to the instrument aperture. A potential problem this may pose is the so called stray-light absorption, or out-of-field absorption, problem. It is worth noting that the CERES science team has decided to resize the field stop to minimize this problem [9]. It is important to note that beyond an incident angle of 2.0 deg, the amount of radiative energy absorbed decreases precipitously as already determined in References 8 and 12. Therefore, it is assumed that isoplanatism as described in Section 2.2.2 is valid for incident angles up to two degrees.

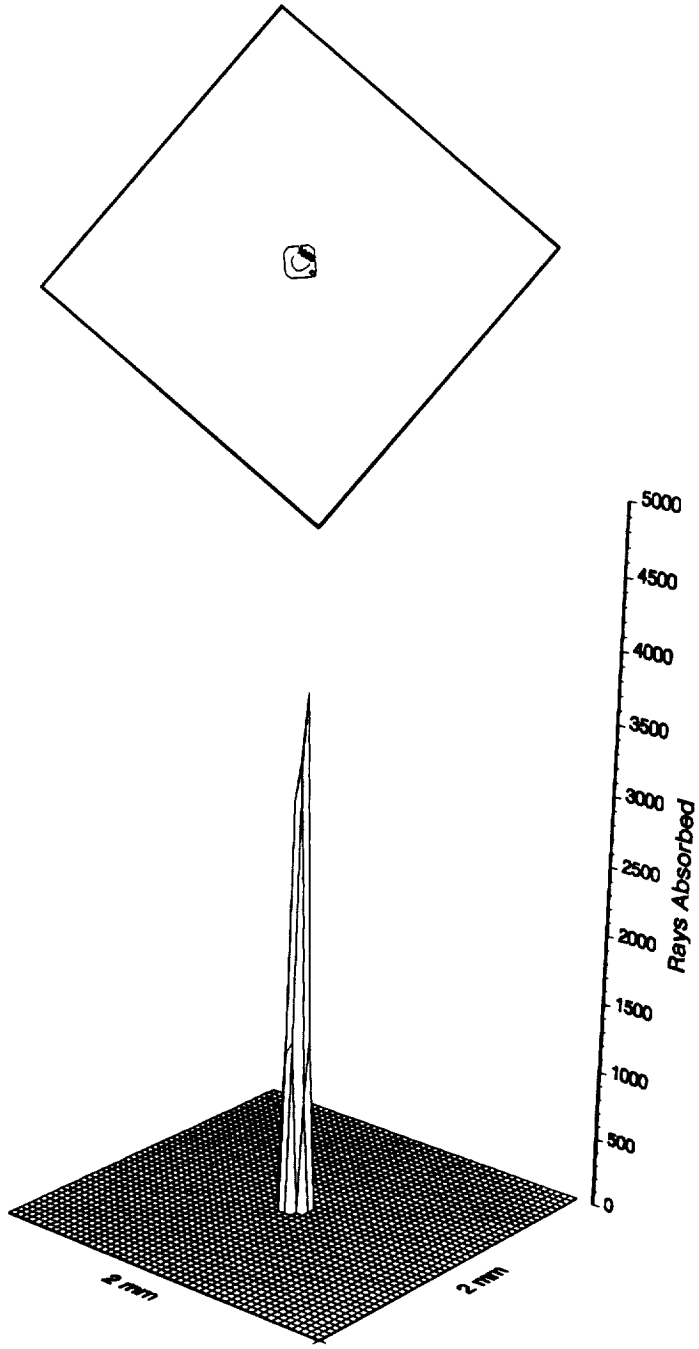


Fig. III.15 PSF at Zenith and Azimuth Angles of 0 deg.

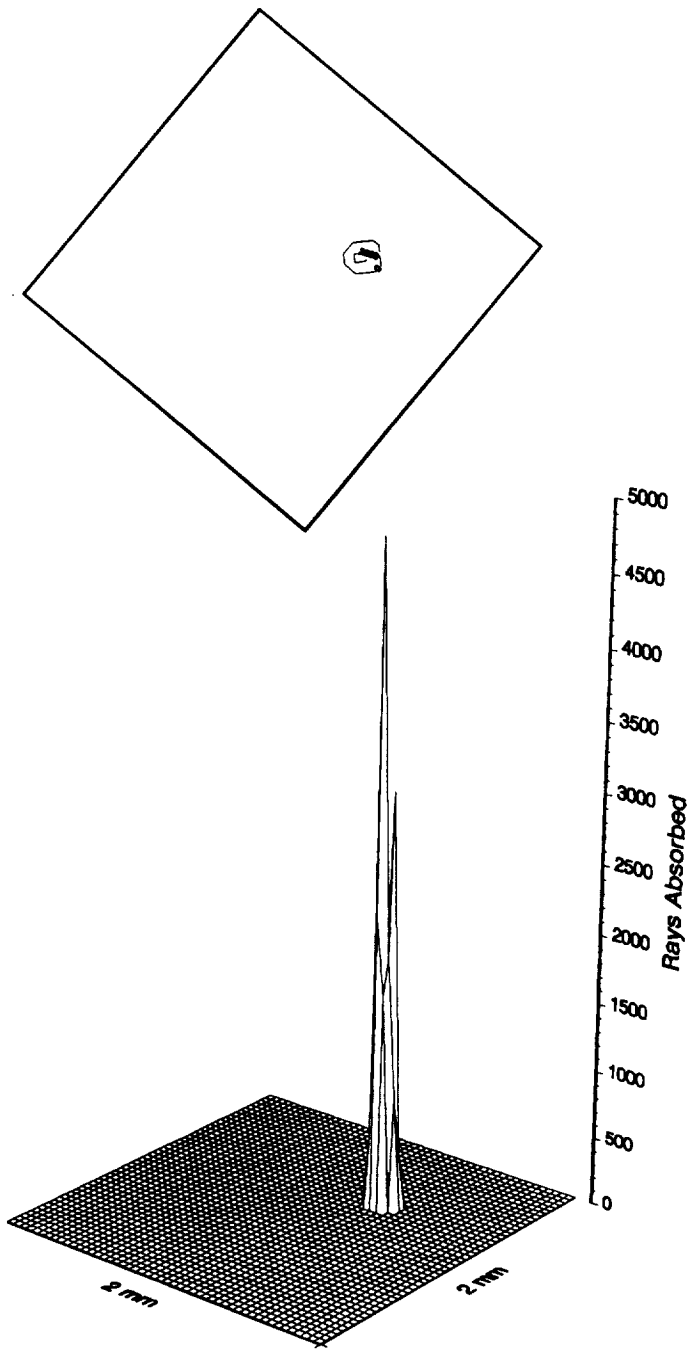


Fig. III.16 PSF at a Zenith Angle of 1 deg and an Azimuth Angle of 0 deg.

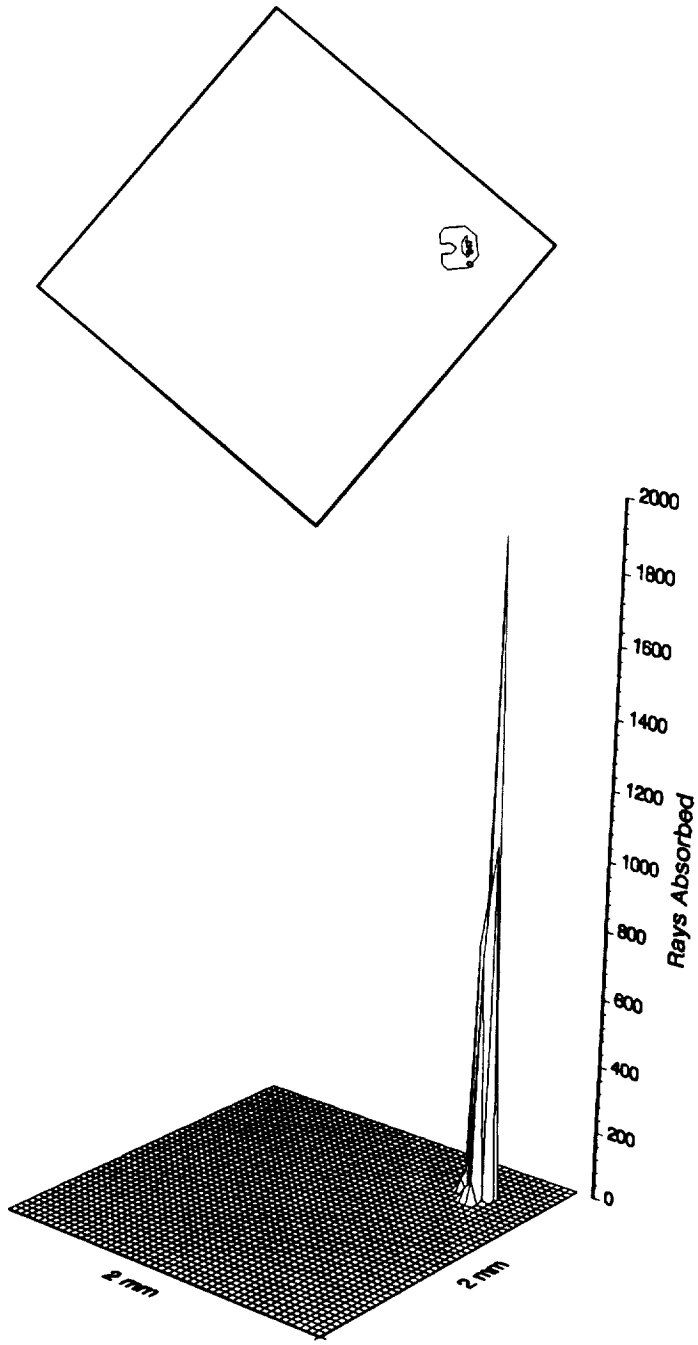


Fig. III.17 PSF at a Zenith Angle of 2.0 deg and an Azimuth Angle of 0 deg.

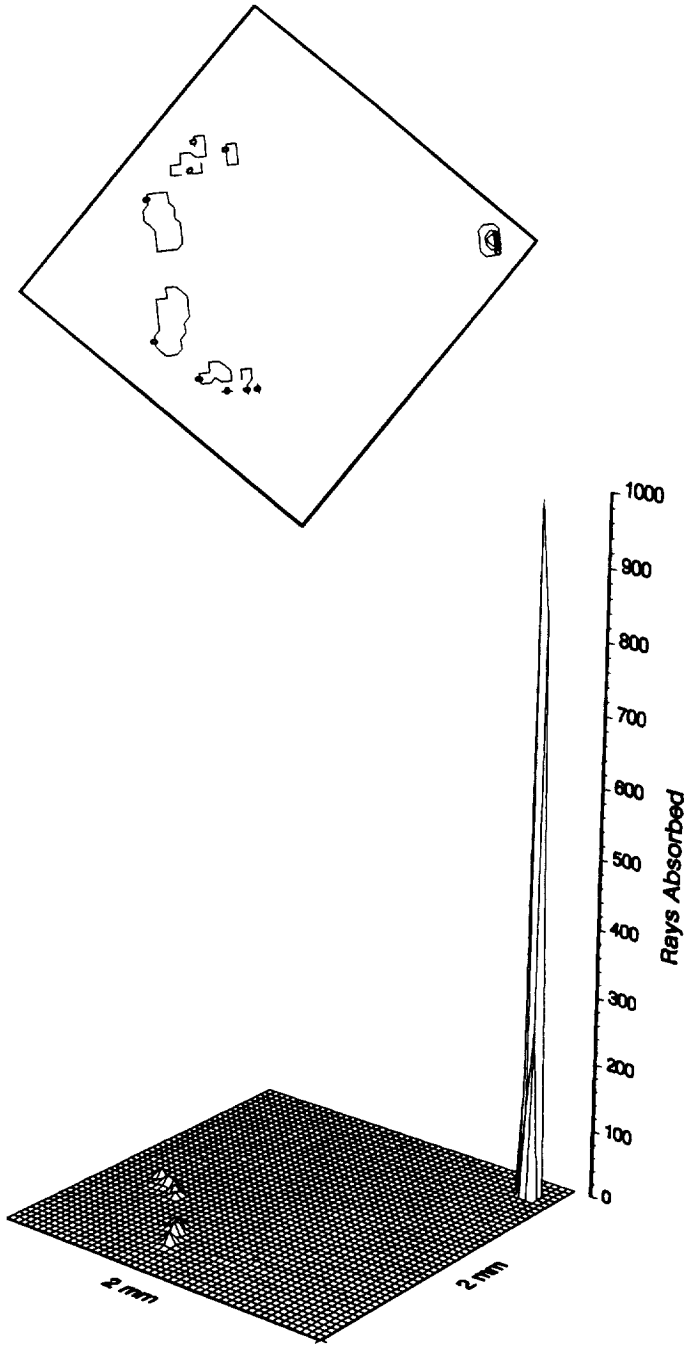


Fig. III.18 PSF at a Zenith Angle of 2.5 deg and an Azimuth Angle of 0 deg.

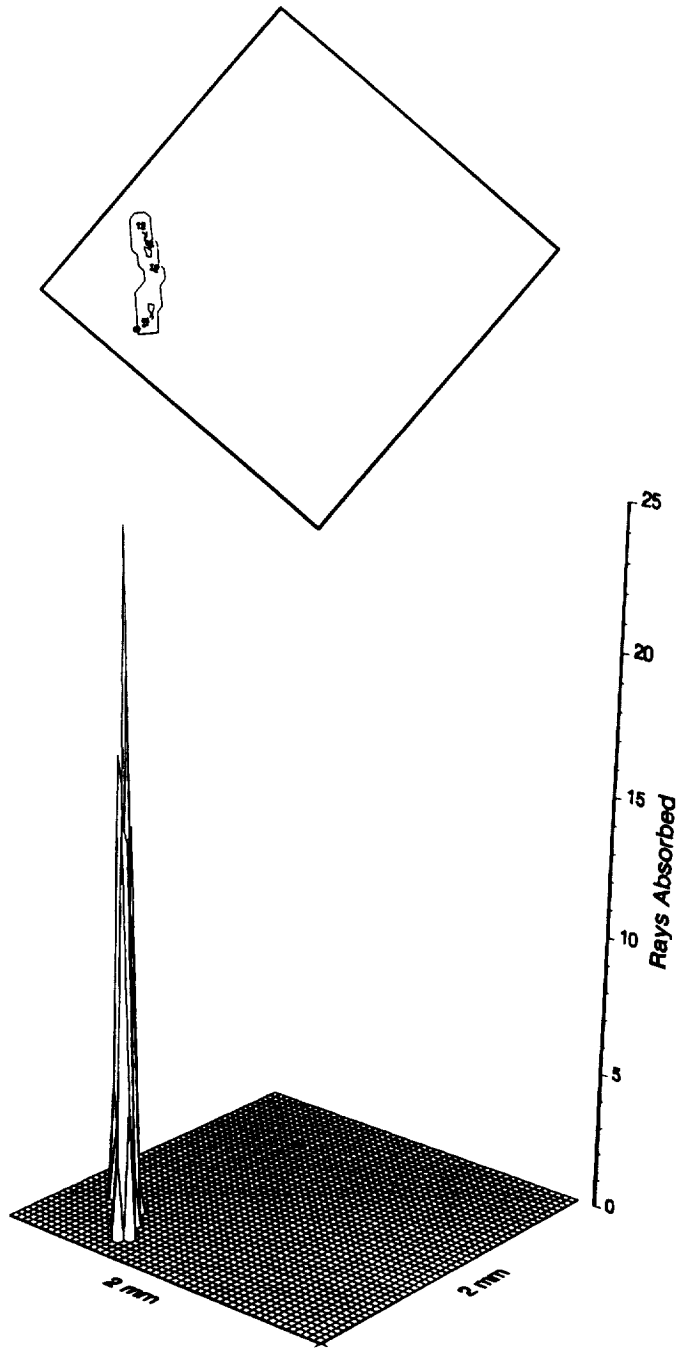


Fig. III.19 PSF at a Zenith Angle of 3.0 deg and an Azimuth Angle of 0 deg.

5.2.2 PSF for Random Angles of Incidence

In the previous section isoplanatism was shown to prevail for angles of incidence below about 2.0 deg. This means the "cumulative" point spread function, as stated by Eq. III.3, can be generated by taking an infinite number of point sources within this angular range. Alternatively, collimated radiation can be allowed to enter the optical system at many angles of incidence with respect to and around the optical axis, provided that the 2.0-deg limit is respected. This has been done and the result is shown in Fig. III.20. The corresponding optical transfer function (OTF), which is the Fourier transform of the PSF, is shown in Fig. III.23. Note that the cumulative PSF three-dimensional profile is perfectly symmetric and with a circular base and that all radiation is incident to the active flake. Hence there is no evidence that the diamond-shaped field stop is obscuring any of the radiation approaching the flake. On the other hand, if the angle limit is extended to 3.0 deg (see Fig. III.21), it is observed that the base of the PSF takes the general shape of the hexagonal field stop, shown in Fig. III.4. The distribution of radiation is more spread and some of it misses the flake causing a reduction of the PSF height.

Figure III.22 shows the point spread function for a limiting angle of 2.25 deg. It is clear that there is a very small difference, mainly at the base, between the shape of this PSF and that of Fig. III.20. Because of this negligible difference it is safe to extend the assumption of isoplanatism to include the dimensions of the precision aperture field stop angles (3.0 x 4.5 deg). The only possible problem is that it is assumed that there is no absorption at the edges of the flake, an assumption which, if true, ensures proper functioning of the thermistor bolometer [49]. In order to avoid this problem the proposed

CERES scanner will have a field stop angle of 1.12 x 2.8 deg. The PSF for the this design is shown in Fig. III.24, where it is clear that all of the radiation reaching the active flake is absorbed well within its boundary. Its corresponding OTF is shown in Fig. III.25.

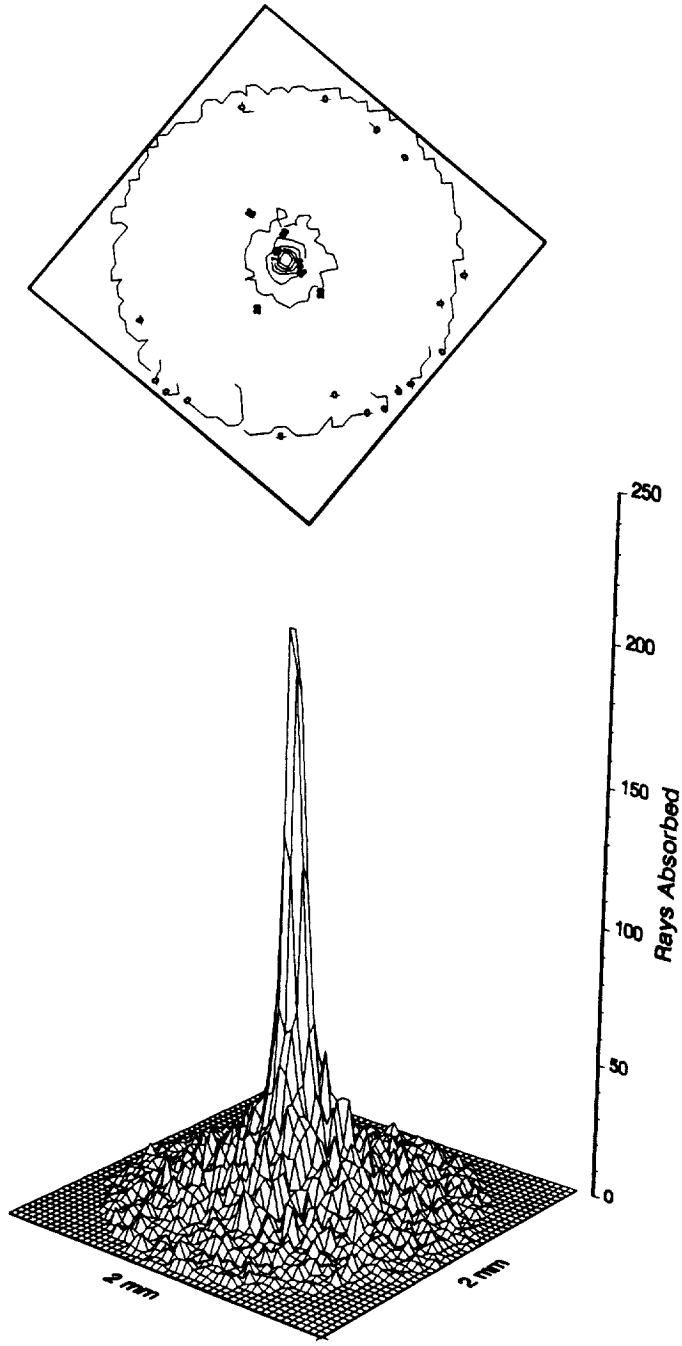


Fig. III.20 Cumulative PSF at Source Angle Limit of 2.0 deg.

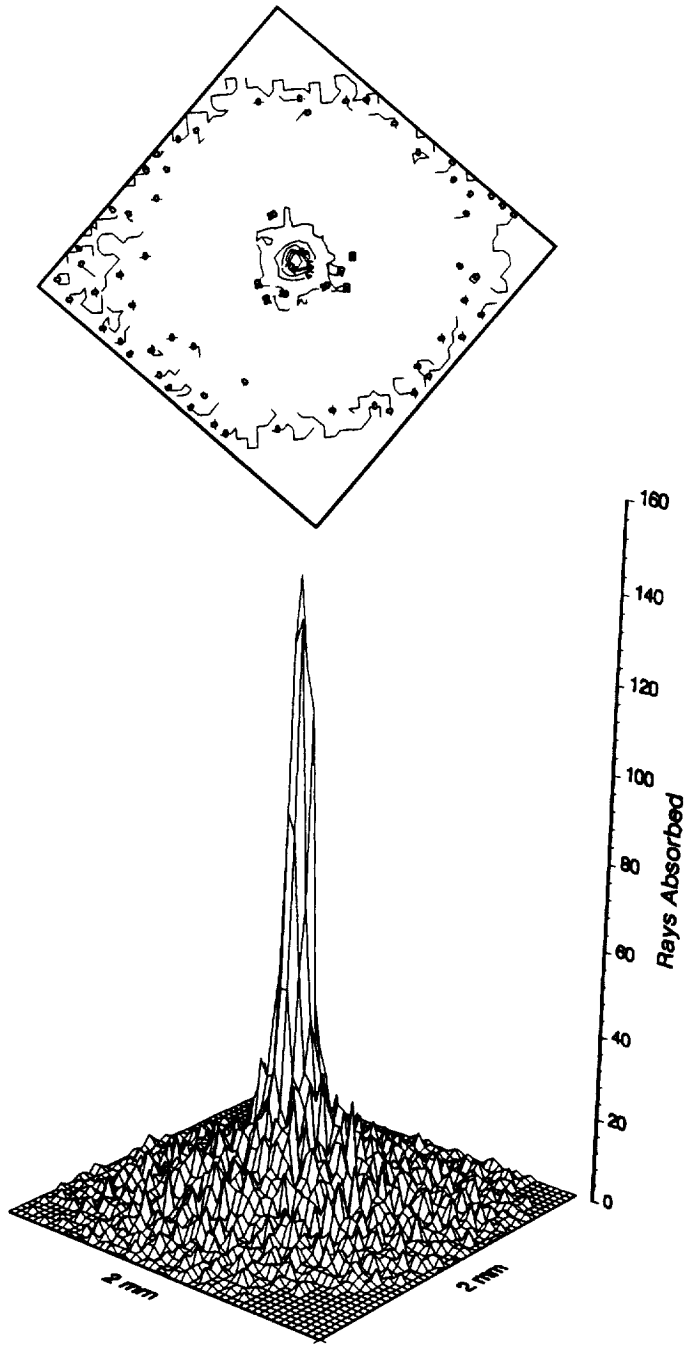


Fig. III.21 Cumulative PSF at Source Angle Limit of 3.0 deg.

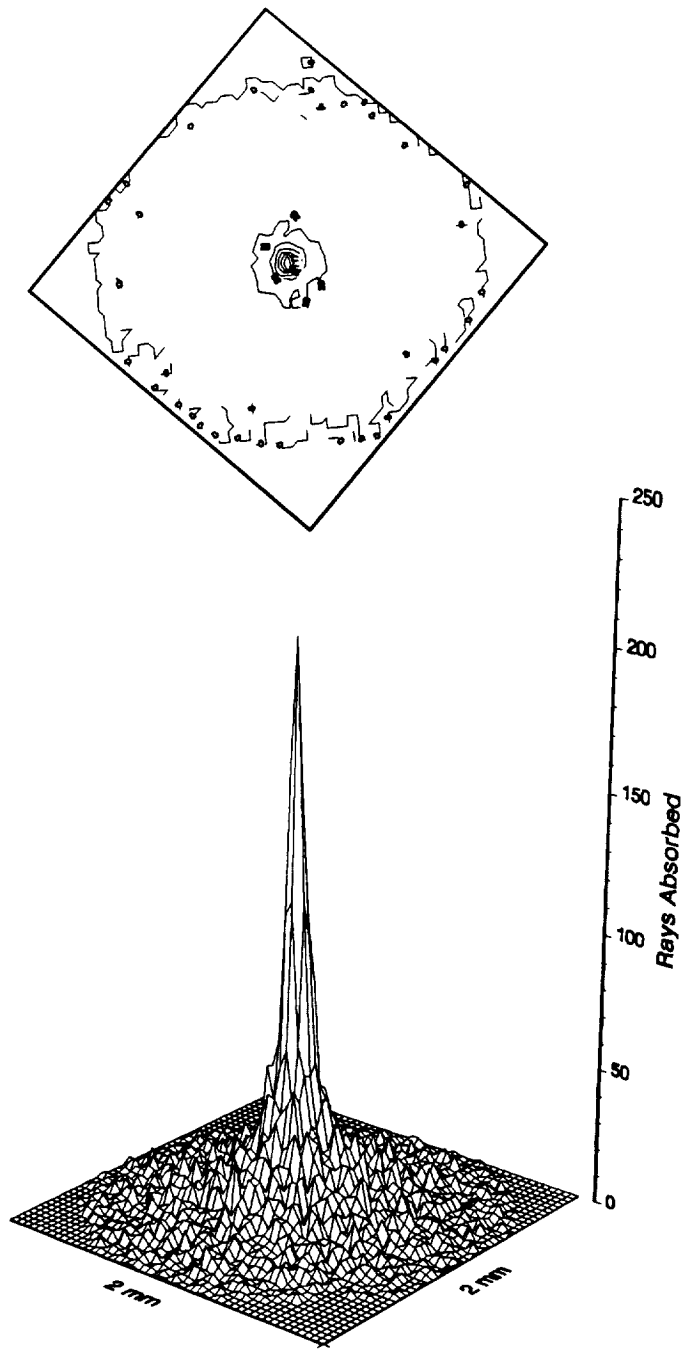


Fig. III.22 Cumulative PSF at Source Angle Limit of 2.25 deg.

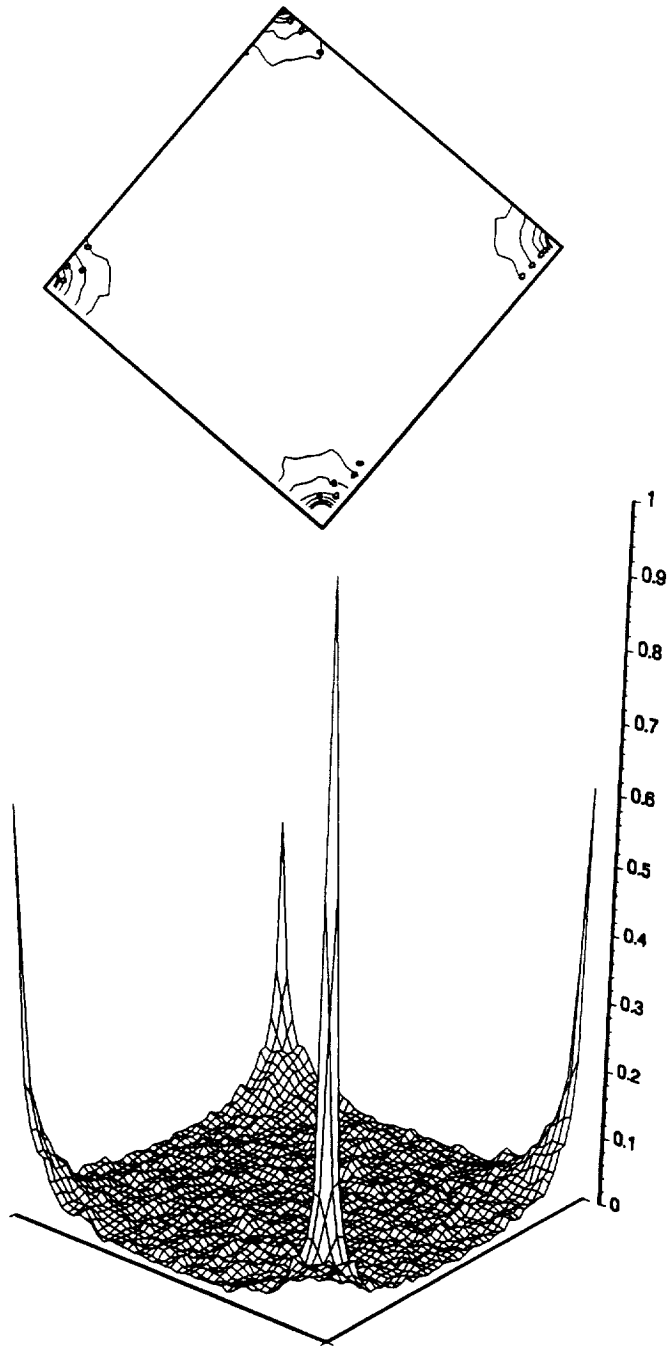


Fig. III.23 The OTF Corresponding to the PSF at Source Angle Limit of 2.25 deg.

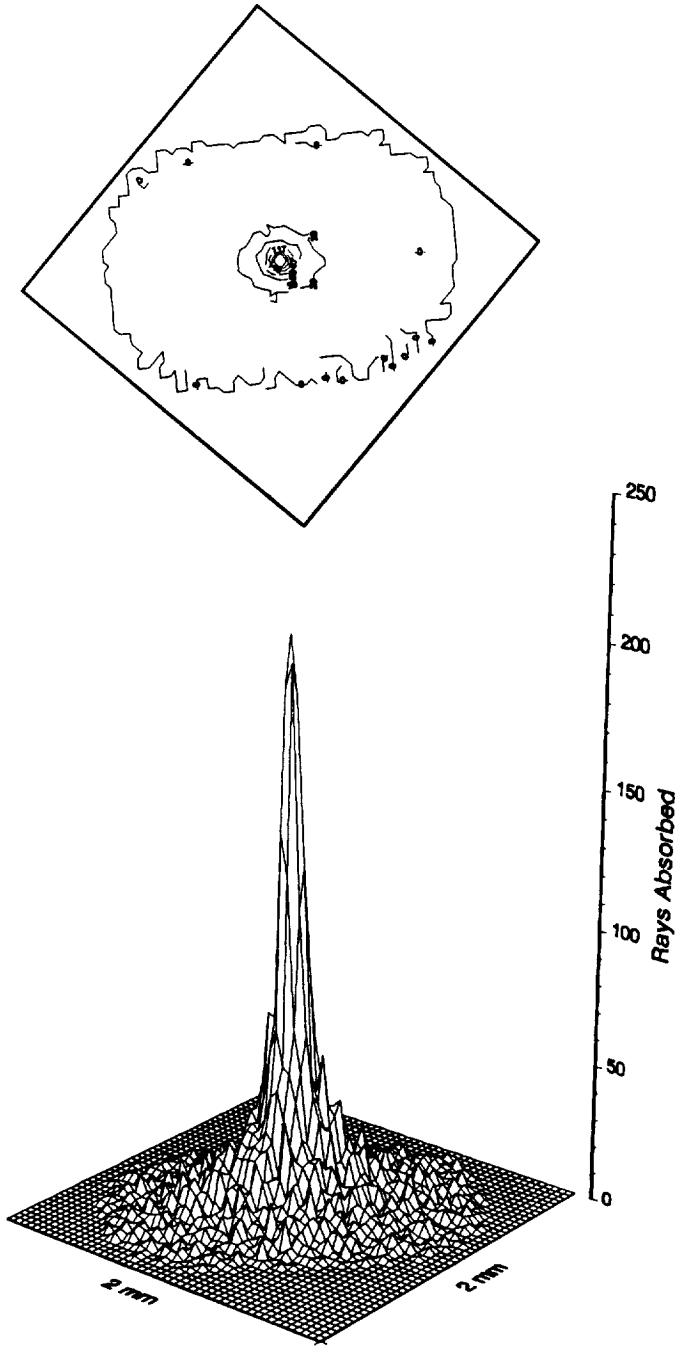


Fig. III.24 Cumulative PSF at Source Angle Limit of 2.0 deg (CERES Design).

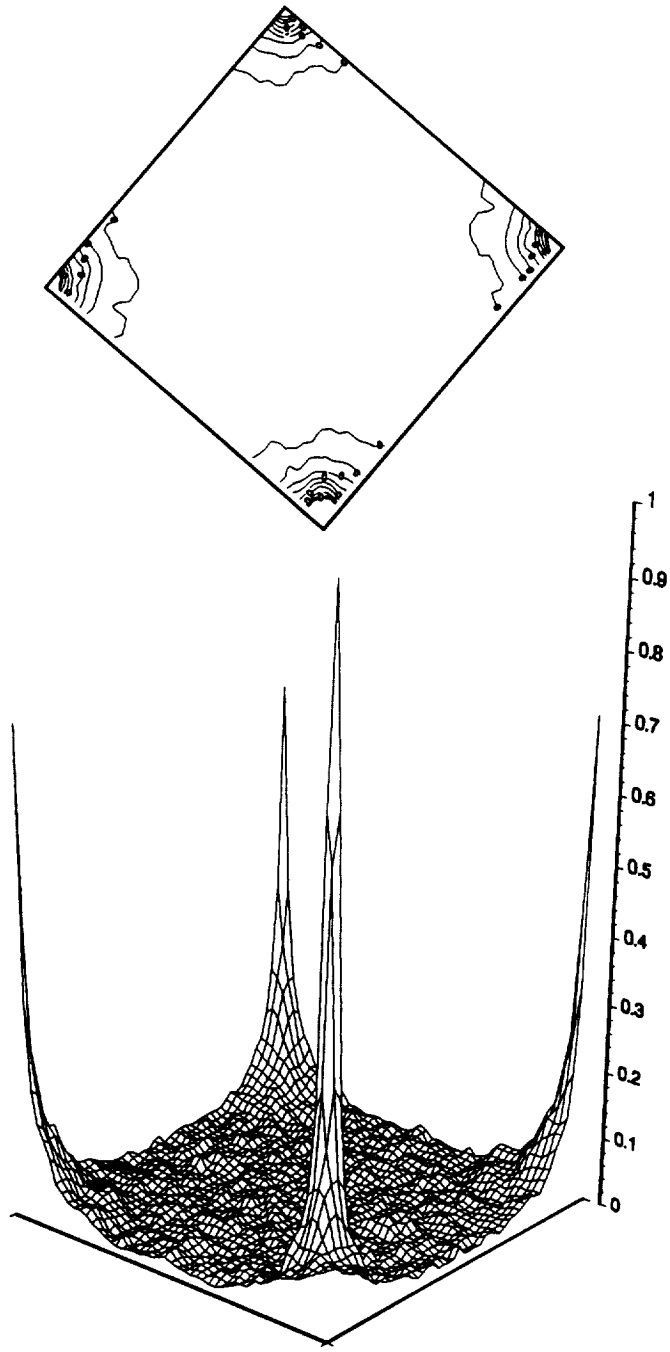


Fig. III.25 The OTF Corresponding to the PSF at Source Angle Limit of 2.0 deg (CERES Design).

6.0 Conclusions and Recommendations

The followings conclusions may be drawn from the research presented in this part:

1. A ray-trace-based analysis code has been demonstrated to be capable of generating the point spread function (PSF) for a Cassegrain-type telescope of the design used for the Earth Radiation Budget Experiment (ERBE) and Clouds and Earth's Radiant Energy System (CERES) systems.

2. The focus for minimum spot size is verified to be the best choice since we are interested in a compact and relatively uniform distribution of radiation on the flake rather than in the quality of the image itself.

3. Isoplanatism is confirmed at angles up to 2.25 deg.

4. The PSF can be used to estimate the input function by deconvolution of Eq. (III.3).

The following recommendations are made based on the research reported in this part:

1. A thermal study of the flake should be conducted to prove the validity of the assumption that the bolometer thermistor response is independent of the radiation distribution on it.

2. Using the computed point spread function, the optical transfer function (the Fourier transform of the PSF) could be investigated to study the frequency response of the scanner Cassegrain-type radiometer.

C-3

References

1. Shaw, N., **Manual of Meteorology**, Cambridge University Press, New York, 1926.

2. Hunt, G. E., R. Kendal, and A. T. Mecherikunnel, *A History of Presatellite Investigation of the Earth's Radiation Budget*, **Reviews of Geophysics**, Vol. 24, No. 2, 351-356 (1986).

3. 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 and Space Physics**, Vol. 24, No. 2, 354-377 (1986).

4. Wielicki, B. A., and B. R. Barkstrom, *Clouds and Earth's Radiant Energy System (CERES): an Earth Observing System experiment*, **Proceedings of the Second Symposium on Global Change Studies**, 11-16, January 13-18, 1990, New Orleans LA.

5. Barkstrom, B. R., *Earth Radiation Budget Measurements: Pre-ERBE, ERBE, and CERES*, **SPIE Proceedings**, Vol. 1299, 52-60 (1990).

6. Barkstrom, B. R., *The Earth Radiation Budget Experiment (ERBE)*, **Bulletins of the American Meteorological Society**, Vol. 65, No. 11, 1170-1180 (1984).

7. Luther, M. R., J. E. Cooper and G. R. Taylor, *The Earth Radiation Budget Experiment Nonscanning Instrument*, **Reviews of Geophysics and Space Physics**, Vol. 24, No. 2, 391-399 (1986).

8. Kopia, L. P., *Earth Radiation Budget Experiment Scanner Instrument*, **Reviews of Geophysics and Space Physics**, Vol. 24, No. 2, 400-406 (1986).

9. Anon., Updated Conceptual Review, **Clouds and the Earth's Radiant Energy System (CERES), NASA Earth Observation System (EOS) Meeting Report** , April 30 - May 2, 1991.

10. Tira, N. E., J. R. Mahan and R. B. Lee III, *An improved electrothermal model for the ERBE nonscanning radiometer: comparison of the predicted and measured behavior during solar observations*, **SPIE Proceedings**, Vol. 1109, Session on Optical Measurements II, Orlando FL., March 21-27, 1989, pp. 195-204.

11. Tira, N. E., J. R. Mahan and R. B. Lee III, *A dynamic electrothermal model for the ERBE nonscanning radiometer with application to solar observations and thermal noise*, **Optical Engineering**, Vol. 29 No. 4, April 1990, pp. 351-358.

12. Meekins, J. L., **Optical Analysis of the ERBE Scanning Thermistor Bolometer Radiometer Using the Monte Carlo Method,** M. S. Thesis, Mechanical Engineering Department, Virginia Polytechnic Institute and State University, Blacksburg, VA (1990).

13. Luther, M. R., R. B. Lee III, B. R. Barkstrom, J. E. Cooper, R. D. Cess, and R. H. Duncan, *Solar calibration results from two earth radiation budget experiment nonscanner instruments*, **Applied Optics**, Vol. 25, No. 4, 540-545 (1986).

14. 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, VA (1987).

15. Mahan, J. R. and N. E. Tira, *Dynamic Simulation of the Total Nonscanning Channel of the Earth Radiation Budget Experiment (ERBE) During Solar Calibration*, **International Radiation Symposium**, Lille, France, August 18-24, 1988.

16. 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, 1327-1337 (1989).

17. Mahan, J. R. and L. D. Eskin, *Application of the Monte Carlo Technique to Transient Thermal Modeling of Cavity Radiometers Having Diffuse-Specular Surfaces*, **Proceedings of the Fourth Conference on Atmospheric Radiation**, Toronto, Canada, June 16-18 (1981).

18. Mahan, J. R., F. Kowsary, N. Tira and B. D. Gardiner, *Transient Conduction-Radiation Analysis of an Absolute Active Cavity Radiometer Using Finite Elements*, **ASME Winter Annual Meeting**, 39-47, Boston, MA. December 13-18 (1987).

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

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

21. Siegel L. R. and J. R. Howell, **Thermal Radiation Heat Transfer**, McGraw Hill, New York, pp. 341-371,1972.

22. Reddy, J. N., **An Introduction to Finite Element Method**, McGraw Hill, New York, 1984.

23. Zienkiewicz, O. C. and R. L. Taylor, **The Finite Element Method**, Fourth Edition, Vol. 1, McGraw Hill, New York, 1977.

24. Blackman R. B. and J. W. Tukey, **The Measurement of Power Spectra**, Dover, New York, (1958).

25. Bendat, J. S. and A. G. Piersol, **Engineering Applications of Correlation and Spectral Analysis**, John Wiley & Sons, New York, (1980).

26. Bendat, J. S. and A. G. Piersol, **Random Data: Theory and Applications**, John Wiley & Sons, New York (1988).

27. Marple, S. L. Jr., **Digital Spectral Analysis with Applications**, Prentice Hall, Englewood Cliffs, N.J. (1987).

28. Harris F. J., *On the Use of Windows for Harmonic Analysis with the DFT*, **Proc. IEEE**, Vol. 66, No. 1 (January, 1978).

29. Roden, M. S., **Analog and Digital Communication Systems**, Sec. Ed., Prentice Hall, Englewood Cliffs, N.J. (1985).

30. Simpson, R. S. and R. C. Houts, **Fundamentals of Analog and Digital Communication Systems**, Allyn and Bacon, Inc., Boston, MA (1971).

31. Cooley J. W. and J. W. Tukey, *An Algorithm for the Machine Computation of Complex Fourier Series*, **Mathematics of Computation**, Vol. 19, 297-301 (1965).

32. Oppenheim, A. V., and R. W. Schafer, **Digital Signal Processing**, Prentice Hall, Englewood Cliffs, N.J. (1975).

33. D'Souza, A. F., **Design of Control Systems**, Prentice Hall, Englewood Cliffs, N.J. (1988).

34. Holman, J. P., **Experimental Methods for Engineers**, Third Ed., McGraw-Hill, New York (1978), pp. 74-77.

35. Carnahan, B., H. A. Luther, and J. O. Wilkes, **Applied Numerical Methods**, John Wiley & Son, New York (1969), pp. 319-329.

36. Jenkins, G. M. and D. G. Watts, **Spectral Analysis and its Applications**, Holden-Day Inc., San Francisco (1968).

37. Schwartz, M., **Information Transmission, Modulation and Noise: A Unified Approach**, McGraw-Hill, New York (1980), pp. 40-50.

38. Boyd, R. W., **Radiometry and the Detection of Optical Radiation**, John Wiley and Sons, New York (1983).

39. DeWitt, D. P., and G. D. Nutter, **Theory and Practice of Radiation Thermometry,** John Wiley and Sons, Inc., New York, 1988.

40. Williams, C. S., and A. B. Becklund, **Introduction to the Optical Transfer Function,** John Wiley and Sons, Inc., New York, 1989.

41. O'Shea, D. C., **Elements of Modern Optical Design,** John Wiley and Sons, Inc., New York, 1985.

42. Manalo, N. D., and G. L. Smith, *Spatial Sampling Errors for a Satellite-Borne Scanning Radiometer*, **SPIE's International Symposium on Optical Engineering and Photonics in Aerospace Sensing**, Orlando FL April 1-5, 1991.

43. Halliday, D., and R. Resnick, **Fundamentals of Physics,** Second Edition, Extended Version, John Wiley and Sons, Inc., New York, 1981.

44. Papoulis, A., **Systems and Transforms with Applications in Optics**, Robert E. Krieger Publishing Company, Inc., Malabar, Florida (1968).

45. Anon., Oriel Corporation, **Optics and Filters**, Stradford, C.T. 1990.

46. Stavroudis, O. N., **The Optics of Rays, Wavefronts, and Caustics**, Academic Press, Inc., New York (1972).

47. 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 Radiation Budget: Development of Algorithms for the ERBE Mission*, **Reviews of Geophysics**, Vol. 24, No. 2, May 1986, pp 407-421.

48. Kopia, L. P., NASA Langley Research Center, *Personal Communication*, November 1990.

49. Smith, G. L., NASA Langley Research Center, *Personal Communications*, November 1990 and June 1991.

Appendix A.

NOISE PROGRAM

```
*****************************************************************************
*                    ========                                               *
*                   | FOVMC  |                                              *
*                    ========                                               *
*  This program computes the distribution factors arrays for the           *
*  Field-of-View limiter geometry, for thermal noise.                      *
*  ----------------------------------------------------------------------- *
*                                                                          *
*                      NOMENCLATURE                                        *
*  ----------------------------------------------------------------------- *
*                                                                          *
*  AAP ........ Absorptivity of the primary aperture (washer) surface.     *
*  AFOV ....... Absorptivity of the Field-of-View (FOV) Limiter.           *
*  DF(I) ...... Distribution factors for :                                 *
*     I = 1     Secondary aperture.                                        *
*     I = 2     Primary aperture (Cavity).                                 *
*     I = 3     Primary aperture plane.                                    *
*     I = 4     Field-of-View limiter surface.                             *
*  DH ......... FOV limiter height.                                        *
*  IFLAG(I) ... Flag referring to the source of radiation.                 *
*     I = 1     Collimated beam through FOV limiter.                       *
*     I = 2     Collimated beam through solar port.                        *
*     I = 3     FOV limiter is the diffuse source.                         *
*     I = 4     Washer is the diffuse source.                              *
*     I = 5     Cavity is the diffuse source.                              *
*  NEB ........ Number of energy bunbles                                   *
*  NCY ........ Number of rings on the cylinder.                           *
*  RRAP ....... Reflectivity ratio  of the washer surface.                 *
*  RRFOV ...... Reflectivity ratio of FOV Limiter.                         *
*  RI ......... Cavity radius.                                             *
*  RIF ........ Radius of the secondary aperture.                          *
*  RIP ........ Precision aperture radius.                                 *
*  RIS ........ FOV sphere radius.                                         *
*  RSP ........ Solar port radius.                                         *
*  RN ......... Randum number.                                             *
*  SPH ........ Solar port height.                                         *
*  THETA ...... Angle of collimated beam w/r to aperture normal.           *
*  ----------------------------------------------------------------------- *
*  Calls: RNSET,RNUN or RANDOM                                             *
*  Calls: PICKP,PICHPC,SLINT2,PICKPA,SLINE,                                *
*         SPEC1,SPEC2,SPEC3                                                *
*****************************************************************************
C
C  Request double precision for all real values execpt the random
C  number and dimension the arrays.
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION IFLAG(5),Z(10,3),P0(3),P1(3),P2(3),DF(4)
      REAL*4 RN(3)
      EXTERNAL RNSET,RNGET,RNUN,RNSPH
c     EXTERNAL RANDOM
      EXTERNAL PICKP,PICKPA,PICKPC,SLINE,SLINT2,SPEC1,SPEC2,SPEC3
      open(1,FILE='fov.dat',status='old')
      open(2,FILE='fov.out',status='new')
      PI = 3.141592654D0
C
C  Read case parameters
C
```

```
      READ(1,*) RI,RIP,RIS,DH
      READ(1,*) RSP,SPH
      READ(1,*) NEB
      READ(1,*) AFOV,RRFOV,AAP,RRAP
      READ(1,*) (IFLAG(I), I = 1, 5)
      NTH = 1
      IF ((IFLAG(1).EQ.1).OR.(IFLAG(2).EQ.1)) READ(1,*) THETAD,DTHE,NTH
C
      EMAX = 0.1D-04
      ISEED = 1234567
      K  = 3
      LDZ  = 20000
      RIF = DSQRT(RIS*RIS - DH*DH)
C
      DO 1000 ITHE = 1,NTH
      THETA = PI*THETAD/180.D0
      DO 1 I = 1,4
 1     DF(I) = 0.D0
C
C  Loop over the number of energy bundles.
C
      CALL RNSET(ISEED)
C      CALL RNSPH(NEB,K,Z,LDZ)
C
      DO 200 IEB = 1,NEB
C
C  Pick one random number for y-, and z-components of point of origin
C
  300 CONTINUE
      CALL RNUN(2,RN)
c      CALL RANDOM(ISEED,2,RN)
C
C  Call the the subroutine corresponding to the diffuse source.
C
      IF (IFLAG(3).EQ.1) THEN
        CALL PICKP(RIS,P0,RN)
        IF (P0(3).GT.DH) GOTO 300
        GOTO 400
      ENDIF
C
C  Collimated radiation through the solar port.
C  If the incident ray falls outside the FOV limiter, try again.
C
      IF (IFLAG(2).EQ.1) THEN
        CALL PICKPC(RIS,RSP,SPH,P0,RN)
        RT =  DSQRT( P0(1)*P0(1) + (P0(2)/DCOS(THETA))**2 )
        IF (RT.GT.RIF) GOTO 300
        CALL SLINT2(RIS,RSP,SPH,THETA,P0,P1,P2)
        GOTO 500
      ENDIF
C
C  Collimated radiation through the FOV Limiter.
C
      IF (IFLAG(1).EQ.1) THEN
        CALL PICKPC(RIS,RIF,DH,P0,RN)
        CALL SLINT2(RIS,RIF,DH,THETA,P0,P1,P2)
        GOTO 500
      ENDIF
```

```
      IF (IFLAG(4).EQ.1)  CALL PICKPA(RIS,RIP,P0,RN)
      IF (IFLAG(5).EQ.1)  CALL PICKPC(RIS,RIP,0.D0,P0,RN)
      RN(3) = 1.D0
      GOTO 450
C
 400  CONTINUE
C
C  Pick three random numbers to find the receiving point.
C
       CALL RNUN(3,RN)
c      CALL RANDOM(ISEED,3,RN)
C
 450  CONTINUE
      CALL PICKP(RIS,P1,RN)
      IF (RN(3).LT.(0.5D0) ) P1(3) = -P1(3)
 500  CONTINUE
C
C  If Z1 is greater then DH, point P1 is above hemisphere.
C  Increment the DF escaped.
C
      IF (P1(3).GT.DH) THEN
        DF(1) = DF(1) + 1.D0
        GOTO 200
      ENDIF
C
C  If Z1 is negative, point P1 is below hemisphere base.  Then there
C  are two possiblity:
C    1) Energy bundle goes through the cavity,
C    2) Energy bundle hits the aperture plane,
C
      IF (P1(3).LT. (0.D0) ) THEN
C
C  First P1 on aperture plane needs to be found.
C
        CALL SLINE(RIS,P0,P1,P2)
        RT =  DSQRT(P1(1)*P1(1) + P1(2)*P1(2))
        IF (RT.LT.RIP) THEN
          DF(2) = DF(2) + 1.D0
        ELSE
C
C  At the aperture plane the energy bundle is either obsorbed or
C  reflected either diffusily or specularly.
C
           CALL RNUN(1,RN)
c          CALL RANDOM(ISEED,1,RN)
          IF (RN(1).LE.AAP) THEN
            DF(3) = DF(3) + 1.D0
          ELSE
             CALL RNUN(1,RN)
c            CALL RANDOM(ISEED,1,RN)
            IF (RN(1).GT.RRAP) THEN
               CALL RNUN(2,RN)
c              CALL RANDOM(ISEED,2,RN)
              RN(3) = 1.0
              DO 15 I = 1,3
 15              P0(I) = P1(I)
              GOTO 450
            ELSE
```

```
            CALL SPEC1(RIS,P0,P1,P2)
            GOTO 500
          ENDIF
        ENDIF
      ENDIF
      GOTO 200
    ENDIF
C
C   Last possiblity is that the energy bundle hits the inside of the
C   hemisphere.  Pick another random number and compare it to the
C   absorptivity of the FOV limiter
C
     CALL RNUN(1,RN)
c    CALL RANDOM(ISEED,1,RN)
C
C   If absorption has taken place, increment the FOV DF
C
    IF (RN(1).LE.AFOV) THEN
      DF(4) = DF(4) + 1.D0
    ELSE
C
C   Pick another random number to check the type of reflection
C   Diffuse or specular:
C
       CALL RNUN(1,RN)
c      CALL RANDOM(ISEED,1,RN)
      IF (RN(1).GT.RRAP) THEN
      DO 20 I = 1,3
 20     P0(I) = P1(I)
        GOTO 400
      ELSE
        R = DSQRT( P0(1)*P0(1) + P0(2)*P0(2) + P0(3)*P0(3) )
        RER = DABS(R - RIS)
        IF (RER.LE.EMAX) THEN
          CALL SPEC2(RIS,P0,P1,P2)
        ELSE
          CALL SPEC3(RIS,P0,P1,P2)
        ENDIF
        GOTO 500
      ENDIF
    ENDIF
 200 CONTINUE
C
C   At this point all energy bundles for the one FOV limiter node are
C   accounted for.  Write out the results.
C
    IF (IFLAG(1).EQ.1) WRITE(2,105) THETAD
    IF (IFLAG(2).EQ.1) WRITE(2,106) THETAD
    IF (IFLAG(3).EQ.1) WRITE(2,111)
    IF (IFLAG(4).EQ.1) WRITE(2,112)
    IF (IFLAG(5).EQ.1) WRITE(2,113)
    IF (ITHE.EQ.1) THEN
    WRITE(2,120) NEB
    WRITE(2,130) AFOV,RRFOV
    WRITE(2,140) AAP,RRAP
    WRITE(2,150)
    ENDIF
    WRITE(2,100) (DF(I)/NEB ,I = 1, 4 )
```

```
      THETAD = THETAD + DTHE
 1000 CONTINUE
      STOP
C
C            *** FORMAT STATEMENTS ***
C
  100 FORMAT(1X,6D13.5)
  105 FORMAT(3X,'Collimated Beam at an Angle of ',F5.2,' Deg. (FOV limit
     &er)',/)
  106 FORMAT(3X,'Collimated Beam at an Angle of ',F5.2,' Deg. (Solar por
     &t)',/)
  111 FORMAT(3X,'The Diffuse Source is the FOV Limiter.',/)
  112 FORMAT(3X,'The Diffuse Source is the Aperture Plane.',/)
  113 FORMAT(3X,'The Diffuse Source is the Cavity Openning.',/)
  120 FORMAT(3X,'The Number of Energy Bundles used is :',I8,/)
  130 FORMAT(3X,'For the FOV Limiter: Alpha =    ',F5.2,2X,'and Rr = ',F
     &5.2,/)
  140 FORMAT(3X,'For the Aperture Plane: Alpha = ',F5.2,2X,'and Rr = ',F
     &5.2,/)
  150 FORMAT(3X,'The Distribution Factors of the FOV Openning, the Cavit
     &y Openning,',/,3X,'the Aperture Plane, and the FOV Surface are :
     &',/)
      END

*************************************************************************
*                                                                       *
*                    SUBROUTINE PICKP                                   *
*                                                                       *
*  This subroutine picks a point at random on the surface of the        *
*  upper hemisphere.                                                    *
*                                                                       *
*************************************************************************
C
      SUBROUTINE PICKP(RIS,P,RN)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION P(3)
      REAL*4 RN(3)
      PI = 3.141592654D0
C
      SQNT =  SQRT(RN(1))
      THETA = DASIN( SQNT )
      PHI = 2.D0*PI*RN(2)
      P(1) = RIS*DSIN(THETA)*DCOS(PHI)
      P(2) = RIS*DSIN(THETA)*DSIN(PHI)
      P(3) = RIS*DCOS(THETA)
      RETURN
      END

*************************************************************************
*                                                                       *
*                    SUBROUTINE PICKPC                                  *
*                                                                       *
*  This subroutine picks a point at random on the cavity openning.      *
*                                                                       *
*************************************************************************
C
```

```
      SUBROUTINE PICKPC(RIS,R0,Z0,P,RN)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION P(3)
      REAL*4 RN(3)
C
      PI = 3.141592654D0
      PHI = 2.D0*PI*RN(2)
      RN(1) = SQRT(RN(1))
      P(1) = R0*RN(1)*DCOS(PHI)
      P(2) = R0*RN(1)*DSIN(PHI)
      P(3) = Z0
      RETURN
      END

*****************************************************************************  *
*                                                                             *
*                        SUBROUTINE SLINT2                                    *
*                                                                             *
*  This subroutine finds the points of intersection between a line            *
*  passing through P0 and P1 and the surface of a sphere.                     *
*  P0 and P1 become the points of intersection.                               *
*                                                                             *
*****************************************************************************  *
C
      SUBROUTINE SLINT2(RIS,R0,H,THETA,P0,P1,P2)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION P0(3),P1(3),P2(3)
C
      EMAX = .1D-04
      P1(1) = P0(1)
C
      IF (THETA.LT.EMAX) THEN
        P0(3) = DSQRT(RIS*RIS - P0(1)*P0(1) - P0(2)*P0(2))
        P1(2) = P0(2)
        P1(3) = -P0(3)
      ELSE
      P1(2) = 0.D0
      P1(3) = H - P0(2)/DTAN(THETA)
C
      DEN1 = P1(3) - P0(3)
      A = (P1(1) - P0(1))/DEN1
      B = (P1(2) - P0(2))/DEN1
C
      C1 = A*A + B*B
      DEN2 = C1 + 1.D0
      C2 = P0(3)*P0(3)*C1
      C3 = A*P0(1) + B*P0(2)
      BP = (P0(3)*C1 - C3)/DEN2
      C4 = -2.D0*P0(3)*C3 + P0(1)*P0(1) + P0(2)*P0(2)
      CP = (C2 + C4 - RIS*RIS)/DEN2
C
C  Solve for Z-coordinate of intersection points, P0 and P1.
C
      DET = BP*BP - CP
      DET = DSQRT(DET)
      Z2 =  BP + DET
      Z2P = BP - DET
```

```
C
C  Solve for X AND Y of points P0 and P1.
C
      C5 =  (Z2 - P0(3))/DEN1
      X2 =  C5*(P1(1) - P0(1)) + P0(1)
      Y2 =  C5*(P1(2) - P0(2)) + P0(2)
      C5P =  (Z2P - P0(3))/DEN1
      X2P =  C5P*(P1(1) - P0(1)) + P0(1)
      Y2P =  C5P*(P1(2) - P0(2)) + P0(2)
C
      P0(1) = X2
      P0(2) = Y2
      P0(3) = Z2
      P1(1) = X2P
      P1(2) = Y2P
      P1(3) = Z2P
      ENDIF
      RETURN
      END

*****************************************************************************
*                                                                           *
*                      SUBROUTINE PICKPA                                    *
*                                                                           *
*  This subroutine picks a point at random on the aperture plane.           *
*                                                                           *
*****************************************************************************
C
      SUBROUTINE PICKPA(RIS,RIP,P,RN)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION P(3)
      REAL*4 RN(3)
      PI = 3.141592654D0
C
      DR = RIS - RIP
      PHI = 2.D0*PI*RN(2)
      RN(1) = SQRT(RN(1))
      P(1) = (DR*RN(1) + RIP)*DCOS(PHI)
      P(2) = (DR*RN(1) + RIP)*DSIN(PHI)
      P(3) = 0.D0
      RETURN
      END

*****************************************************************************
*                                                                           *
*                      SUBROUTINE SLINE                                     *
*                                                                           *
*  This subroutine finds point P2 on the aperture plane.  P0 and P1         *
*  are on the sphere. All three points form a straight line.                *
*  At the end, P1 is set equal to P2.                                       *
*                                                                           *
*****************************************************************************
      SUBROUTINE SLINE(RIS,P0,P1,P2)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION P0(3),P1(3),P2(3)
C
```

```
      C1 = -P0(3)/(P1(3) - P0(3))
      P2(1) = C1*(P1(1) - P0(1)) + P0(1)
      P2(2) = C1*(P1(2) - P0(2)) + P0(2)
      P2(3) = 0.D0
      DO 10 I = 1,3
   10 P1(I) = P2(I)
      RETURN
      END

*****************************************************************************
*                                                                           *
*                    SUBROUTINE SPEC1                                       *
*                                                                           *
*  This subroutine finds point P2 on sphere.                                *
*  P0 ........ Original Point on sphere.                                    *
*  P1 ........ Reflecting point on the aperture plane                       *
*  P2 ........ Reflected Point on sphere.                                   *
*                                                                           *
*****************************************************************************
C
      SUBROUTINE SPEC1(RIS,P0,P1,P2)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION P0(3),P1(3),P2(3)
C
      CALL SLINT(RIS,P0,P1,P2)
      P2(3) = -P2(3)
      DO 10 I = 1,3
         P0(I) = P1(I)
   10    P1(I) = P2(I)
      RETURN
      END

*****************************************************************************
*                                                                           *
*                    SUBROUTINE SLINT                                       *
*                                                                           *
*  This subroutine finds the point of intersection between a line           *
*  and the surface of a sphere. P2 is on the sphere.                        *
*                                                                           *
*****************************************************************************
C
      SUBROUTINE SLINT(RIS,P0,P1,P2)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION P0(3),P1(3),P2(3)
C
      EMAX = .1D-04
      DEN1 = P1(3) - P0(3)
      A = (P1(1) - P0(1))/DEN1
      B = (P1(2) - P0(2))/DEN1
C
      C1 = A*A + B*B
      DEN2 = C1 + 1.D0
      C2 = P0(3)*P0(3)*C1
      C3 = A*P0(1) + B*P0(2)
      BP = (P0(3)*C1 - C3)/DEN2
      C4 = -2.D0*P0(3)*C3 + P0(1)*P0(1) + P0(2)*P0(2)
```

```
      CP = (C2 + C4 - RIS*RIS)/DEN2
C
C  Solve for Z-coordinate of point P2.
C
      DET = BP*BP - CP
      DET = DSQRT(DET)
      P2(3) = BP + DET
C
C  Look for the right P2
C
      ERR1 = DABS(P2(3) - P0(3))
      ERR2 = DABS(P2(3) - P1(3))
      IF ((ERR1.LT.EMAX).OR.(ERR2.LT.EMAX)) P2(3) = BP - DET
C
C  Solve for X AND Y of point P2.
C
      C5 =  (P2(3) - P0(3))/DEN1
      P2(1) =  C5*(P1(1) - P0(1)) + P0(1)
      P2(2) =  C5*(P1(2) - P0(2)) + P0(2)
      RETURN
      END

*****************************************************************************
*                                                                           *
*                      SUBROUTINE SPEC2                                     *
*                                                                           *
*  This subroutine finds point P2 on sphere.                                *
*  P0 ........ Original Point on sphere.                                    *
*  P1 ........ Reflecting point on the sphere.                              *
*  P2 ........ Reflected Point on sphere.                                   *
*                                                                           *
*****************************************************************************
C
      SUBROUTINE SPEC2(RIS,P0,P1,P2)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION P0(3),P1(3),P2(3)
C
      CALL SPECS(RIS,P0,P1,P2)
C
C  Assign new points to old pints
C
      DO 10 I = 1,3
        P0(I) = P1(I)
 10     P1(I) = P2(I)
      RETURN
      END

*****************************************************************************
*                        ========                                           *
*                       | SPECS |                                           *
*                        ========                                           *
*  This subroutine determines the point of a specular reflection            *
*  within a the surface of a sphere.                                        *
*****************************************************************************
C
```

```
      SUBROUTINE SPECS(RIS,P0,P1,P2)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION P0(3),P1(3),P2(3)
C
      EMAX = 0.1D-04
      DEN = P0(2)*P1(3) - P0(3)*P1(2)
      CA = (-P0(1)*P1(3) + P0(3)*P1(1))/DEN
      CB = (-P0(2)*P1(1) + P0(1)*P1(2))/DEN
      CC = P0(1)*P1(1) + P0(2)*P1(2) + P0(3)*P1(3)
C
      DEN = P1(2)/P1(1) - CA
      A1 = CC/(P1(1)*DEN)
      A2 = (CB - P1(3)/P1(1))/DEN
C
      B1 = (CC - A1*P1(2))/P1(1)
      B2 = -(A2*P1(2) + P1(3))/P1(1)
C
      DEN = A2*A2 +B2*B2 + 1.D0
      BP = (A1*A2 + B1*B2)/DEN
      CP = (A1*A1 + B1*B1 - RIS*RIS)/DEN
C
C  Solve for the Z-coordinate of point P2
C
      DET = BP*BP - CP
      DET = DSQRT(DET)
      P2(3) = -BP + DET
C
C  Look for the right P2
C
      ERR1 = DABS(P2(3) - P0(3))
      ERR2 = DABS(P2(3) - P1(3))
      IF ((ERR1.LT.EMAX).OR.(ERR2.LT.EMAX))  P2(3) = -BP - DET
C
C  Solve for Y AND X of point P2
C
      P2(2) = A1 + A2*P2(3)
      P2(1) = B1 + B2*P2(3)
      RETURN
      END

*************************************************************************
*                                                                       *
*                       SUBROUTINE SPEC3                                *
*                                                                       *
*  This subroutine finds point P2 on sphere.                            *
*  P0 ........ Original Point on the aperture plane.                    *
*  P1 ........ Reflecting point on the sphere.                          *
*  P2 ........ Reflected Point could be anywhere.                       *
*                                                                       *
*************************************************************************
C
      SUBROUTINE SPEC3(RIS,P0,P1,P2)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION P0(3),P1(3),P2(3)
C
C  Find P0' which becomes P2, then set P2 to P0.
C
```

```
      CALL SLINT(RIS,P0,P1,P2)
      DO 10 I = 1,3
10       P0(I) = P2(I)
C
C  Find P2', then P2
C
      CALL SLINT(RIS,P0,P1,P2)
      CALL SPECS(RIS,P0,P1,P2)
C
C  Assign new points to old points
C
      DO 15 I = 1,3
        P0(I) = P1(I)
 15     P1(I) = P2(I)
      RETURN
      END
```

Appendix B.

FOV PROGRAMS

```
*****************************************************************************
*                                                                           *
*                        Program SPA_S                                      *
*                                                                           *
*    This program calculates the illuminated area inside the                *
*    cavity for a collimated beam at each angle of insidence.               *
*    The circular apertures have an eccentricity or Y-sift.                 *
*                                                                           *
*    Calls SPAREA_S                                                         *
*                                                                           *
*---------------------------------------------------------------------------*
*                        NOMENCLATURE                                       *
*---------------------------------------------------------------------------*
*    AREA ........ Illuminated area inside cavity                           *
*    PHI ......... Angle of incidence                                       *
*    PHIMX ....... Maximum Angle of incidence                               *
*    IMAX ........ Number of angle steps                                    *
*    RI .......... Cavity radius                                            *
*    RIP ......... Precision aperture radius                                *
*    RSP ......... Solar port radius                                        *
*    SFT ......... Y-shift between Solar port and precision aperture        *
*    SPH ......... Solar port hight from aperture                           *
*                                                                           *
*****************************************************************************
C
      IMPLICIT REAL*8 (A-H,O-Z), INTEGER*2 (I-N)
      CHARACTER*1 YorN
      open(1,file='spas1.in',status='old')
      open(2,file='spas1.out')

      PI = 3.1415926535897932D0
      ACF = PI/180.D0
      READ(1,*) RIP,RSP,SPH,SFT,PHIMX,IMAX
C
C  Display the current geometry parameters and ask for new ones
C
      WRITE(6,110) RIP,RSP,SPH,SFT,PHIMX,IMAX
      WRITE(6,120)
      READ(5,101) YorN
C
C  Read the FOV  dimensions if there are any changes.
C
      IF (YorN.EQ.'Y' .OR. YorN.EQ.'y') then
        WRITE(6,1)
        WRITE(6,2)
        READ(5,*) RIP
        WRITE(6,3)
        READ(5,*) RSP
        WRITE(6,4)
        READ(5,*) SPH
        WRITE(6,5)
        READ(5,*) SFT
        WRITE(6,6)
        READ(5,*) PHIMX
        WRITE(6,7)
        READ(5,*) IMAX
      ENDIF

      PHIMX = PHIMX*ACF
      DPHI = PHIMX/IMAX
      PHI = -(PHIMX + DPHI)
      N = 2*IMAX + 1
```

```
C
C  Loop over the number of angles
C
      DO 10  I = 1,N
         PHI = PHI + DPHI
         APHI = DABS(PHI)
         CALL  SPAREA_S(RIP,RSP,SPH,SFT,APHI,AREA)
C
C  Write output
C
         WRITE(2,100) PHI/ACF,AREA
   10 CONTINUE
      CLOSE(1)
      CLOSE(2)
      STOP
C
C  The format statements follow:
C
  100  FORMAT(1X,6E13.7)
  101  FORMAT(A1)
  110  FORMAT(//,1X,'The current parameters are:',//,3X,
     & 'Primary aperture radius                    = ',F10.5,' mm',/,3X,
     & 'Seconday aperture radius                   = ',F10.5,' mm',/,3X,
     & 'Vertical distance between the two apertures = ',F10.5,' mm',/,3X,
     & 'Y-Shift between the two aperture           = ',F10.5,' mm',/,3X,
     & 'Maximum angle of incidence                 = ',F10.5,' mm',/,3X,
     & 'Number of angles of incidence              = ',I10)
  120  FORMAT(/,1X,'Would you like to make any changes? (Y/N)')
    1 FORMAT(///,10X,41('*'),/,10X,'*',7X,'Field of view geometry   ',
     &       7X,'*',/,10X,41('*'),///,1X,'THE F-O-V DIMENSIONS ',
     &       'FOLLOW',/,1X,32('='),/)
    2 FORMAT(1X,'Primary aperture radius?  (mm)')
    3 FORMAT(1X,'Seconday aperture radius?  (mm)')
    4 FORMAT(1X,'Vertical distance between the two apertures?  (mm)')
    5 FORMAT(1X,'Y-Shift between the two aperture?  (mm)')
    6 FORMAT(1X,'Maximum angle of incidence?  (mm)')
    7 FORMAT(1X,'Number of angles of incidence? ')
      END

*****************************************************************************
*                                                                           *
*                        Subroutine SPAREA_S                                *
*                                                                           *
*   This program calculates the illuminated area inside the                 *
*   cavity for a collimated beam at each angle of insidence.                *
*   The two circular apertures have an eccentricity or shift.               *
*                                                                           *
*   Called by SPA_S                                                         *
*                                                                           *
*---------------------------------------------------------------------------*
*                        NOMENCLATURE                                       *
*---------------------------------------------------------------------------*
*   AREA ........ Illuminated area inside cavity                            *
*   PHI ......... Angle of incidence                                        *
*   PHIC1 ....... Angles between which aperture is seen as a complete       *
*   PHIC2 ....... ellipse                                                   *
*   PHIS ........ Angle at which both X-axes coincide                       *
*   PHI01 ....... Angles before and after which aperture is not seen        *
*   PHI02 .......                                                           *
*   RI .......... Cavity radius                                             *
*   RIP ......... Precision aperture radius                                 *
```

```
*   RSP ......... Solar port radius                                          *
*   SFT ......... Y-shift between Solar port and precision aperture          *
*   SPH ......... Solar port hight from aperture                             *
*                                                                            *
*****************************************************************************
C
      SUBROUTINE SPAREA_S(RIP,RSP,SPH,SFT,PHI,AREA)
      IMPLICIT REAL*8 (A-H,O-Z), INTEGER*2 (I-N)
      PI = 3.1415926535897932D0
      PHI = DABS(PHI)

      IF (RIP.GT.RSP) THEN
        AREA = 0.D0
        write(6,* ) 'Cannot do this case: R2>R1'
        RETURN
      ENDIF
C
C  Find the angles beforeand after which the aperture is not seen
C
      PHI01 = (RSP + RIP + SFT)/SPH
      PHI01 =  DATAN(PHI01)
      PHI02 = (SFT - RSP - RIP )/SPH
      PHI02 =  DATAN(PHI02)
      IF ( (PHI.GE.PHI01) .OR. (PHI.LE.PHI02) ) THEN
        AREA = 0.D0
        RETURN
      ENDIF
C
C  Find the angle PHIC1 and PHIC2 between which the aperture is seen
C  as a complete ellipse.
C
      PHIC1 = (RSP - RIP + SFT)/SPH
      PHIC1 =  DATAN(PHIC1)
      PHIC2 = (RIP - RSP + SFT)/SPH
      PHIC2 =  DATAN(PHIC2)
      IF ( (PHI.LE.PHIC1) .AND. (PHI.GE.PHIC2) ) THEN
        AREA = PI*RIP**2*DCOS(PHI)
        RETURN
      ENDIF
C
C  Find the angle PHIC at which both X-axes coincide
C
      PHIS =  SFT/SPH
      PHIS =  DATAN(PHIS)
C
C  Find Y-coordinate of the intersection of the 2 ellipses.
C
      COSP = DCOS(PHI)
      Y = SPH*DSIN(PHI) - SFT*COSP
      Y = Y + ((RIP**2 - RSP**2)*COSP**2 )/Y
      Y = 0.5D0*Y
C
C  Find X-coordinate of the intersection of the 2 ellipses.
C
      X = DSQRT(RIP**2 - (Y/COSP)**2)
C
C  Find the area outside the intersection of the 2 ellipses.
C
      CON1 = 2.D0*X*(SPH*DSIN(PHI) - SFT*COSP)
      CON2 = DSQRT(RSP**2 - X**2)
      CON3 = DSQRT(RIP**2 - X**2)
      CON4 = RSP**2*DASIN(X/RSP)
```

```
      CON5 = RIP**2*DASIN(X/RIP)
C
C  X-axis of larger ellipse is at Y>0.
C
      IF (PHI.GT.PHIS) THEN
C
C  Find the the illuminated area when:
C  1) Intersection is below X-axis of smaller ellipse, i.e. Y<0.
C
        IF (Y.LT.0.D0) THEN
          AS1 =  CON1 + COSP*(X*(CON3 - CON2) + CON5 - CON4)
          AREA = PI*RIP**2*COSP - AS1
C
C  2) Intersection is above X-axis of smaller ellipse, i.e. Y>0.
C
        ELSE
          AREA = -CON1 + COSP*(X*(CON3 + CON2) + CON5 + CON4)
        ENDIF
      ELSE
C
C  X-axis of larger ellipse is at Y<0.
C
C  Find the the illuminated area when:
C  1) Intersection is below X-axis of smaller ellipse, i.e. Y<0.
C
        IF (Y.LT.0.D0) THEN
          AREA =  CON1 + COSP*(X*(CON3 + CON2) + CON5 + CON4)
C
C  2) Intersection is above X-axis of smaller ellipse, i.e. Y>0.
C
        ELSE
          AS2 = -CON1 + COSP*(X*(CON3 - CON2) + CON5 - CON4)
          AREA = PI*RIP**2*COSP - AS2
        ENDIF
      ENDIF
C
      RETURN
      END
```

Appendix C.

PSF PROGRAMS

```
*****************************************************************************
**************        J. L. Meekins and  N. E. Tira        ***************
 ********************************************************************
  **                                                              **
   **                ERBE SCANNING INSTRUMENT                    **
   **                 MONTE CARLO ANALYSIS                       **
  **                                                              **
 ********************************************************************
**************        Advisor: J. Robert Mahan        ****************
*****************************************************************************
```


Virginia Polytechnic Institute & State University
Mechanical Engineering Department
Blacksburg, Virginia 24061
May 1991

This system uses the Monte Carlo Method to determine the distribution of radiation within the optics module of the ERBE scanning radiometer.
Diffuse or collimated radiation may be chosen.
Either a gray or a wavelength-dependant analysis may be performed.

The output file MOUDULE.DAT gives the number of energy bundles absorbed by each part of the instrument.
The file PSFOUT.DAT gives the point spread function on the active flake.

```
*****************************************************************************
*                                                                           *
*                          PROGRAM WAVELGNM                                 *
*                                                                           *
* 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.                                     *
*                                                                           *
* Calls: INPUT, RNSET,RANDOM                                                *
* Calls: VECTOR1, BBFCN, OPTICSM, PRINT                                     *
*                                                                           *
* Called by: none                                                           *
*****************************************************************************
*
      PROGRAM WAVELGNM
*
* Request double precision variables.
*
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*4 R
*
* Define storage parameters and variable types.
*
      INTEGER*4 SHTCNT, ABSSHT
      PARAMETER (NP1=40,NP2=37,NP3=3000,NP4=20,NP5=25,NP6=100,NP9=100)
      REAL*8 KAPPA1,KAPPA2
      INTEGER*4 DF(NP1,NP1), PSF(NP6,NP6)
      REAL*8 XN(NP4),YN(NP4),ZN(NP4),X1N(NP4),Y1N(NP4),Z1N(NP4)
      INTEGER*2 NDIV
      external gettim
      COMMON /PSFSTOR/ NDIV,PSF
*
* 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 /COORDS/ XN,YN,ZN,X1N,Y1N,Z1N
      COMMON /DISTR/ DF,II
      COMMON /CHANNEL/ ALAMBDA,NCHANEL,NFLTRSW,NSPCTRM
      COMMON /SPEC/ NUMHITS,MIRTYPE
      COMMON /VECT/ INFLAG
      DATA      NRADTYP/     0/
C
C Open I/O files
C
      OPEN(3,FILE='PSFOUT.DAT')
      OPEN(9,FILE='MODULE.DAT')
*
* Initialize all variables (dimensions).
*
      PI = DACOS(-1.D0)
      ACONV = PI/180.0D0
```

```
      CALL INPUT

      Call gettim(ih,im,is,ihs)
      Cpu =  (ih*60 +im)*60 + is + ihs/100.0
      CALL RNSET(2325745)
*
* Select type of incident radiation (collimated beam or diffuse)
*
      WRITE(6,*)' Enter the number corresponding to the type of radiatio
     &n'
      WRITE(6,*)' you wish to be incident on the scanner optics module.'
      WRITE(6,100)
      WRITE(6,110)
      WRITE(6,120)
      READ(5,*) NRADTYP
*
* Accept a zenith angle
*
      IF (NRADTYP.EQ.2) THEN
        WRITE(6,*)' Enter maximum Zenith angle, ( = 3° for ERBE)'
        READ(5,*) ZTHETA
        ZTHETA =  ACONV*ZTHETA
      ENDIF
*
* 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(6,*)' Specify the angles of the incident collimated radiati
     &on.'

        WRITE(6,*)'  Zenith angle (deg)'
        READ(5,*) PPHI1
        WRITE(6,*)'  Azimuth angle (deg)'
        READ(5,*) PPHI2
C
C Convert angles to radian
C
        THETA = PPHI1 * ACONV
        PPHI2 = PPHI2 * ACONV
      ENDIF
*
* Specify type of analysis to be performed.
*
      WRITE(6,*)' Specify the type of analysis to be performed'
      WRITE(6,170)
      WRITE(6,180)

      READ(5,*) NSPCTRM
*
* Select radiometric channel.
*
 500  WRITE(6,*)' Enter the number corresponding to which'
      WRITE(6,*)' channel of the scanner to be modeled.'
      WRITE(6,*)' If you selected gray analysis, choose total channel on
     &ly.'
      WRITE(6,200)
      WRITE(6,210)
      WRITE(6,220)
      READ(5,*) NCHANEL
      IF (NRADTYP.EQ.0 .AND. NCHANEL.GT.0) GOTO 500
```

```
*
* Select type of mirrors. (100% specular or wavelength-dependent)
*
 510  WRITE(6,*)' Select the type of mirrors used in your analysis'
      WRITE(6,*)' If you selected gray analysis, choose user '
      WRITE(6,*)' defined specularity only.'
      WRITE(6,240)
      WRITE(6,250)
      READ(5,*) MIRTYPE
      IF (NRADTYP.EQ.0 .AND. MIRTYPE.GT.0) GOTO 510
*
* Enter the temperature of the observed scene.
*
      IF(NSPCTRM.EQ.1) THEN
        WRITE(6,*) ' Enter the temperature of the scene (K)'
        WRITE(6,*) '    300 K for earth scene'
        WRITE(6,*) '   5800 K for solar scene'
        READ(5,*) TEMP
      ENDIF
*
* Enter the number of energy bundles to be emitted from the
* aperture of the optics module.
*
      WRITE(6,*)' Enter the number of energy bundles to be emitted'
      WRITE(6,*)' from the aperture of the optics module '
      READ(5,*) NSHOTS
C*
C* Enter the number of divisions on flake (for PSF calculation)
C*
      WRITE(6,*) ' To compute the Point Spread Function, the flake is di
     &vided into NxN squares.'
      WRITE(6,*) '   Enter the number N < 101'
      READ(5,*) NDIV
*
* Initialize necessary data.
*
      IN = NP9 + 1
      II = 1
      ABSSHT = 0
*
* Initialize arguments to be sent to subprograms.
*
      DO 5 I=1,NP1
      DO 5 J=1,NP1
    5   DF(I,J) = 0

      DO 20 J=1,NP4
        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

      DO 30 I = 1, NDIV
      DO 30 J = 1, NDIV
   30     PSF(I,J) = 0
*
* Monitor energy bundle counter.
*
      DO 1 SHTCNT = 1, NSHOTS
```

```
      IF(SHTCNT.EQ.1) WRITE(6,*) 'Running ...'
*
* Find point of emission.
*
        CALL RANDOM (R)
        RCTRD = DREF/2.0D0 * SQRT(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.2) THEN
        CALL RANDOM (R)
        THETA = ZTHETA * R
        CALL RANDOM (R)
        PPHI2 = 2.0D0 * PI * R
      ENDIF
*
* If collimated radiation enters the aperture at a set angle.
*
      IF(NRADTYP.GE.1) THEN
        XS = DSIN(THETA)*DCOS(PPHI2)
        YS = DSIN(THETA)*DSIN(PPHI2)
        ZS = DCOS(THETA)

        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 1
*
* 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 OPTICSM(X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
    1 CONTINUE
*
* Print results
*
      WRITE(9,102)
      WRITE(9,101)
      IF (NRADTYP.EQ.0) WRITE(9,100)
      IF (NRADTYP.EQ.1) THEN
        WRITE(9,110)
        WRITE(9,130) PPHI1
        WRITE(9,140) PPHI2/ACONV
      ENDIF
      IF (NRADTYP.EQ.2) THEN
        WRITE(9,120)
        WRITE(9,150) ZTHETA/ACONV
      ENDIF
      WRITE(9,160)
      IF (NSPCTRM.EQ.0) WRITE(9,170)
      IF (NSPCTRM.EQ.1) THEN
       WRITE(9,180)
       WRITE(9,181) TEMP
      ENDIF
      WRITE(9,190)
      IF (NCHANEL.EQ.0) WRITE(9,200)
      IF (NCHANEL.EQ.1) WRITE(9,210)
      IF (NCHANEL.EQ.2) WRITE(9,220)
      WRITE(9,230)
      IF (MIRTYPE.EQ.0) WRITE(9,240)
      IF (MIRTYPE.EQ.1) WRITE(9,250)
      WRITE(9,290)
      CALL PRINT (ABSSHT,NSHOTS)
*
* Print out non-zero values of PSF
*
      K = -1
      DO 40 I = 1, NDIV
      DO 40 J = 1, NDIV
        IF (PSF(I,J) .NE. 0.D0) THEN
          WRITE(3,300) I, J, PSF(I,J)
        ENDIF
   40 CONTINUE
      CLOSE(3)
      CLOSE(9)
      Call gettim(ih,im,is,ihs)
      Cpu =  (ih*60 +im)*60 + is + ihs/100. -cpu
      write(6,310) cpu
      STOP
*
********************** Format Statements ****************************
*
  102 FORMAT(1X,15X,'*** INPUT INFORMATION ***')
```

```
101 FORMAT(1X,/,5X,'Type of radiation is:')
100 FORMAT(1X,15X,'0 = diffuse')
110 FORMAT(1X,15X,'1 = collimated radiation at user defined angle')
120 FORMAT(1X,15X,'2 = collimated radiation at random angles')
130 FORMAT(1X,15X,'Zenith angle  : ',F6.2,' (deg)')
140 FORMAT(1X,15X,'Azimuth angle : ',F6.2,' (deg)')
150 FORMAT(1X,15X,'Maximum Zenith angle : ',F6.2,' (deg)')
160 FORMAT(1X,/,5X,'Type of analysis is:')
170 FORMAT(1X,15X,'0 = gray')
180 FORMAT(1X,15X,'1 = spectral')
181 FORMAT(1X,15X,'Scene temperature : ',F8.2,' (K)')
190 FORMAT(1X,/,5X,'Type of channel used:')
200 FORMAT(1X,15X,'0 = total channel')
210 FORMAT(1X,15X,'1 = longwave channel')
220 FORMAT(1X,15X,'2 = shortwave channel')
230 FORMAT(1X,/,5X,'Type of mirrors used is:')
240 FORMAT(1X,15X,'0 = user-defined specularity')
250 FORMAT(1X,15X,'1 = wavelength dependent')
290 FORMAT(1X,/,15X,'*** END INPUT INFORMATION ***',/)
300 FORMAT(1X,2I4,I6)
310 FORMAT(1X,'CPU time = ',F8.2)
    END

*************************************************************************** *
*                                                                           *
*                    SUBROUTINE INPUT                                       *
*                                                                           *
* This subroutine queries the user for all applicable dimensions            *
* associated with the optics module of the ERBE scanning radiometer.        *
*                                                                           *
* Called by: WAVELGN                                                        *
* Calls:     CHECK                                                          *
*                                                                           *
*************************************************************************** *
*
    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
C
C Open I/O files
C
    OPEN(7,FILE='MEASURE.DAT')
```

```
*
* Read all data into an array necessary for making any changes in the
* dimensions of the optics module.
*
      READ(7,*) (A(I), I = 1,62)
*
* 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
C      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 values of the absorptivity and reflectivity ratio for the'
     &,/,10X,'reflective conical surfaces are:',//,
     &10X,'(10) abs =',F5.2,5X,'(11) refr =',F5.2,//,10X,
     &'The values of the absorptivity and reflectivity ratio for the'
     &,/,10X,'black conical surfaces are:',//,
     &10X,'(12) abs =',F5.2,5X,'(13) refr =',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 values of the absorptivity and reflectivity ratio for the'
  &,/,10X,'reflective surfaces are:',//,
  &10X,'(22) abs =',F5.2,5X,'(23) refr =',F5.2,//,10X,
  &'The values of the absorptivity and reflectivity ratio for the'
  &,/,10X,'black surfaces are:',//,
  &10X,'(24) abs =',F5.2,5X,'(25) refr =',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)',/,10
  &X,'(30) ref. angle (conical surface)           =',F6.2,' (deg)',//,10
  &X,'The values of the absorptivity and reflectivity ratio for the'
  &,/,10X,'detector housing surfaces are:',//,
  &10X,'(31) abs =',F5.2,5X,'(32) refr =',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 values of the absorptivity and reflectivity ratio for the'
  &,/,10X,'mirrored surface are:',//,
  &10X,'(37) abs =',F5.2,5X,'(38) refr =',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 values of the absorptivity and reflectivity ratio for the'
  &,/,10X,'secondary mirror are:',//,
  &10X,'(43) abs =',F5.2,5X,'(44) refr =',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 values of the absorptivity and reflectivity ratio for the'
  &,/,10X,'insert are:',//,
  &10X,'(48) abs =',F5.2,5X,'(49) refr =',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 values of the absorptivity and reflectivity ratio for the'
  &,/,10X,'reflective surfaces are:',//,
  &10X,'(52) abs =',F5.2,5X,'(53) refr =',F5.2,//,10X,
  &'The values of the absorptivity and reflectivity ratio for the'
  &,/,10X,'black surfaces are:',//,
  &10X,'(54) abs =',F5.2,5X,'(55) refr =',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 values of the absorptivity and reflectivity ratio for
     &the flake are',//,
     &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 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
      PARAMETER (NP9=100)
      REAL*4  Z(NP9), R

      IF(IN.EQ.NP9+1) GO TO 10
      IN = IN+1
      IF(IN.LE.NP9) GO TO 20
   10 CONTINUE
      IN = 1
      NR = NP9
      CALL RNUN (NR,Z)
   20 CONTINUE
      R = Z(IN)
c     open(1,file='rand.tst')
c     wRite(1,*) 'IN, R ',IN ,R
      RETURN
      END
```

```
*****************************************************************************
*                                                                           *
*                    PROGRAM RNUN                                           *
*                                                                           *
*    This program uses the 32-bits based memory cell of a computer          *
*    to generate an N random numbers.                                       *
*    Range: -2147483648 to 2147483647 for a 32-bit memory cell              *
*    Range: -32768 to 32767 for a 16-bit memory cell                        *
*                                                                           *
* IX ... Seed number                                                        *
* IN ... number of desired random points not to exceed N                    *
* RN ... random number dimensioned N                                        *
*                                                                           *
* Called by RANDOM                                                          *
* Calls  RNSET                                                              *
*****************************************************************************
      SUBROUTINE RNUN(NR,RN)
      PARAMETER (NP9=100)
      REAL*4  RN(NP9), FRAND, RQ

      CALL RNSET(ISEED)
      DO 10 I = 1, NR
        RN(I) = FRAND(ISEED,IZ)
        IRN = 6*RN(I) + 1
        ISEED = IZ
        RQ = FRAND(ISEED,IZ)
        IRQ = 6*RQ + 1
        IT = IRN + IRQ
        ISEED = IZ
   10 CONTINUE
      RETURN
      END


      FUNCTION FRAND(IBEG,ITER)
      REAL*4  FRAND
      INTEGER  IBEG,ITER,noflo
      DATA NOFLO/ 2147483647/

      ITER = IBEG*899
      IF (ITER.LT.0) ITER = ITER + NOFLO + 1
      FRAND = FLOAT(ITER)/FLOAT(NOFLO)
      RETURN
      END

      SUBROUTINE RNSET(SEED)
      INTEGER*4 SEED
      SEED = 2325745
      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.                                                        *
*                                                                           *
* Called BY:  WAVELGNM                                                      *
*                                                                           *
*****************************************************************************
```

```
*
      SUBROUTINE VECTOR1 (X1,Y1,Z1,UNX,UNY,UNZ,VOX,VOY,VOZ)
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*4 R
      DATA  THETAM/ 14.314D0/

      PI = DACOS(-1.0D0)
      ZETA = THETAM*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 OPTICSM                                 *
*                                                                       *
* 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.                                                   *
*                                                                       *
* CALLED BY WAVELGNM                                                    *
*                                                                       *
* CALLS: CAP, SPIDER, FILTER, MIDCYL, MIRROR,MOUNT,DETCYL,CONE          *
* CALLS: PRABLA, INSERT, SECMIR, FOVM, HEADER, COORD, RHOSPEC           *
*                                                                       *
*************************************************************************
*
      SUBROUTINE OPTICSM(X1,Y1,Z1,X,Y,Z,VOX,VOY,VOZ)
*
```

```
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
c     REAL*4 R
*
* 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,NFLTRSW,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 PRABLA (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 FOVM(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.                                                        *
*                                                                       *
*  Called by OPTICSM                                                    *
*  Calls  CAPCHK, COORD, RANDOM, REFLECT, VECTOR                        *
*************************************************************************
*
      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)
      REAL*4 R
*
* Define size and type of storage variables.
*
      INTEGER*2 FLAG
      PARAMETER (NP1=40)
      REAL*8 KAPPA1,KAPPA2,APEX(8),V(2)
      INTEGER*4  DF(NP1,NP1)
*
* 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
        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)
      PARAMETER  (NP1=15)
*
* Define storage variables, size and type.
*
      DIMENSION XN(NP1),YN(NP1),ZN(NP1),X1N(NP1),Y1N(NP1),Z1N(NP1)
*
* 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.                                                            *
*                                                                           *
* Called by OPTICSM                                                         *
* Calls    COORD, RANDOM, REFLECT, VECTOR                                   *
*****************************************************************************
*
      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.
*
      PARAMETER (NP1=40)
      INTEGER*2 FLAG
      INTEGER*4  DF(NP1,NP1)
      REAL*8 KAPPA1,KAPPA2
      REAL*4 R
*
* 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
          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
          ELSE
            DF(II,MM+10) = DF(II,MM+10) + 1
          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 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.                                                          *
*                                                                       *
* CALLED BY OPTICSM                                                     *
*                                                                       *
* CALLS: XMIT, COORD, RANDOM, REFLECT                                   *
*************************************************************************
*
      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
      REAL*4 R
*
* 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 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,NFLTRSW,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
*
* Open channel unit file
*
     IF(NUNIT.EQ.11) OPEN(NUNIT,FILE='LWFILTER.DAT')
     IF(NUNIT.EQ.12) OPEN(NUNIT,FILE='SWFILTER.DAT')
     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 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)
      REAL*4 R
*
* Specify storage parameters, size and type.
*
      PARAMETER (NP1=40)
      INTEGER*2 FLAG
      INTEGER*4 DF(NP1,NP1)
      REAL*8 KAPPA1,KAPPA2,T(2)
*
* 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
        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)
    REAL*4 R
*
* Define storage variables, size and type.
*
    PARAMETER (NP1=40)
    INTEGER*2 FLAG
    INTEGER*4 DF(NP1,NP1)
    REAL*8 KAPPA1,KAPPA2,T(2)
*
* 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
        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)
      REAL*4 R
*
* Define storage variables, size and type.
*
      PARAMETER (NP1=40)
      INTEGER*2 FLAG
      INTEGER*4 DF(NP1,NP1)
      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 /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
        FLAG=0
        RETURN
      ENDIF

      IF(R.LE.ABS4) THEN
        DF(II,18+MM) = DF(II,18+MM) + 1
        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)
      REAL*4 R
*
* Define storage variables, size and type.
*
      PARAMETER (NP1=40)
      INTEGER*2 FLAG
      INTEGER*4 DF(NP1,NP1)
      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 /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
        FLAG = 0
        RETURN
      ENDIF

      IF(Z.LE.HDET1) THEN
        DF(II,22) = DF(II,22) + 1
        FLAG = 0
        RETURN
      ELSE
        DF(II,25) = DF(II,25) + 1
        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)
    REAL*4 R
*
* Define storage variables and size.
*
    PARAMETER (NP1=40)
```

```
      INTEGER*2 FLAG
      INTEGER*4 DF(NP1,NP1)
      REAL*8 KAPPA1,KAPPA2,APEX(2)
*
* 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
       FLAG = 0
       RETURN
      ELSE
       DF(II,24) = DF(II,24) + 1
       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 PRABLA                                  *
*                                                                       *
* 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 PRABLA (X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*4 R
*
* Define storage variables, size and type.
*
      PARAMETER (NP1=40)
      INTEGER*2 FLAG
      INTEGER*4 DF(NP1,NP1)
      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 /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
        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
        FLAG = 0
        RETURN
      ELSE
        DF(II,26) = DF(II,26) + 1
        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)
     REAL*4 R
*
* Define storage variables, size and type.
*
     PARAMETER (NP1=40)
     INTEGER*2 FLAG
     INTEGER*4 DF(NP1,NP1)
     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 /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
        FLAG = 0
        RETURN
      ELSE
        DF(II,27) = DF(II,27) + 1
        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)
    REAL*4 R
*
* Define storage variables, size and type.
*
    PARAMETER (NP1=40)
    INTEGER*2 FLAG
    INTEGER*4 DF(NP1,NP1)
    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 /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,NFLTRSW,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
        FLAG = 0
        RETURN
      ENDIF

      CALL RANDOM (R)
      IF(R.LE.ABS7) THEN
        DF(II,21) = DF(II,21) + 1
        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 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)
      REAL*4 R
*
* Specify storage parameters, size and type.
*
      INTEGER*2 FLAG
      PARAMETER (NP1=40,NP3=3000,NP5=25)
c     REAL*8 KAPPA1,KAPPA2,POINT(2,NP3),DF(NP1,NP1)
      INTEGER*4 DF(NP1,NP1)
      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
```

```
c     DIMENSION NSPLOT(NP5,NP5)
c     COMMON /PLOT/ POINT,NSPLOT
      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
          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
          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 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)
      REAL*8 LLAMBDA, LREFR
*
* Interpolate the value of REFR for the wavlelength of the energy bundle
*
      LLAMBDA = 0.0D0
      LREFR = 0.0D0

    1 CONTINUE
*
```

```
* Open mirror unit file
*
      OPEN(13,FILE='MIRREFL.DAT')
      READ(13,*) ULAMBDA,UREFR
*
* Interpolate to find value for TAU
*
      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 FOVM                                    *
*                                                                       *
* This subprogram was modified to include PSF calculation               *
* 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.                                                        *
*                                                                       *
* CALLED BY: OPTICSM                                                    *
*                                                                       *
* CALLS: FOV2, COORD,RANDOM, REFLECT, VECTOR, PSFSUB                    *
*                                                                       *
*************************************************************************
*
      SUBROUTINE FOVM(X1,Y1,Z1,X,Y,Z,FLAG,VOX,VOY,VOZ)
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*4 R
*
* Specify storage parameters, size and type.
*
      INTEGER*2 FLAG
      PARAMETER (NP1=40,NP3=3000,NP5=25)
c     REAL*8 KAPPA1,KAPPA2,POINT(2,NP3),DF(NP1,NP1)
      INTEGER*4  DF(NP1,NP1)
      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

c     DIMENSION NSPLOT(NP5,NP5)
c     COMMON /PLOT/ POINT,NSPLOT
      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( (DABS(CHK1).LE.CONST).AND.(DABS(CHK2).LE.CONST)) THEN
C      IF(CHK1.LE.CONST.AND.CHK2.LE.CONST.AND.CHK1.GE.-CONST.AND.
C     &  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
          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
*
* Record points on the active flake that energy bundles strike.
* May be used to plot image developed on flake.
*
          JJJ =  DF(II,30)
          CALL PSFSUB(FLAKDIM,X,Y)
```

```
          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.                                                        *
* Called by FOVM                                                        *
* Calls  COORD, RANDOM, REFLECT, VECTOR                                 *
*************************************************************************
*
      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)
      REAL*4 R
*
* Specify storage parameters, size and type.
*
      INTEGER*2 FLAG
      PARAMETER (NP1=40,NP3=3000,NP5=25)
c     REAL*8 KAPPA1,KAPPA2,POINT(2,NP3),DF(NP1,NP1)
      INTEGER*4  DF(NP1,NP1)
      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 /DISTR/ DF,II
c     DIMENSION NSPLOT(NP5,NP5)
c     COMMON /PLOT/ POINT,NSPLOT
      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
          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
            CALL PSFSUB(FLAKDIM,X,Y)
          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 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 PSFSUB                                  *
*                                                                       *
* This subroutine computes the point spread function                    *
* on the flake.                                                         *
* Added by N. Tira.                                                     *
*                                                                       *
* CALLED BY: FOVM.                                                      *
*                                                                       *
*************************************************************************
*
    SUBROUTINE PSFSUB(FLAKDIM,X,Y)
```

```
*
* Declare all real and integer variables.
*
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (NP6=100)
      INTEGER*2 NDIV, I, J
      INTEGER*4 PSF(NP6,NP6)
      COMMON /PSFSTOR/ NDIV,PSF

*
* Divide up the flake into equal increments DELX.
*
      DELX  = FLAKDIM/FLOAT(NDIV)
      DELXH = DELX*0.5D0
      XMAX  = FLAKDIM*0.5D0 - DELXH
*
* Transform x and y thru a 45 deg rotation
*
      CONST = DSQRT(2.D0)/2.D0
      XT =  X*CONST + Y*CONST
      YT = -X*CONST + Y*CONST
*
* Loop over the locations on the flake from left to right and bottom up
*
      DO 10 I = 1, NDIV
        XI = -XMAX + (I-1)*DELX
        DO 10 J = 1, NDIV
          XJ = -XMAX + (J-1)*DELX
          IF ( (XT.GE.XI-DELXH) .AND. (XT.LT.XI+DELXH) .AND.
     &         (YT.GE.XJ-DELXH) .AND. (YT.LT.XJ+DELXH) ) THEN
            PSF(I,J) = PSF(I,J) + 1
          ENDIF
 10   CONTINUE
      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.                                          *
*                                                                       *
* Called by: WAVELGN                                                    *
*                                                                       *
* CALLS: ROMB, NEWTON, XUFLOW                                           *
*************************************************************************
*
      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
c I found 1 CONTINUE to be missing
  1   CONTINUE
      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 &mu.m-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 &mu.m-K instead of 1897.8 &mu.m-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.
*
cc    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
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
      SUBROUTINE ROMB(FCN,A,B,TOL,RESULT)
      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(6,200)
*
*     Print intermediate results
*
 50    I = I+1
       IF(IPRINT.EQ.0) GOTO 80
       DO 70 L = 1,I
        WRITE(6,203) (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 WERE ')
 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.
*
cc    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).EQ.TOL) THEN
        OLDERR = ERROR
        NFLAG = 1
      ENDIF

      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.                                                               *
*                                                                           *
*  Called by WAVELGNM                                                       *
*****************************************************************************
*
      SUBROUTINE PRINT (ABSSHT,NSHOTS)
*
* Declare all real variables double precision.
*
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (NP1=40)
*
* Specify size and type of storage parameters.
*
      INTEGER*4 DF(NP1,NP1), ABSSHT,ESCSHT,COUNT1,COUNT2,COUNT3,NSUM
*
* 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)

      COUNT3 = DF(II,26) + DF(II,33)
      WRITE(9,600) COUNT3,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(1X,/,10X,'NUMBER OF ENERGY BUNDLES ABSORBED',/)
  200 FORMAT(5X,'Reflector cap, part',I3,':',I6)
  300 FORMAT(5X,'Perimeter of middle of secondary mirror mount :',
     &I6,/,5X,'Perimeter of secondary mirror :',16X,I6,/,5X,
     &'Conical portion of secondary mirror mount :',4X,I6,/,
     &5X,'Cylindrical portion of secondary mirror mount :',I6,/,
     &5X,'Rim of secondary mirror mount towards flake :  ',
     &I6,/,5X,'Detector housing, cylinder 1 : ',I6,/,5X,
     &'Detector housing, cone 1 :',5X,I6,/,5X,
     &'Detector housing, cone 2 :',5X,I6,/,
     &5X,'Detector housing, cylinder 2 : ',I6,/,5X,
     &'Primary insert :',15X,I6,/,5X,
     &'Top of secondary mirror mount :',I6)
  400 FORMAT(5X,'Top of spider leg number',I3,':',3X,I6)
  500 FORMAT(5X,'Bottom of spider leg number',I3,':',I6)
  600 FORMAT(5X,'Primary mirror :',15X,I6,/,
     &5X,'Secondary mirror :',13X,I6)
  700 FORMAT(5X,'Field of view limiter, top :',3X,I6,/,5X,
     &'Field of view limiter, bottom :',I6,/,5X,
     &'Active Flake :',I6,/)
  800 FORMAT(5X,'Number of energy bundles absorbed :',I8,/,5X,
     &       'Number of energy bundles escaped : ',I8,/)
  900 FORMAT(5X,'Aperture :',I6)
  999 FORMAT(5X,'Number of shots missing flake :',6X,I6)
      RETURN
      END
```

Vita

Nour Eddine Tira was born May 22, 1961 in Menzel-Temine, Tunisia. In June of 1981, he obtained his *baccalauréat* from the *Lycée Technique de Nabeul* with *mention bien*. Because of his impressive performance in the *baccalauréat*, he was first offered a scholarship to study in Montpelier, France. He declined this offer when he was granted another scholarship to study in Baton Rouge, Louisiana, USA. He obtained a Bachelor of Science in December 1985, from the Mechanical Engineering Department at Louisiana State University and A. & M. College. In January of 1986, he began his graduate studies at Virginia Polytechnic Institute and State University (Virginia Tech), and earned a Master of Science from the Mechanical Engineering Department in December of 1987. He has been conducting research in support of NASA's Earth Radiation Budget Experiment (ERBE) under the supervision of Professor J. R. Mahan since entering graduate school at VPI & SU. Dr. Tira has been the author and coauthor of several papers on radiometry. During the year 1989, he served as the mechanical engineering department representative to the Graduate Student Assembly. Finally, in December of 1991 he obtained a Ph. D. degree from the same department at Virginia Tech.

Nour Eddine Tira