To the Graduate Council:

I am submitting herewith a dissertation written by Itacil Chiari Gomes entitled "A Generalized Albedo Option for Forward and Adjoint Monte Carlo Calculations." I have examined the final copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Nuclear Engineering.

Paul N. Stevens, Major Professor

We have read this dissertation
and recommend its acceptance:

Accepted for the Council:

Associate Vice Chancellor
and Dean of The Graduate School

# A GENERALIZED ALBEDO OPTION FOR FORWARD AND ADJOINT MONTE CARLO CALCULATIONS

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee

Itacil Chiari Gomes

May 1991

Dedicated to the Memory of my Mother

Lady Chiari Gomes

## ACKNOWLEDGMENTS

The author wishes to thank his major professor Dr. Paul N. Stevens for his guidance, understanding, and encouragement throughout this work. Dr. Shannon and the Fusion Engineering Design Center staff provided support and a friendly working environment. Also, Oak Ridge National Laboratory research scientists M. B. (Peggy) Emmett and Wayne Rhoades provided very important technical assistance during critical phases of this work.

This research was sponsored by a subcontract between the Department of Nuclear Engineering of the University of Tennessee and the Fusion Engineering Design Center of Oak Ridge National Laboratory.

**ABSTRACT**

The advisability of using the albedo procedure for the Monte Carlo solution of deep penetration shielding problems which have ducts and other penetrations is investigated. It is generally accepted that the use of albedo data can dramatically improve the computational efficiency of certain Monte Carlo calculations - however the accuracy of these results may be unacceptable because of lost information during the albedo event and serious errors in the available differential albedo data. This study has been done to evaluate and appropriately modify the MORSE/BREESE package, to develop new methods for generating the required albedo data, and to extend the adjoint capability to the albedo-modified calculations. The major modifications include an option to save for further use information that would be lost at the albedo event, an option to displace the emergent point during an albedo event, and an option to read spatially-dependent albedo data for both forward and adjoint calculations - which includes the emergent point as a new random variable to be selected during an albedo reflection event. The theoretical basis for using TORT-generated forward albedo information to produce adjuncton-albedos is

derived. The MORSE/STORM code was developed to perform both forward and adjoint modes of analysis using spatially-dependent albedo data. The results obtained using the MORSE/STORM code package for both forward and adjoint modes were compared with benchmark solutions - excellent agreements along with improved computational efficiencies were achieved demonstrating the full utilization of the albedo option in the MORSE code.

## TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

# INTRODUCTION

The albedo for a reflecting medium is defined as the ratio between the emergent current and the incident current. The use of this concept can simplify the solution of radiation transport problems by allowing part of the geometry to be represented by albedo surfaces. Significant savings in computer time can be achieved as well as a better description of the radiation field within specific regions of interest.

Monte Carlo calculations can benefit from the albedo concept since the computation-time-intensive tracking of particles is not performed inside albedo regions such as the walls of ducts and cavities. But this advantage is not realized without some cost, the generation of albedo data is very expensive and the amount of the data can be prohibitively large.

The albedo procedure is one of the options available in the MORSE [1] Monte Carlo code. The multigroup energy format of the MORSE code facilitates the coupling of the albedo information generated by discrete ordinates with the Monte Carlo code.

On the basis of early experiences with applications

of the albedo procedure to Monte Carlo analyses, it was recommended that the use of albedos should be restricted to problems in which the characteristic dimensions of the penetrations are large when compared with the radiation particle's mean-free-path in the wall material. To better understand the reasons behind this restriction and to develop a basis for overcoming this problem, a set of benchmark-quality full-transport calculations was performed using the MORSE and MCNP [2] codes for a wide range of problems. The results were compared with results from Monte Carlo/albedo calculations. It was found that the albedo driven results, despite achieving solutions with essentially zero standard deviations and requiring reasonable computer times, performed very poorly in terms of accuracy. The albedo driven solutions were found to consistently underestimate the benchmark results - in some cases by orders-of-magnitude. As a first step to generalize the albedo procedure, the standard MORSE/BREEZE [3] package was modified to recover some of the information lost at the albedo events. The accuracies of the results calculated with the modified version of the code were greatly improved, however there remained a consistent deviation of about 20 percent (low) with respect to the benchmark results. A plausible reason which could explain this deviation was that the albedo data used were not correctly representing the true reflection process. A series of MCNP calculations was performed to benchmark the standard

albedo data. The Monte Carlo results and the standard albedo azimuthal distributions were found to be significantly different. The probable reason for this is that the DOT angular flux files contained negative angular fluxes for a few combinations of energy groups and directions of the quadrature set.

The TORT [4] three-dimensional discrete ordinates code was modified and used to generate a new set of albedo data, which includes a spatial variable for the point-of-emergence. The characteristic method for the flux sweep was used to ensure positive numerical values for the angular flux for all energy groups and discrete directions. An experimental package of codes called MORSE/STORM was developed to perform the formatting, manipulation and utilization of the generated albedo data. When using spatially-dependent albedo data from the modified version of the TORT code, the subroutines and interface codes which comprise the BREESE system of the MORSE/BREESE package are replaced by the STORM system routines.

The new procedures described in this dissertation significantly broaden the applicability of albedo procedures. The fact that the use of biasing techniques was not formally investigated in this research offers the real prospect for future improvements in the figure-of-merit through the use of appropriate biasing techniques.

In Chapter I, a historical review of the use and

calculation of the albedo information is described. Chapter II presents the fundamentals associated with the albedo concept along with descriptive definitions of various kinds of albedos. In Chapter III, the available albedo data are described and sampling procedures are discussed. Chapter IV describes of the modifications introduced into the MORSE/BREESE package, the new method for calculating spatially-dependent albedo data, the development of the MORSE/STORM package, the derivation of the reciprocity relationship between forward and adjoint albedos, and the implementation of the adjoint capability into the MORSE/STORM package. In Chapter V, the calculations performed and their results are presented. Finally, the conclusions of this research and recommendations for future work are presented in Chapter VI.

# CHAPTER I

## HISTORICAL REVIEW

Albedos are cited in several academic books [5,6,7] in which solutions of the transport equation are described. The albedo concept has been used to simplify some analytical solutions of the diffusion equation as well as being an important tool in transport theory analyses. Analytical functions for albedos have been proposed either based on simplified theoretical considerations or by mathematically fitting calculated or experimental data. Wade E. Selph [8] provides an overview of early efforts made to obtain albedo data both experimentally and through Monte Carlo or discrete ordinates calculations. Fitting functions for albedos are mostly derived for a particular range of energy and a few materials. The albedo boundary condition is available as a standard input option in most discrete ordinates codes. The generation of point albedo data is also provided as an option in the DORT discrete ordinates code [9].

The implementation of the albedo procedure in Monte Carlo codes has been investigated for several years. The first studies for calculating albedo data for use in the

Monte Carlo random walk were carried out during the early sixties. At that time the limitations in terms of computer size and speed were important constraints. Studies conducted by Maerker and Muckenthaler [10] and by Allen, Futterer, and Wright [11] showed the importance of using a calculation method which provides a detailed description of the albedo reflection process. The Maerker and Muckenthaler's studies have incorporated not only the angular dependence of the albedo data but also the energy dependence. Coleman [12] studied albedo data for neutrons of intermediate energy generated with the OR5 Monte Carlo code [13]. In this study a formal description and analysis of the variables influencing the albedo data are provided as well as the formal description of the Monte Carlo next-event estimation for generating albedo data.

In the early seventies the MORSE/BREESE package was developed. This work brought to light the importance, necessity, and difficulties of having a complete description of the energy as well as the angular dependence of the albedo data. The spatial dependence of the point-of-emergence was not included due to the use of a two-dimensional discrete ordinates code to generate the albedo data. The MORSE Monte Carlo code was modified to accommodate the albedo information generated by the DOT discrete ordinates code [14]. A complete code package was developed which included CARP [15] - a code which handles the angular flux file generated by

the DOT code; BREESE - a code which uses the output binary file from the CARP code to construct the probability tables in a form suitable for use in the MORSE code; and the MORSE code itself - which was modified to reflect the particles reaching an albedo surface according to the albedo probability tables. Several studies were conducted to benchmark the results produced by the albedo procedure [16,17]. The analyses done in these studies indicated that the albedo driven results are very sensitive to modeling schemes and to match experimental results was not a straight forward task - thus establishing that the MORSE/BREESE albedo option is less than completely reliable.

During the eighties, the Japanese have shown an increasing interest in the albedo procedure [18,19]. Some of the studies have lead to a modified version of the MORSE/BREESE package called MORSE-ALB [20]. The MORSE-ALB code system comprises a modified version of the MORSE/BREESE package, the one-dimensional transport code SLDN-SL3 [21] for generating albedo data, as well as interface codes to couple forward Monte Carlo albedo calculation and discrete ordinates adjoint calculation to estimate effect-of-interests at regions far away from the exit of the duct/cavity under consideration. The use of a one-dimensional code to generate the albedo data implicitly neglects the azimuthal dependence, as well as neglecting the location of the emergent point (assuming the point-of-emergence coincident with

the point-of-incidence).

Most albedo methods developed so far make use of the so called "point albedo", which is generated such that the point-of-emergence from an albedo event is the same as the point-of-incidence. In Coleman's research, a Monte Carlo study was carried out to characterize the distribution of distances between the points of incidence and emergence. For incident neutron energies below 0.2 Mev, it was shown that most of the particles emerge at distances less than 10 cm from the point-of-incidence and that the emergent distances were a function of the incident energy and therefore this effect was neglected. A literature search indicated that no further studies were conducted to better define the influence of the emergent distance on the Monte Carlo estimate of the effect-of-interest.

The adjoint mode of Monte Carlo analysis which includes the albedo random walk capability is not provided by any available code, and its underlying formal basis has not been discussed in the literature. The availability of this option will facilitate solutions of problems with penetrations and different source terms at the entrance of the penetration. The MORSE code has the adjoint mode already implemented, but the MORSE/BREESE package does not take advantage of this option due to limitations in the basic albedo data set (this point is discussed in details in chapter IV).

## CHAPTER II

## ALBEDOS

Unlike the reflection of the light which is mainly a surface phenomenon, neutrons and gamma-rays are scattered back from inside the material - sometimes from depths of several mean-free-paths and after several collisions. Figure II.1 shows a few possible tracks which a particle may experience inside a material medium. For neutrons, the scattering from nuclides can be elastic or inelastic, and Compton scattering from electrons is dominant for photons. As a result the particle can emerge through the same surface it has entered. This can be described in the macroscopic sense by a reflection coefficient usually referred to as the albedo. The albedo for a reflecting medium is defined as the ratio between the current out to the current in

$$\alpha(E_0,\theta_0;E,\theta,\phi) = \frac{J(E,\theta,\phi)}{J(E_0,\theta_0)} , \qquad \text{(II.1)}$$

where $J(E_0,\theta_0)$ is the incident current, $J(E,\theta,\phi)$ is the emergent current and $\alpha(E_0,\theta_0,E,\theta,\phi)$ is the albedo as shown in Figure II.2.

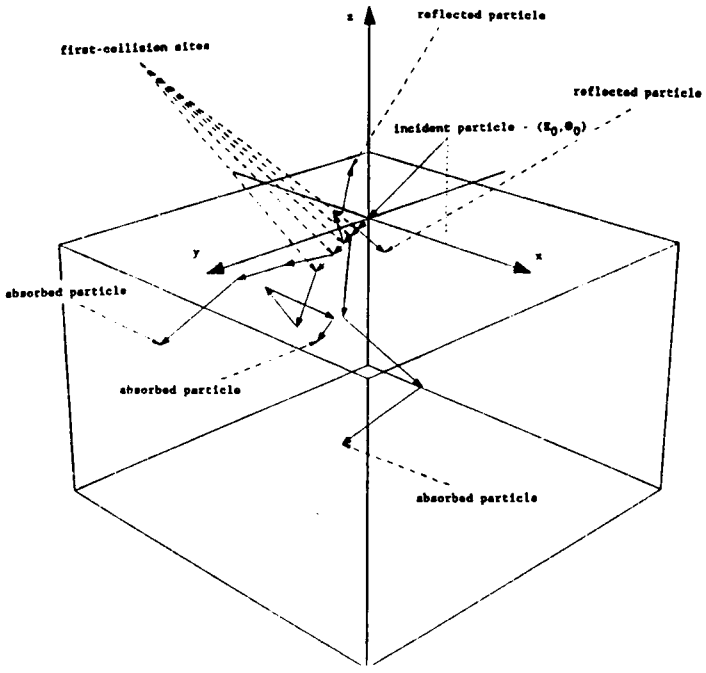



Figure II-1. Few Possible Tracks Which a Particle May Experience Inside a Material Medium.

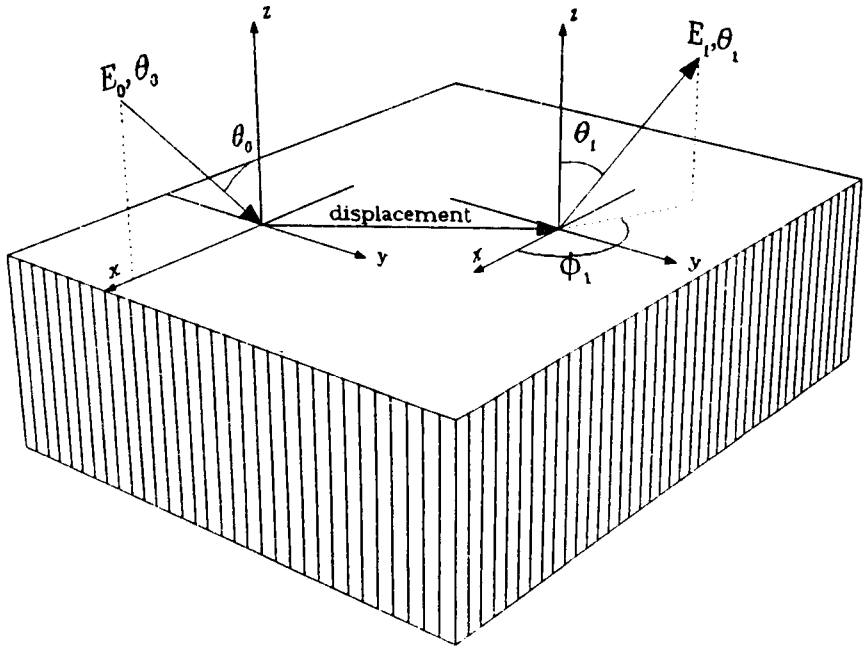



Figure II-2. Albedo Reflection Process and Related Phase Space Coordinates.

If the spatial dependence of the emergent point is included, the equation for the albedo becomes:

$$\alpha(E_0,\mathbf{r}_0,\theta_0;E,\mathbf{r},\theta,\phi) = \frac{J(E,\mathbf{r},\theta,\phi)}{J(E_0,\mathbf{r}_0,\theta_0)}, \qquad \text{(II.2)}$$

where $\mathbf{r}$ represents the point of emergence and $\mathbf{r}_0$ represents the point of incidence.

Albedos are available in several forms. The form used in Equation II.1 is doubly-differential; that is, differential with respect to both the reflected energy E and the reflected direction (as described by $\theta$ and $\phi$). Equation II.2 represents a triply-differential albedo which includes the dependence on the spatial point of emergence. The doubly-differential albedo may be obtained from the triply-differential albedo through integration over the spatial variable. A singly-differential albedo results when a doubly-differential albedo is integrated over either energy or direction.

The portion of the surface through which the backscattered particles leave the reflecting medium can be called the reflecting region. If a beam of particles is incident on the surface of a reflecting medium, the reflecting portion of this surface is always larger than the intersection of the beam with the surface. If the dimensions of the reflecting portion of the surface are considerably smaller than the distance from the reflecting region to the

detector, the dimensions of the reflecting region can be neglected and the reflecting region may be considered coincident with intersection of the incident beam with the surface. In such cases, the point-of-emergence can be considered coincident with the incident point and a knowledge of the doubly-differential albedo is sufficient. The available albedo data are in fact "point" albedos. Point albedos are doubly-differential and should be used only for problems in which the distance to the detector is large as compared with the dimensions of the reflecting region. This restricts the application of doubly-differential albedos to regions which are far way from the detectors, or, in the case of ducts and cavities, to problems for which the diameter of the duct is large enough to comply with the above mentioned constraint.

Other representations of albedo data are:
the differential number albedo

$$\alpha_{Nt}(E_0,\theta_0;\theta,\phi) = \int_{E=0}^{E=E_0} \alpha(E_0,\theta_0,E',\theta,\phi)\, dE', \qquad (II.3)$$

the polar differential number albedo

$$\alpha_{Na}(E_0,\theta_0,\theta) = \int_{\phi=0}^{\phi=2\pi} \alpha_{Nt}(E_0,\theta_0,\theta,\phi)\, d\phi\ , \qquad (II.4)$$

and the total number albedo

$$\alpha(E_0,\theta_0) = \int_{\mu=0}^{\mu=1} \alpha_{Na}(E_0,\theta_0,\theta)\, d\mu \ , \qquad \text{(II.5)}$$

where $\mu = \cos\theta$.

The use of these albedos which by their nature are less descriptive, can speed up the sampling process in a Monte Carlo calculation if the process of selection of the variables is performed using marginal and conditional probability distribution tables ( this point is discussed in more detail in the Chapter 4).

## CHAPTER III

## ALBEDO DATA AND SAMPLING PROCEDURES

Albedo data are made available in several different forms. Empirical or analytical formulae would be very convenient for Monte Carlo calculations, but due to the detail /resolution required for the accurate solution of complex problems the available albedo formulations are not suitable. The most common way to obtain differential albedo information is through two-dimensional discrete ordinates calculations. The problem usually solved consists of a reflecting semi-infinite medium with an incident monodirectional beam of particles having energies within a particular energy group. A run is performed for each combination of discrete incident polar direction and energy group. The angular and energy distributions of the reflected particles are calculated and these distributions are expressed as a reflected current per unit incident current. A Monte Carlo code, such as the MORSE code can use this albedo information to reflect particles which reach an albedo surface. The phase space coordinates of the reflected particle are selected according to probability distribution tables which are constructed

from the angular flux output of the discrete ordinates code. These point albedo data do not have any information about the spatial distribution of the emergent particles.

The MORSE/BREESE package was developed to make use of the albedo information to simplify the Monte Carlo generated random walks. The BREESE module comprises of the subroutines ALBIN, ALBDO, ALBDOE, and THETO as well as a separate program which writes a tape of albedo data in the proper format for input into the MORSE code. The subroutine ALBIN reads the albedo tape and sets the pointers for the common blank. The subroutine ALBDO performs the reflection process, sampling from the probability tables for a given incident polar angle (as defined by the subroutine THETO), the outgoing energy group and direction. The subroutine ALBDOE estimates the statistically defined next-event contribution to the detectors from each albedo event. The cumulative distribution functions (cdf's) are constructed for each of the statistically-dependent variables (outgoing polar angle, azimuthal angle and energy), to avoid spending too much computer time in searching a single joint cdf describing the albedo probabilities.

The total albedo $\alpha(E_0,\theta_0)$ may be described as the probability of having a particle reflected from the reflecting medium (independent of the outgoing energy and angle). The $\alpha(E_0,\theta_0)$ is used to adjust the weight of the particle which undergoes an albedo event. The weight of the incident

particle is multiplied by the total albedo. This correction of the weight of the particle is statistical in nature, and represents the average number of particles emerging from a reflecting medium due to an incident particle with a particular energy and direction $(E_0,\theta_0)$.

To select the outgoing phase space coordinates (group g, $\theta$, $\phi$) the following cdf's are constructed. The marginal cdf for the outgoing polar angle is given by

$$cdf_\theta(E_0,\theta_0;\theta) = \frac{\int_{\mu'=0}^{|\mu|} d|\mu'|\ \alpha_{Na}(E_0,\theta_0,\theta)}{\alpha(E_0,\theta_0)}, \quad \text{(III.1)}$$

where $\mu = \cos\theta$, $\alpha_{Na}(E_0,\theta_0;\theta)$ is the polar differential number albedo (as defined by Equation II.5), and $\alpha(E_0,\theta_0)$ is the total albedo.

The conditional cdf for the azimuthal angle within the polar angle sampled from the previous cdf is defined as:

$$cdf_\phi(E_0,\theta_0,\theta;\phi) = \frac{\int_0^{\phi} d\phi'\ \alpha_{Nt}(E_0,\theta_0;\theta,\phi)}{\alpha_{Na}(E_0,\theta_0,\theta)} \quad \text{(III.2)}$$

where $\alpha_{Nt}(E_0,\theta_0;\theta,\phi)$ is the differential number albedo (as defined by Equation II.4), and $\alpha_{Na}(E_0,\theta_0;\theta)$ is the polar differential number albedo.

The doubly-conditional cdf for the energy for the previously defined outgoing direction is:

$$cdf_E(E_0,\theta_0,\theta,\phi;E) = \frac{\int_{E'=0}^{E} dE'\ \alpha(E_0,\theta_0;E,\theta,\phi)}{\alpha_{Nt}(E_0,\theta_0;\theta,\phi)} \qquad \text{(III.3)}$$

where $\alpha(E_0,\theta_0;E,\theta,\phi)$ is the doubly conditional differential albedo.

The above definitions for the cdf's are for continuous variables, but for albedos generated with discrete ordinates codes the energy and angular variables are discretized, thus values associated to discrete segments of those variables are sampled instead.

In the MORSE/BREESE package, the subroutine ALBIN reads the cdf's and the subroutine ALBDO makes the selection of the outgoing phase space coordinates. First, the subroutine THETO calculates the cosine of the incident polar angle $\theta$ as given by

$$\cos\theta = u_n\,u + v_n\,v + w_n\,w\ , \qquad \text{(III.4)}$$

where $u_n,v_n,w_n$ are defining cosines of the normal to the surface, and u,v,w are direction cosine of the incoming particle.

The incident polar angle "$\theta$" is compared with the limits of each incoming angular polar bin of the albedo

data, and the bin to which $\theta$ belongs is identified. The discrete form of the $cdf_{\theta}$ (Equation III.1) is used to select an outgoing polar angle bin. The outgoing polar angle is then selected from an equally probable distribution between the limits of the selected polar angle bin. The outgoing azimuthal angle $\phi$ bin is selected from the conditional cumulative probability distribution $cdf_{\phi}$ (Equation III.2). The outgoing azimuthal angle is, as for the polar angle, selected from an equally probable distribution between the limits of the selected azimuthal angle bin. Finally, the outgoing energy bin is selected from the doubly-conditional $cdf_{E}$ (Equation III.3). In case the group structure for the cross-sections is different from that of the albedo data, the outgoing energy group is sampled from an equally probable distribution over lethargy.

A limitation of the sampling scheme implemented in the BREESE code is that energy cutoff can not be used because the angular distribution from which the sampling process is initiated is a summation over all possible outcomes, and the fair game is not preserved when using energy cutoff unless the angular distribution for each outgoing energy group is exactly the same. It is important to notice that the albedo data are usually generated for coupled (neutron-gamma ray) problems. Then the use of an angular distribution which is associated with the summation over all energy groups as the starting point for the sampling process

limits the applicability of energy cutoff as well as energy biasing during the random walk. Angular biasing is still a fair game with the implemented the sampling scheme.

From an analysis of an albedo data set distributed by the Radiation Shielding Information Center, the presence of negative angular fluxes in the basic data set was noticed. These negative angular fluxes cannot be avoided due to the low order expansion of the scattering cross sections. These negative fluxes show up mainly in the high energy groups for the forward component of the outgoing angular flux. This fact leads to an undesirable distortion in the description of the reflection process and to an underestimation of the forward component - which is usually the most important component of the radiation field being analyzed. The BREESE routines treat these negative fluxes as if they were a part of the true solution. Since the negative fluxes lead to negative probabilities, the sign of the particle's statistical weight is reversed in ranges where the probability density function is negative.

# CHAPTER IV

## MODIFIED RANDOM WALK USING ALBEDO DATA

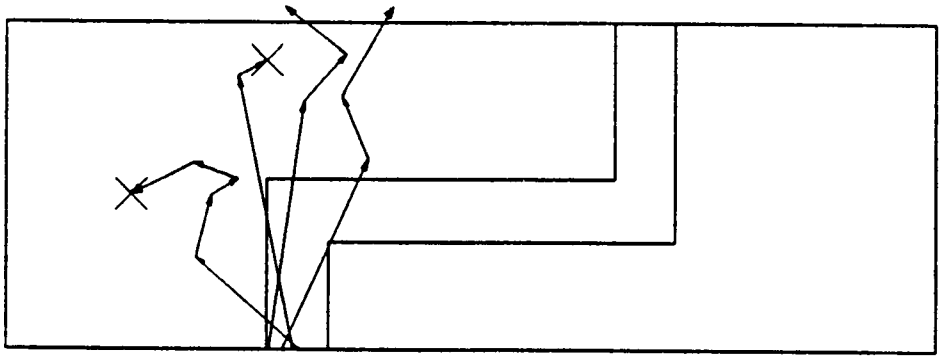
The original MORSE/BREEZE package was designed to perform the additional computational steps necessary to accomplish the albedo-modified random walk. The procedure consists of following the particles only within zones which are enclosed by the albedo surfaces. The computational time is minimized because of the simplified tracking (ray tracing) between albedo events and because much of the random walk is in effect accomplished deterministically. Problems in which streaming through ducts and passageways is the most important component of the radiation field, can be solved very efficiently using the albedo option.

The available MORSE/BREEZE code package is able to handle forward calculations which combine albedo reflection and full transport for selected regions. The estimation of effects of interest (responses of detectors that view an albedo surface) may be accomplished using either next-event estimation or analog boundary crossing. At an albedo surface, the particle suffers a collision-like event in which the weight of the incident particle is reduced by the total

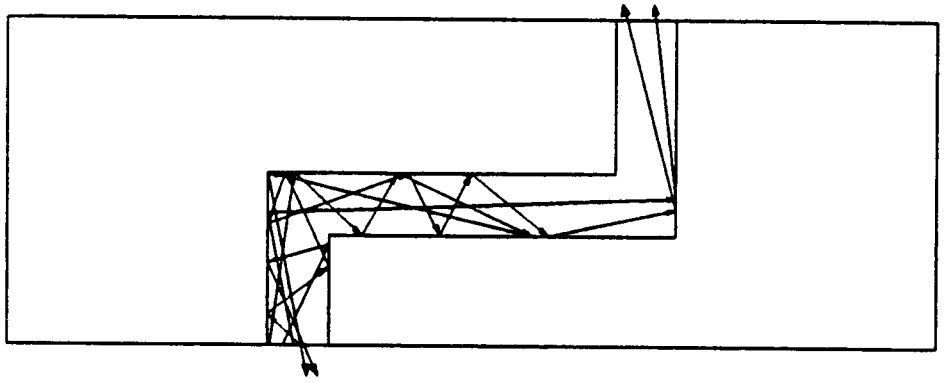
albedo and the outgoing energy and direction are selected according to the doubly-differential albedo. The albedo-modified random walk procedure used by the MORSE code does not allow particle transport inside albedo regions. As a consequence, estimation of the radiation field at positions behind albedo surfaces using the statistical estimators at ALBDOE is incorrectly done. To correct this problem, a modification was introduced into the MORSE code which allows particles (pseudo-particles) to cross the albedo surfaces. A schematic comparison of the three possible transport modes (full, albedo, and pseudo-particle) is shown in Figure IV-1. The pseudo-particle modification increases the computational time, but gains may be obtained in terms of more precise results and a more complete description of the radiation field for use in statistical estimation. The computer time required can be much less than for the full transport case because of the advantage gained when particles are driven deterministically by the albedo reflections through the duct with pseudo-particles being created at positions not easily reached by normal transport.

## MODIFICATIONS INTRODUCED INTO THE MORSE/BREESE PACKAGE

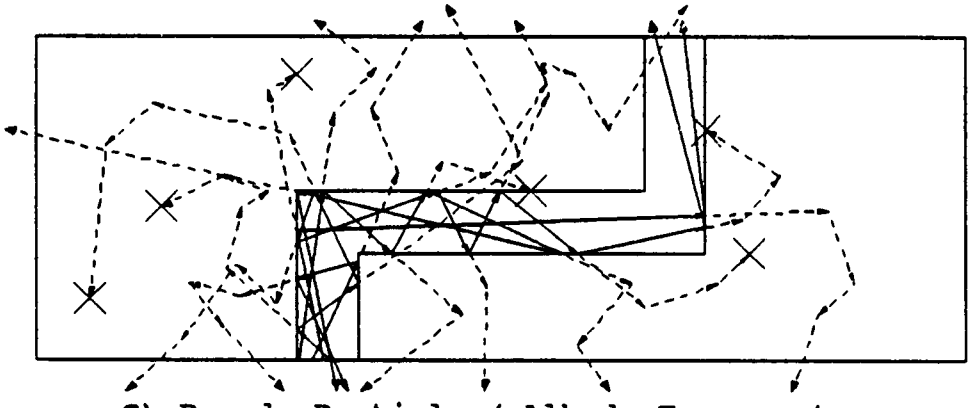
The standard version of the MORSE/BREESE package is unable to estimate to points behind an albedo surface.



A) Full Transport



B) Albedo Transport



C) Pseudo-Particle / Albedo Transport.

Figure IV-1. Graphic Comparison of A) Full Transport, B) Albedo Transport, C) Pseudo-Particle/Albedo Transport Mode.

This limitation is due to the fact that collisions that could have occurred inside the albedo medium and statistically contributed to effects-of-interest outside the albedo surface envelope cannot be represented by the reflected portion of the particle. To overcome this problem the pseudo-particle modification was introduced into the MORSE code to allow the transport of particles inside an albedo medium and the estimation of effects-of-interest outside the albedo surface envelope. The albedo reflection process is kept unmodified and a pseudo-particle may be created at each albedo event depending on the survival probability associated with that particular albedo event. As soon as the pseudo-particle is created, it is stored in the neutron bank with the same direction, position and energy as the incident albedo-transported particle and the tracking of the albedo reflected particle is resumed. The tracking of the pseudo-particles are performed at the end of each batch, after the source particles are completely processed. The Monte Carlo transport of a pseudo-particle is performed as it would be in a full transport calculation, allowing the use of any of the many available variance reduction biasing techniques.

The new procedure was implemented by specifying a second set of medium numbers for the albedo media. This second set of medium numbers are used when following the pseudo-particles within the albedo medium. The user may control the generation of pseudo-particles through several

mechanisms such as the accumulated average weight of the albedo-transported particles, the importance by direction, or the specification of region- and energy-dependent probabilities for the generation of pseudo-particles.

The comparison of the weight of the albedo-transported particle with the accumulated average weight of the albedo-transported particles by region and energy is the default procedure in the modified version of the MORSE code. The weight of an albedo-transported particle which suffers an albedo event is compared with the average weight of all particles which have previously experienced albedo events within that portion of phase space. The purpose of this procedure is to control the number of particles generated by preventing large fluctuations in the weights of the generated pseudo-particles. To provide an initial guess of the average weight at the start of a run, a dummy batch of albedo-transported particles is processed for a user specified number of particles. During the albedo-modified Monte Carlo analysis, the average weight of the albedo-transported particles is updated after each batch. The average weight by region and by energy is used in a fair game procedure to determine if an albedo-collision will produce a pseudo-particle or not. At each albedo event, the current weight of the albedo-transported particle is compared with the "average weight" associated with the region and energy group of the incident particle. If the weight of the albedo-trans-

ported particle is less than the average weight, the Russian-roullete game is played. The probability for survival is

$$P_s = \frac{\text{WGTINC}}{\text{WGTAVG}(E_{in})} \quad , \qquad \text{(IV-1)}$$

where WGTINC is the weight of the incident albedo-transported particle and WGTAVG($E_{in}$) is the batch estimate of the appropriate "average weight". If a random number is less than or equal to $P_s$, a pseudo-particle is banked with a weight equal to WGTAVG($E_{in}$). Also, if the weight of the pseudo-particle is larger than the average weight a pseudo-particle is banked with its current weight.

The importance-by-direction procedure for generating pseudo-particles is based on the angle $\theta_D$ between the direction of the incoming albedo-transported particle and the vector connecting the albedo-event point to a "point-of-interest". The biasing is accomplished in terms of $[\cos(\theta_D/2)]^n$ where "n" is an integer provided by the user - "n" can be set equal to zero to turn off this biasing procedure. The code determines $\theta_D$ by the following formula:

$$\bar{n}\cdot\bar{a} = || \bar{n} || \, . \, || \bar{a} || \cos( \theta_D ), \qquad \text{(IV-2)}$$

where $\bar{n}$ is the inward normal to the plane where the albedo event occurs and $\bar{a}$ is the vector connecting the point of

the albedo event to the point-of-interest. Equation (IV-2) can also be rewritten as,

$$\cos \theta_D = \frac{u.(x_{imp}-x) + v.(y_{imp}-y) + w.(z_{imp}-z)}{dist}, \qquad \text{(IV-3)}$$

where

$$dist = \sqrt{(x_{imp}-x)^2 + (y_{imp}-y)^2 + (z_{imp}-z)^2}\ , \qquad \text{(IV-4)}$$

u,v,w are the direction cosines of the albedo particle, x,y,z are the coordinates of the albedo event, and $x_{imp}, y_{imp}, z_{imp}$ are the coordinates of the point of interest.

The survival probability is based on the half angle "$\theta_D/2$". This assures that the cosine of $\theta_D/2$ will assume a value between zero and one. A Russian roulette game is played by comparing a random number with $[\cos(\theta_D/2)]^n$. If the random number is greater than $[\cos(\theta_D/2)]^n$, the pseudo-particle is not generated. A surviving pseudo-particle is stored in the bank with its weight corrected by the factor $[1/\cos(\theta/2)]^n$. To prevent the occurrence of particles with very large or even infinite weights, a limit is established for the weight correction. If $[\cos(\theta/2)]^n$ is less than 0.2, a weight correction factor equal to 10 is applied to surviving pseudo-particles.

The modified input file also provides an option for

assigning probabilities for generating pseudo-particles by region and by energy group. The generation probability associated with the incident albedo-transported particle is compared to a random number. If the random number is smaller than the generation probability, a pseudo-particle is generated with the weight of the albedo-transported particle multiplied by the inverse of the probability.

Any of the above described biasing procedures can be turned off by the user. It is important to point out that these procedures may generate pseudo-particles with very large weights - the user must therefore exercise some caution in the choice of biasing parameters.

## FAIR GAME TRANSPORT OF THE PSEUDO-PARTICLES

An albedo-transported particle is reflected with probability one when it encounters an albedo surface. Its statistical weight is modified by the total albedo in order to correctly represent the statistical nature of the reflection process. The pseudo-particle is created with the same phase space coordinates as the incident particle. To preserve the fair game, a pseudo-particle is not permitted to cross back through the surface of the albedo medium. This is due to the fact that an albedo reflected particle is only one of the possible outcomes from the full transport of a

particle within the albedo medium. If for any reason the pseudo-particle attempts to cross back through the albedo surface, it must be killed because this event is deterministically represented by the reflection of the albedo particle.

To implement the process of eliminating pseudo-particles which are trying to cross back through the albedo surface, a check is made at each collision site to determine if the ray connecting two successive collision sites intersects the albedo surface. In the event that an intersection of the albedo surface is detected, the pseudo-particle is killed. When using several albedo surfaces the check is performed only with respect to the albedo surface at which the pseudo-particles was created. To accomplish this, two different hemispheres are associated with each albedo surface and within each hemisphere only one type of particle is allowed to exist. Also when using next-event estimation, pseudo-particles are not allowed to contribute to detectors for which the straight line connecting the collision point to the detector intersects the albedo surface.

The albedo data can be generated for planar, cylindrical and spherical surfaces. For the planar albedo surface, which is the only case studied here, the equation which defines the relative position of a point to a surface is given by

$$SIGN = \vec{n} \cdot \vec{c}, \qquad \text{(IV-5)}$$

where $\bar{n}$ is the normal to the albedo plane and $\bar{c}$ is the vector connecting the collision point to the point where the pseudo-particle was created. Equation (IV-5) can be rewritten as

$$SIGN = u_n (x_0-x) + v_n (y_0-y) + w_n (z_0-z) , \qquad (IV-6)$$

where $u_n, v_n, w_n$ are the direction cosines of the normal to the plane, $x_0, y_0, z_0$ are the coordinates of a point on the plane, and x,y,z are the coordinates of the point for which the relative position to the plane is unknown. A positive value for SIGN indicates that the point (x,y,z) is within the hemisphere as the positive normal of the surface. In applying this equation to the Monte Carlo transport of pseudo-particles, the direction cosines ($u_n, v_n, w_n$) represents the normal to the albedo surface at the hemisphere where albedo-particles are tracked, ($x_0, y_0$, and $z_0$) represent the coordinates of the albedo event which generated the pseudo-particle, and (x,y,z) are the coordinates of the pseudo-particle's next collision site. As long as the value of SIGN is negative the tracking of the pseudo-particle continues. If a positive value of SIGN is calculated the pseudo-particle is killed.

For next event estimation, essentially the same procedure is used. The coordinates of the detector are used in place of the coordinates of the collision point ( x,y,z) in the equation for SIGN. The pseudo-particle will not make

a contribution, if SIGN is positive.

In the case of cylindrical and spherical geometries the only check performed is to define if the distance from the collision point to the axis of the cylinder on the radial plane or the center of the sphere, respectively, is larger than the defining radius of the body under consideration. Since albedo media are usually placed at the external region of the cylinder or sphere, the pseudo-particle will be killed if it tries to enter the body.

When using next-event estimation, the pseudo-particle is killed if it is unable to make contributions to any of the detectors specified in the problem. This follows since the sign of SIGN can only changed if the pseudo-particle crosses back through the albedo surface - this however results in a pseudo-particle kill. This procedure prevents the waste of computing time following pseudo-particles which cannot contribute.

## EMERGENT POINT FROM AN ALBEDO EVENT

The traditional point albedos are generated considering the point of emergence from an albedo type collision to be coincident with the point of incidence. A "rule" for the use of point albedos being considered acceptable/reasonably-accurate is that the distance traveled by the particles

between two successive albedo events must be large as compared with the radiation's mean-free-path in the albedo medium. In other words, as stated in Chapter II, the dimensions of the reflecting region must be small compared with the distance between the albedo event and the detector.

The use of point albedos in problems in which the radiation field has streaming components has the effect of retarding the transport of the albedo-transported particles. As a consequence, underestimation of the detector response downstream is likely to occur. This effect can be expected on phenomenological grounds - because of preferential scattering in the forward direction, the reflected particle will have a higher probability of emerging at positions downstream from the point of incidence.

The most common applications of the albedo concept are problems in which particles stream inside ducts and cavities. The aforementioned "rule" restricts the number of such problems which can be solved using point albedo data. Only configurations in which the internal diameter of the duct (or cavity) is large compared with the radiation particle's mean free path in the wall material can be solved using the point albedo data.

Two different approaches are used in this research to mitigate the limitations imposed by the "rule". The first is the generation of a new set of albedo data with the advantage of correcting the distortions which exist in the avail-

able sets of albedo data. The second is to infer from the true behavior within the reflecting region (as described in the Chapter II) appropriate corrections to the available point albedo data. The formulation of an empirical relation to describe the displacement of the emergent particle relative to the point of incidence is discussed in this subsection.

The major problem in incorporating this modification into the albedo modified random-walk is that the necessary information which would permit the calculation of the "average displacement" is generally unavailable. To overcome this limitation of the previously calculated albedo data, new calculations are required in order to provide the necessary information. In this subsection calculations performed with the MCNP code for this purpose are discussed. A set of albedo-type calculations was performed for various combinations of the angle and energy of the incident source particles. The weight of the reflected particles were counted at the albedo surface within surface areas. Figure IV-2 shows a schematic representation of the surface bins used in the calculations. It is important to point out that only the total albedo at each surface area bin was estimated, thus the angle and energy were not binned in the detectors. The results for each surface bin were projected onto the direction defined by the intersection of the albedo plane with the plane defined by the normal to the albedo plane and the

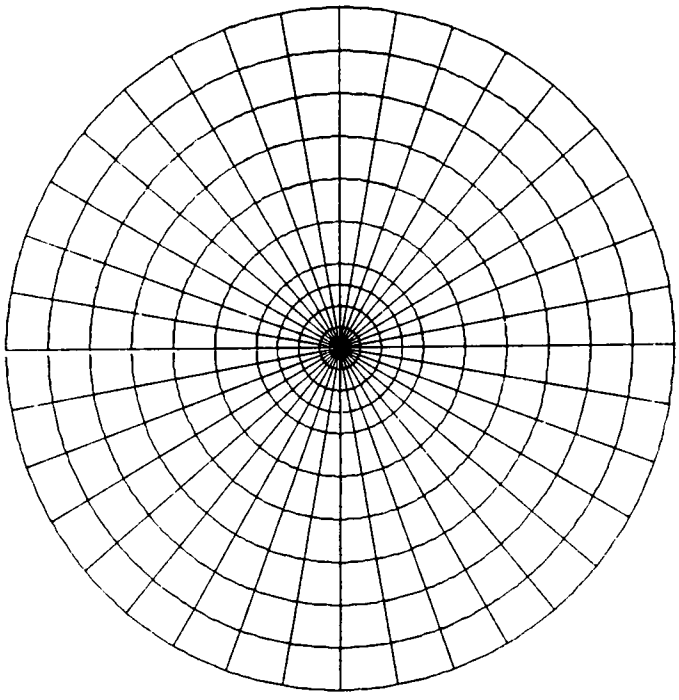


Figure IV-2. Schematic Representation of the Surface Bins Used in the MCNP Calculations Performed to Obtain Spatial Description of the Albedo Reflection Process for Concrete.

direction of the incident source particles (the projection was performed as if each bin was represented by a vector with its origin at the point of incidence of the source particles, with direction defined by the relative position of the bin to the line of projection and a magnitude equal to the distance from the point of incidence multiplied by the total albedo computed at the bin). Performing the summation over all projected "vectors", a single distance in mean-free-paths between the point of incidence and the "average point of emergence" was obtained for each incident angle and energy combination. Figure IV-3 displays the plot

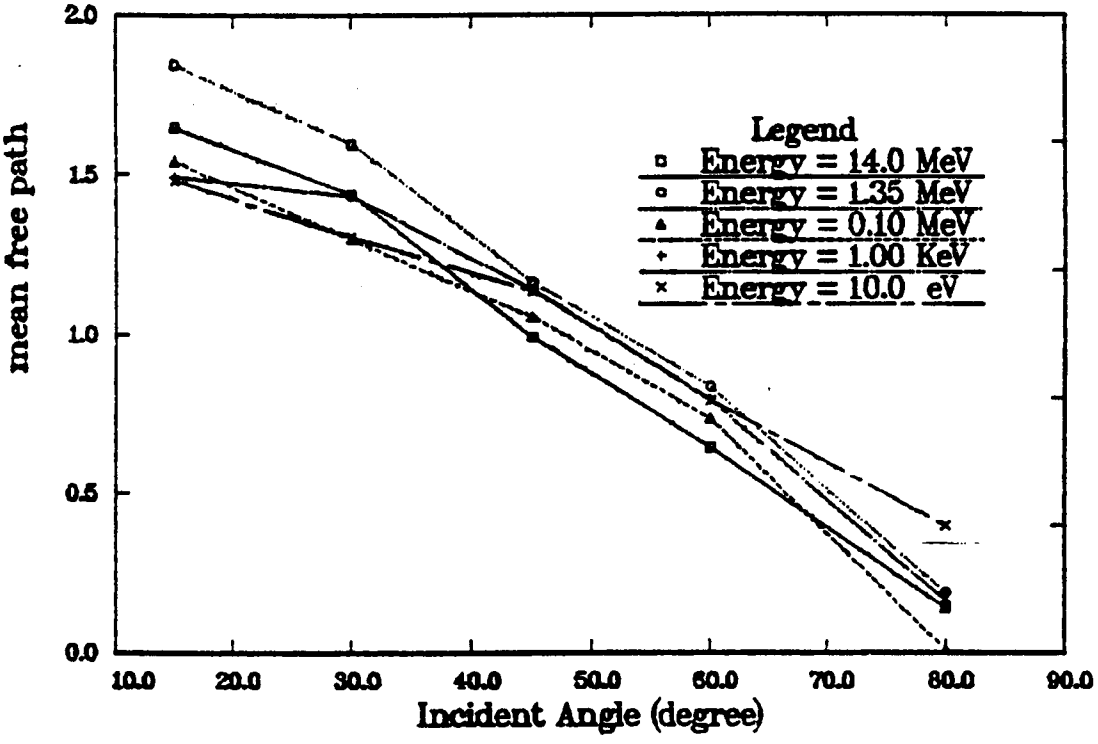



Figure IV-3. Displacement in Number of Mean-Free-Paths as a Function of the Incident Angle for Different Incident Energies for Ordinary Concrete.

of the average distance (measured in mean-free-paths) as a function of the angle of incidence for each of the energies considered. It can be noticed that a new average distance, which only depend on the angle of incidence, can be determined. Figure IV-4 displays the plot of the incident energy of the particle as a function of the average number of mean-free-paths from the incident point for each of the incident angles. It can be seen in this plot that the average number of mean-free-paths is not independent of the incident energy of the particle - but the dependence is clearly weak when compared with the dependence on the angle of incidence. Therefore, the dependence on incident energy was neglected thus keeping the empirical results and required Monte Carlo procedures as simple as possible.

Using the results presented in Figure IV-3, a fitting function was derived. The parameters for the function were determined by minimizing the deviations in relation to the calculated points. The derived formula for the displacement of the point of emergence for ordinary concrete is the following:

$$\rho = 0.025 \times ( 90 - \theta_o ) / \Sigma_T \quad , \qquad \text{(IV-7)}$$

where:

$\rho$ is the displacement in centimeters,

$\theta_o$ is the incident angle of the particle measured from

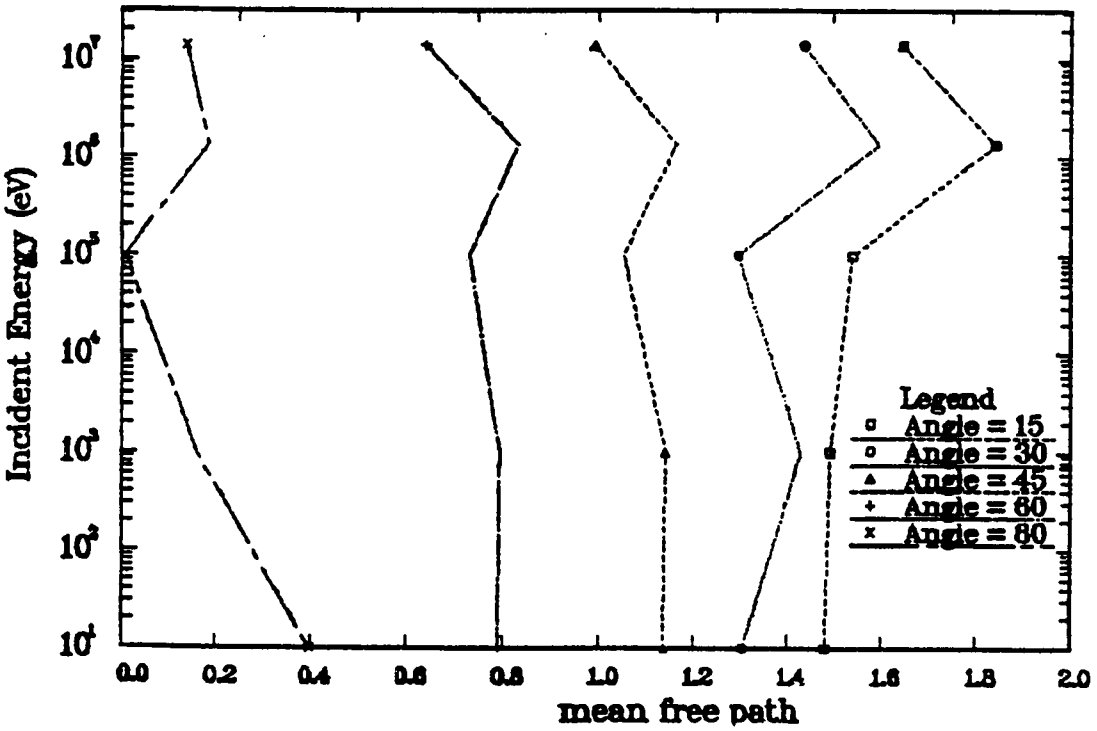



Figure IV-4. Displacement in Number of Mean-Free-Paths as a Function of the Incident Energy for Different Incident Angles for Ordinary Concrete.

the albedo plane, and

$\Sigma_T$ is the total cross section corresponding to the energy group of the incident particle ( reciprocal of the mean free path ).

The displacement of the emergent point in relation to the point-of-incidence was performed using the fitting function given by Equation IV-7. This simplified approach was incorporated into the albedo procedure to demonstrate the influence of the point-of-emergence on the final results of a calculation. This emergent point represents the average displacement from the point-of-incidence, over all possible outcomes. Symmetry of the diffusion process inside a medium leads to the observation that the displacement vector which defines the so called "average point-of-emergence" must lie in the plane defined by the projected incident particle's direction onto the albedo surface and by the normal to the albedo plane.

## IMPLEMENTATION OF THE POINT-OF-EMERGENCE IN THE RANDOM WALK

The MORSE code was modified to make available the generated point-of-emergence information in the albedo-modified random walk. Using the fitting function and the particle's incident angle to the albedo surface plane, it is

possible to calculate the average displacement for each albedo collision. The subroutine ALBDO of the MORSE code was modified to accommodate the displacement during the random-walk. A new subroutine was written in order to transfer the particle from the point-of-incidence to the point-of-emergence, including the determination of the new region and medium number at the point-of-emergence. This new subroutine called GOMES is presented in the appendix A. Adding a displacement to the emergent point may result in the creation of an albedo-displaced pseudo-particle inside an albedo region. If this occurs, the emergent particle is banked as if it was a pseudo-particle and its tracking would be resumed together with the other pseudo-particles. An exception is made in that the albedo-displaced pseudo-particle is not killed by any albedo plane and its tracking is performed in a full transport mode - the albedo-particle which was being followed would have its tracking terminated as if it was absorbed by the albedo medium and the next source particle in the bank would be processed. As a consequence up to two pseudo-particles may be created at the same albedo event, one at the point of incidence heading in the same direction as the incident particle and a second at the point-of-emergence heading in the direction dictated by the albedo reflection process. If the pseudo-particle option is turned off, the emergent particle which is displaced to points inside of an albedo region is killed.

## GENERATION OF A NEW SET OF ALBEDO DATA WITH THE TORT CODE

The discrete ordinates three-dimensional TORT code has the advantage over Monte Carlo codes by providing a detailed description of the radiation field in terms of the easily accessible discrete ordinates fluxes. To obtain the same detail in a Monte Carlo calculation, much more computer time would be required and the calculation would be prohibitively expensive. To calculate albedo data, the boundary angular fluxes are required. The version of the TORT code distributed by Radiation Shielding Information Center does not have this feature and additional work was necessary to modify the code to make this information available.

The TORT code has the capability of several modes for the flux sweep, it has the linear or Diamond difference model, the step model, and the semi-analytical methods (nodal and characteristic). The calculation of albedo data is a very difficult problem for discrete ordinate codes due to the punctual, monoenergetic, mono-directional characteristic of the source term. To avoid the propagation or even the occurrence of negative angular fluxes in the results, a study was carried out to identify which mode would be the best for generating albedo data. It was clear that the method which leads to overall positivity of the angular fluxes was the characteristic method using the highest order of flux expansion available in the code. The diamond differ-

ence and the step models, even when using negative flux fix-up, resulted in the calculation of negative scalar fluxes. The nodal method did not present any negative scalar fluxes for high- order expansions of the flux, but occasional negative angular fluxes were observed. The characteristic method was the only procedure to exhibit an overall positivity in the scalar and angular fluxes and with a flux shape distribution which was acceptable. The option of using the characteristic method is not without some penalty - it is a very expensive method (it requires about three times more computer time than the diamond difference option for this problem). Considering the number of runs that are required to calculate a set of albedo data, this additional computational expense makes a dramatic difference in the overall cost.

It should be noticed that this is a preliminary study in generating albedo data using the TORT code, and that improvements can be made to specialize the code for efficient albedo generation (some of these improvements will reduce the required computer time by a factor of two or more). Nevertheless a set of albedo data was generated for concrete using $S_{10}$ quadrature (145 directions) and a $P_3$ set of cross sections with 25 neutron groups and 12 photons groups. A concrete block, 2 meters by 2 meters by 1.5 meters deep, was modeled with a 13x13 x-y spatial mesh and 9 mesh spaces in the direction of the inward normal of the albedo

plane.

The directional flux, N, was written as it was calculated for each direction, energy, and spatial position on the albedo plane. To transform the directional flux into corresponding albedo data, the following relationship is used;

$$\alpha_{m_0,g_0,m,i,j,g} = \frac{N_{m,i,j,g}\, W_m\, \mu_m}{S_{m_0,i_0,j_0,g_0}\, W_{m_0}\, A_{i_0,j_0}\, \mu_{m_0}}, \quad \text{(IV-8)}$$

where $\alpha_{m_0,g_0,m,i,j,g}$ is the triply-differential albedo for particles incident at the quadrature defined direction "$m_0$", within the energy group "$g_0$", which emerge at the quadrature defined direction "m", at the spatial mesh (i,j), and within the energy group g; $N_{m,i,j,g}$ is the directional flux; $W_m$ is the quadrature weight for direction "m"; $\mu_m$ is the cosine of "$\theta$"; $S_{m_0,i_0,j_0,g_0}$ is the source strength; and $A_{i_0,j_0}$ is the surface area of the spatial mesh into which the source is input.

The value of the albedo for each combination of the incident energy and direction is computed and added to create a joint cumulative probability distribution. The next subsection explains the steps which are necessary to manipulate and format the data into forms suitable for use by the modified version of the MORSE code.

## PROCESSING OF THE SPATIALLY-DEPENDENT ALBEDO DATA

The mechanics of processing and using a set of spatially-dependent albedo data differs considerably from that of its two-dimensional counterpart. The spatial distribution of the points of emergence is added to the albedo data as well as the symmetry of the angular flux to the plane defined by the incident direction and the normal to the albedo plane is no longer true for the whole domain of the spatial variable.

A interface code called ALBFAC3D was developed to read the TORT files which contain the angular distribution of the emergent flux at the albedo plane and also generate the probability distribution tables. ALBFAC3D writes a file with the albedo information in a format suitable for input into a modified version of the subroutine ALBIN of the MORSE/BREESE package. The marginal-to-conditional probability distribution sampling scheme was changed to accommodate the adjoint mode - procedurally not possible with the conventional scheme used in the BREEZE code. The marginal probability distribution for the reflected albedo particles in the ALBFAC3D code is the energy distribution, the singly-conditional distribution is for the spatial variable, the doubly-conditional is for the polar angle, and the triply-conditional probability distribution is for the azimuthal probability distribution. This modification in the order of

sampling the variables results in a modified normalization procedure through the use of a different scheme of summation. The following set of equations represent the implemented procedure in the ALBFAC3D code.

The total albedo for an incident energy ($E_0^{i_0}$) and $\eta$-level ($\theta_0^{m_0}$) is calculated by

$$\alpha_T(E_0^{i_0},\theta_0^{m_0}) = \sum_{i=i_0}^{I} \sum_{1}^{J} \sum_{1}^{K} \sum_{1}^{M} \sum_{1}^{N} \alpha(E_0^{i_0},\theta_0^{m_0};E_1^i,X_j,Y_k,\theta_1^m,\phi_n), \quad \text{(IV-9)}$$

where $\alpha_T(E_0^{i_0},\theta_0^{m_0})$ is the total albedo; $E_0^{i_0}$ and $\theta_0^{m_0}$ are the incident energy group and incident $\eta$-level in the quadrature set, respectively. $E_1^i$ represents i-th outgoing energy group for which "I" represents the total number of outgoing energy groups, $X_j$ and $Y_k$ are displacements components parallel and perpendicular respectively to the projection of the incident direction of the particle on the albedo plane, for which "J" and "K" are the total number of emergent points considered for the parallel and perpendicular components respectively. The angle $\theta_1^m$ represents the m-th outgoing $\eta$-level for which "M" is the total number of outgoing $\eta$-levels. The angle $\phi_n$ is the n-th outgoing azimuthal angle for which "N" is the maximum number of azimuthal angles in the quadrature set.

The selection of the particle's outgoing energy group from an albedo event is performed based on the marginal probability distribution table

$$\mathrm{cdf}_E(E_0^{i_0},\theta_0^{m_0};E_1^{i_1}) = \frac{\sum_{i=i0}^{i1}\sum_{1}^{J}\sum_{1}^{K}\sum_{1}^{M}\sum_{1}^{N}\alpha(E_0^{i_0},\theta_0^{m_0};E_1^{i},X_j,Y_k,\theta_1^{m},\phi_n)}{\alpha_T(E_0^{i_0},\theta_0^{m_0})}, \tag{IV-10}$$

where $\mathrm{cdf}_E$ represents the cumulative marginal probability of a particle being scattered from energy group $i_0$ into one of the energy groups $i_0$ to $i_1$.

The cumulative conditional distribution function for the spatial displacement in the direction parallel to the projection of the particle's incident direction on the albedo plane is given by

$$\mathrm{cdf}_X(E_0^{i_0},\theta_0^{j_0},E_1^{i_1};X_{j_1}) = \frac{\sum_{1}^{j_1}\sum_{1}^{K}\sum_{1}^{M}\sum_{1}^{N}\alpha(E_0^{i_0},\theta_0^{m_0},E_1^{i};X_j,Y_k,\theta_1^{m},\phi_n)}{\sum_{1}^{J}\sum_{1}^{K}\sum_{1}^{M}\sum_{1}^{N}\alpha(E_0^{i_0},\theta_0^{m_0},E_1^{i};X_j,Y_k,\theta_1^{m},\phi_n)}, \tag{IV-11}$$

and the cdf for the displacement in the direction perpendicular to the projection of the incident direction on the albedo plane can be expressed as

$$\mathrm{cdf}_Y(E_0^{i_0},\theta_0^{j_0},E_1^{i_1};Y_{k_1}) = \frac{\sum_{1}^{k_1}\sum_{1}^{J}\sum_{1}^{M}\sum_{1}^{N}\alpha(E_0^{i_0},\theta_0^{m_0},E_1^{i};X_j,Y_k,\theta_1^{m},\phi_n)}{\sum_{1}^{J}\sum_{1}^{K}\sum_{1}^{M}\sum_{1}^{N}\alpha(E_0^{i_0},\theta_0^{m_0},E_1^{i};X_j,Y_k,\theta_1^{m},\phi_n)}. \tag{IV-12}$$

It can be noted that the cdf's $cdf_X$ and $cdf_Y$ are mutually independent.

The cumulative conditional distribution functions for the emergent $\eta$-level and emergent azimuthal angle are conditional with respect to 3 other variables. The number of discrete values of a variable which have to be stored is equal the number of discrete values of the considered variable multiplied by the matrix of discrete values of all the variable upon which the considered variable depends. To minimize the storage requirement for the discrete values of the emergent $\eta$-levels and emergent azimuthal angles these two variables may have their distributions collapsed into super-regions of the spatial variables. This can be accomplished by indicating through an input parameter to the ALBFAC3D code, the number of spatial mesh which comprise a super-region. In the following formulation for the $\eta$-level and azimuthal cdf's the notation "JK" is used to denote the total number of super-regions, "XY" to denote the super-region, and jk the index of the super-region. Then the cumulative distribution function for the outgoing $\eta$-level can be expressed as

$$cdf_{\theta}(E_0^{i_0},\theta_0^{m_0},E_1^{i_1},XY_{jk_1};\theta_1^{m_1}) = \frac{\sum_{1}^{m_1}\sum_{1}^{N}\alpha(E_0^{i_0},\theta_0^{m_0},E_1^{i},XY_{jk_1},\theta_1^{m},\phi_n)}{\sum_{1}^{M}\sum_{1}^{N}\alpha(E_0^{i_0},\theta_0^{m_0},E_1^{i},XY_{jk_1},\theta_1^{m},\phi_n)}, \quad \text{(IV-13)}$$

and the cdf for the azimuthal angle

$$cdf_{\phi}(E_0^{i_0},\theta_0^{j_0},E_1^{i_1},XY_{jk_1},\theta_1^{m_1};\phi_{n_1}) = \frac{\sum_{1}^{n_1}\alpha(E_0^{i_0},\theta_0^{m_0},E_1^{i},XY_{jk_1},\theta_1^{m_1},\phi_n)}{\sum_{1}^{N}\alpha(E_0^{i_0},\theta_0^{m_0},E_1^{i},XY_{jk_1},\theta_1^{m_1};\phi_n)}. \quad \text{(IV-14)}$$

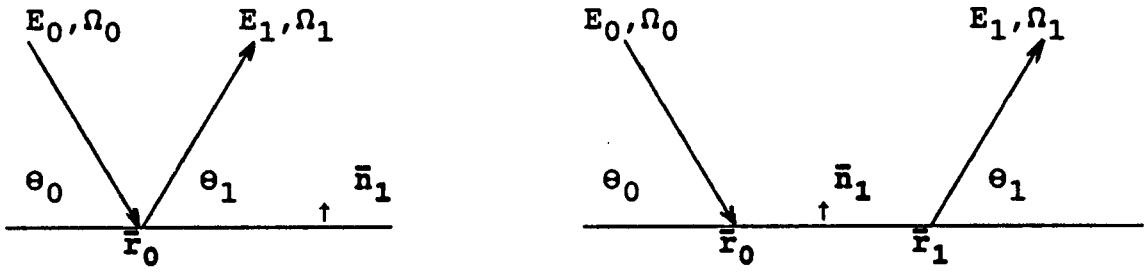
The ALBFAC3D code can collapse the number of spatial positions for which the emergent angular distribution is calculated to any number specified by the user. To allow the code to correctly collapse the data, the user must provide an input matrix which relates the original mesh space distribution to the super-regions. During the sampling process implemented in the modified version of the subroutine ALBDO, the emergent position (j,k indexes of the emergent mesh) is sampled before selection of the emergent direction, then ALBDO at the time of sampling the emergent direction has all the necessary information, which are the spatial indices j and k of the emergent mesh and input matrix, to select the emergent direction from the conditional probability distribution for the super-regions. This collapsing capability of the ALBFAC3D code is very useful in reducing the memory space requirement for the MORSE run, without losing any detail of the description of the emergent point. It is important to point out that the direction distribution is less sensitive to collapsing than the spatial distribution.

To make the MORSE code compatible with the spatially-dependent albedo data generated by the ALBFAC3D code, a new version of the subroutine ALBIN was written. The name of the new subroutine is ALBIN3D thereby allowing the user to choose either point or spatially-dependent albedos through a new input parameter. The subroutine ALBDO which performs the albdo reflection process was also accordingly modified and its name was changed to ALBDO3D. The subroutine ALBDO3D allows both forward and adjoint modes of analysis and is compatible with the albedo data set generated by the modified TORT code and processed by the ALBFAC3D code.

## RECIPROCITY RELATIONSHIP FOR THE FORWARD AND ADJUNCTON ALBEDOS

A reciprocity relationship for the forward and adjuncton albedos will be developed to demonstrate that with proper interpretation and processing, the forward albedo data have a simple relationship with their adjoint counterparts thereby eliminating the need for separate albedo calculations of the adjuncton albedos. The derivation to follow considers a semi-infinite medium with a radiation particle incident on the albedo surface at the spatial position $\bar{r}_0$. This albedo model assumes that the particle emerges from albedo surface at the spatial position $\bar{r}_1$. In

general, discrete ordinates calculated albedos are "point" albedos rather than the phenomenologically correct "spatially-dependent" albedos. The two albedo models are schematically illustrated in the figure IV-5.




a. Point Albedo Model

b. Spatially-Dependent Albedo

Figure IV-5. Schematic Representation of the Reflection Process Using Point and Spatially-Dependent Albedo Data.

In order to establish a general reciprocity relationship between the forward and adjuncton albedos, the forward transport equation is written for the angular flux with phase space coordinates $(\bar{r},E,\bar{\Omega})$ due to a unit surface source at $(\bar{r}_0,E_0,\bar{\Omega}_0)$,

$$\bar{\Omega}\cdot\nabla\,\Phi(\,\bar{r},E,\bar{\Omega};\,\bar{r}_0,E_0,\bar{\Omega}_0)\;+\;\Sigma_T(\bar{r},E)\Phi(\bar{r},E,\bar{\Omega};\bar{r}_0,E_0,\bar{\Omega}_0)\;=$$

$$=\int_0^{\infty}\int_0^{4\pi}dE'd\bar{\Omega}'\Phi(\bar{r},E',\bar{\Omega}';\bar{r}_0,E_0,\bar{\Omega}_0)\Sigma_s(E'\Rightarrow E,\bar{\Omega}'\Rightarrow\bar{\Omega}),\quad\text{(IV-15)}$$

In a similar fashion, the transport equation for the adjoint flux with phase space coordinates $(\bar{r},E,\bar{\Omega})$ due to a unit adjoint surface source at $(\bar{r}_1,E_1,\bar{\Omega}_1)$ may be written as

$$-\bar{\Omega}\cdot\nabla\,\Phi^+(\bar{r},E,\bar{\Omega};\bar{r}_1,E_1,\bar{\Omega}_1) + \Sigma_T(\bar{r},E)\Phi^+(\bar{r},E,\bar{\Omega};\bar{r}_1,E_1,\bar{\Omega}_1) =$$

$$= \int_0^\infty \int_0^{4\pi} dE'd\bar{\Omega}'\Phi^+(\bar{r},E',\bar{\Omega}';\bar{r}_1,E_1,\bar{\Omega}_1)\Sigma_s(E\Rightarrow E',\bar{\Omega}\Rightarrow\bar{\Omega}'). \quad \text{(IV-16)}$$

It is important to notice that the above formulations consider the source terms as being external to the system volume and therefore represented as surface sources. Equation (IV-15) is multiplied by $\Phi^+$ and Equation (IV-16) by $\Phi$ - the resulting equations are subtracted and integrated over the system volume:

$$\int_V (\Phi^+\bar{\Omega}\cdot\nabla\Phi + \Phi\bar{\Omega}\cdot\nabla\Phi^+)dv =$$

$$= \int_V \int_0^\infty \int_0^{4\pi} d\bar{\Omega}'dE'dv\ \Phi^+(\bar{r},E,\bar{\Omega};\bar{r}_1,E_1,\bar{\Omega}_1)\Sigma_s(E'\Rightarrow E,\bar{\Omega}'\Rightarrow\bar{\Omega})$$

$$x\ \Phi(\bar{r},E',\bar{\Omega}';\bar{r}_0,E_0,\bar{\Omega}_0) \quad - \quad \int_V \int_0^\infty \int_0^{4\pi} d\bar{\Omega}'dE'dv\ \Phi(\bar{r},E,\bar{\Omega};\bar{r}_0,E_0,\bar{\Omega}_0)$$

$$x\ \Sigma_s(E\Rightarrow E',\bar{\Omega}\Rightarrow\bar{\Omega}')\Phi^+(\bar{r},E',\bar{\Omega}';\bar{r}_1,E_1,\bar{\Omega}_1). \quad \text{(IV-17)}$$

According to vector calculus, it is noted that

$$\int_V (\Phi^+ \bar{\Omega}\cdot\nabla\Phi + \Phi\bar{\Omega}\cdot\nabla\Phi^+)dv = \int_V \bar{\Omega}\cdot\nabla\,\Phi\Phi^+ dv = \int_S \bar{n}\cdot\bar{\Omega}\Phi\Phi^+ ds \qquad \text{(IV-18)}$$

and Equation (IV-17) can be rewritten as

$$\int_S \bar{n}\cdot\bar{\Omega}\Phi\Phi^+ ds =$$

$$= \int_V \int_0^\infty \int_0^{4\pi} d\bar{\Omega}' dE' dv \; [\Phi^+(\bar{r},E,\bar{\Omega};\bar{r}_1,E_1,\bar{\Omega}_1)\Sigma_s(E'=>E,\bar{\Omega}'=>\bar{\Omega}) \; x$$

$$x \; \Phi(\bar{r},E',\bar{\Omega}';\bar{r}_0,E_0,\bar{\Omega}_0)] - \int_V \int_0^\infty \int_0^{4\pi} d\bar{\Omega}' dE' dv \; [\Phi(\bar{r},E,\bar{\Omega};\bar{r}_0,E_0,\bar{\Omega}_0)$$

$$x \; \Sigma_s(E=>E',\bar{\Omega}=>\bar{\Omega}')\Phi^+(\bar{r},E',\bar{\Omega}';\bar{r}_1,E_1,\bar{\Omega}_1)]. \qquad \text{(IV-19)}$$

Since in the formulation of both Equations (IV-15) and (IV-16) the source terms were considered to be external to the system and represented as surface sources, the product $\Phi\Phi^+$ does not vanish on the surface. The value of $\Phi(\bar{r}_s)$ for $\bar{n}\cdot\bar{\Omega} < 0$ is,

$$\Phi(\bar{r}_s)_{\bar{n}\cdot\bar{\Omega}<0} = \frac{1}{\cos\theta_0} S(\bar{r}_0,E_0,\bar{\Omega}_0) = \frac{1}{|\bar{n}\cdot\bar{\Omega}_0|}\,\delta(\bar{r}_s-\bar{r}_0)\,\delta(\bar{\Omega}\cdot\bar{\Omega}_0)\,\delta(E-E_0) \qquad \text{(IV-20)}$$

Since $\Phi^+(\bar{r})$ is the value of a radiation particle at $\bar{r}$, the value of $\Phi^+(\bar{r})$ for $\bar{n}\cdot\bar{\Omega}>0$ is

$$\Phi^{+}_{\bar{n}\cdot\bar{\Omega}>0} = \frac{1}{\cos\theta_1} S^{+}(\bar{r}_1, E_1, \bar{\Omega}_1) = \frac{1}{|\bar{n}\cdot\bar{\Omega}_1|} \delta(\bar{r}_s - \bar{r}_1)\delta(\bar{\Omega}\cdot\bar{\Omega}_1)\delta(E-E_1) \tag{IV-21}$$

It follows that,

$$\int_S \bar{n}\cdot\bar{\Omega}\Phi\Phi^{+}ds = -\int_S \delta(\bar{r}_s-\bar{r}_0)\ \delta(\bar{\Omega}\cdot\bar{\Omega}_0)\ \delta(E-E_0)\ \Phi^{+}\ ds +$$

$$+ \int_S \delta(\bar{r}_s-\bar{r}_1)\ \delta(\bar{\Omega}\cdot\bar{\Omega}_1)\ \delta(E-E_1)\ \Phi\ ds =$$

$$= -\delta(\bar{\Omega}\cdot\bar{\Omega}_0)\delta(E-E_0)\Phi^{+}(\bar{r}_0, E, \bar{\Omega}; \bar{r}_1, E_1, \bar{\Omega}_1) +$$

$$+ \delta(\bar{\Omega}\cdot\bar{\Omega}_1)\delta(E-E_1)\Phi(\bar{r}_1, E, \bar{\Omega}; \bar{r}_0, E_0, \bar{\Omega}_0). \tag{IV-22}$$

Substituting Equation (IV-22) into Equation (IV-17) and integrating over energy and solid angle yields

$$-\int_0^{\infty}\int_0^{4\pi} d\bar{\Omega}dE\ \delta(\bar{\Omega}\cdot\bar{\Omega}_0)\delta(E-E_0)\Phi^{+}(\bar{r}_0, E, \bar{\Omega}; \bar{r}_1, E_1, \bar{\Omega}_1) +$$

$$+\int_0^{\infty}\int_0^{4\pi} d\bar{\Omega}dE\ \ \delta(\bar{\Omega}\cdot\bar{\Omega}_1)\delta(E-E_1)\Phi(\bar{r}, E, \bar{\Omega}; \bar{r}_0, E_0, \bar{\Omega}_0)$$

$$= \int_V\int_0^{\infty}\int_0^{4\pi}\int_0^{\infty}\int_0^{4\pi} d\bar{\Omega}'dE'd\bar{\Omega}dEdv[\{\Phi^{+}(\bar{r}, E, \bar{\Omega}; \bar{r}_1, E_1, \bar{\Omega}_1)\Sigma_s(E'\Rightarrow E, \bar{\Omega}'\Rightarrow\bar{\Omega})$$

$$\times\ \Phi(\bar{r}, E', \bar{\Omega}'; \bar{r}_0, E_0, \bar{\Omega}_0)\} - \{\Phi(\bar{r}, E', \bar{\Omega}'; \bar{r}_0, E_0, \bar{\Omega}_0)\Sigma_s(E'\Rightarrow E, \bar{\Omega}'\Rightarrow\bar{\Omega})$$

$$\times\ \Phi^{+}(\bar{r}, E, \bar{\Omega}; \bar{r}_1, E_1, \bar{\Omega}_1)\}]. \tag{IV-23}$$

The last integral in Equation (IV-23) is identically zero and evaluating the energy and solid angle integrals results in the relationship

$$\Phi(\bar{r}_1,E_1,\bar{\Omega}_1;\bar{r}_0,E_0,\bar{\Omega}_0) = \Phi^+(\bar{r}_0,E_0,\bar{\Omega}_0;\bar{r}_1,E_1,\bar{\Omega}_1). \quad \text{(IV-24)}$$

The $\Phi$ term in Equation (IV-24) is usually interpreted physically as an angular flux associated with the differential phase space $dP_1 = d\bar{r}_1 dE_1 d\bar{\Omega}_1$

$\Phi(\bar{r}_1,E_1,\bar{\Omega}_1;\bar{r}_0,E_0,\bar{\Omega}_0)\ dP_1 \equiv$ angular flux in $dP_1$ about $\bar{r}_1,E_1,\bar{\Omega}_1$ due to a unit source at $\bar{r}_0,E_0,\bar{\Omega}_0$ = probability that a source particle located on the albedo surface at $\bar{r}_0,E_0,\bar{\Omega}_0$ will contribute to the reflected flux at $\bar{r}_1,E_1,\bar{\Omega}_1$.

It follows that the current of reflected particles due to a unit source current is given by $J = \Phi\ |\cos\theta_1|\ |\cos\theta_0|$ and may be identified as the albedo $\alpha$ :

$\alpha(\bar{r}_1,E_1,\bar{\Omega}_1;\bar{r}_0,E_0,\bar{\Omega}_0)dP_1 = \Phi(\bar{r}_1,E_1,\bar{\Omega}_1;\bar{r}_0,E_0,\bar{\Omega}_0)|\cos\theta_0||\cos\theta_1| =$ angular albedo in $dP_1$ about $\bar{r}_1,E_1,\bar{\Omega}_1$ due to a unit incident current at $\bar{r}_0,E_0,\bar{\Omega}_0$.

The $\Phi^+$ term in equation (IV-24) may be interpreted as a value or importance function, that is

$\Phi^+(\bar{r}_0,E_0,\bar{\Omega}_0;\bar{r}_1,E_1,\bar{\Omega}_1)dP_1$ ≡ the value associated with an incident particle with phase space coordinates $(\bar{r}_0,E_0,\bar{\Omega}_0)$ that results in a particle of unit value that emerges in $dP_1$ about $(\bar{r}_1,E_1,\bar{\Omega}_1)$.

A more useful interpretation of $\Phi^+$ is in terms of adjunctons, the hypothetical particles transported in the $\hat{\Omega} \equiv -\bar{\Omega}$ directions during the generation of random walks in a adjoint Monte Carlo calculation. The adjuncton albedo $\alpha^+$ corresponding to $\Phi^+$ is given by

$\alpha^+(\bar{r}_0,E_0,\hat{\Omega}_0;\bar{r}_1,E_1,\hat{\Omega}_1) = \Phi^+(\bar{r}_0,E_0,\bar{\Omega}_0;\bar{r}_1,E_1,\bar{\Omega}_1)\ |\cos\theta_0||\cos\theta_1|$

= the adjuncton current emerging at $\bar{r}_0,E_0,\bar{\Omega}_0$ due to an incident unit adjuncton current at $\bar{r}_1,E_1,\bar{\Omega}_1$.

Multiplying both sides of Equation (IV-24) by $|\cos\theta_0||\cos\theta_1|$ and identifying the forward and adjuncton albedos yields

$$\alpha(\bar{r}_1,E_1,\bar{\Omega}_1;\bar{r}_0,E_0,\bar{\Omega}_0) = \alpha^+(\bar{r}_0,E_0,\bar{\Omega}_0;\bar{r}_1,E_1,\hat{\Omega}_1). \quad \text{(IV-23)}$$

According the Equation (IV-23), the forward albedo which describes the $(\bar{r}_0,E_0,\bar{\Omega}_0 \Rightarrow \bar{r}_1,E_1,\bar{\Omega}_1)$ reflection is identical to the adjuncton albedo which describes the $(\bar{r}_1,E_1,\bar{\Omega}_1 \Rightarrow$

$\bar{r}_0, E_0, \bar{\Omega}_0$) reflection. Therefore, a transformation of the forward albedo data to create the adjuncton albedos should be possible - the direct calculation of the adjuncton albedos should not be required. The forward albedo data are similar in format to that of the differential scattering cross section data and a similar transpose of the forward albedo data during its loading into subroutine ALBIN of the BREESE program will create the $\alpha^+$ data set for subsequent use during the reflection of adjunctons. ALBIN also calculates certain derived quantities such as the total albedos and the cumulative distribution functions for selection of reflected energy group and direction. These quantities will necessarily be different from their forward counterparts because the various summations required during normalization procedures are over the $(E_0, \bar{\Omega}_0)$ space rather than over the $(E_1, \bar{\Omega}_1)$ space - as would be expected.

## ADJOINT MODE WITH THE SPATIALLY-DEPENDENT ALBEDO DATA SET

The use of the reciprocity relationship of forward and adjuncton albedos provides the formal basis for using the same angular flux file for generating both forward and adjoint albedos. The interface code ALBFAC3D has a standard input option for generating adjoint albedo data. The procedure used and the manipulation of the data are explained in

this subsection.

The calculation of the total adjuncton albedo is performed in manner similar to its forward counterpart, but the indexes in the summation over i are modified to correctly describe the flow of an adjoint calculation giving as a result, an $\alpha_T^+$ that is different from $\alpha_T$. The procedure implemented in the ALBFAC3D code uses the following formulation to calculate the adjoint total albedo

$$\alpha_T^+(E_1^{i_1},\theta_1^{m_1}) = \sum_{i=1}^{i_1} \sum_{1}^{J} \sum_{1}^{K} \sum_{1}^{M} \sum_{1}^{N} \alpha(E_0^i,\theta_0^m;E_1^{i_1},X_j,Y_k,\theta_1^{m_1},\phi_n). \quad \text{(IV-24)}$$

It can be noticed that the summation is performed over the incident energy groups and polar angles. The index for the azimuthal angle varies as it does for the forward mode.

The mechanics of selecting the outgoing energy of an adjuncton experiencing an albedo event is very similar to its forward counterpart. The most important difference is that adjunctons scatter upwards in energy - that is from low energy groups to a higher energy group. The adjuncton albedo energy cumulative distribution table reflects this difference between the adjoint mode and the forward mode. The forward angular flux generated by a discrete ordinates code is used for generating the probability table but the indices for the summation are modified to reflect the backwards flow of the adjunctons. The formulation used in the

ALBFAC3D code for constructing the cumulative distribution function for adjuncton's outgoing energy is given by

$$cdf_E^+(E_1^{i_1},\theta_1^{m_1};E_0^{i_0}) = \frac{\sum_{i=i_0}^{i_1}\sum_{1}^{J}\sum_{1}^{K}\sum_{1}^{M}\sum_{1}^{N}\alpha(E_0^{i_0},\theta_1^{m_1};E_1^{i_1},X_j,Y_k,\theta_1^{m_1},\phi_n)}{\alpha_T^+(E_1^{i_1},\theta_1^{m_1})} \tag{IV-25}$$

It is important to notice that the energy index $i_1$ which is the upper limit for the summation performed over the energy groups is fixed and index $i_0$ which is the lower limit of the summation varies. The summation over the "$\eta$-level", represented by the variables $\theta_1^{m_1}$ and $\theta_0^m$, is performed in the reverse order, this means that the summation is over the incoming "$\eta$-level" and not over the outgoing "$\eta$-level".

The calculation of the adjoint displacement cumulative distribution function is performed using the following formulation

$$cdf_X^+(E_1^{i_1},\theta_1^{m_1},E_0^{i_0};X_{j_0}) = \frac{\sum_{1}^{j_1}\sum_{1}^{K}\sum_{1}^{M}\sum_{1}^{N}\alpha(E_0^{i_0},\theta_0^m,E_1^{i_1};X_j,Y_k,\theta_1^{m_1},\phi_n)}{\sum_{1}^{J}\sum_{1}^{K}\sum_{1}^{M}\sum_{1}^{N}\alpha(E_0^{i_0},\theta_0^m,E_1^{i_1};X_j,Y_k,\theta_1^{m_1},\phi_n)}, \tag{IV-26}$$

where the same modification of indices is performed, and the displacement selection procedure is performed for the ad-

juncton as it would have been to the forward particle. Selection from the $cdf_Y^+$ follows the same procedure and will not be presented here.

The cdf to select the outgoing $\eta$-level is derived based on the following formula

$$cdf_{\theta}^{+}(E_1^{i_1},\theta_1^{m_1},E_0^{i_0},XY_{jk_0};\theta_0^{m_0})=\frac{\sum_{1}^{m_0}\sum_{1}^{N}\alpha(E_0^{i_0},\theta_0^{m},E_1^{i_1},XY_{jk_0},\theta_1^{m_1},\phi_n)}{\sum_{1}^{M}\sum_{1}^{N}\alpha(E_0^{i_0},\theta_0^{m},E_1^{i_1},XY_{jk_0},\theta_1^{m_1},\phi_n)}, \tag{IV-27}$$

and the cdf for the azimuthal angle is

$$cdf_{\phi}^{+}(E_1^{i_1},\theta_1^{m_1},E_0^{i_0},XY_{jk_0},\theta_0^{m_0};\phi_{n_0})=\frac{\sum_{1}^{n_0}\alpha(E_0^{i_0},\theta_0^{m_0},E_1^{i},XY_{jk_1},\theta_1^{m_1},\phi_n)}{\sum_{1}^{N}\alpha(E_0^{i_0},\theta_0^{m_0},E_1^{i},XY_{jk_1},\theta_1^{m_1};\phi_n)}. \tag{IV-26}$$

It is easy to see that the cumulative probability distribution for the adjuncton is exactly the same as for its forward counterpart, except that the angles have opposite signs due to the fact that the adjuncton's direction is opposite to that of its forward counterpart. A schematic representation of the azimuthal reflection process is presented in Figure IV-6. It can be noticed that what used to be the incident direction for the forward mode is now, for the

adjoint mode, the emergent direction. The probability of reflection between the two directions remain the same (for both forward and adjoint modes) as stated earlier in this chapter, but the direction of measuring the angles is inverted. This later statement does not have any effect on the polar angle, which is measured from the normal to the plane, but it does have an effect on the sign of the azimuthal angle.

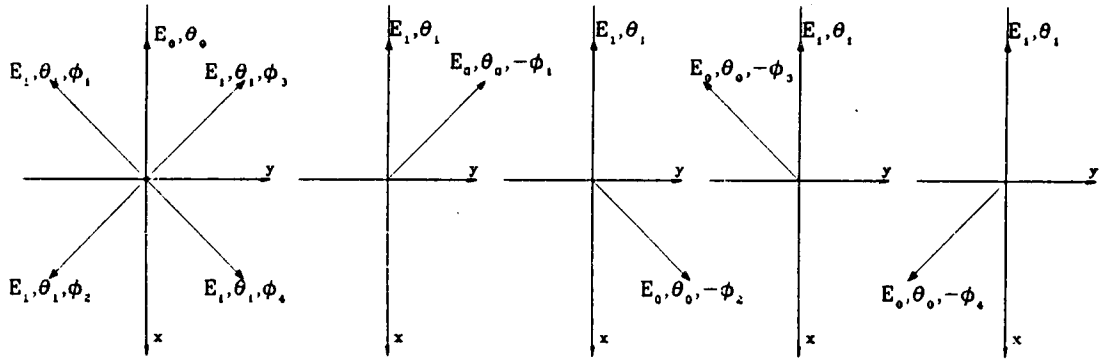



Figure IV-6. Forward and Adjoint Azimuthal Albedo Distribution, for Fixed Incident Energy and Angle, and Fixed Outgoing Energy and Polar Angle.

## NORMALIZATION FACTORS FOR ADJOINT CALCULATIONS

The MORSE code is designed to perform adjoint calculations in a manner very similar to the forward calculations. The input instructions and the cross-section sets are given in the forward structure and the code is able to make the necessary inversions. Adjoint source and estimation procedures that can be described by standard input parameters produce responses which are intrinsically made compatible by the code to their forward counterparts. If estimation or source routines are provided by the user, a careful analysis must be performed to ensure that the adjoint responses are correctly normalized.

The use of adjuncton-albedo data during the course of a Monte Carlo adjoint mode of analysis does not require any additional normalization factors and the same normalization procedures used for a normal adjoint calculation can be applied.

The Monte Carlo estimated effect-of-interest $\lambda$ for the forward mode is given by the functional

$$\lambda_{for} = \int_E \int_{\bar{r}} \int_{\bar{\Omega}} P^{\Phi}(\bar{r},E,\bar{\Omega})\ \Phi(\bar{r},E,\bar{\Omega})\ d\bar{r}\ dE\ d\bar{\Omega}, \qquad \text{(IV-27)}$$

where

$P^{\Phi}(\bar{r},E,\bar{\Omega})$ is the response function of the effect-of-interest due to a unit angular flux with direction $\bar{\Omega}$, energy E, and position $\bar{r}$, and

$\Phi(\bar{r},E,\bar{\Omega})$ is the time-independent angular flux.

The evaluation of this functional is automatically performed by the MORSE code and the result is normalized to provide an answer in terms of a unit source particle. The total effect-of-interest is then accomplished by multiplying the Monte Carlo estimated effect-of-interest by the strength of the source term.

In the adjoint mode, the effect of interest $\lambda$ is estimated in a fashion similar to that used in the forward case. The functional which describes the relationship between the adjuncton angular flux $\chi^*(\bar{r},E,\bar{\Omega})$ and the adjoint response function $S(\bar{r},E,\bar{\Omega})$ can expressed as

$$\lambda_{adj} = \int_E \int_{\bar{r}} \int_{\hat{\Omega}} S(\bar{r},E,\hat{\Omega})\ \chi^*(\bar{r},E,\hat{\Omega})\ d\bar{r}\ dE\ d\hat{\Omega}, \qquad \text{(IV-28)}$$

where

$S(\bar{r},E,\hat{\Omega})$ is the forward source term with direction $\hat{\Omega}$, energy E, and position $\bar{r}$, and

$\chi(\bar{r},E,\hat{\Omega})$ is the time-independent adjoint angular flux due to the adjoint source term $P^{\Phi}(\bar{r},E,\hat{\Omega})$.

Equation (IV-28) can be rewritten in terms of energy groups, by replacing the integral over the energy by the summation

over the energy groups. Then the equation (IV-28) becomes

$$\lambda_{adj} = \sum_{g=1}^{G} \int_{\bar{r}} \int_{\hat{\Omega}} S_g(\bar{r},\hat{\Omega})\ \chi_g^*(\bar{r},\hat{\Omega})\ d\bar{r}\ \ d\hat{\Omega}, \qquad \text{(IV-29)}$$

where

$S_g(\bar{r},\hat{\Omega})$ is the forward source term for energy group g, with direction $\hat{\Omega}$, and position $\bar{r}$,

$\chi_g^*(\bar{r},\hat{\Omega})$ is the time-independent adjuncton angular flux for the energy group g, with direction $\hat{\Omega}$, and position $\bar{r}$.

The Monte Carlo estimated effect-of-interest $\lambda$ should have exactly the same expected values for both the forward or adjoint modes of analysis - but due to the code normalization procedures, this may not be true for all cases.

To evaluate the normalization factors which the code includes in the final answer and additional factors which should be applied by the user, it is important to perform a step by step analysis of the normalization procedure. Accordingly, brief descriptions of the units and the nature of the physical variables used in evaluating the functionals are presented.

Analyzing equation (IV-27), it can be noticed that the response function $P^{\Phi}$ is a value function and as such carries the units of response per unit fluence. The units of

the effect-of-interest for the forward mode is the response associated with $P^{\Phi}$.

The source term of the forward Monte Carlo mode of analysis is a differential distribution and is given by

$$S_{for} = S(\bar{r},E,\bar{\Omega}). \qquad \text{(IV-30)}$$

When the functional of the effect-of-interest in the adjoint mode is evaluated, the same normalization procedures used in the forward mode are applied by the code (except for the energy, which will be discussed later). The evaluation of the functional for the effect-of-interest in the adjoint mode is performed (as given by Equation IV-29) using the adjuncton angular flux and the adjoint response function - which is the forward source term. It is well known that the adjoint flux is not a differential distribution and has the units of response per unit particle.

The MORSE Monte Carlo code when performing an adjoint mode calculation executes the same procedures as those for the forward mode. The normalization factor applied to the adjoint source is

$$N^{S}_{adj} = \int S^{*}(P)\ dP \qquad \text{(IV-31)}$$

This follows because the MORSE code is designed to normalize the source term to unity regardless of units. As an example, consider a point adjoint source (which correspond to a point

detector in the forward mode) for which the forward group response functions are set equal to unity (isotropic flux detector) for G energy groups. In this case the adjoint source normalization factor is

$$N^{s}_{adj} = \int P^{\phi}(E,\vec{\Omega})\ dE\ d\vec{\Omega} = \Sigma \int_{4\pi} d\vec{\Omega}$$

$$= 4\pi\ G. \qquad \text{(IV-32)}$$

As a consequence, the results of an adjoint calculation must be multiplied by the factor $4\pi G$. The user enters the source energy spectrum with the E1 card in the MORSE input file and the code automatically multiplies the Monte Carlo estimation by the summation of the values entered by the E1 card - in this example this summation equals G.

Therefore, the adjoint estimate can be made to yield essentially the same answer as the forward calculation (response per unit source particle) if the adjoint estimate is divided by the source normalization factor which is given by

$$N^{d}_{adj} = \int R_{adj}(P)\ dP. \qquad \text{(IV-33)}$$

The MORSE code in the adjoint mode sees the forward source S(P) as a response function $R_{adj}(P)$ and does not automatically normalize it. As an example, consider a forward isotropic point source defined over N energy groups which in the forward case a uniform probability of being sampled.

Then Equation (IV-33) can be evaluated as

$$N^{d}_{adj} = \int R_{adj}(P)\ \delta(\bar{r}-\bar{r}_0)\ dP = \int_E \int_{4\pi} R_{adj}(E',\bar{\Omega})\ dE'\ d\bar{\Omega}$$

$$= \sum_{n=1}^{N} \int_{4\pi} d\Omega = 4\pi\ N. \qquad \text{(IV-34)}$$

Then the $4\pi N$ normalization factor for the forward source does in fact multiply the adjoint estimate. Therefore, the adjoint MORSE Monte Carlo answer must be divided by this factor. It is important to notice that in this example the $4\pi$ factors cancel out, but this may not be the case for different calculations.

# CHAPTER V

## RESULTS

This chapter presents the results obtained using the original version of the MORSE/BREESE package, the modified versions of the MORSE/BREESE package, and the new MORSE/STORM package. The first subsection presents results obtained using the unmodified version of the MORSE/BREESE package. Results from a modified version of the MORSE/BREESE package which includes the tracking of pseudo-particles are presented in the second subsection. The third subsection presents results from a modified version of the MORSE/BREESE package in which the displacement of the point of emergence is included as well as the tracking of pseudo-particles. The forth subsection presents the comparison of albedo data generated using various Monte Carlo and discrete ordinates procedures. The fifth subsection presents forward mode results using the new MORSE/STORM package, and the sixth subsection presents the adjoint mode results using the MORSE/STORM package.

## RESULTS USING THE STANDARD MORSE/BREESE PACKAGE

The MORSE Monte Carlo random-walk is modified by the albedo event such that particle transport is not performed within albedo media. Next-event estimation is performed at each albedo event which is equal to the probability that the particle would be reflected in the direction of the detector multiplied by the probability for uncollided transport of the particle to the detector. The standard version of the MORSE/BREEZE package makes contributions only to detectors which are in the hemisphere of the incident particle - this precludes including certain potentially very important contributions from other albedo media. This assumption might produce acceptable results if the effect of interest is not measurably affected by contributions from collisions inside the albedo region. It therefore follows that a careful use of the next-event estimator in conjunction with albedo-modified random walks is required.

The boundary crossing estimator receives contributions from particles which actually cross designated surfaces and it does not allow estimation at positions other than those enclosed by albedo surfaces.

A study was conducted for the shield configuration shown in Figures V-1 and V-2 to evaluate the performance of both estimators. The concrete composition used and the energy group structure of the cross-sections are presented

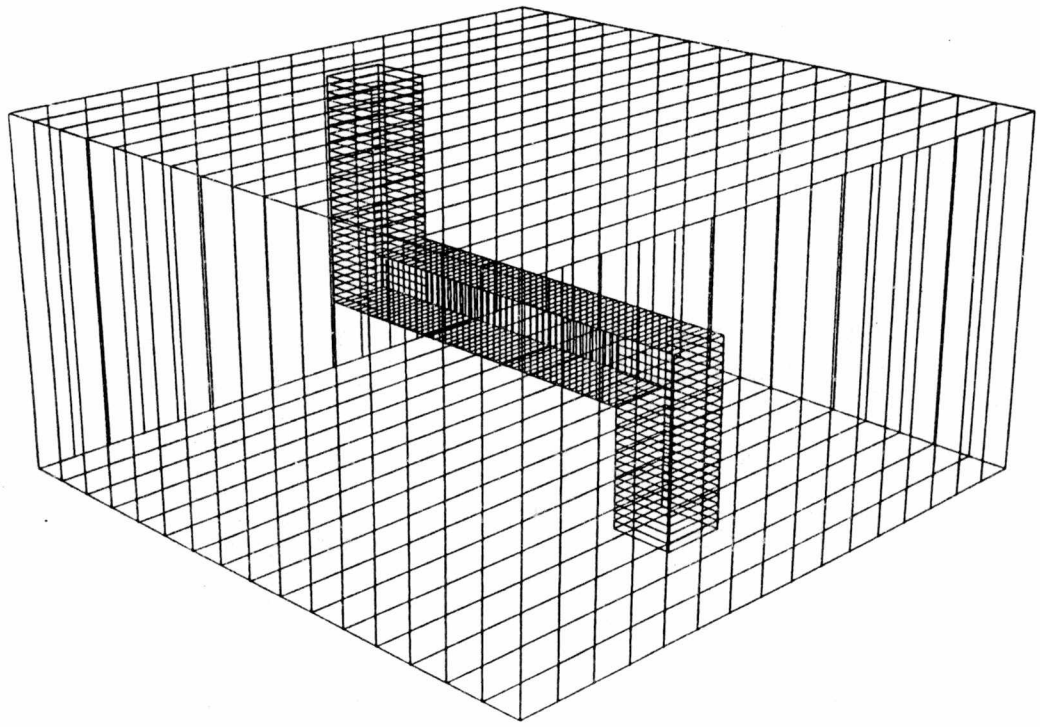


Figure V-1. Three-Dimensional View of the Geometric Configuration of the Example Problem.

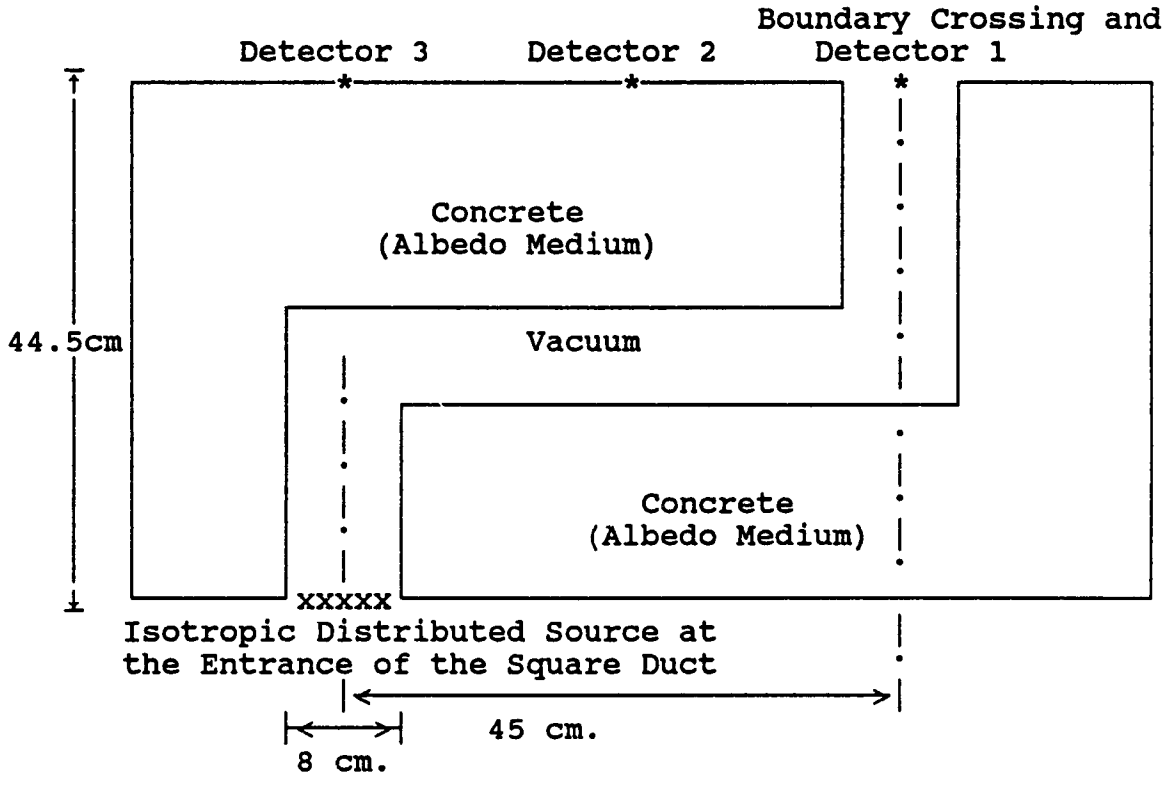



Figure V-2. Shield Configuration and Detector Locations for the Example Problem.

in the Appendix C. Monte Carlo solutions were obtained for each type of estimator and comparisons made with full transport MORSE and MCNP benchmarks for this problem. Table (V-1) presents the results, the fractional standard deviations, and the figure-of-merits for each case. It can be seem that the MORSE/BREESE calculations underestimate the true answer even though the fractional standard deviations are relatively small. It can be seen that the next-event and the boundary crossing estimators do not yield the same results, and that the standard deviation for the boundary crossing estimator is large even after 5,000,000 particles have been processed. It can be concluded that the MORSE/BREESE code package does not faithfully model the example problem.

The problem solved presents a complex geometry - a three-legged streaming path for the radiation and a small physical size. The traditional "corner effect" could be blamed for the large discrepancies observed in the results. The reason behind this assertion is that the albedos calculated for semi-infinite media do not correctly represent the full transport behavior near the intersections of albedo surfaces. Nonphysical distortions of the radiation field occur when two adjacent albedo surfaces intersect to form a corner. To possibly mitigate this problem, calculations were performed with full transport within a concrete wedge at each corner. Table (V-2) presents results using point albedo data with full transport at the corners compared with the

Table V-1. Results for the Benchmark Case (Full Transport MORSE and MCNP) and for the Standard MORSE/BREESE Package Using Next-Event and Boundary Crossing Estimators.

| | Benchmark | Benchmark | Next-event | Boundary-crossing |
|---|---|---|---|---|
| | MORSE | MCNP | MORSE/BREESE | MORSE/BREESE |
| D[a] | 500,000[b] | 3,000,000 | 1,000,000 | 5,000,000 |
| 1 | 1.0618e-05[c]<br>(0.0161)[d]<br>[3.67][e] | 1.15490e-05<br>(0.0267) | 3.9303e-07<br>(0.0013)<br>[17,584] | 5.8597e-08<br>(0.12015)<br>[1.19] |
| 2 | 3.3060e-05<br>(0.0310)<br>[0.99] | 3.30359e-05<br>(0.0272) | 2.0154e-06<br>(0.0034)<br>[2,571] | — |
| 3 | 6.8373e-05<br>(0.0240)<br>[1.65] | 7.35721e-05<br>(0.0330) | 8.1034e-06<br>(0.00001)<br>[2.97e+08] | — |

[a] Location of the detectors are indicated in the Figure V-1.

[b] Total number of source particles processed in the run.

[c] Total flux with units = neutrons/cm**2/source neutron/sec.

[d] Fractional standard deviation associated with the result.

[e] Figure of merit = 1./(T $\sigma$**2), where T is the computer time and $\sigma$**2 is the variance associated with the result.

Table V-2. MORSE/BREESE Results With Full Transport at the Corners.

| | Benchmark | Next-event | Boundary-crossing |
|---|---|---|---|
| | MORSE | MORSE/BREESE | MORSE/BREESE |
| Detector[a] | 500,000[b] | 500,000 | 1,000,000 |
| 1 | 1.0618e-05[c]<br>(0.0161)[d]<br>[3.67][e] | 2.7672e-06<br>(0.0462)<br>[4.18] | 8.6265e-07<br>(0.1195)<br>[1.56] |
| 2 | 3.3060e-05<br>(0.0310)<br>[0.99] | 4.4297e-06<br>(0.00453)<br>[435.] | — |
| 3 | 6.8373e-05<br>(0.0240)<br>[1.65] | 1.1727e-05<br>(0.00179)<br>[2,786] | — |

[a] Location of the detectors are indicated in the Figure V-1.
[b] Total number of source particles processed in the run.
[c] Total flux with units = neutrons/cm**2/source neutron/sec.
[d] Fractional standard deviation associated with the result.
[e] Figure of merit.

MORSE benchmark. It can be noticed that these results are much closer to the benchmark results than those calculated without full transport at the corners. The next-event estimator detector and the boundary crossing estimator located at the exit of the duct had significant gains due to full transport at the corners - however the results are still significantly lower than the benchmark answers. Detectors at other positions did not demonstrate the same improvements. The problem of estimating to detectors outside the albedo surface envelope remains unresolved primarily because of the very restricted nature of the albedo statistical estimation procedure.

## RESULTS USING PSEUDO-PARTICLES

Tables V-3 and V-4 present results for the example problem using the modified version of the MORSE code which creates and tracks pseudo-particles inside albedo regions. The mechanisms of producing, tracking and killing pseudo-particles were described in detail in the Chapter IV. The results were obtained without employing any biasing technique when tracking pseudo-particles or when albedo-particles are tracked in a full-transport mode at the corners. Also, these results were obtained without displacement of the point-of-emergence with respect to the albedo event.

Table V-3. Results Using Pseudo-Particles Without Displacement of the Point-of-Emergence.

| | Benchmark | Case # 1[a] | Case # 2[b] | Case # 3[c] | Case # 4[d] |
|---|---|---|---|---|---|
| | MORSE | | | | |
| D[e] | 500,000[f] | 200,000 | 500,000 | 200,000 | 200,000 |
| 1 | 1.0618e-05[g]<br>(0.0161)[h]<br>[3.67][i] | 8.0767e-06<br>(0.03126)<br>[9.22] | 8.1946e-06<br>(0.01998)<br>[10.5] | 7.7923e-06<br>(0.03093)<br>[24.0] | 8.3649e-06<br>(0.03488)<br>[23.3] |
| 2 | 3.3060e-05<br>(0.0310)<br>[0.99] | 2.2687e-05<br>(0.04545)<br>[4.36] | 2.4332e-05<br>(0.03703)<br>[3.06] | 2.4683e-05<br>(0.04169)<br>[13.2] | 2.4145e-05<br>(0.07777)<br>[4.68] |
| 3 | 6.8373e-05<br>(0.0240)<br>[1.65] | 3.5440e-05<br>(0.03162)<br>[9.01] | 3.7203e-05<br>(0.02371)<br>[7.47] | 3.7110e-05<br>(0.03669)<br>[17.1] | 3.8629e-05<br>(0.05251)<br>[10.3] |

[a] Case # 1 has the generation of the pseudo-particles performed without any biasing.

[b] Case # 2 has the generation of pseudo-particles biased with the average weight of albedo-particles by region and by energy group.

[c] Case # 3 has the generation of pseudo-particles biased with the cosine of the half of the angle between the incident particle's direction and the vector connecting the point-of-incidence to an input point-of-interest.

[d] Case # 4 has the generation of pseudo-particles biased with probabilities assigned by the user for each energy group and each geometry region.

[e] Detectors - locations are indicated in Figure V-1.

[f] Total number of source particles processed in this run.

[g] Total flux with units = neutrons/cm**2/source neutron/sec.

[h] Fractional standard deviation associated with the result.

[i] Figure of merit.

Table V-4. Results Using Pseudo-Particles Without Displacement of the Point-of-Emergence and Performing Full Transport at the Corners.

| | Benchmark | Case # 1[a] | Case # 2[b] | Case # 3[c] | Case # 4[d] |
|---|---|---|---|---|---|
| | MORSE | | | | |
| D[e] | 500,000[f] | 200,000 | 500,000 | 200,000 | 200,000 |
| 1 | 1.0618e-05[g]<br>(0.0161)[h]<br>[3.67][i] | 4.6289e-06<br>(0.04082)<br>[9.02] | 4.6471e-06<br>(0.02368)<br>[11.2] | 4.2984e-06<br>(0.03488)<br>[20.1] | 4.5780e-06<br>(0.04473)<br>[13.9] |
| 2 | 3.3060e-05<br>(0.0310)<br>[0.99] | 1.7947e-05<br>(0.05482)<br>[5.00] | 1.8271e-05<br>(0.02635)<br>[9.06] | 1.8302e-05<br>(0.03789)<br>[17.0] | 1.7908e-05<br>(0.04772)<br>[12.2] |
| 3 | 6.8373e-05<br>(0.0240)<br>[1.65] | 4.0787e-05<br>(0.04082)<br>[9.02] | 3.9320e-05<br>(0.02301)<br>[11.9] | 4.0302e-05<br>(0.04321)<br>[13.1] | 3.8367e-05<br>(0.03429)<br>[23.6] |

[a] Case # 1 has the generation of pseudo-particles performed without any biasing.

[b] Case # 2 has the generation of pseudo-particles biased with the average weight of the albedo particles by region and energy group.

[c] Case # 3 has the generation of pseudo-particles biased with the cosine of the half of the angle between the incident particle's direction and the vector from the point-of-incidence to an input point-of-interest.

[d] Case # 4 has the generation of pseudo-particles biased with the user specified probabilities by energy group and by geometry region.

[e] Detector - locations are indicated in Figure V-1.

[f] Total number of source particles processed in this run.

[g] Total flux with units = neutrons/cm**2/source neutron/sec.

[h] Fractional standard deviation associated with the result.

[i] Figure of merit.

Several survival biasing techniques were used in the generation of pseudo-particles. Tables V-3 and V-4 present results for the MORSE benchmark case, a calculation without biasing when generating pseudo-particles, a case using the average weight of the albedo-transported particles for biasing the generation of pseudo-particles, a case using cosine biasing, and finally a calculation in which probabilities are assigned by energy group and by geometry region for generating pseudo-particles. In comparing these results with those already presented, it can be seen that significant improvements were achieved for all detectors in particular for the detectors located behind albedo surfaces (detectors 2 and 3). It can also be noticed that biasing the generation of pseudo-particles improves the figure-of-merit. The cosine biasing scheme was shown to be the best option for this problem. The probability by energy group and region biasing procedure could have been more effective if the input probabilities were based on adjoint information.

Table V-4 presents results for the same cases presented in Table V-3 except that the results in Table V-4 were obtained with full-transport at the corners. It should be noted that the results for detector 1 from the calculations with full-transport for albedo-particles at the corners are about 50 percent lower than those without full transport. This difference can be explained by the position of the albedo surface behind the concrete wedge at the

corner closest to the source. Pseudo-particles which would have been created at this surface would not be permitted to contribute to any detector, thereafter no pseudo-particle was created at this surface. Results for detector 2 are only about 20 percent lower when a wedge of concrete is placed at the corners - what would be expected due to the lesser importance of the pseudo-particles at positions behind the corner-albedo-surface would have for detector 2 as compared with detector 1. Detector 3 has an improved performance due to albedo-particles which are scattered back from the corner-albedo-surface nearest the source. What this implies is that the pseudo-particles may represent the transport through corners very well and that placing a concrete wedge at the corners with an albedo surface behind the wedge has an adverse effect on the accuracy of the estimates.

## MONTE CARLO CALCULATED AVERAGE POINT-OF-EMERGENCE

The MCNP code was used to estimate the average displacement of the point of emergence with respect to the position of an albedo event. The geometry used and the methodology adopted in calculating the average displacement are presented in Chapter IV. Figures V-3 and V-4 show the radial distance and angular position distributions of the total albedo for a 14-Mev neutron entering a concrete block

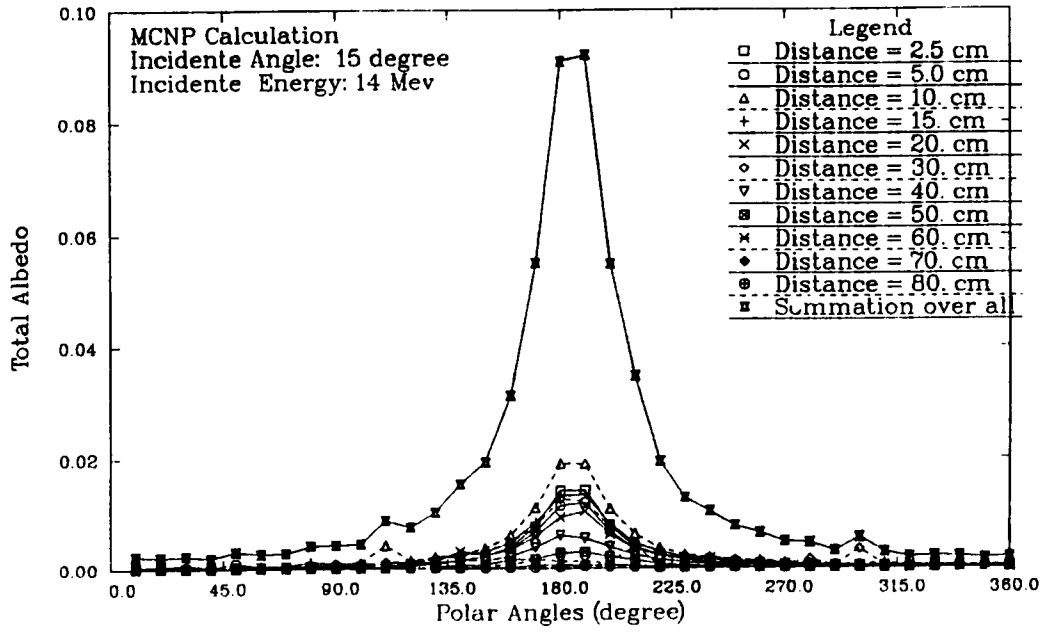



**Figure V-3. Angular Position of the Emergent Particles in Relation to the Point-of-Incidence.**

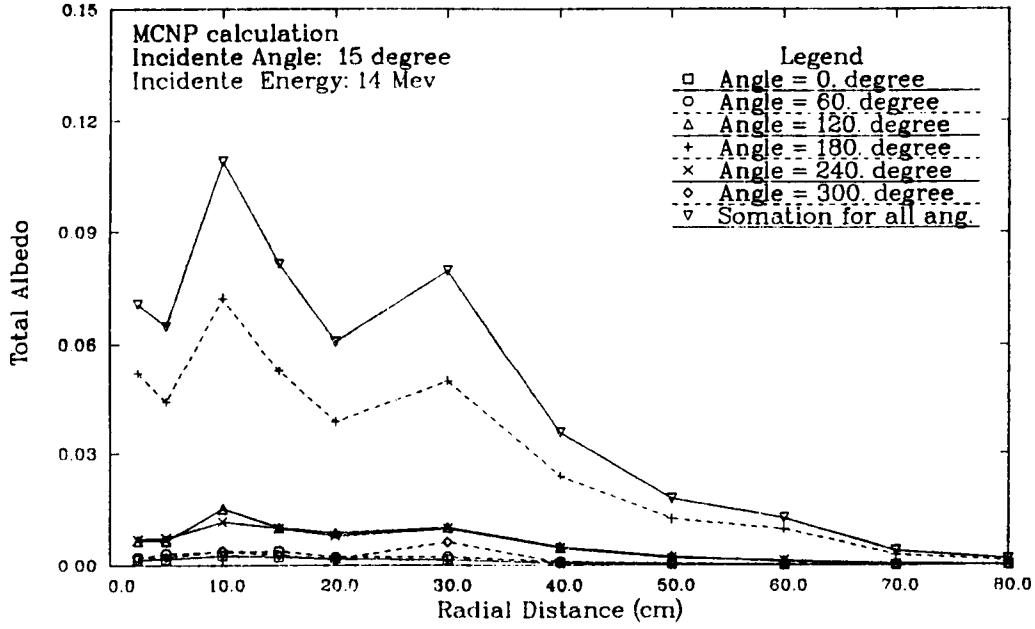



**Figure V-4. Radial Distance from the Point-of-Incidence to the Point-of-Emergence.**

with a polar angle of 75°. The emergent directional distribution shown in Figure V-3 is highly peaked in the forward direction. Therefore the emergent position is more likely to be downstream from the point-of-incidence. Figure V-4 shows that the displacement of the emergent point is significant and should not be neglected, and that the most important component of the total displacement is the forward direction (180°).

## RESULTS USING AN AVERAGE DISPLACEMENT FOR THE EMERGENT POINT

Table V-5 presents the results obtained with the modified version of the MORSE/BREESE package which includes the displacement of the point-of-emergence in relation to the point of incidence at an albedo event. The inclusion of a displacement for the point-of-emergence has improved the accuracy of the results relative to the MORSE benchmark from the previous 25% (low) for the results for pseudo-particles without displacement to approximately 10% (low). It can also be noticed that this deviation remains constant for all detectors used in the problem suggesting a better characterization of the radiation field - which was not the case for the results presented in the previous subsections.

Table V-6 presents the results of calculations using pseudo-particles, the displacement of the point-of-emer-

Table V-5. Results Using Pseudo-Particles With Displacement of the Point-of-Emergence in the Direction of the Incident Particle Projected on the Albedo Plane.

| | Benchmark | Case # 1[a] | Case # 2[b] | Case # 3[c] | Case # 4[d] |
|---|---|---|---|---|---|
| | MORSE | | | | |
| D[e] | 500,000[f] | 200,000 | 500,000 | 200,000 | 200,000 |
| 1 | 1.0618e-05[g] (0.0161)[h] [3.67][i] | 8.9060e-06 (0.02663) [12.7] | 9.2179e-06 (0.01692) [14.7] | 9.1657e-06 (0.02763) [18.3] | 9.0989e-06 (0.03920) [15.3] |
| 2 | 3.3060e-05 (0.0310) [0.99] | 3.0318e-05 (0.05083) [3.49] | 2.7009e-05 (0.02601) [6.21] | 2.6266e-05 (0.04614) [6.55] | 2.7367e-05 (0.04817) [10.2] |
| 3 | 6.8373e-05 (0.0240) [1.65] | 5.8924e-05 (0.03376) [7.90] | 5.9686e-05 (0.02638) [6.04] | 5.6107e-05 (0.02856) [17.1] | 6.0418e-05 (0.04256) [13.0] |

[a] Case # 1 has the generation of pseudo-particles performed without any biasing.

[b] Case # 2 has the generation of pseudo-particles biased with the average weight of the albedo-particles by geometry region and energy group.

[c] Case # 3 has the generation of pseudo-particles biased with the cosine of the half of the angle between the particle's incident direction and the vector connecting the point-of-incidence to the input point-of-interest.

[d] Case # 4 has the generation of pseudo-particles biased with the input probabilities by geometry region and by energy group.

[e] Detector - locations are indicated in Figure V-1.

[f] Total number of source particles processed in this run.

[g] Total flux with units = neutrons/cm**2/source neutron/sec.

[h] Fractional standard deviation associated with the result.

[i] Figure of merit.

Table V-6. Results Using Pseudo-Particles With Displacement in the Direction of the Incident Particle Projected on the Albedo Plane and Performing Full Transport at the Corners.

| | Benchmark | Case # 1[a] | Case # 2[b] | Case # 3[c] | Case # 4[d] |
|---|---|---|---|---|---|
| | MORSE | | | | |
| D[e] | 500,000[f] | 200,000 | 500,000 | 200,000 | 200,000 |
| 1 | 1.0618e-05[g]<br>(0.0161)[h]<br>[3.67][i] | 7.4751e-06<br>(0.02740)<br>[15.7] | 7.5386e-06<br>(0.02003)<br>[9.23] | 7.3664e-06<br>(0.03304)<br>[13.6] | 7.3700e-06<br>(0.03855)<br>[15.9] |
| 2 | 3.3060e-05<br>(0.0310)<br>[0.99] | 2.6459e-05<br>(0.03783)<br>[6.72] | 2.6511e-05<br>(0.02217)<br>[8.85] | 2.6462e-05<br>(0.03724)<br>[10.7] | 2.6158e-05<br>(0.06207)<br>[6.14] |
| 3 | 6.8373e-05<br>(0.0240)<br>[1.65] | 5.6919e-05<br>(0.03676)<br>[7.16] | 5.7592e-05<br>(0.01685)<br>[15.3] | 5.6680e-05<br>(0.02556)<br>[22.7] | 5.7810e-05<br>(0.03148)<br>[23.9] |

[a] Case # 1 has the generation of pseudo-particles performed without any biasing.

[b] Case # 2 has the generation of pseudo-particles biased with the average weight of albedo-particles by geometry region and energy group.

[c] Case # 3 has the generation of pseudo-particles biased with the cosine of the half of the angle between the incident particle's direction and the vector connecting the point-of-incidence to the input point-of-interest.

[d] Case # 4 has the generation of pseudo-particles biased with the input probabilities by energy group and by geometry region.

[e] Detectors - locations are indicated in Figure V-1.

[f] Total number of source particles processed in this run.

[g] Total flux with units = neutrons/cm**2/source neutron/sec.

[h] Fractional standard deviation associated with the result.

[i] Figure of merit.

gence, and full-transport at the corners. The same behavior as observed early is again presented - adding a concrete wedge produces results that deviate (low) from the benchmark results. This consistency in the trends demonstrates that the pseudo-particles better represent the particle transport through the corner than the inclusion of a concrete wedge. In this case, the results for calculations with full transport at the corners are higher than those of Table V-4 due to the downstream displacement at the albedo surfaces behind the concrete wedges.

Another computational feature observed is the improvement in the figure-of-merit for the calculations which included the displacement of the point of emergence. In general the figure-of-merit increased by 20% to 30% when compared with similar calculations. The significance of this improvement in the figure-of-merit is that the additional computer time spent in performing the displacement phase of the albedo-transported particles was accompanied by a significant decrease in the variance - probably due to a more uniform profile of the contributions to the detectors. In some cases, the figure-of-merits for results which included the displacement of the point-of-emergence decreased in value.

Table V-7 presents results similar to those reported in Table V-5 but using a different approach in displacing a particle which undergoes an albedo event. The procedure

Table V-7. Results Using Pseudo-Particles With Displacement of the Point-of-Emergence in the Direction of the Axis of the Duct.

| D[e] | Benchmark MORSE 500,000[f] | Case # 1[a] 200,000 | Case # 2[b] 500,000 | Case # 3[c] 200,000 | Case # 4[d] 200,000 |
|---|---|---|---|---|---|
| 1 | 1.0618e-05[g]<br>(0.0161)[h]<br>[3.67][i] | 9.5988e-06<br>(0.02599)<br>[17.2] | 9.2716e-06<br>(0.01572)<br>[19.4] | 9.6434e-06<br>(0.02633)<br>[26.9] | 9.9202e-06<br>(0.04025)<br>[15.9] |
| 2 | 3.3060e-05<br>(0.0310)<br>[0.99] | 2.8096e-05<br>(0.06266)<br>[2.96] | 2.7029e-05<br>(0.02754)<br>[6.31] | 2.4593e-05<br>(0.03589)<br>[14.5] | 2.9133e-05<br>(0.05094)<br>[9.93] |
| 3 | 6.8373e-05<br>(0.0240)<br>[1.65] | 5.0004e-05<br>(0.04073)<br>[7.01] | 4.9007e-05<br>(0.02099)<br>[10.9] | 5.2432e-05<br>(0.03782)<br>[13.1] | 4.8521e-05<br>(0.04694)<br>[11.7] |

[a] Case # 1 has the generation of pseudo-particles performed without any biasing.

[b] Case # 2 has the generation of pseudo-particles biased with the average weight of albedo-particles by geometry region and energy group.

[c] Case # 3 has the generation of pseudo-particles biased with the cosine of the half of the angle between the incident particle's direction and the vector connecting the point-of-incidence to the input point-of-interest.

[d] Case # 4 has the generation of pseudo-particles biased with the input probabilities by energy group and by geometry region.

[e] Detectors - locations are indicated in Figure V-1.

[f] Total number of source particles processed in this run.

[g] Total flux with units = neutrons/cm**2/source neutron/sec.

[h] Fractional standard deviation associated with the result.

[i] Figure of merit.

consists of projecting the displacement-vector (which has a magnitude equal to the absolute length of the displacement and a direction equal to that of the incident particle projected onto the albedo plane) onto the direction of the axis of the duct. Some improvement was obtained using this displacement procedure in particular for the detector at the exit of the duct. It is important to point out that in this procedure, no weight correction is applied to the albedo particle due to this "bias" in the direction of the displacement. The results indicated that this assumption is acceptable for the example problem but a universal application of this procedure may be questionable due to the tendency for underestimating the results at positions other than at the exit of the duct (as can be seen by comparing the results presented in Tables V-5 and V-7).

Table V-8 presents results for calculations with the displacement in the direction of the axis of the duct and full transport at the corners. It can be noted that these results deviate (low) from results reported in Table V-7 by about the same percentage as the deviations observed between Tables V-5 and V-6. The variations in the deviations follow the general trends of the results of other cases with full transport at the corners and pseudo-particle generation.

Table V-8. Results Using Pseudo-Particles With Displacement of the Point-of-Emergence in the Direction of the Axis of the Duct and Performing Full Transport at the Corners.

| D[e] | Benchmark MORSE 500,000[f] | Case # 1[a] 200,000 | Case # 2[b] 500,000 | Case # 3[c] 200,000 | Case # 4[d] 200,000 |
|---|---|---|---|---|---|
| 1 | 1.0618e-05[g] (0.0161)[h] [3.67][i] | 8.7858e-06 (0.02651) [17.6] | 9.1419e-06 (0.01935) [13.8] | 8.9308e-06 (0.03259) [18.8] | 8.8041e-06 (0.03466) [20.2] |
| 2 | 3.3060e-05 (0.0310) [0.99] | 2.2602e-05 (0.03649) [9.27] | 2.4055e-05 (0.02750) [6.84] | 2.3731e-05 (0.03549) [15.8] | 2.3969e-05 (0.05021) [9.63] |
| 3 | 6.8373e-05 (0.0240) [1.65] | 5.0181e-05 (0.03031) [13.4] | 4.9267e-05 (0.02100) [11.7] | 4.9078e-05 (0.02565) [30.3] | 4.8682e-05 (0.03045) [26.2] |

[a] Case # 1 has the generation of pseudo-particles performed without any biasing.

[b] Case # 2 has the generation of pseudo-particles biased with the average weight of albedo-particles by geometry region and energy group.

[c] Case # 3 has the generation of pseudo-particles biased with the cosine of the half of the angle between the incident particle's direction and the vector connecting the point-of-incidence to the input point-of-interest.

[d] Case # 4 has the generation of pseudo-particles biased with the input probabilities by energy group and by geometry region.

[e] Detectors - locations are indicated in Figure V-1.

[f] Total number of source particles processed in this run.

[g] Total flux with units = neutrons/cm**2/source neutron/sec.

[h] Fractional standard deviation associated with the result.

[i] Figure of merit.

## COMPARISON OF ALBEDO DATA GENERATED WITH DIFFERENT METHODS

Considering that essentially all of the "lost" information for a particle experiencing an albedo event using the standard albedo data is recovered and that the results are still consistently low - then the validity of the standard albedo data must be questioned. The assumption that the albedo data correctly describe the true reflection process was not supported by the present calculations. A new set of calculations was performed with the MCNP code to benchmark the angular distribution of the albedo data for a semi-infinite concrete medium in plane geometry. The polar angular distribution of the point albedos calculated with the MCNP code did agree with the distribution of the standard albedo data, but the azimuthal distribution of the available albedo data was shown to be more isotropic and the scattering more preferential in the backward directions than indicated by MCNP distributions. Figures V-5, and V-6 show comparisons of the azimuthal distribution calculated with the MCNP code and with that from a standard albedo data set for two different combinations of incident polar angle and energy. The plots are displayed for the five $\eta$-levels - "level 1" corresponds to the largest cosine of the polar angle and "level 5" represents the smallest cosine of the polar angle in the quadrature set. Within each level, smaller abscissa values correspond to more forward azimuthal

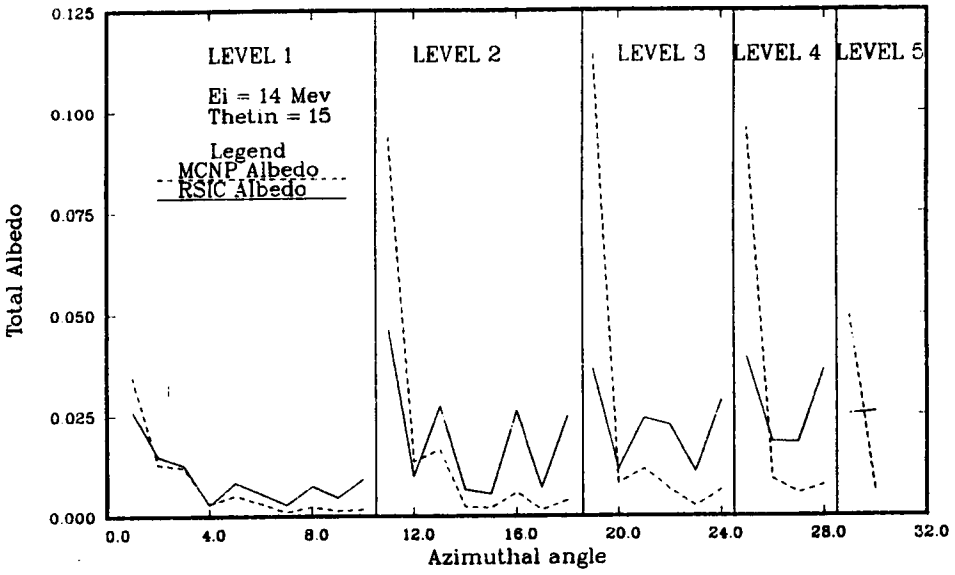



Figure V-5. Angular Azimuthal Distribution of the Available Albedo Data and of the MCNP Calculated Data. The Incident Polar Angle is 75° and the Neutron Energy is 14 Mev.

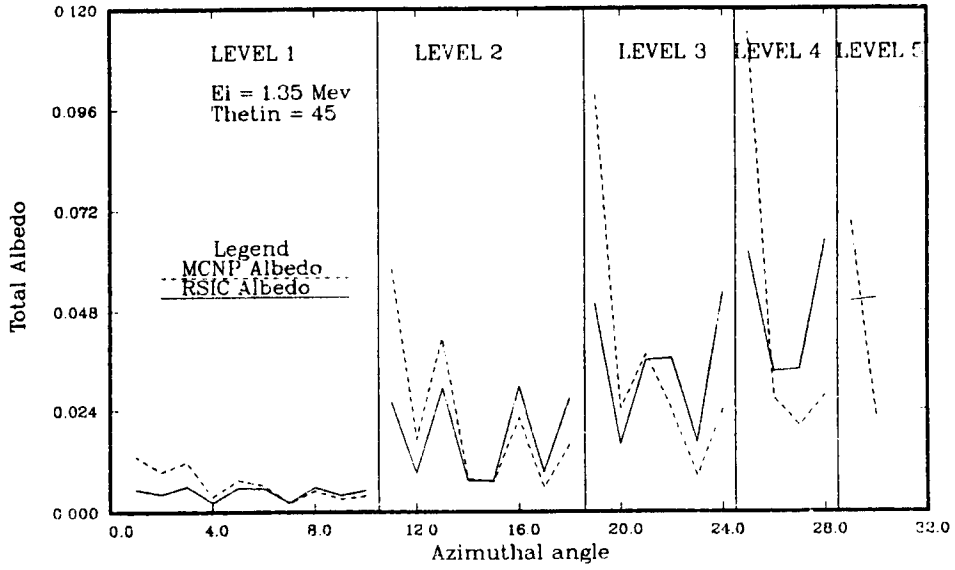



Figure V-6. Angular Azimuthal Distribution of the CARP Albedo Data and of the MCNP Albedo Information. The Incident Polar Angle is 45° and the Neutron Energy is 1.35 Mev.

angles. To better understand why the available albedo data exhibit poor correlations with the MCNP-estimated azimuthal distributions, a study was carried out to analyze the angular flux distributions generated by the discrete ordinates DOT code - which is the main tool for the calculation of albedo data. It was found that negative fluxes were calculated for many discrete directions and that the negative fluxes occurred more frequently for high neutron energy groups. These negative fluxes are in most cases associated with the forward directions and they are at least in part due to the low order expansion of the cross sections. These negative fluxes are set equal to zero in the BREESE code thereby reducing the magnitudes of the forward-directed components. Inconsistencies can be observed in the value of the total albedo as given by the CARP code and that produced by the BREESE code. After the negative fluxes are set equal to zero by the BREESE code, the value for the total albedo more than doubles for several combinations of incident energy group and direction. Another procedural issue that can be addressed is the sampling scheme implemented in the CARP/BREEZE system. Point albedos are characterized in terms of three phase space coordinates of the emergent particle - energy, eta-level, and azimuthal angle. The sampling scheme used in the BREEZE code was to first select an eta-level from the marginal probability distribution for the polar angle, then an azimuthal angle is selected from the condi-

tional probability distribution for the azimuthal angle, and finally the energy is selected from the doubly-conditional energy distribution. The problem for concrete is that the thermal neutron component is very large due to the presence of hydrogen. Then the angular distribution, which is independent of the outgoing energy in the sampling scheme used in BREESE, is almost completely dominated by the nearly isotropic angular distribution of the thermal neutrons. This explains in part the isotropic character of the angular distribution of the DOT generated albedo data and because of the negative fluxes in the basic data, this distortion can not be corrected.

Based on these results, a decision was made to generate a new set of albedo data. The usual discrete ordinates procedures for generating albedo data were inappropriate because they only calculate point albedos. The use of Monte Carlo codes for generating albedo data is not practical due to the prohibitively large computer time required to obtain detailed albedo information with acceptable precision. The only viable option was to use a three-dimensional discrete ordinates code to provide a detailed description of the radiation field while requiring an acceptable amount of computer time. The TORT three-dimensional discrete ordinates code was used to generate a new set of albedo data as described in Chapter IV. The problems related to negative angular fluxes were addressed and input options were select-

ed which provide positive fluxes for all combinations of mesh spaces, quadrature directions, and energy groups.

Figure V-7 displays plots of the angular distribution of the CARP generated point albedo, the MCNP calculated albedo information (with the spatial variable collapsed to a point), and the TORT calculated albedo data with P3 and P8 cross-sections, that also had the spatial variable collapsed to a point for display as point albedos. The order-numbers displayed on the x-axis of the Figure V-7 are related to the azimuthal angle in such a way that smaller numbers within each level correspond to more negative cosines of the azimuthal angle. It should be noted that the quadrature set used for generating TORT albedo data is different from that used for generating CARP and MCNP albedos. Table V-9 presents the limits of the polar angles for the two quadrature sets used - the level numbers presented in the header of the table are the same as the level numbers displayed in Figures V-5, V-6, and V-7. The limits of the azimuthal angle within each "$\eta$-level" for each quadrature set are presented in Table V-10.

Figure V-8 displays a plot of the spatial distributions of the TORT albedos with P3 cross-sections.

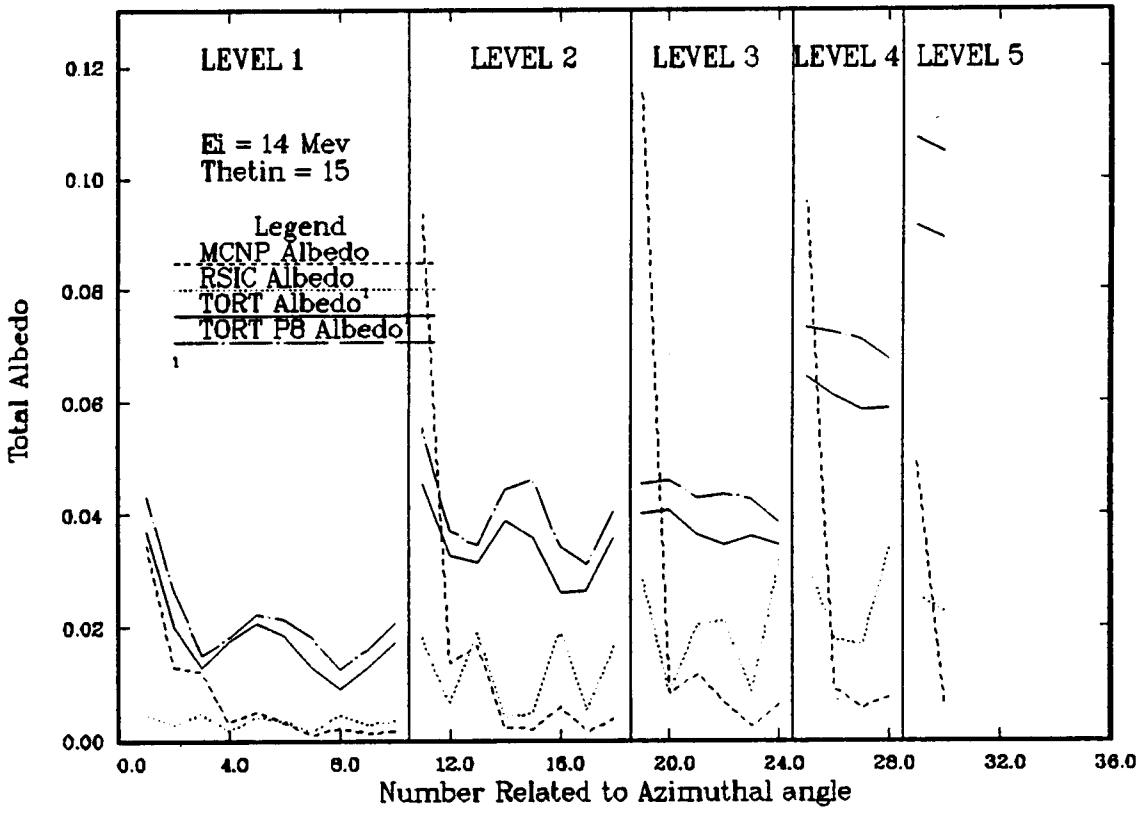


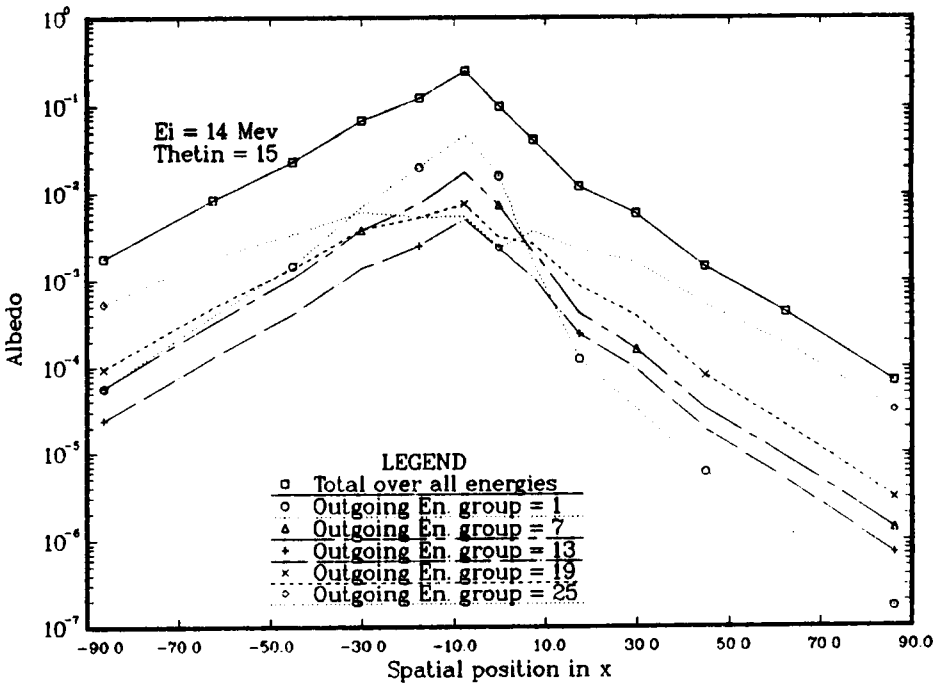

Figure V-7. Comparison of the Angular Distribution of Various Albedo Data Sets.

Table V-9. Limits of the Polar Angles of the CARP and MCNP Albedo Data Compared With the Limits of the Polar Angles of the TORT Albedo Data.

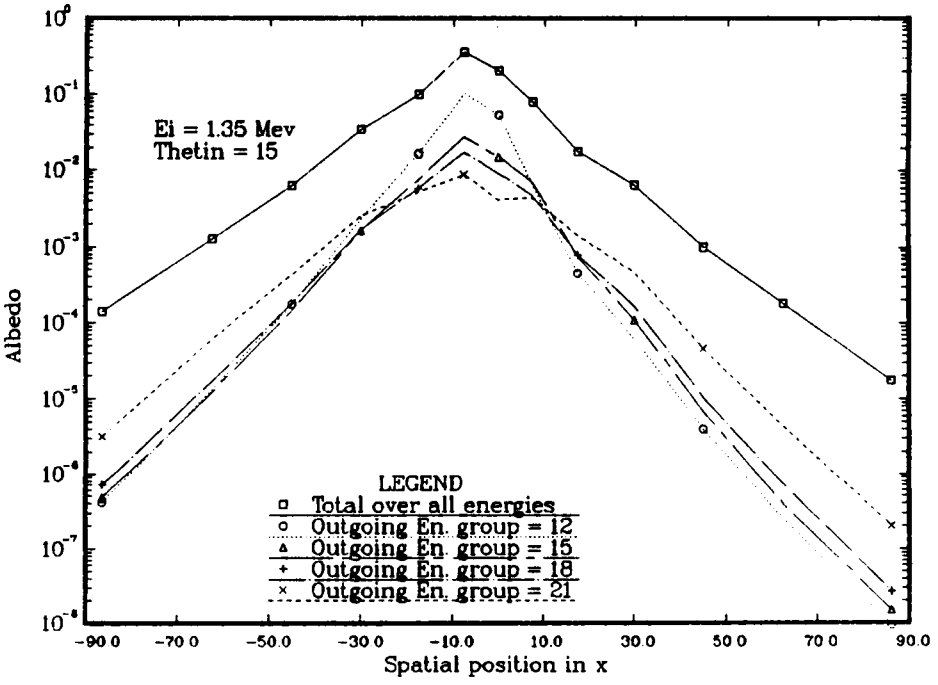
| | cosθ Limits | | | | |
|---|---|---|---|---|---|
| | Level 5 | Level 4 | Level 3 | Level 2 | Level 1 |
| CARP MCNP | 1.0 | .93333 | .78388 | .56479 | .29552 | 0. |
| TORT | 1.0 | .90800 | .76825 | .61728 | .37313 | 0. |

Table V-10. Limits of the Azimuthal Angles of the CARP and MCNP Albedo Data Compared With the Limits of the Azimuthal Angles in the TORT Albedo Data.

| cosϕ Limits | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Level 5 | | Level 4 | | Level 3 | | Level 2 | | Level 1 | |
| CARP | TORT | CARP | TORT | CARP | TORT | CARP | TORT | CARP | TORT |
| -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| 0.0000 | 0.0000 | -0.5169 | -0.7071 | -0.7147 | -0.8709 | -0.8417 | -0.9006 | -0.9379 | -0.9259 |
| 1.0000 | 1.0000 | 0.0000 | 0.0000 | -0.5235 | -0.4915 | -0.7180 | -0.7071 | -0.8086 | -0.7767 |
| | | 0.5169 | 0.7071 | 0.0000 | 0.0000 | -0.1593 | -0.4346 | -0.5084 | -0.6299 |
| | | 1.0000 | 1.0000 | 0.5235 | 0.4915 | 0.0000 | 0.0000 | -0.3779 | -0.3777 |
| | | | | 0.7147 | 0.8709 | 0.1593 | 0.4346 | 0.0000 | 0.0000 |
| | | | | 1.0000 | 1.0000 | 0.7180 | 0.7071 | 0.3779 | 0.3777 |
| | | | | | | 0.8417 | 0.9006 | 0.5084 | 0.6299 |
| | | | | | | 1.0000 | 1.0000 | 0.8086 | 0.7767 |
| | | | | | | | | 0.9379 | 0.9259 |
| | | | | | | | | 1.0000 | 1.0000 |




A) Incident Energy 14 MeV. and Incident Polar Angle 75°




B) Incident Energy 1.35 MeV. and Incident Polar Angle 75°

Figure V-8. Spatial Distributions of the Tort Albedos.

## RESULTS USING SPATIALLY-DEPENDENT ALBEDOS - FORWARD MODE

The MORSE code was modified to accommodate the three-dimensional albedo data. The ALBFAC3D interface code was written to format the albedo information generated by the TORT code into a file which can be read by the MORSE/STORM code package. The same example problem as solved for several biasing procedures used in generating pseudo-particles. Table V-11 presents results obtained with the MORSE/STORM code using spatially-dependent albedos.

It can be noticed that the Monte Carlo estimation of the effect-of-interests for the detectors 1 and 2 using three-dimensional albedos are essentially the same as those of the MORSE benchmark case. The results for detector 3 are slightly higher than for the benchmark case. However if this result is compared with that presented on Table V-1 for the MCNP benchmark case, an excellent agreement is found. Detector 1 has only 17% of the total contributions coming from albedo events, the remaining 83% of the contributions come from real collisions of the pseudo-particles. This breakdown between the two components of the response explains the agreement observed between the MORSE benchmark response and the MORSE/STORM response for this detector. It is noted that the next-event statistical estimations as well as the scattering of the pseudo-particles are performed using the standard procedures implemented in the MORSE code - and as

Table V-11. MORSE/STORM Results With Spatially-Dependent Albedos Are Compared with MORSE Benchmark Results.

| D[e] | Benchmark MORSE 500,000[f] | Case # 1[a] 500,000 | Case # 2[b] 500,000 | Case # 3[c] 500,000 | Case # 4[d] 500,000 |
|---|---|---|---|---|---|
| 1 | 1.0618e-05[g] (0.0161)[h] [3.67][i] | 1.0332e-05 (0.02007) [16.3] | 1.0298e-05 (0.02254) [12.9] | 1.0332e-05 (0.02305) [12.4] | 1.0301e-06 (0.02192) [13.1] |
| 2 | 3.3060e-05 (0.0310) [0.99] | 3.1782e-05 (0.02883) [7.92] | 3.0206e-05 (0.03138) [6.68] | 2.9669e-05 (0.02751) [8.69] | 3.0157e-05 (0.03016) [6.92] |
| 3 | 6.8373e-05 (0.0240) [1.65] | 7.2280e-05 (0.01959) [17.1] | 7.1326e-05 (0.01743) [21.7] | 7.3833e-05 (0.01823) [19.8] | 7.2335e-05 (0.01684) [22.2] |

[a] Case # 1 has the generation of pseudo-particles performed without any biasing.

[b] Case # 2 has the generation of pseudo-particles biased with the average weight of albedo-particles by geometry region and energy group.

[c] Case # 3 has the generation of pseudo-particles biased with the cosine of the half of the angle between the incident particle's direction and the vector connecting the point-of-incidence to the input point-of-interest.

[d] Case # 4 has the generation of pseudo-particles biased with the input probabilities by energy group and by geometry region.

[e] Detectors - locations are indicated in Figure V-1.

[f] Total number of source particles processed in this run.

[g] Total flux with units = neutrons/cm**2/source neutron/sec.

[h] Fractional standard deviation associated with the result.

[i] Figure of merit.

it would be expected the MORSE/STORM results are closer to the MORSE benchmark results than to the MCNP benchmark results. The difference observed for detector 1 between the MORSE and MCNP benchmarks can be explained by the cross-section treatments as well as differences in scattering and estimation procedures. The MORSE benchmark calculation was performed using P3 cross-sections and the standard scattering procedure - the scattering process is described by two outgoing angles. This procedure does not effect the results after few collisions, but due to the small size of the example problem the first- and second-collided components of the radiation field are dominant. This leads to the conclusion, for this case, that some of the deviations observed between the MORSE and MCNP benchmark results are due to the MORSE treatment of the scattering event. The next-event contributions to the effect-of-interest are also affected by the low order of truncation of the Legendre polynomials. For this particular problem, due to a highly localized region-of-importance for each detector, the P3 representation of the scattering process may not be sufficiently accurate.

Detector 3, which is aligned with the source through the axis of the first leg of the penetration, receives 22% of its contributions from albedo events and 8% coming from the uncollided component. The fact that the results for this detector are closer to the MCNP benchmark suggests that for

this particular detector ray-like effects were present in the MORSE full-transport case.

## RESULTS USING SPATIALLY-DEPENDENT ALBEDOS - ADJOINT MODE

The MORSE code can perform both, forward and adjoint modes of analysis. The adjuncton albedos can be obtained from the same forward angular fluxes used in generating the forward albedos. The ALBFAC3D interface code is able to process the forward angular flux file generated by the TORT code and to synthesize either forward or adjuncton albedo data. The MORSE/STORM code is also able to read the albedo data file and to perform calculations in either the forward mode or the adjoint mode. Table V-12 presents the results of calculations performed using adjuncton albedos and the standard adjoint mode of analysis in the MORSE code. The geometric configurations used in these calculations are the same as those used in the forward calculations. The number of energy groups for the adjoint calculation was reduced to three top groups. The adjoint source is located at the position of the detector 1 (Figure V-2) and the adjoint estimation is performed to an adjoint point detector located at center of the entrance of the penetration - an approximation since the source in the forward calculation is uniformly distributed over the entrance. The energy spectrum of the

forward source (adjoint response function) is the same as that of the forward calculations (14-MeV neutron source). It can be noticed from the results presented in Table V-12 that the adjoint calculation performed using adjuncton-albedos yields statistically the same result as the full transport calculation and that the figure-of-merit is 50% larger when adjuncton-albedos are used. Both calculations were performed without the use of any biasing techniques, which shows that the use of adjuncton-albedos improves the computational efficiency.

Table V-12. Results of Adjoint Calculations Using the MORSE/STORM Code Package for the 3-Group Example Problem.

| | Full Transport[a] | Adjuncton Albedo[b] |
|---|---|---|
| | MORSE | MORSE/STORM |
| Detector[c] | 600,000[d] | 500,000 |
| 1 | 2.20371-06[e]<br>(0.02622)[f]<br>[32.1][g] | 2.26881e-06<br>(0.02315)<br>[48.3] |

[a] Standard MORSE code adjoint mode calculation performing full transport of the adjunctons (without any biasing).

[b] Standard MORSE/STORM adjoint mode calculation using adjuncton albedos and without biasing the generation of pseudo-adjuncton-particles.

[c] Point adjoint source location as indicated in Figure V-1. The point adjoint detector location is at the center of the entrance of the first leg (forward planar source replaced by a point source).

[d] Total number of source particles processed in this run.

[e] Total flux with units = neutrons/cm**2/source neutron/sec.

[f] Fractional standard deviation associated with the result.

[g] Figure of merit.

Table V-13 presents a comparison of the MORSE/STORM adjuncton-albedo calculation for the standard (complete) example problem with MORSE and MCNP forward calculated benchmarks. This calculation is not a fair evaluation of the adjuncton-albedo calculation because the adjoint response, a 14-MeV point neutron source, creates a inherently difficult problem to solve in the adjoint mode. Also, the decision not to use biasing procedures when performing any of the Monte Carlo random-walks precluded the optimization of the adjoint calculation - which for the example problem would involve energy biasing.

Table V-13. Comparison of the Adjuncton-Albedo Calculation with the MORSE and MCNP Benchmark Results for the Example Problem.

| Detector[d] | Benchmark[a] MORSE 500,000[e] | Benchmark[b] MCNP 3,000,000 | Adjoint[c] MORSE/STORM 4,200,000 |
|---|---|---|---|
| 1 | 1.0618e-05[f]<br>(0.0161)[g]<br>[3.67][h] | 1.15532e-05<br>(0.0079)<br>[24.0] | 1.23179e-05<br>(0.0635)<br>[3.74] |

[a] Standard MORSE code forward mode calculation performing full transport of the particles (without any biasing).

[b] Standard MCNP code forward mode calculation performing full transport of the particles (without any biasing).

[c] Standard MORSE/STORM adjoint mode calculation using adjuncton albedos and without biasing the generation of pseudo-adjuncton-particles.

[d] Point adjoint source location (the same as the forward point detector) as indicated in Figure V-1. In the adjoint mode the forward planar source is replaced by a point adjoint detector located at the center of the entrance of the first leg.

[e] Total number of source particles processed in this run.

[f] Total flux with units = neutrons/cm**2/source neutron/sec.

[g] Fractional standard deviation associated with the result.

[h] Figure of merit.

# CHAPTER VI

## CONCLUSIONS AND FUTURE WORK

This research demonstrated the new procedures that permit a more generalized utilization of the albedo methodology in both forward and adjoint modes of Monte Carlo calculation. With the use of "pseudo-particles", it was possible to recover an essential part of the information that would have been lost at the albedo event - this recovered information was used to calculate a missing and very important component in the statistical estimation procedure. The spatial variable was included in the albedo procedure either through Monte Carlo generated supplementary information or using directly calculated spatially-dependent albedo data. It was also demonstrated that, due to the limitations of the point-albedos in truly representing the angular distribution of the reflection process, the modified version of the MORSE/BREESE code package which includes both the displacement of the point-of-emergence of an albedo event and the tracking of pseudo-particles does not yield the same results as the benchmark calculations. The MORSE/STORM code package, which was developed to process and use spatially-

dependent albedo data calculated by the three-dimensional TORT discrete ordinates code, demonstrated close agreements with the benchmark results to within the statistical uncertainties.

An important conclusion from this research is that the spatially-dependent albedo data associated with the generation of pseudo-particles can be used in any class of problems. It was demonstrated that all significant components of the radiation field can be correctly represented by a MORSE/STORM calculation when using the spatially-dependent albedos.

The example problem was carefully designed to make it a valid test for certifying the new albedo procedures. The Monte Carlo estimation of the effect-of-interest is strongly influenced by several components of the radiation field. The transmission of particles through the material medium is on balance with the streaming through the duct. The small size of the problem allows a large fraction of particles to escape from the system after few collisions. The fact that the problem does not present a single dominant component of the radiation field ensures that traditional albedo Monte Carlo would not yield satisfactory results. The procedures implemented in the MORSE/STORM code, for both forward and adjoint modes, were demonstrated to efficiently characterize the radiation field. The results of both forward and adjoint calculations were found to agree with the benchmark results.

Important improvements have been observed in the figure-of-merits for both modes when compared with the figure-of-merits of full-transport Monte Carlo calculations.

The biasing procedures, implemented in the MORSE/STORM code, have satisfactorily controlled the number of pseudo-particles created in different parts of the geometry. The optimal use of these procedures may greatly improve the Monte Carlo figure-of-merit for calculations in which the importance function is strongly dependent on the phase space coordinates.

The use of standard biasing procedures applied during the tracking of particles, either albedo- or pseudo-particles, was not included in this research. Also the implementation of biasing procedures with respect to the spatially-dependent albedo data is an almost straightforward task which may lead to important improvements in the figure-of-merit. For example, biasing the displacement of albedo-particles can reduce the number of pseudo-particles created due to displacement into regions of low importance. A concept similar to the "non-leakage" biasing procedure can be used to keep the displaced albedo particles inside a duct when solving some classes of problems in which the only important component is the streaming component.

The generation of spatially-dependent albedo data was performed using a standard production version of the TORT code - many improvements should be possible. The implementa-

tion of the necessary modifications in the "sweeping of the spatial mesh" procedure to take advantage of the spatial symmetry of the albedo generation calculation is one of the possible improvements which deserves attention. Another point which would reduce the computer time required to generate three-dimensional albedo data would be a study of the sensitivity of the accuracy of the generated albedo information to the modeling of the spatial mesh. Also, the angular and spatial distributions of the albedo data can benefit from an optimization of the number and/or the size of the spatial mesh.

# BIBLIOGRAPHY

## BIBLIOGRAPHY


1. M. B. Emmett, "The MORSE Monte Carlo Code Radiation Transport Code System," ORNL 4972/R2, Oak Ridge National Laboratory (1984).

2. J. F. Briesmeister et al., "MCNP - A General Monte Carlo Code for Neutron and Photon Transport," LA-7396-M,Rev.2, Los Alamos National Laboratory (1986).

3. V. R. Cain and M. B. Emmett, "BREESE-II: Auxiliary Routines for Implementing the Albedo Option in the MORSE Monte Carlo Code," ORNL/TM-6807, Oak Ridge National Laboratory (1979).

4. W. A. Rhoades and R. L. Childs, "The TORT Three-Dimensional Discrete Ordinates Neutron/Photon Transport Code," ORNL-6268, Oak Ridge National Laboratory (1987).

5. J. J. Duderstadt, and L. J. Hamilton, Nuclear Reactor Analysis, John Wiley & Sons, Inc., New York (1976).

6. S. Glasstone and M. C. Edlund, The Elements of Nuclear Reactor Theory, Van Nostrand, New York, 1952.

7. S. Chandrasekhar, Radiative Transfer, Clarendon Press, Oxford, 1950.

8. W. E. Selph, " Neutron and Gamma-Ray Albedos," ORNL-RSIC-21, Oak Ridge National Laboratory (1968).

9. W. A. Rhoades, and R. L. Childs, " DORT Two-Dimensional Discrete Ordinates Transport Code," ORNL/TM-5851, Oak Ridge National Laboratory (1982).

10. R. E. Maerker and F. J. Muckenthaler, " Calculation and Measurement of the Fast-Neutron Differential Dose Albedo for Concrete," Nucl. Sci. Eng. 22, 455 (1965).

11. F. J. Allen, A. Futterer, and W. Wright, Neutron Reflection and Flux Versus Depth for Nevada Test Site Soil, BRL-1190 Ballistic Research Laboratories Report (1963).

12. W. A. Coleman, A Determination of Intermediate-Energy Neutron Albedo Data for Concrete Using Monte Carlo, Thesis, University of Tennessee (1965).

13. D. C. Irving, R. M. Freestone, Jr., and F. B. K. Kam, "05R, A General-Purpose Monte Carlo Neutron Transport Code," ORNL-3633 (1965)

14. F. R. Mynatt et al., " The DOT III Two-Dimensional Discrete Ordinates Transport Code," ORNL/TM-4280, Oak Ridge National Laboratory (1973).

15. M. B. Emmett and W. A. Rhoades, "CARP, A Computer Code and Albedo Data Library for Use by BREESE, the MORSE Albedo Package," ORNL/TM-6503, Oak Ridge National Laboratory (1978).

16. R. Howell, V. Cain, P. N. Stevens, "Monte Carlo Calculation of the TSF Two-Legged Concrete Duct Experiment," 1-138-02-034-00, prepared for Engineering Physics Division, Oak Ridge National Laboratory (1978).

17. R. Howell, V. Cain, P. N. Stevens, "CRBR Coolant Pipe Chaseway," 1-138-02-034-00, prepared for Engineering Physics Division, Oak Ridge National Laboratory (1978).

18. M. Kawai et al., " Application of the Albedo Monte Carlo Method to FBR Neutron Streaming Analysis," Presented at the Sixth Int. Conf. on Radiation Shielding, Paper 6a-2, Tokyo (1983).

19. M. Kawai et al., " Development of Shielding Calculation Method Using the Albedo SN Method," Report on the Results of the Research Sponsored by PNC, SJ 201 82-07 (1982)

20. N. Ohtani, M. Kawai, Y. Hayashida, and M. Yamauchi, "Three-Dimensional Albedo Monte Carlo Code System for Calculating Radiation Flux Distribution," JAPFNR-780, Power Reactor and Nuclear Fuel Development Corporation, Tokyo, Japan (1983).

21. M. Kawai, M. Yamauchi, Y. Hayashida, and J. Aratomei, "Radiation Flux Distribution Calculation Code System of the Three- Dimensional Albedo Monte Carlo Method," JAPFNR-779, Japan Atomic Energy Enterprises, Inc., Kawasaki, Japan (1983).

# APPENDIXES

## APPENDIX A

### MODIFIED SUBROUTINES OF THE MORSE CODE

The albedo-modified random-walk as implemented in the standard MORSE/BREESE package is performed by replacing some standard Monte Carlo procedures of the MORSE code with procedures which perform the albedo reflection and estimation processes. Subroutines such as COLISN and RELCOL are replaced by ALBDO and ALBDOE respectively. The standard Monte Carlo random walk is interrupted if an albedo surface is reached and the BREESE subroutines are called.

The modifications implemented in the MORSE/BREESE package to allow the generation of pseudo-particles and the displacement of the point-of-emergence at an albedo-event require the modification of some of the standard MORSE routines. The MORSE/STORM code uses the same modified versions of the subroutines of the MORSE code as was used by the modified version of the MORSE/BREESE code package.

The subroutines of the standard MORSE code which were modified to allow the generation of pseudo-particles at the albedo event are MORSE, SETNT, GETNT, STORNT and XSEC. The input parameters for the pseudo-particles generation proce-

dure are read by the subroutine PSEDIN which is called by the XSEC subroutine when the albedo option is selected in the standard input file. The input instructions for the PSEDIN subroutine are presented in the listing of the fortran source of the PSEDIN subroutine in this appendix.

The subroutine ALBDO was modified to perform the displacement of the point-of-emergence at an albedo event and the subroutine GOMES, which is a version of the subroutine GOMST, is specialized to perform the displacement of the point-of-emergence.

Presented below are the fortran card image of those subroutines used by the modified version of the MORSE/BREESE code package and by the MORSE/STORM code package (except ALBDO).

## MODIFIED VERSION OF THE MORSE SUBROUTINE

```
      subroutine morse(nlft)
      real random
      integer blznt,blzon,iblz,iiblzn
c
      common /nxt/ mxalb,locpsd,lprpsd,lpepsd,ialb,dokill,un0,vn0,wn0,
     * x0,y0,z0,ncos,mregbs,iblznt,ximp,yimp,zimp,kmizxt,ndum,iiblzn,
     * locneu,locnne,locggm,locngm,ialbwt,ialbcs,ialbrg,ialbeg,jyl
     * ,nduct,ndct(50),uaxis(50),vaxis(50),waxis(50),irrprl,im,jm,nxy
c
      common /pdet/ nd,nne,ne,nt,na,nresp,nex,nexnd,nend,ndnr,ntnr,ntne,
     1 nane,ntndnr,ntnend,nanend,locrsp,locxd,locib,locco,loct,locud,
     2 locsd,locqe,locqt,locqte,locqae,lmax,efirst,egtop
      common /apollo/ agstrt,ddf,deadwt(5),eta,etath,etausd,uinp,vinp,
     1 winp,wtstrt,xstrt,ystrt,zstrt,tcut,xtra(10),
     2   i0,il,media,iadjm,isbias,isour,iters,itime,itstr,locwts,locfwl,
     3 locepr,locnsc,locfsn,maxgp,maxtim,medalb,mgpreg,mxreg,nalb,
```

```
     4 ndead(5),newnm,ngeom,ngpqt1,ngpqt2,ngpqt3,ngpqtg,ngpqtn,nits,
     5 nkcalc,nkill,nlast,nmem,nmgp,nmost,nmtg,noleak,normf,npast,
     6 npscl(13),nquit,nsigl,nsour,nsplt,nstrt,nxtra(10)
      common /albcon/ ntheti, inrgys, nmgpa,  nino,   ninonp, igamg,
     1lsumenad,igalbold, locice, locenr, locvel, loccor, locthi, lntho,
     2  idnt, lsumen, lsntho, loctho, locnp,  lsnpki, lsnpi,  locphi,
c     3        locmxy, loctot, locang, locgam, locume, locump, locumt,
c     4                ialbdo, maxalb, lenalb, medal(10), icomb
     3 locnij, locmxy, loctot, loccen, locpxy, locpet, locpaz, locadj,
     4 lsumadj,maxija, ialbdo, maxalb, lenalb, medal(10),icomb,lsumenr
      common /nutron/ name,namex,ig,igo,nmed,medold,nreg,u,v,w,uold,vold
     1 ,wold,x,y,z,xold,yold,zold,wate,oldwt,wtbc,blznt,blzon,age,oldage
      common /fisbnk/ mfistp,nfisbn,nfish,ftotl,fwate,watef
      common/locsig/istart,isccog,inabog,igabog,ifporg,ifngp,ifspog,
     1       idsgog,iprbng,iprbgg,iscang,iscagg,isporg,isport,inpbuf,
     2       isigog,infpog,iabsog,itotsg,ngp,nds,ngg,ndsg,ingp,inds,
     3   nxxmed,nelem,nmix,ncoef,nsct,nxxts,ntg,ndsngp,ndsngg,iadj,
     4 nme,loc,ings,insg,ixxl,ixx0, kkk,ixtape,idel,iteml,itemg,irsg,
     5       irdsg,istr,iprin,ifmu,imom,idtf,istat,ipun
     6       ,nus,ngn,iht,inus,inusn,ingn,ingnp,innn,iggg
      common /normal/ un,vn,wn
      common /kitsx/ kits
      common /kmkk2/kmk2
      common /check/igcheck
      dimension nts(1)
         dimension wtbyrg(200)
      common wts(1)
      equivalence(wts(1),nts(1))
c     call msgr(i0)
c  begin new problem
 10   nlast=nlft
       call second(timew)
       timew=timew/60.0
      ialb=0
      ialbs=0
      call input
c       reads cards a thru d - calls sorin for cards e if isour .le. zer
c       reads cards f thru o - calls jomin, xsec and scorin
      if (ndum.eq.0) ndum=100
      ntstw=nkill+nsplt
      ngpreg = nmtg*mxreg
      ncomb = ngpqtn*ngpqtg
      ia = locfwl + 1
      iu = locfwl + mxreg
      do 15 i=ia,iu
      ib = i + mxreg
 15   wts(ib) = wts(i)
      iaw = locwts + 1
      iuw = locwts + 3*mgpreg
      do 20 i=iaw,iuw
      ib = i + 12*mgpreg
 20   wts(ib) = wts(i)
```

```
      inits = nits
      iruns = nquit
      indx=0
       call second(timev)
       timev=timev/60.0
       timeinp=timev-timew
       write(i0,1010) timeinp
 1010 format ('unweighted cpu time required for input was ',1pe10.3)
c    begin new run
 25   kits=0
      if (mxalb.eq.0 .or. ialbwt.eq.0) kits=1
      nits = inits
      call bankr(-1)
      do 30 i=ia,iu
      ib = i + mxreg
 30   wts(i) = wts(ib)
      do 35 i=iaw,iuw
      ib = i + 12*mgpreg
 35   wts(i) = wts(ib)
      iters=nits
      itstr=0
c  begin new batch
 40   nmem=nstrt
      kits=kits+1
      if (kits.le.1) nmem=ndum
      if(nmem.le.0) goto 40
      if (itstr) 45,50,45
 45   nmem = nfish
c
c   if the particles stored in the bank are pseudo-particles; then ialb
c   is larger than zero.       ICG - 06/02/89
c
 50    if (ialb) 51,51,52
 52     nmemx=nstrt
        nmem=ialbs
        kmiz=locpsd+1
        xialbs=ialbs
        ialbs=0
        kalo=1
        xkillp=0.
        goto 60
c
51    continue
       wtprod = 0.
       xcrea = 0.
       xkilld = 0.
       kmiz = kmizxt
59    call bankr(-2)
c       calls stbtch
      kalo=0
      nbatch=nits-iters+1
       random=ranget(0)
```

```
c      call rndout(random)
       if (kits.le.1) goto 360
       write (i0,1015) nbatch,random
 1015 format (15h0***start batch ,i4,25x,7hrandom=,o22/12h0source data)
360    call msour
       if (iters) 195,195,55
 55    call outpt(1)
c  begin new history
 60    if (ialb) 61,61,62
 62    call getnt(nmem,2)
       call getnt(nmem,3)
       do 99 iykl=1,maxalb
       if(nmed.eq.medal(iykl))write(6,852)medal(iykl)
852    format(5x,' WARNING *** a pseudo-particle was stored inside an alb
      *edo medium ***',/,5x,'medal(i) = ',i10)
       if(nmed.eq.medal(iykl))nmem=nmem-1
       if(nmem.le.0) goto 170
       if(nmed.eq.medal(iykl)) goto 62
99     continue
c
c                          ICG -06/17/89
c
       kmiz=locpsd+1
c
       goto 63
 61    call getnt(nmem,1)
       igcheck = ig
       kmiz=kmizxt
c
c     now it makes on fly estimation of the average weight by region
c
       wtbias=0.
       iblzx=iblznt
       if (ialbwt.eq.0) goto 63
       if (ig.le.nmgp .and. nts(locnne+iblzx).ne.0 )
      * wtbias=wts(locneu+iblzx)/float(nts(locnne+iblzx))
       if (ig.gt.nmgp .and. nts(locngm+iblzx).ne.0 )
      * wtbias=wts(locggm+iblzx)/float(nts(locngm+iblzx))
 63    nmem = nmem - 1
       nalb = 0
       ngpqt = ngpqt1
 65    if (wate) 70,165,70
 70    if (ig-ngpqt) 90,90,75
 75    if (ig-ngpqt2) 160,160,80
 80    if (ig-ngpqt3) 85,85,160
 85    ngpqt=ngpqt3
 90    igo=ig
       uold=u
       vold=v
       wold=w
       oldwt=wate
       xold=x
```

```
      yold=y
      zold=z
      blzon=blznt
      medold=nmed
      oldage=age
      mk=1
      if (ntstw.gt.0) call testw
      if (wate) 100,95,100
 95   ndead(1)=ndead(1)+1
      deadwt(1)=deadwt(1)+oldwt
c  r r kill
      go to 165
 100  dokill=0.
      call nxtcol(kits)
      if (dokill.eq.0.) goto 197
c
c  the pseudo-particle has tried to cross the plane on which it was
c  created, thus it is killed. ICG 06/02/89
c
 734  xkill=xkill+1
      wtkill=wtkill+wate
c   it stores the weight of the particles killed during the batch.
      goto 165
 197  if (nalb.le.0.or.ialb.eq.0) goto 198
      if (nreg.le.mregbs) goto 198
      write(i0,872)nreg,mregbs
872   format(/,5x,'the region number  nreg is greater than the maximum n
     *umber of regions defined for pseudo-particles',/,5x,'an albedo eve
     *nt is going to occur with a pseudo-particle and the medium number
     *of the region is not defined',/,5x,'blznt =',i5,3x,'mregbs =',i5)
      call error
198    continue
      if (nreg-mxreg) 102,102,101
 101  write (i0,1016) nreg,mxreg
 1016 format  (6h0nreg=,i5,8h, mxreg=,i5,81h, mxreg on card i must be ge
     1 to the number of regions described in geometry input)
      call exit
 102  if (tcut.le.0..or.age.lt.tcut) go to 110
      ndead(4)=ndead(4) +1
      deadwt(4) =deadwt(4)+oldwt
c  age kill
      npscl(10) = npscl(10) + 1
c     call bankr(10)  for time-kill analysis
      go to 165
 110  if (wate) 120,115,120
 115  ndead(2)=ndead(2)+1
      deadwt(2)=deadwt(2)+wtbc
c  escape
      go to 165
c     albedo handling
c
c if the particle being tracked is a pseudo-particle, albedo reflection
```

```
c is not allowed.                    ICG - 06/02/89
c
 120  continue
      if (nalb.le.0) go to 130
c      nalb > 1000   -  specular reflection media
      if (nalb.lt. 1000) go to 125
      call spculr(u,v,w,nreg)
c  spculr handles specular reflection boundaries
c   this restores the region number of previous region (prior to albedo)
      nreg = nxtra(10)
      go to 65
 125  isct = locnsc + 2*ngpreg + (nreg-1)*nmtg + ig
      nts(isct) = nts(isct) + 1
      isct = isct + ngpreg
      wts(isct) = wts(isct) + wate
      if(ialb.gt.0) goto 851
       watex = wate
       xx = x
       yx = y
       zx = z
       blxx = iiblzn
       nmxx = nts(locpsd+iblznt)
       ux = u
       vx = v
       wx = w
       igx = ig
       kmk2 = 0
       iblzx = iblznt
c
c     albdo handles real albedo boundaries      *      *     *    *    *
c
851    mk=1
       call albdo(ig,u,v,w,wate,nmed,nreg,mk,x,y,z)
c
       if(mk.ne.-1.and.mk.ne.3) goto 859
       if(ialb.eq.0) xkilld = xkilld + 1
       if(ialb.ne.0) xkillp = xkillp + 1
c
       goto 165
c     mk=-1 indicate escape from the system
859    continue
       if(ialb.gt.0) goto 241
c     mk=3 indicate specular reflection (not allowed in the actual version)
c
c     the following commands are introduced here to save the information
c     related to the particle which is going to suffer an albedo event.
c     ICG 06/01/89
c
 229   watel = watex
       if (ialb) 121,121,240
121    if ( kits.le.1 .or. mxalb.le.0 ) goto 240
       if (ialbs.lt.mxalb) goto 239
```

```
      iover=iover+1
      goto 228
239   continue
c      if(kmk2.gt.0) goto 4
      if (ialbwt.eq.0) goto 330
      if ( watel.ge.wtbias ) goto 330
      r0=fltrnf(0)
      if (wtbias.le.0) goto 330
      wtx=watel/wtbias
      if (r0.gt.wtx) goto 228
      watel=wtbias
c
c     average weight biasing for generating pseudo-particles
c
330   if (ialbrg.eq.0) goto 340
      r1=fltrnf(0)
      if (wts(lprpsd+iblznt).le.r1) goto 228
      watel=watel/wts(lprpsd+iblznt)
c
c     importance by region biasing for generating pseudo-particles
c
340   if (ialbcs.eq.0) goto 350
      dista=sqrt((ximp-x)*(ximp-x)+(yimp-y)*(yimp-y)+(zimp-z)*(zimp-z))
      albim=(u*(ximp-x)+v*(yimp-y)+w*(zimp-z))/dista*(-1.)
      angulo=acos(albim)/2.
      albim=cos(angulo)
      albim=albim**ncos
      if (albim.le.0.2) albim=0.1
      r2=fltrnf(0.)
      if (albim.le.r2) goto 228
      watel=watel/albim
c
c     angular biasing for generating pseudo-particles
c
350   if (ialbeg.eq.0) goto 230
      r3=fltrnf(0.)
      if (wts(lpepsd+igx).le.r3) goto 228
      watel = watel/wts(lpepsd+igx)
c
c     energy biasing for generating pseudo-particles
c
c
 230  continue
      do 31 i=1,nd
      if(kmk2.gt.0) goto 4
      ia = locxd+i
      xe = wts(ia)
      ye = wts(ia+nd)
      ze = wts(ia+2*nd)
c
c   check if the collision point and the detector are on the same side
c   of the plane on which the pseudo-particle was created.
```

```
c                                                          ICG - 06/02/89
      balbl = un*(xe-x)+vn*(ye-y)+wn*(ze-z)
      if (balbl.le.0.) goto 4
31    continue
      if(mk.eq.2)goto 228
      goto 241
 4    ialbs = ialbs+1
      wtxl = wate
      igxl = ig
      wate = watel
      ig = igx
c
      wtprod = wtprod + wate
c
      nnmx = nmed
      nmed = nmxx
      iblz = blznt
      blznt = blxx
      ipp = blznt/32768
      ippl = blznt-ipp*32768
      xa = x
      ya = y
      za = z
      x = xx
      y = yx
      z = zx
      ua = u
      va = v
      wa = w
      u = ux
      v = vx
      w = wx
      call stornt(ialbs,2)
      call stornt(ialbs,3)
      wate = wtxl
      blznt = iblz
      nmed = nnmx
      x = xa
      y = ya
      z = za
      u = ua
      v = va
      w = wa
      ig = igxl
228   continue
      if(mk.ne.2)goto 240
c
c     the following commands are introduced to save the reflected pseudo-
c     particles, which were created during the displacement ( note that
c     if only the original pseudo-particle is created when the particle is
c     displaced to points inside an albedo medium, then these particles
c     are not able to make contributions to detectors at the other side
```

```
c       of the surface at which they were created).
c
        kmk2=kmk2+1
        un=-un
        vn=-vn
        wn=-wn
        watex = wate
        ux = u
        vx = v
        wx = w
        xx = x
        yx = y
        zx = z
        igx = ig
        nmxx = nts(locpsd +iblznt)
        iblzx = iblznt
        blxx = iiblzn
        if(kmk2-1) 229,229,240
c
c      now, it saves the weight of the particles suffering albedo events
c      by region
c
240     continue
        if(mk.ne.2)goto 249
        xcrea = xcrea + 1
        goto 241
249     if (mxalb.le.0 .or.ialb.ne.0 ) goto 241
        if ( ialbwt.eq.0 ) goto 393
        if ( igx1 - nmgp ) 391,391,392
391     nts(locnne+iblzx) = nts(locnne+iblzx) + 1
        wts(locneu+iblzx) = wts(locneu+iblzx) + watex
        goto 393
392     nts(locngm+iblzx) = nts(locngm+iblzx) + 1
        wts(locggm+iblzx) = wts(locggm+iblzx) + watex
393     continue
        if (kits.le.1) goto 394
c
c      now, it stores the normal of the surface where the pseudo-particle
c      was created. ICG - 06/02/89
c
c
241    npscl(6) = npscl(6) + 1
       call bankr(6)
       if(mk.eq.2.and.ialb.gt.0) wate=0.0
       if(ialb.gt.0) goto 394
       if(mk.ne.2) goto 394
       xkilld=xkilld+1
       goto 165
c    this restores the region number of previous region (prior to albedo)
394    nreg = nxtra(10)
       go to 65
 130   call gtmed(nmed,imed)
```

```
      if (mfistp) 140,140,135
 135  if (wts(locfsn+(imed-1)*nmtg+ig)) 140,140,136
 136  call fprob
 140  if (ncomb) 155,155,145
 145  if (wts(locfsn+(2*media+imed-1)*nmtg+ig)) 155,155,150
 150  call gprob
c
 155  if(kits.le.1) goto 156
      isct = locnsc + (nreg-1)*nmtg + ig
      nts(isct) = nts(isct) + 1
      isct = isct + ngpreg
      wts(isct) = wts(isct) + wate
      isct = locnsc + 8*ngpreg + imed
      nts(isct) = nts(isct) + 1
 156  call colisn( ig,u,v,w,wate,nmed,nreg)
c
      if(kits.le.1) goto 65
      npscl(5) = npscl(5) + 1
      call bankr(5)
c       calls relcol
      if (dokill.ne.0) goto 734
      go to 65
 160  ndead(3)=ndead(3)+1
      deadwt(3)=deadwt(3)+wate
      npscl(9) = npscl(9) + 1
c     call bankr(9)  for e-cut analysis
c  energy cutoff
 165  if (nmem) 170,170,60
c  end of history
170       if (ialbs) 171,171,191
171    if (ialb.ne.0) write(6,1919)wtprod,ttkill,xialbs,xkill,xcrea,
     *xkilld,xkillp
1919    format(/,5x,'total weight of the pseudo-particles created =',
     *1pe15.7,/,5x, 'total weight of the pseudo-particles killed  =',
     *1pe15.7//,5x, 'total number of normal pseudos created       =',
     *1pe15.7,/,5x, 'total # pseudos killed due to killing planes =',
     *1pe15.7,/,5x, 'total # pseudos create due to displacement   =',
     *1pe15.7,/,5x, 'total # alb-part killed during displacement  =',
     *1pe15.7,/,5x, 'total # pseudos killed during displacement   =',
     *1pe15.7)
c      write(4,873) nmem
873   format('  after writting pseudos etc.. - nmem = ',i5)
      if(ialbwt.eq.0) goto 843
       do 842 imreg=1,mregbs
       wtbyrg(imreg)=0.
       if(nts(locnne+imreg).ne.0)
     * wtbyrg(imreg) = wts(locneu+imreg)/nts(locnne+imreg)
842    continue
       write(6,841) (imreg,wtbyrg(imreg),imreg=1,mregbs)
841    format(5x,'average weight by region used for creating pseudos',/,
     *' reg# av. weight',' reg# av. weight',' reg# av. weight',
     *' reg# av. weight',/, (4(i4,e12.5)))
```

```
843         call bankr(-3)
c        calls nbatch
        call outpt(2)
         call second(timeval)
         timeval=timeval/60.0
         tmaxtim=float(maxtim)
         if (timeval-timev-tmaxtim) 180,180,175
 175   nits = nits - iters + 1
       iters = 0
       nquit = nquit - iruns
       iruns = -nquit
       write (i0,1020) iruns,inits,nits
 1020  format(1h0/39h0run terminated by execution time limit
      1                    /i8,8h runs of,i3,8h batches,16h   and 1  run
      2of,i3,19h batches completed./)
 180   if (kits.le.1) goto 185
       iters = iters - 1
       if (iters) 195,195,185
 185   if (nsour) 191,191,190
 190   itstr=1
c
c  end of batch
c
c       the following lines are introduced to print out a warning in case
c       of it does not have enough room to store all pseudo-particles
c       created at albedo events. ICG - 06/01/89
c
191     ishould=mxalb+iover
        if (iover.gt.0) write(6,1200)iover,mxalb,ishould
1200    format(5x,i10,3x,59hparticles were not created at albedo events b
     *ecause mxalb =,i10,3x,25h,and should be at least =,i10)
        iover=0
        ialb=0
        if (ialbs.eq.0) goto 40
        ialb=1
        xkill=0.
        wtkill=0.
        write(6,1499)
1499    format(5x,'it is begining to follow pseudo-particles')
        goto 40
c  end of batch
 195   call bankr(-4)
c        calls nrun
       nquit=nquit-1
        call second(timv)
        timv=timv/60.0
        write(i0,1030) nits,timv
 1030  format ('cpu time required for the preceding',i4,' batches was ',
      1 1pe10.3)
c  end of nits batches
       if (nquit) 200,200,25
 200   call outpt(3)
```

```
c  end of run
       call second(ftime)
       ftime=ftime/60.0
      write(i0,1040) ftime
 1040 format ('0total cpu time for this problem was ',1pe10.3,
     1'minutes.')
       return
       end
```

## MODIFIED VERSION OF THE SETNT SUBROUTINE

```
      subroutine setnt(nlast,nmost,kopt)
      common/nnnnno/nno,nn1,nn2
c
c   This subroutine was modified to accomodate the pseudo-particles
c   generated at albedo-events.
c                                ICG - 06/07/89
      goto(100,200,300),kopt
100   nno = nlast
      nlast=nlast+14*nmost
      return
200   nn1 = nlast
      nlast=nlast+14*nmost
      return
300   nn2 = nlast
      nlast = nlast+4*nmost
      return
      end
```

## MODIFIED VERSION OF THE STORNT SUBROUTINE

```
      subroutine stornt(n,kopt)
      common/kmkk2/kmk2
      common/nnnnno/nno,nn1,nn2
      common/nutron/name,namex,ig,igo,nmed,medold,nreg,u,v,w,uold,vold,
     1  wold,x,y,z,xold,yold,zold,wate,oldwt,wtbc,blznt,blzon,age,oldage
```

```
      common /normal/un,vn,wn
c      level 3,  bc
      common   bc(1)
      dimension i2b(1)
      equivalence(bc(1),i2b(1))
c
c   The following modifications were introduced to accomodate the
c   pseudo-particles into the common blank.
c                                                  ICG - 06/07/89
      if(kopt.eq.1)nnx = nno
      if(kopt.eq.2)nnx = nn1
      if(kopt.eq.3)goto 100
      if(kopt.eq.4)goto 200
      k=nnx+14*n-13
      i2b(k)=ig
      k=k+1
      bc(k) = u
         k = k + 1
      bc(k) = v
         k = k + 1
      bc(k) = w
         k = k + 1
      bc(k) = x
         k = k + 1
      bc(k) = y
         k = k + 1
      bc(k) = z
         k = k + 1
      bc(k) = wate
         k = k + 1
      bc(k) = age
         k = k + 1
      bc(k) = blznt
      k4=k+1
      i2b(k4) = name
         k4 = k4 + 1
      i2b(k4) = namex
         k4 = k4 + 1
      i2b(k4) = nmed
         k4 = k4 + 1
      i2b(k4) = nreg
      return
100      k=nn2+4*n-3
      bc(k)=un
         k=k+1
      bc(k)=vn
         k=k+1
      bc(k)=wn
         k=k+1
      bc(k)=kmk2
      return
200      k=nn2+4*n-3
```

```
      bc(k)=u
         k=k+1
      bc(k)=v
         k=k+1
      bc(k)=w
         k=k+1
      bc(k)=0
      return
      end
```

## MODIFIED VERSION OF THE GETNT SUBROUTINE

```
      subroutine getnt(n,kopt)
c
      common/kmkk2/kmk2
      common /nxt/ mxalb,locpsd,lprpsd,lpepsd,ialb,dokill,un0,vn0,wn0,
     * x0,y0,z0,ncos,mregbs,iblznt,ximp,yimp,zimp,kmizxt,ndum,iiblzn,
     * locneu,locnne,locggm,locngm,ialbwt,ialbcs,ialbrg,ialbeg,jyl
     * ,nduct,ndct(50),uaxis(50),vaxis(50),waxis(50),irrprl,im,jm,nxy
c
      common/nnnnno/nno,nn1,nn2
      common /nutron/ name,namex,ig,igo,nmed,medold,nreg,u,v,w,uold,vold
     1 ,wold,x,y,z,xold,yold,zold,wate,oldwt,wtbc,blznt,blzon,age,oldage
      dimension i2b(1)
c      level 3,  bc
      common   bc(1)
      equivalence (bc(1),i2b(1))
      if(kopt.eq.1)nnx = nno
      if(kopt.eq.2)nnx = nn1
      if(kopt.eq.3)goto 100
      k=nnx+14*n-13
      ig=i2b(k)
      k=k+1
      u = bc(k)
         k = k + 1
      v = bc(k)
         k = k + 1
      w = bc(k)
         k = k + 1
      x = bc(k)
      if(kopt.eq.2) x0 = x
         k = k + 1
      y = bc(k)
      if(kopt.eq.2) y0 = y
```

```
         k = k + 1
      z = bc(k)
      if(kopt.eq.2) z0 = z
         k = k + 1
      wate = bc(k)
         k = k + 1
      age = bc(k)
         k = k + 1
      blznt = bc(k)
      k4=k+1
      name = i2b(k4)
         k4 = k4 + 1
      namex = i2b(k4)
         k4 = k4 + 1
      nmed = i2b(k4)
         k4 = k4 + 1
      nreg = i2b(k4)
      return
100     k=nn2+4*n-3
      un0=bc(k)
        k=k+1
      vn0=bc(k)
        k=k+1
      wn0=bc(k)
        k=k+1
      kmk2=bc(k)
      return
      end
```

## MODIFIED VERSION OF THE XSEC SUBROUTINE

```
      subroutine xsec(iadjm,locepr,medalb,media,nlast,nmgp,nmtg,nleft,
     1 io,in)
c
c     this routine is the primary interface between cross section module
c         and the rest of morse   *   *   *    *     *     *     *
c      it reads cross section information and sets up storage locations
c        a call to this routine completely sets up  the cross sections *
      common/locsig/istart,isccog,inabog,igabog,ifporg,ifngp,ifspog,
     1       idsgog,iprbng,iprbgg,iscang,iscagg,isporg,isport,inpbuf,
     2       isigog,infpog,iabsog,itotsg,ngp,nds,ngg,ndsg,ingp,inds,
     3       nmed,nelem,nmix,ncoef,nsct,nts,ntg,ndsngp,ndsngg,iadj,
     4       nme,loc,ings,insg,il,i0, kkk,ixtape,idel,iteml,itemg,irsg,
     5       irdsg,istr,iprin,ifmu,imom,idtf,istat,ipun
```

```
     6       ,nus,ngn,iht,inus,inusn,ingn,ingnp,innn,iggg
      common /gtscl/npt(16),nxsect(17),ndsgp,id(2),io6rt,igqpt
     1,negps,espd(100) ,nnic,nxpm,ninc
      common sigt(1)
      dimension nsig(1),title(20)
      data ikn/0/
      equivalence (sigt(1),nsig(1))
      loc = locepr
      il=in
      i0=io
      ngn=0
      nmxsec = 0
      if (medalb) 10,15,15
 10   medalb = - medalb
      write (i0,1010)
 1010 format (70h0this is an albedo only problem - normal xsect input wi
     111 be bypassed.)
      call albin(nlast,nleft,i0)
      return
 15   read (il,1020) title
 1020 format(20a4)
      write(i0,1030) title
 1030 format(1hl,20a4)
      iadj=iadjm
      read(il,1040)ngp,nds,ngg,ndsg,ingp,itbl,isgg,nmed,nelem,nmix,ncoef
     1,nsct,istat,iht
c                    5  10  15   20   25   30   35   40    45   50   55
c      are pair production cross sections present
      if(ncoef) 16,16,17
   16 ncoef = -ncoef
      write(i0,2229)
 2229 format(1h0, ' *** pair production and compton data are present')
c      pair production xsec are present
      ikn = 1
  17  continue
 1040 format(14i5)
      if(iht.eq.0) iht=3
      write(i0,1050)ngp,nds,ngg,ndsg,ingp,itbl,iht,isgg,nmed,nelem,nmix,
     1 ncoef,nsct,istat,iadj
 1050 format(31h0number of primary groups (ngp)       6x  ,i6/
     1       37h number of primary downscatters (nds)     ,i6/
     2       33h number of secondary groups (ngg)     4x  ,i6/
     3       40h number of secondary downscatters (ndsg)  ,i3/
     4       33h number of prim+sec groups (ingp)    7x   ,i3/
     5       20h table length (itbl)                 20x  ,i3/
     e       27h loc of total (sig t) (iht)    ,     10x,  i6/
     6       36h loc of within group (sig gg) (isgg)      ,i7/
     7       23h number of media (nmed)              14x  ,i6/
     8       33h number of input elements (nelem)     4x  ,i6/
     9       32h number of mixing entries (nmix)      5x  ,i6/
     a       31h number of coefficients (ncoef)       6x  ,i6/
     b       24h number of angles (nsct)             16x  ,i3/
```

```
     c       22h restore coeff (istat)              15x  ,i6/
     d       28h adjoint switch (from morse)         9x  ,i6/)
      read(i1,1040)irdsg,istr,ifmu,imom,iprin,ipun,idtf ,ixtape,jxtape
     1,io6rt,igqpt
      write(i0,1060)
 1060 format(21h0input/output options  )
      write(i0,1070)irdsg,istr,ifmu,imom,iprin,ipun,idtf,      ixtape
     1,jxtape,io6rt
 1070 format(5x,16h irdsg (as read) ,i10  /
     1    5x,16h istr (as store)  ,i10/5x,11h ifmu (mus) ,7x,i8/
     2    5x,15h imom (moments)  ,i11 /5x,20h iprin (angles,prob) ,i6 /
     3    5x,23h ipun (impossible coef) ,i3/5x,19h card format (idtf),i7
     4   /5x,20h input tape (ixtape) ,i6/ 5x,21h morsec tape (jxtape),i5
     5 /  5x,17h o6r tape (io6rt) ,i9 )
c     ngg =0 if not doing gamma transport
c     istat  gt 0 saves legendre coeff for statistical estimation
      if(ixtape)245,20,20
 20   nme=1
      inds=itbl-isgg+1
      nus=isgg-iht-1
      if(ngp)25,25,30
 25   ngp=ngg
      nds=ndsg
      nme=ingp-ngg+1
      ngg=0
      ndsg=0
 30   continue
      ntg   =ngp+ngg
      nts   =nds+ndsg
      if (media-nmed+nmgp-ngp+nmtg-ntg) 35,40,35
 35   write(i0,1080)media,nmed,nmgp,ngp,nmtg,ntg
 1080 format(17h1media in input =,i3,16h, nmed in xsec =,i3,/7h nmgp = ,
     110x,i3,7h, ngp =,9x,i3,/7h nmtg =,10x,i3,7h, ntg =,9x,i3)
  37  call xschlp(0,4hxsec)
      call exit
 40   if(iadj)55,55,45
 45   if(ngg)55,55,50
 50   nte=ngg
      ngg=ngp
      ngp=nte
      nte=nds
      nds=ndsg
      ndsg=nte
 55   irsg =nlast
      ings=irsg+3*nmix
      insg=ings+ngp
      idel=insg+ngg
      nec=0
      nts=nts+nus
      if(ixtape)65,65,60
 60   nec= nelem*ncoef
      idm=idel+nec
```

```
      rewind ixtape
      ide=idel+1
      read(i1,1090) (nsig(m),m=ide ,idm)
1090  format(14i5)
      write(i0,1100) (nsig(m),m=ide ,idm)
1100  format(33h1     elements from library tape  , /12h identifiers ,
     1 14i5 , 4(/12x,14i5))
65    continue
      innn=idel+nec
      iggg=innn+ngp
      istart=iggg+ngg
      ndsngg=0
      ndsngp = 0
      nl=nus+nds
      do 90 i=1,ngp
      nsig(ings+i)=ndsngp
      ndsk=ngp-i+1+nus
      if(ndsk-nl)75,75,70
70    ndsk=nl
      go to 85
75    if(ndsk-nus)80,80,85
80    ndsk=nus+1
85    nsig(i+innn)=ndsk
      ndsngp = ndsngp + ndsk
90    continue
      ndsngp=0
      nl=nus+ndsg
      if ( ngg ) 125, 125, 95
95    do 120 i=1,ngg
      nsig(insg+i)=ndsngp
      ndsk=ngg-i+1 +nus
      if(ndsk-nl)105,105,100
100   ndsk=nl
      go to 115
105   if(ndsk-nus)110,110,115
110   ndsk=nus+1
115   nsig(i+iggg)=ndsk
120   ndsngp=ndsngp+ndsk
125   if(ixtape) 160,160,130
130   ispor=nleft
      call gtndsk(ispor)
      if(iadj)135,135,140
135   nl=1
      n2=ngp
      go to 150
140   if(ngg)135,135,145
145   nl=ngp+1
      n2=ntg
150   continue
      n3=nl+1
      nsig(ings+nl)=0
      do 155 i=n3,n2
```

```
 155  nsig(ings+i)=nsig(ings+i-1)+nsig(innn+i-1)
 160  continue
      if(ngg)170,170,165
 165  ndsngg=nsig(insg+ngg)+ nsig(iggg+ngg)
 170  ndsngp=nsig(ings+ngp)+ nsig(innn+ngp)
      isccog=istart+ntg
      inabog=isccog+ntg
      igabog=inabog+ntg
      ifporg=igabog+ngp
      inus=ifporg+ntg
      ingn =inus+ntg
      ingnp=ingn
      if(ikn-1) 172,171,172
 171  ingnp = ingn+2*ntg
c     ingn is starting location of pair prod and compton
 172  ifngp = ingnp
c * * ifngp=ingnp+ngg*ngp
      ifspog=ifngp +ngp*ngg
      idsgog=ifspog+ndsngp
      iprbng=idsgog+ndsngg
      iprbgg=iprbng+ndsngp*nsct
      iscang=iprbgg+ndsngg*nsct
      iscagg=iscang+ndsngp*nsct
      isporg=iscagg+ndsngg*nsct
      isport=isporg*nmed   -(nmed-1)*istart
      ispm=isporg-istart
      ngp3=ingp  +iht+nus
      if(ispm*nmed-ingp*ngp3)175,180,180
 175  isport=ingp*ngp3+istart
c     buffer region for input cross sections starts at istart
 180  inpbuf=istart
      ibuf=nelem*(ntg*(nts+2)+(ndsngp+ndsngg)*(ncoef-1))
      if(ikn-1) 181,182,181
  182 ibuf= ibuf+2*nelem*ntg
  181 continue
      ispor=      nleft-ibuf
      ntemp=isport-ispor
      if(ntemp)190,190,185
 185  write(i0,1110) ntemp
 1110 format (56h0temporary cross-section storage exceeds blank common b
     1y,i7)
      call xschlp(0,4hxsec)
      call exit
 190  isport=ispor
      isigog=isport+ntg*nts*nelem
      infpog=isigog+ntg*nelem
      iabsog= infpog+ntg*nelem
      if ( ikn-1) 192,191,192
 191  ingnp = iabsog
      iabsog = ingnp+2*ntg*nelem
 192  continue
c  *   allow temporary storage for pair production and compton xsec
```

```
      itemg =iabsog+ndsngp*(ncoef-1)
      itotsg=itemg +ndsngg*(ncoef-1)
      iteml=itotsg-iabsog
      nmxsec=iabsog+iteml*nelem
      isporg = isporg-istart
      if(istat)195,195,200
195   nlast=isporg*nmed+istart
      ibuf= isport-nlast
      go to 205
200   ispm=nmed*iteml
      nlast=isporg*nmed+istart+ispm
      ibuf=isport-(isporg*nmed+istart)
205   write(i0,1120)irsg,nlast,isport,nmxsec,ibuf
1120  format(1h0,19hstorage allocations  /
     1       1h ,26hcross sections start at   ,i7, /
     2          27h last location used (perm) ,i7, /
     3          27h temp locations used       ,i7,4h to ,i7,/
     4          27h excess storage (temp)     ,i7  )
      if(istat)220,220,210
210   ibuf=isporg*nmed+istart
      write(i0,1130)ibuf
1130  format(27h start of legendre coeff   ,i7)
      ntemp=nlast- nleft
      if(ntemp)220,220,215
215   write(i0,1140)ntemp
1140  format (56h0permanent cross-section storage exceeds blank common b
     ly,i7)
      call xschlp(0,4hxsel)
      call exit
220   call jnput
      if(jxtape)230,230,225
225   call xstape(2,jxtape,nlast,title,nleft)
230   if (medalb) 240,235,240
235   medalb = 7777
      go to 250
240   call albin(nlast,nleft,i0)
      call psedin(nlast,nleft,i0,ngp,ngg,in)
      go to 250
245   ixtape=iabs(ixtape)
      call xstape(1,ixtape,nlast,title,nleft)
      if(locepr .ne.loc) loc=locepr
      if(iadj.eq.1.and.ngg.ne.0) go to 270
      if(media-nmed+nmgp-ngp+nmtg-ntg) 35,230,35
 270  if(media-nmed+nmgp-ngg+nmtg-ntg)36,230,36
 36   write(i0,1150)media,nmed,nmgp,ngg,nmtg,ntg
1150  format(67hlin a combined adjoint problem using a processed cross s
     lection tape/17h media in input = ,    i3,16h, nmed in xsec = ,i3,/
     2 7h nmgp = ,10x,i3,7h, ngg = ,9x,i3/7h nmtg = ,10x,i3,7h, ntg = ,
     3 9x,i3)
      go to 37
250   if(io6rt )260,260,255
255   call gtsct(nlast,nleft)
```

```
 260  return
      end
```

**THE PSEUDO-PARTICLES INPUT INSTRUCTIONS SUBROUTINE**

```
      subroutine psedin(nlastx,nleftx,i0x,ngpqtn,ngpqtg,in)
c
      common /nxt/ mxalb,locpsd,lprpsd,lpepsd,ialb,dokill,un0,vn0,wn0,
     * x0,y0,z0,ncos,mregbs,iblznt,ximp,yimp,zimp,kmizxt,ndum,iiblzn,
     * locneu,locnne,locggm,locngm,ialbwt,ialbcs,ialbrg,ialbeg,jyl
     * ,nduct,ndct(50),uaxis(50),vaxis(50),waxis(50),irrprl,im,jm,nxy
c
      common bc(1)
      dimension nc(1)
      equivalence (bc(1),nc(1))
c
c     mxalb - maximum number of pseudo-particles that can be created
c             (if eq zero all modifications related to pseudo-particles
c              are skipped).
c     ncos - exponent for the cosine biasing the angular distribution
c            of the generation of pseudo-particles.
c     ialbwt - flag indicating that the generation of pseudo-particles
c              is to be biased with the average weight by region.
c     ialbcs - flag indicating that the generation of pseudo-particles
c              is to be biased in function of the direction of the particle.
c     ialbrg - flag indicating that the generation of pseudo-particles
c              is to be biased with the probabilities given by region.
c     ialbeg - flag indicating that the generation of pseudo-particles
c              is to be biased with the probabilities given by energy.
c     ndum   - number of particles to be followed in the dummy batch
c              (batch which initialize the on fly estimation of the average
c              weight of the albedo particles by region). It is set
c              equal 100 if equal zero.
c     mregbs - number of regions described in the geometry input for
c              which probabilities for generating pseudo-particles
c              will be assigned.
c
c     ximp,yimp,zimp - point toward which the vector from the position of the
c                      albedo event points to define the most important direction
c                      for the angular biasing.
c
c     albias(i),i=1,mregbs -          /probabilities for generating pseudo-part.
c     bc(i),i=lprpsd+1,lprpsd+mregbs \assigned for each region of the geometry.
c
```

```
c     alben(i),i=1,(ngpqtn+ngpqtg) =  / probabilities for generating
c     bc(i),i=lpepsd+1,               \ pseudo-particles by group being
c             lpepsd+(ngpqtn+ngpqtg)   \   analised.
c
      read(in,100)mxalb,ialbwt,ialbrg,ialbcs,ialbeg,ndum,ncos,mregbs
100   format(14i5)
      write(6,101)
      write(6,102)mxalb,ialbwt,ialbrg,ialbcs,ialbeg,ndum,ncos,mregbs
101   format(//,5x,34hreading input for pseudo-particles,/)
102   format(//,5x,8hmxalb  =,i5,/,5x,8hialbwt =,i5,/,5x,8hialbrg =,i,,
     */,5x,8hialbcs =,i5,/,5x,8hialbeg =,i5,/,5x,8hndum   =,i5,/,5x,
     *8hncos   =,i5,/,5x,8hmregbs =,i5)
c
c  the next is used only when the displacement is to be made in the
c      direction of the axis of the duct.
c
      read(in,190)nduct
      write(6,301)nduct
301   format(//,5x,'number of ducts described in the geometry = ',i5)
190   oormat(i5)
      read(in,191)(ndct(ik),ik=1,nduct)
      write(6,302)(ndct(ik),ik=1,nduct)
302   format(/,5x,'geometry number of the ducts =',/,5x,14i5)
191   format(14i5)
      read(in,192)(uaxis(ik),vaxis(ik),waxis(ik),ik=1,nduct)
      write(6,303)(ik,uaxis(ik),vaxis(ik),waxis(ik),ik=1,nduct)
303   format(//,5x,'direction cosines of the ducts',/,5x,'duct #',5x,
     *'u-direction',5x,'v-direction',5x,'w-direction',/,(5x,i6,5x,f11.5,
     *5x,f11.5,5x,f11.5))
192   format(3f10.5)
c
c  end duct
c
      if(mxalb.le.0) goto 51
      if(ialbcs.eq.0)goto 20
c
      read(in,110)ximp,yimp,zimp
      write(6,111)ximp,yimp,zimp
110   format(3f10.7)
111   format(//,5x,6hximp =,f10.5,2x,6hyimp =,f10.5,2x,6hzimp =,f10.5)
20    locpsd = nlastx
      ia = locpsd+1
      ib = ia + mregbs - 1
c
c     now read the region number which corresponds to cross-section sets.
c
      read(in,100)(nc(i),i=ia,ib)
      write(6,201)(nc(i),i=ia,ib)
201   format(/,5x,38hregion number for the pseudo-particles,/,
     *(5x,14i5))
c
      if(ialbrg.eq.0)goto 30
```

```
      lprpsd = ib
      ia = ib + 1
      ib = ia + mregbs - 1
c
c     now read the probabilities for generating pseudo-particles by region.
c                                        ICG - 06/09/89
      read(in,200)(bc(i),i=ia,ib)
200   format(8f10.5)
      write(6,202)(bc(i),i=ia,ib)
202   format(/,5x,55hprobabilities for generating pseudo-particles by re
     *gion,/,(5x,8f10.5))
      lpepsd=ib
c
 30   if(ialbeg.eq.0) goto 40
      ia = ib + 1
      ib = ia + ngpqtn + ngpqtg - 1
c
c     now read the probabilities for generating pseudo-particles by energy
c     group.                             ICG - 06/09/89
c
      read(in,200)(bc(i),i=ia,ib)
      write(6,203)(bc(i),i=ia,ib)
203   format(/,5x,'probabilities for generating pseudo-particles by ener
     *gy',/,(5x,8f10.5))
c
 40   if(ialbwt.eq.0)goto 50
      locneu = ib + 1
      locnne = locneu + mregbs
      locggm = locnne + mregbs
      locngm = locggm + mregbs
      nlastx = locngm + mregbs - 1
      do 70 klm=locneu,locnne-1
70    bc(klm) = 0.
      do 71 klm=locnne,locggm-1
71    nc(klm) = 0
      do 72 klm=locggm,locngm-1
72    bc(klm) = 0.
      do 73 klm=locngm,nlastx
73    nc(klm) = 0
      return
50    nlastx=ib
51    continue
      return
      end
```

## MODIFIED VERSION OF THE ALBDO SUBROUTINE

```
      subroutine  albdo(ig,u,v,w,wate,nmed,nreg,mark,x,y,z)
c *****
c *****
c ***** no debugging has been done on provisions for gamma-rays (as seco
c *****
c *****
      common /nxt/ mxalb,locpsd,lprpsd,lpepsd,ialb,dokill,un0,vn0,wn0,
     * x0,y0,z0,ncos,mregbs,iblznt,ximp,yimp,zimp,kmizxt,ndum,iiblzn,
     * locneu,locnne,locggm,locngm,ialbwt,ialbcs,ialbrg,ialbeg
     * ,nduct,ndct(50),uaxis(50),vaxis(50),waxis(50),irrpr1
c
      common /albcon/ ntheti, inrgys, nmgpa,  nino,   ninonp, igamg,
     1                locice, locenr, locvel, loccor, locthi, lntho,
     2                lsntho, loctho, locnp,  lsnpki, lsnpi,  locphi,
     3                loctot, locang, locgam, locume, locump, locumt,
     4                ialbdo, maxalb, lenalb, medal(10), icomb
      common /user/   agstrt, wtstrt, xstrt,  ystrt,  zstrt,  dff,
     1                ebotn,  ebotg,  tcut,   i0,     il,     iadjm,
     2                ngpqt1, ngpqt2, ngpqt3, ngpqtg, ngpqtn, nits,
     3                nlast,  nleft,  nmgp,   nmtg,   nstrts
      common /normal/ un, vn, wn
      common bc(1)
      dimension nc(1)
      equivalence (bc(1),nc(1))
      data nnegt/0/, nnegp/0/, acons/1.e-4/
c * * iin = index of incident polar angle
c * *   k = index of reflected polar angle
c * *   l = index of reflected azimuthal angle
      igold = ig
      igalb = ig
      if (locice .ne. locenr) igalb = nc(locice + ig)
      ngpn=ngpqtn
      ngpg=ngpqtg
      if (nmgp.gt.ngpqtn)ngpn=ngpqtn+1
      if ((nmtg-nmgp).gt.ngpqtg .and. ngpqtg.ne.0)ngpg=ngpqtg+1
      if (locice .eq. locenr) goto 8
      igmaxn = nc(locice + ngpn)
      igmaxg = nc(locice + nmgp + ngpg)
      iggaml = nc(locice + nmgp)
      goto 5
  8   igmaxm = ngpn
      igmaxg = nmgp + ngpg
      iggaml = nmgp
c
  5   call thet0(iin,4halbo,u,v,w,dumy,scosto)
c
c     the following command calculates the displacement of the point
```

```
c     of emergence in relation to the point of incidence for ordinary
c      concrete - this approximate fitting was obtained from several
c      MCNP calculations.              ICG - 09/06/89
c
      kprim=kprim+1
      if(kprim.le.1)write(6,700)
700   format(5x,'************ WARNING : You are using the displacement',
     */,15x,'of the point of emergence for ordinary concrete **********
     ***')
      if(ialb.ne.0) goto 702
      do 701 ik0=1,nduct
      if(irrprl.eq.ndct(ik0))goto 702
701   continue
      write(6,721)x,y,z,u,v,w,mark
721   format(5x,' x,y,z= ',3f10.5,' u,v,w= ',3f10.5,' mark= ',i5)
      mark=-1
        write(6,703)irrprl,(ndct(ii0),ii0=1,nduct)
703   format(5x,'There is not any zone in the input which corresponds to
     * the duct in which the particle is ',/,5x,'*** Then => STOP ***',
     */,5x,'irrprl= ',i4,' nduct(i)= ',(10i5))
      stop
702   call gtmed(imed,med)
      call nsigta(ig,1,tsig,pn)
      displ=1.65*sqrt(1.-scosto*scosto)/tsig
c
c     now calculate the componentes of the vector in the albedo plane
c
      ul=u-scosto*un
      vl=v-scosto*vn
      wl=w-scosto*wn
      fnorm=sqrt(ul*ul+vl*vl+wl*wl)
c
c     now normalize the vector
c
      ul=ul/fnorm
      vl=vl/fnorm
      wl=wl/fnorm
c
c     this version calculates the displacement on the direction of the
c     projection of the incident direction on the albedo plane.
c
      if(displ.eq.0) goto 705
      mysign=1
      if(displ.lt.0.)mysign=-1
      displ=abs(displ)
      if(ialb.eq.0) goto 855
855   call gomes(mark,ul,vl,wl,displ,mysign)
      if(mark.eq.(-1)) goto 706
      if(mark.eq.1.or.mark.eq.2) goto 705
      if(mark.eq.3) write(6,707)x,y,z
707   format(5x,'*** exit called by albdo ****',/,5x,
     *'*** a specular reflective surface was encountered during the disp
```

```
     *lacement of ***',/,5x,'*** the point of emergence at an albedo eve
     *nt ***',/,5x,' x= ',e15.7,'  y= ',e15.7,'  z= ',e15.7)
      stop
705   imin = (iin-1)*inrgys
      wate = wate*bc(loctot + (ialbdo-1)*lenalb + igalb + imin)
      jmo = igalb - 1
      ini = inrgys + 1
      nthets = nc(lsntho + iin)
      ntheto = nc(lntho + iin)
      indt = locumt + (ialbdo-1)*lenalb + inrgys*nthets + jmo*ntheto--
c * * cumtet(k) = bc(indt+k)
      ra = fltrnf(ra)
      do 10 k=1,ntheto
      if (ra - abs(bc(indt+k))) 15,15,10
 10   continue
      call error
 15   if (bc(indt+k)) 20,25,25
 20   wate = 0.0\
      nnegt = nnegt + 1
      return
 25   indth = loctho + nthets + iin - 1 + k
      coso = bc(indth) + fltrnf(ra)*(bc(indth+1)-bc(indth))
c * * 0 .lt. coso .lt. 1.0       -- outgoing polar angle
      sino = sqrt(1.0 - coso**2)
      nsnpi = nc(lsnpi + iin)
c *****
      ind1 = nc(lsnpki + nthets + ntheto + iin)
      ind2 = nc(lsnpki + nthets + k + iin - 1)
      indp = locump + (ialbdo-1)*lenalb + inrgys*nsnpi + jmo*ind1 + ind2
c * * cumphi(l) = bc(indp+1)
      ra = fltrnf(ra)
      nphio = nc(locnp + nthets + k)
      do 30 l=1,nphio
      if (ra-abs(bc(indp+l))) 35,35,30
 30   continue
      call error
 35   if (bc(indp+l)) 40,45,45
 40   wate = 0.0
      nnegp = nnegp + 1
      return
 45   lp = locphi + nsnpi + nthets + k - 1 + l+ind2
      cosphi = bc(lp) + fltrnf(ra)*(bc(lp+1) - bc(lp))
      sinphi = sqrt(1.0 - cosphi**2)
      if (sflraf(ra)) 50,50,55
 50   sinphi = - sinphi
 55   mmax = inrgys - igalb + 1
      inde = locume + (ialbdo-1)*lenalb + ini*inrgys/2*nsnpi
     1              + (jmo*inrgys-(jmo-1)*jmo/2)*ind1
     2              + (inrgys-jmo)*(ind2+1-1)
c * * cume(m) = bc(inde+m)
c     Now the emergent energy group is biased.
ct
```

```
      xbias0=1.
      xbiasg=0.
      ra = fltrnf(ra)
      if(igalb.gt.igmaxn)goto 6
c
c bc(inde+igmaxn-igalb+1) is the albedo reflection probability for the group ig
c
      xbiasn = bc(inde + igmaxn - igalb+1)
      if(ngpqtg.gt.0)
     *xbiasg = bc(inde + igmaxg - igalb+1) - bc(inde + iggaml - igalb+1)
      xbias0 = xbiasn + xbiasg
      igtotn = igmaxn - igalb+1
c
c   now select the outgoing  energy group
c
      do 61 m=1,igtotn
      if ( ra-abs( bc(inde+m) / xbias0) ) 65,65,61
61    continue
      if (ngpqtg.le.0) goto 66
      ml = iggaml+1
c
c     iggaml is  the last neutron group number in the albedo structure
c
      igtotg = igmaxg
      do 62 m = ml,igtotg
      xig = ( bc(inde + igmaxn - igalb+1) + bc(inde + m - igalb+1)
     * - bc(inde + iggaml - igalb+1) ) / xbias0
      if (ra - abs(xig)) 98,98,62
62    continue
66    write(6,63)xig,xbias0,xbiasn,xbiasg
63    format(5x,'*** Error at the biasing normalization ****',/,
     *5x,'xig = ',e15.7,' xbias0 = ',e15.7,' xbiasn = ',e15.7,' xbiasg =
     * ',e15.7)
      call error
6     mmax = igmaxg - igalb+1
      xbias0 = bc(inde + mmax)
      do 60 m=1,mmax
      if (ra-abs( bc(inde + m) / xbias0 ) ) 65,65,60
 60   continue
      write(6,64)xbias0
64    format(5x,' *** Error at the energy biasing normalization ***',/,
     *5x,' xbias0 = ',e15.7)
      call error
 98   if ( xig ) 70,70,75
 65   if ( bc(inde + m) / xbias0 ) 70,70,75
 70   wate = -wate
c
c   define the new energy group
c
 75   igalb = igalb + m - 1
c
      wate = wate*xbias0
```

```
c
c    end of the changes I.C.G. 7/20/89
c
      ig = igalb
      if (locice .eq. locenr) go to 105
c * igalb must be converted to xsec structure as ig * * * * * * *
      npg = 0
      do 80 ige=igold,nmtg
      if (nc(locice+ige) .ne. igalb) go to 80
      if (npg .eq. 0) nfpg = ige
      npg = npg + 1
 80   continue
c * npg is no. of possible xsec grps corresponding to igalb based on ini
c   locenr is zero address of albedo energy array
      indx = locenr + igalb
c * select a value for the actual energy
      r = fltrnf(0)
      if (icomb .ne. 0  .and.  igalb .gt. nmgpa) go to 85
c * this selection is uniform in lethargy from the lower limit, bc(indx+
c   to the upper limit, bc(nfpg)
      energy = bc(indx+1)*(bc(nfpg)/bc(indx+1))**r
      go to 90
 85   indx = indx + 1
c * * this selection is uniform in energy, same limits as above
      energy = bc(indx+1) + r*(bc(nfpg)-bc(indx+1))
 90   ng = nfpg + npg - 1
c  * now which xsec energy does it correspond to?
      do 95 i=nfpg,ng
 95   if (energy .gt. bc(i)) go to 100
 100  ig = i - 1
 105  continue
      ssint  = sqrt(1.0-scosto**2)
      if (ssint - acons) 115,110,110
 110  a = u - scosto*un
      b = v - scosto*vn
      c = w - scosto*wn
      go to 130
 115  call gtiso(a,b,c)
 120  dotn = a*un + b*vn + c*wn
      a = a - dotn*un
      b = b - dotn*vn
      c = c - dotn*wn
      if (abs(a)+abs(b)+abs(c)-acons) 115,115,125
 125  dmag = sqrt(a*a + b*b + c*c)
      a=a/dmag
      b=b/dmag
      c=c/dmag
      ssint =1.0
c  *  * select direction cosines and normalize
 130  u = coso*un + sino*cosphi*a/ssint  + sino*sinphi*(c*vn-b*wn)/ssint
      v = coso*vn + sino*cosphi*b/ssint  + sino*sinphi*(a*wn-c*un)/ssint
      w = coso*wn + sino*cosphi*c/ssint  + sino*sinphi*(b*un-a*vn)/ssint
```

```
      fnorm = sqrt(u*u+v*v+w*w)
      u= u/fnorm
      v=v/fnorm
      w= w/fnorm
706   continue
      return
      end
```

## SUBROUTINE GOMES

```
      subroutine gomes(mark,ul,vl,wl,etath,mysign)
c
      common /nxt/ mxalb,locpsd,lprpsd,lpepsd,ialb,dokill,un0,vn0,wn0,
     * x0,y0,z0,ncos,mregbs,iblznt,ximp,yimp,zimp,kmizxt,ndum,iiblzn,
     * locneu,locnne,locggm,locngm,ialbwt,ialbcs,ialbrg,ialbeg,jyl
     * ,nduct,ndct(50),uaxis(50),vaxis(50),waxis(50),irrprl,im,jm,nxy
c
      common /nutron/ name,namex,ig,igo,nmed,medold,nreg,u,v,w,uold,vold
     1 ,wold,x,y,z,xold,yold,zold,wate,oldwt,wtbc,blznt,blzon,age,oldage
      common/parem/xb(3), wb(3), ir,    wp(3),  xp(3)  ,idbg  ,
     1             irprim,nasc   ,lsurf  ,nbo   ,lri   ,lro   ,
     2             rin   ,rout   ,kloop ,loop   ,itype ,pinf  ,
     3             noa   ,dist
      common /orgi/ markg,dist0,nmedg,nblz,blzold
      common/gomloc/ kma,kfpd,klcr,knbd,kior,kriz,krcz,kmiz,kmcz,
     1  kkrl,kkr2,knsr,kvol,nadd,ldata,ltma,lfpd,numr,irtru,numb,nir
      common /albcon/ ntheti, inrgys, nmgpa,  nino,   ninonp, igamg,
     1lsumenad,igalbold, locice, locenr, locvel, loccor, locthi, lntho,
     2  idnt, lsumen, lsntho, loctho, locnp,  lsnpki, lsnpi,  locphi,
     3 locnij, locmxy, loctot, loccen, locpxy, locpet, locpaz, locadj,
     4 lsumadj,maxija, ialbdo, maxalb, lenalb, medal(10),icomb,lsumenr
      common /normal/ un, vn, wn
      common nstor(1)
      dimension wts(1)
      equivalence  (wts(1),nstor(1))
      integer blznt,blzold
      kmiz=kmizxt
      if(ialb.ne.0)kmiz=locpsd+1
      dist=0.0
      x=x+0.1*un
      y=y+0.1*vn
      z=z+0.1*wn
      u2=ul*mysign
```

```
      v2=v1*mysign
      w2=w1*mysign
      klp=0
      irprim=blznt/32768
      irppp=irprim
      ir=blznt-irprim*32768
      blzold=blznt
      irold=ir
    5 nasc=-1
      xb(1)=x
      xb(2)=y
      xb(3)=z
      wb(1)=u2
      wb(2)=v2
      wb(3)=w2
   10 dist0=dist+etath
      mkk=0
      call gl(s,nstor(kma),nstor(kfpd),nstor(klcr),nstor(knbd),
     1   nstor(kior),nstor(kkrl),nstor(kkr2))
      if(irprim.ne.(-3)) go to 20
c      some difficulty has occurred in tracking through the eeometry * *
c          treat the particle as an escape and continue * * * * * * * *
      write(6,1001)name,nmed,nreg,x,y,z
 1001 format(6h0name=i5,6h nmed=i5,6h nreg=i5,3h x=1pe14.6,3h y=1pe14.6,
     13h z= 1pe14.6)
c      call pr(1)
      mark=-1
      return
   20 x=xb(1)+wb(1)*dist
      y=xb(2)+wb(2)*dist
      z=xb(3)+wb(3)*dist
c
      if(markg.eq00)klp=klp+1
c
      nmedt=nstor(kmiz+irprim-1)
      blznt=irprim+32768*ir
c    markg = 1, the flight lies completely within one medium          * *
c                          or encountered an albedo boundary          * *
c          = 0, the flight crosses a medium boundary                  * *
c
c   more changes. ICG - 06/13/89
c
      mark = markg
      iblznt=irprim
      iiblzn=blznt
      if(irprim.ne.irppp)mkk=1
      irppp=irprim
      ialbdo = nmedt
      if(nmedt.gt. 1000) go to 40
c      nmedt gt 1000 indicates a specular reflection albedo boundary
      do 25 i=1,maxalb
c   this loop determines if an albedo media was encountered
```

```
      ialbdo=i
      if(nmedt .eq. medal(i) .and. klp .gt. 0 ) mark=2
   25 continue
      nalb=0
      ii=ir
      ir=irprim
      irprim=ii
      nreg=nstor(kriz+ir-1)
      blzold=blznt
      irold=ir
      nmed=nmedt
      etath=etath-s
      if(nmed.le.0)   mark=-1
c        nmed=0 indicates an escape from the system * *  * * * * * * * * *
c        nmed=1000 indicates an internal void encountered* * * * * * * * *
c
      if(mark.eq.(-1)) return
      if(markg.eq.1) goto 11
      goto 10
11    continue
      return
c
c      reflective medium has been encountered  *  *  *  *  *   *   *   *   *
c      set mark =3    *  *  *  *   *   *   *   *
   40 nalb=ialbdo
      mark=3
50    return
      end
```

## APPENDIX B

## STORM SUBROUTINES AND INTERFACES

The STORM subroutines were developed to perform the albedo-modified random walks with TORT-generated spatially-dependent albedo data. The subroutines which comprise the STORM package are ALBDO3D, ALBIN3D, and ALBDOE3D. The differences between the STORM subroutines and their BREESE counterparts are described in this appendix.

The STORM interface code ALBFAC3D reads the albedo information tapes generated by the TORT code and performs all the required steps for generating cumulative distribution tables. The ALBFAC3D interface code has two different versions, one specialized for generating forward albedos and another for generating adjuncton-albedos. Future versions of the code should incorporate the forward and adjoint versions into the same code - which was not possible with this experimental version due to the large amounts of computer memory required in dimensioning the working arrays (this problem can be solved with advanced programing techniques). The differences between this interface code and the BREESE code are primarily in the use of spatially-dependent albedo

information and on the nature of the marginal/conditional scheme used in constructing the probability tables.

The ALBDO3D subroutine performs the albedo reflection selecting the outgoing energy first, then the emergent position from the conditional cdf for the spatial variable, then the outgoing polar angle from the doubly-conditional cdf for the polar angle, and finally the outgoing azimuthal angle from the triply-conditional cdf for the azimuthal angle. This sequence of selecting the variables has an important influence on the estimation procedure in the ALBDOE3D subroutine, since the outgoing energy and position are fixed prior to the selection of the outgoing direction-defining angles.

The following are listings of the two versions of the ALBFAC3D code and of the subroutine ALBDO3D, ALBDOE3D, and ALBIN3D.

## THE ALBFAC3D INTERFACE CODE - FORWARD MODE

```
      dimension huse(2),titl(12),idum(14),iduma(10),x(20),y(20),z(20),
     *ener(100),dumrl(8),fang(70,13,13,37),fange(37),fangij(37),
     *fangtx(13,37),fangty(13,37),fangtm(5,37,16)
      dimension neta(70),etaout(10),
     *fangm(16,37),fangeadj(37,37),nij(169),fko(70,13,13),
     *naz(70),ntheto(5),
     *azi(12),beta(12),theto(10),fangme(5,16,37)
      dimension fangtme(20,5,37,16),totadj(25),nazz(70)
c
      character*8 infile(125)
```

```
c
      common /datas/ w(70),eta(70),xmu(70),energy(37),veloc(37),
     * ntheti,inrgys,nmgpa,maxalb,nazim(5),etain(5),nazimm(5)
c
         data neta/1,1,1,2,2,2,2,2,3,3,3,3,3,3,3,4,4,4,4,4,4,4,4,4,
     * 5,5,5,5,5,5,5,5,5,5,5,5,
     * 1,1,1,2,2,2,2,2,3,3,3,3,3,3,3,4,4,4,4,4,4,4,4,4,
     * 5,5,5,5,5,5,5,5,5,5,5,5/
c
         data naz/1,1,2, 1,1,2,3,4, 1,1,2,3,4,5,6, 1,1,2,3,4,5,6,7,8,
     *1,1,2,3,4,5,6,7,8,9,10, 3,3,4, 5,5,6,7,8, 7,7,8,9,10,11,12,
     *9,9,10,11,12,13,14,15,16, 11,11,12,13,14,15,16,17,18,19,20/
c
         data nazz/ 3,3,4, 5,5,6,7,8, 7,7,8,9,10,11,12,
     *9,9,10,11,12,13,14,15,16, 11,11,12,13,14,15,16,17,18,19,20,
     *1,1,2, 1,1,2,3,4, 1,1,2,3,4,5,6, 1,1,2,3,4,5,6,7,8,
     *1,1,2,3,4,5,6,7,8,9,10/
c
        data infile/'ft2801a1','ft2802a1','ft2803a1','ft2804a1',
     *'ft2805a1','ft2806a1','ft2807a1','ft2808a1','ft2809a1','ft2810a1',
     *'ft2811a1','ft2812a1','ft2813a1','ft2814a1','ft2815a1','ft2816a1',
     *'ft2817a1','ft2818a1','ft2819a1','ft2820a1','ft2821a1','ft2822a1',
     *'ft2823a1','ft2824a1','ft2825a1',
     *'ft2801a2','ft2802a2','ft2803a2','ft2804a2','ft2805a2','ft2806a2',
     *'ft2807a2','ft2808a2','ft2809a2','ft2810a2','ft2811a2','ft2812a2',
     *'ft2813a2','ft2814a2','ft2815a2','ft2816a2','ft2817a2','ft2818a2',
     *'ft2819a2','ft2820a2','ft2821a2','ft2822a2','ft2823a2','ft2824a2',
     *'ft2825a2',
     *'ft2801a3','ft2802a3','ft2803a3','ft2804a3','ft2805a3','ft2806a3',
     *'ft2807a3','ft2808a3','ft2809a3','ft2810a3','ft2811a3','ft2812a3',
     *'ft2813a3','ft2814a3','ft2815a3','ft2816a3','ft2817a3','ft2818a3',
     *'ft2819a3','ft2820a3','ft2821a3','ft2822a3','ft2823a3','ft2824a3',
     *'ft2825a3',
     *'ft2801a4','ft2802a4','ft2803a4','ft2804a4','ft2805a4','ft2806a4',
     *'ft2807a4','ft2808a4','ft2809a4','ft2810a4','ft2811a4','ft2812a4',
     *'ft2813a4','ft2814a4','ft2815a4','ft2816a4','ft2817a4','ft2818a4',
     *'ft2819a4','ft2820a4','ft2821a4','ft2822a4','ft2823a4','ft2824a4',
     *'ft2825a4',
     *'ft2801a5','ft2802a5','ft2803a5','ft2804a5','ft2805a5','ft2806a5',
     *'ft2807a5','ft2808a5','ft2809a5','ft2810a5','ft2811a5','ft2812a5',
     *'ft2813a5','ft2814a5','ft2815a5','ft2816a5','ft2817a5','ft2818a5',
     *'ft2819a5','ft2820a5','ft2821a5','ft2822a5','ft2823a5','ft2824a5',
     *'ft2825a5'/
c
        data nij/8,8,8,8,8,8,8,8, 9,9,9,9,9,9, 7,8,8,8,8,8,8,8, 9,9,9,9,9,9,
     * 7,7,8,8,8,8,8,8, 9,9,9,9,9,9,9, 7,7,7,5,5,5,5, 6,6,6,9,9,9,
     * 7,7,7,4,5,5,5,5, 6,6,6,9,9,9, 7,7,7,4,4,5,5, 6,6,6,9,9,9,
     * 10,10,10,2,2,2,1,  3,3,3,11,11,11, 7,7,7,4,4,5,5, 6,6,6,9,9,9,
     * 7,7,7,4,5,5,5,5, 6,6,6,9,9,9, 7,7,7,5,5,5,5,5, 6,6,6,9,9,9,
     * 7,7,8,8,8,8,8,8, 9,9,9,9,9,9,9, 7,8,8,8,8,8,8,8, 9,9,9,9,9,9,9,
     * 8,8,8,8,8,8,8,8, 9,9,9,9,9,9,9/
c
```

```
       data ntheto/5,5,5,5,5/
c
       data im,jm/ 13,13/
c
      call link('unit6=(alb102,create,text),unit7=(new102,create) //')
      call link('unit8=terminal,unit9=(albft28,create,text) //')
       write(6,1001)
1001   format(5x,' THREE-DIMENSIONAL ALBEDO DATA CALCULATED USING THE TO
     *RT CODE',//,17x,'25 neutron groups and 12 gamma-ray groups',//,5x
     *,'only 25 neutron energy groups are provided as incident energies'
     *,//,10x,'only material present in the library is ordinary concrete
     *',//,10x,'Data generated in September,1990 by Itacil Gomes at FEDC
     *-ORNL')
      maxija = 0
      do 401 ij1=1,169
      if(nij(ij1).gt.maxija) maxija=nij(ij1)
401   continue
      write(8,636) (nij(kl),kl=1,169)
636   format(13i5)
      write(7) 'THREE-DIMENSIONAL ALBEDO DATA'
      write(6,1002) ntheti,inrgys,nmgpa,maxalb,im,jm
1002  format(/////,5x,'ntheti =',i3,', inrgys =',i3,', nmgp =',i3,', max
     *alb =',i3,' im = ',i4,' jm = ',i4)
      write(7) ntheti,inrgys,nmgpa,maxalb,im,jm
      write(7) (energy(i),i=1,inrgys)
      write(7) (veloc(i),i=1,inrgys)
      write(7) (etain(i),i=1,5)
      write(7) (ntheto(i),i=1,5)
c
c  now, define the limits for the polar angles
c
      k=0
      pi = 4.*atan(1.)
      dv = sqrt(pi)
      r = 1./2./dv
      theto(1) = 1.
      do 901 i=1,5
      do 902 j=1,nazim(i)
      k=k+1
      sumw=sumw+2.*w(k)
902   continue
      h = sumw/dv
      theto(i+1)=(r-h)/r
      if(i.eq.5) theto(i+1)=0.
901   continue
      do 950 kth=1,ntheti
      write(6,1003) (theto(i),i=1,ntheto(kth)+1)
1003  format(//,5x,'limits for the outgoing polar angles',//,5x,6f12.5,
     *//)
950   continue
      do 1200 kth = 1,5
      write(7)(theto(i),i=1,6)
```

```
1200  continue
      do 960 kth=1,ntheti
      write(6,1006) (i,nazim(i),i=1,ntheto(kth))
1006  format(//,5x,'number of limits for the azimuthal angles per eta le
     *vel',//,5x,' eta level',5x,'azimuthal limits',/,(i10,i18))
960   continue
      do 1210 kth=1,5
      write(7) (nazim(i),i=1,ntheto(kth))
1210  continue
c
c   now, define the limits for the azimuthal angles
c
       k = 0
       do 909 i0=1,2
       do 910 i = 1,5
       wazi = 0.
       azi(1) = -1.
       k0 = k
       do 911 j = 1,nazim(i)
       k = k + 1
       wazi = wazi + w(k)
911    continue
       azim = pi
       write(6,1004)
1004   format(5x,'theta out',2x,'  phi out',4x,'cosine phi',5x,'cosine b
     *eta')
       k = k0
       do 912 j = 1,nazim(i)
       k = k+1
       delazi = w(k) * pi / wazi
       if(i0.eq.2) delazi = -1.* delazi
       azim = azim + delazi
       azi(j) = cos(azim)
       beta(j) = sin(azim)
       write(6,1005) i,j,azi(j),beta(j)
1005   format(5x,i10,i10,f15.7,f15.7)
 912   continue
       k = k0 + nazim(i)
c
       write(7) (azi(j),beta(j),j=1,nazim(i))
c
 910   continue
 909   continue
       write(6,1102)
1102   format(//,10x,'relation between fine and super meshes numbers',
     *//,5x,'fine mesh  super mesh',5x,'fine mesh  super mesh',5x
     *,'fine mesh  super mesh')
c
       write(7) (nij(i),i=1,169)
c
       do 1101 kcol = 1,57
       if(kcol.eq.57) write(6,1008) kcol,nij(kcol)
```

```
      if(kcol.eq.57) goto 1101
      write(6,1008) kcol,nij(kcol),kcol+57,nij(kcol+57),kcol+113,
     *nij(kcol+113)
1101  continue
1008  format(5x,2i10,5x,2i10,5x,2i10)
c
c     begin loop over all incident directions
c
      do  4 mi0=1,5
c
c     define the location of the source normalization in the eta array
c
      do 104 igg0 = 1,20
      do 104 igg1 = 1,20
      fangeadj(igg0,igg1) = 0.
104   continue
c
      mi01 = nazim(1) * min0( 1,(mi0-1) ) + nazim(2) * min0( 1,max0(0,
     *(mi0-2) )) + nazim(3) * min0( 1,max0(0,(mi0-3))) + nazim(4)
     * * min0( 1,max0(0,(mi0-4) )) + 1
      write(6,891) mi0,mi01,eta(mi01)
891   format(5x,'mi0,mi01,eta(mi01) = ',2i5,f15.7)
c
c     begin loop over all incident energies.
c
      do  5 ig0 = 1,20
c
c     clear the work arrays
c
      fangtot = 0.
c
      do 101 igg = 1,20
      fange(igg) = 0.
      fangij(igg) = 0.
c
      do 102 ik2 = 1,13
      fangtx(ik2,igg) = 0.
      fangty(ik2,igg) = 0.
102   continue
c
      do 103 ik1 = 1,maxija
      fangm(ik1,igg) = 0.
c
      do 103 mk1 = 1,5
      fangtm(mk1,igg,ik1) = 0.
      fangme(mk1,ik1,igg) = 0.
      do 103 mk2 = 1,20
      fangtme(mk2,mk1,igg,ik1) = 0.
103   continue
c
      fangm(ik1,igg) = 0.
101   continue
```

```
      do 925 i=1,25
925   totadj(i) = 0.
c
c     close and delete the last file used as input angular something
c
      if(ig0.gt.1.or.mi0.gt.1) close(28,status='delete')
c
c     define the name of the new file and open it.
c
      nfile = ig0 + (mi0-1) * 25
      open(28,file=infile(nfile),status='old',access='sequential')
c
c     record counter
c
      il=il+1
      write(6,90) il
 90   format(/,5x,'file # ',i5)
c
c     read the headers of the albedo file
c
      read(28) hname,(huse(i),i=1,2),ivers
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      read(28) date,user,charge,case,time,(titl(i),i=1,12)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      read(28) ifmom,ifbnd,idim, igm,neut, im,jm,km,lms,mm, nresp
     * , (idum(n),n=1,14)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      read(28)     (iduma(i),i=1,10)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      read(28) (x(i),i=1,im+1) , (y(j),j=1,jm+1) , (z(k),k=1,km+1)
     * , (ener(ig),ig=1,igm) , emin , eneut
     * , (dumrl(kl),kl=1,8)
      iwr = iwr + 1
      if(iwr.gt.1) goto 1702
      write(6,1013)
1013  format(/,5x,'mesh boundaries in the x-direction')
      write(6,140)(x(i),i=1,im+1)
      write(6,1014)
1014  format(/,5x,'mesh boundaries in the y-direction')
      write(6,140) (y(j),j=1,jm+1)
140   format(5e14.5)
      write(7) (x(i),i=1,im+1) , (y(j),j=1,jm+1)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     begin loop over all available energy group in the albedo file
c
 1702  continue
c
      do 10 ig=ig0,20
c
c     read the angular data from the albedo file
c
```

```
      read(28)(((fko(m,i,j),m=1,70),i=1,13),j=1,13)
      write(9,951) (fko(m,7,7),m=1,70)
951   format(9e12.4)
c
c     begin loop to normalize the data and to determine overall
c     summation for each distribution
c
      do 1 m01=1,70
      do 1 i01=1,13
      do 1 j01=1,13
c
c     normalize to get angular flow
c
c     source term =  area(incident)*eta(incident) = 25. * eta(mi01)
c     normalization to get angular flow from the data =
c         quadrature weight(emergent) * area(emergent) * eta(emergent) =
c       = w(m01) * (x(i01+1)-x(i01)) * (y(i01+1)-y(i01)) * eta(m01)
c
      fang(m01,i01,j01,ig) = fko(m01,i01,j01) * w(m01)
     * * ( x(i01+1) - x(i01)) * ( y(j01+1) - y(j01))
     * * eta(m01) / 25. / abs(eta(mi01))
c
c     add all outgoing component for a given eta, and energy incident
c
      fangtot = fangtot + fang(m01,i01,j01,ig)
c
c     add all outgoing component for a given outgoing energy, and incident
c      eta and energy
c
      fangij(ig) = fangij(ig) + fang(m01,i01,j01,ig)
c
c     define the position inside the colapsing array for angular distribution
c
      ij01 = i01 + (j01-1) * 13
      ij = nij(ij01)
c
c     add all outgoing components in a given portion of the spatial meshes
c     (in which the meshes are going to have the same angular distribution)
c     for a given spatial region, outgoing energy, and incident energy and eta
c
      leveta = neta(m01)
      fangm(ij,ig) = fangm(ij,ig) + fang(m01,i01,j01,ig)
     fangme(leveta,ij,ig) = fangme(leveta,ij,ig) + fang(m01,i01,j01,ig)
c
c     end loop for outgoing directions, and positions for an outgoing energy
c
  1   continue
c
c     end loop for outgoing energies
c
 10   continue
c
```

```
c       write in the unit 6 the total albedo for incident energy and angle
c
        write(6,555)mi0,ig0,fangtot
555     format(//,5x,'total albedo for mi0 = ',i5,' and ig0 = ',i5,' = ',
     * e20.10)
c
c       begin loop to construct cummulative probability distributions
c
        do 20 ig=ig0,20
c
        xnormij=fangij(ig)
c
        do 31 ibeta=1,2
        do 31 m001=1,35
        do 31 i01=1,13
        do 31 j01=1,13
c
        m01 = (ibeta-1)*35 + m001
        ij01=i01+(j01-1)*13
        ij=nij(ij01)
        xfang0 = fang(m01,i01,j01,ig)
c
c      energy probability distribution
c
        xfange = fange(ig)
        fange(ig) = xfange + xfang0 / fangtot
c
c      position probability distribution
c
        na = naz(m01)
c
c       the next command represents the symmetry of the angular flux
c       to the center line in the y-direction. This is used because of
c       the spatial colapse of the angular distribution.
c
        if(j01.gt.7) na = nazz(m01)
        leta = neta(m01)
        xfangi = fangtx(i01,ig)
        xfangj = fangty(j01,ig)
        xfangm = fangtm(leta,ig,ij)
        xfangme = fangtme(na,leta,ig,ij)
c
        fangtx(i01,ig) = xfangi + xfang0 / xnormij
c
        fangty(j01,ig) = xfangj + xfang0 / xnormij
c
        fangtm(leta,ig,ij) = xfangm + xfang0 / fangm(ij,ig)
c
        fangtme(na,leta,ig,ij) = xfangme + xfang0 / fangme(leta,ij,ig)
c
31      continue
20      continue
```

```
      write(6,184)mi0,ig0
184   format(//,5x,' probability distribution tables for incident theta'
     *,i4,' incident energy group',i4)
      write(6,180)ig0,igm,(fange(ig),ig=ig0,20)
c
      write(7) fangtot, (fange(ig),ig=ig0,20)
c
      write(7)(((fangtx(i,ig),i=1,13),(fangty(j,ig),j=1,13)),ig=ig0,20)
c
      do 35 ij=1,maxija
      do 34 ig=ig0,20
      if(ig.gt.(ig0+3)) goto 34
      write(6,183)ij,ig,(fangtm(m,ig,ij),m=1,5)
      do 32 leta=1,5
      write(6,185)leta,ij,ig,(fangtme(nazi,leta,ig,ij),nazi=1,
     *2*nazimm(leta))
32    continue
34    continue
35    continue
c
      write(7) (((fangtm(leta,ig,ij),leta=1,5),ij=1,maxija),ig=ig0,20)
c
      do 931 leta=1,5
      write(7) (((fangtme(nazi,leta,ig,ij),nazi=1,2*nazimm(leta)),
     *ij=1,maxija),ig=ig0,20)
 931    continue
c
 180  format(/,5x,'energy distribution for group ',i5,' to group',i5,/,
     *(5e14.5))
 181  format(/,5x,'spatial distribution in x-direction for energy group'
     *,i4,/,(5e14.5,/,5e14.5,/,3e14.5))
 182  format(/,5x,'spatial distribution in y-direction for energy group'
     *,i4,/,(5e14.5,/,5e14.5,/,3e14.5))
 183  format(/,5x,'polar distribution for region',i4,' energy group'
     *,i4,/,(5e14.5))
 185  format(/,5x,'azimuthal distribution for eta level',i3,' region',i4
     *,' energy group',i4,/,(5e14.5))
c
c     end ig0 loop
c
  5     continue
c
c     end mi0 loop
c
  4     continue
c
9999  continue
      end
      block data
c
      common /datas/ w(70),eta(70),xmu(70),energy(37),veloc(37),
     * ntheti,inrgys,nmgpa,maxalb,nazim(5),etain(5),nazimm(5)
```

```
c
      data ntheti,inrgys,nmgpa,maxalb/ 5,20,20,1/
c
      data nazim/3,5,7,9,11/
c
      data nazimm/2,4,6,8,10/
c
      data etain/-.96225,-.83887,-.69389,-.50918,-.19245/
c
c      data etaout/.96225,.83887,.69389,.50918,.19245/
c
      data w/0.,115005.e-7,115005.e-7,0.,873399.e-8,873399.e-8,
     * 873399.e-8,873399.e-8,0.,6173.e-6,652549.e-8,6173.e-6,6173.e-6,
     * 652549.e-8,6173.e-6,0.,873399.e-8,652549.e-8,652549.e-8,
     * 873399.e-8,873399.e-8,652549.e-8,652549.e-8,873399.e-8,0.,
     * 115005.e-7,873399.e-8,6173.e-6,873399.e-8,115005.e-7,115005.e-7,
     * 873399.e-8,6173.e-6,873399.e-8,115005.e-7,
     * 0.,115005.e-7,115005.e-7,0.,873399.e-8,873399.e-8,
     * 873399.e-8,873399.e-8,0.,6173.e-6,652549.e-8,6173.e-6,6173.e-6,
     * 652549.e-8,6173.e-6,0.,873399.e-8,652549.e-8,652549.e-8,
     * 873399.e-8,873399.e-8,652549.e-8,652549.e-8,873399.e-8,0.,
     * 115005.e-7,873399.e-8,6173.e-6,873399.e-8,115005.e-7,115005.e-7,
     * 873399.e-8,6173.e-6,873399.e-8,115005.e-7/
c
      data xmu/  -27217.e-5,-19245.e-5, 19245.e-5,-54433.e-5,
     * -50918.e-5,-19245.e-5, 19245.e-5, 50918.e-5,-72008.e-5,
     * -69389.e-5,-50918.e-5,-19245.e-5, 19245.e-5, 50918.e-5,
     *  69389.e-5,-86066.e-5,-83887.e-5,-69389.e-5,-50918.e-5,
     * -19245.e-5, 19245.e-5, 50918.e-5, 69389.e-5, 83887.e-5,
     * -98131.e-5,-96225.e-5,-83887.e-5,-69389.e-5,-50918.e-5,
     * -19245.e-5, 19245.e-5, 50918.e-5, 69389.e-5, 83887.e-5,
     *  96225.e-5,
     * -27217.e-5,-19245.e-5, 19245.e-5,-54433.e-5,
     * -50918.e-5,-19245.e-5, 19245.e-5, 50918.e-5,-72008.e-5,
     * -69389.e-5,-50918.e-5,-19245.e-5, 19245.e-5, 50918.e-5,
     *  69389.e-5,-86066.e-5,-83887.e-5,-69389.e-5,-50918.e-5,
     * -19245.e-5, 19245.e-5, 50918.e-5, 69389.e-5, 83887.e-5,
     * -98131.e-5,-96225.e-5,-83887.e-5,-69389.e-5,-50918.e-5,
     * -19245.e-5, 19245.e-5, 50918.e-5, 69389.e-5, 83887.e-5,
     *  96225.e-5/
c
      data eta/ 96225.e-5, 96225.e-5, 96225.e-5,
     *  83887.e-5,  83887.e-5,  83887.e-5,  83887.e-5,  83887.e-5,
     *  69389.e-5,  69389.e-5,  69389.e-5,  69389.e-5,  69389.e-5,
     *  69389.e-5,  69389.e-5,
     *  50918.e-5,  50918.e-5,  50918.e-5,  50918.e-5,  50918.e-5,
     *  50918.e-5,  50918.e-5,  50918.e-5,  50918.e-5,
     *  19245.e-5,  19245.e-5,  19245.e-5,  19245.e-5,  19245.e-5,
     *  19245.e-5,  19245.e-5,  19245.e-5,  19245.e-5,  19245.e-5,
     *  19245.e-5,
     *  96225.e-5, 96225.e-5, 96225.e-5,
     *  83887.e-5, 83887.e-5, 83887.e-5, 83887.e-5, 83887.e-5,
```

```
     *  69389.e-5, 69389.e-5, 69389.e-5, 69389.e-5, 69389.e-5,
     *  69389.e-5, 69389.e-5,
     *  50918.e-5, 50918.e-5, 50918.e-5, 50918.e-5, 50918.e-5,
     *  50918.e-5, 50918.e-5, 50918.e-5, 50918.e-5,
     *  19245.e-5, 19245.e-5, 19245.e-5, 19245.e-5, 19245.e-5,
     *  19245.e-5, 19245.e-5, 19245.e-5, 19245.e-5, 19245.e-5,
     *  19245.e-5/
c
      data veloc/37*1./
c
      end
```

**ALBFAC3D INTERFACE CODE - ADJOINT MODE**

```
      dimension huse(2),titl(12),idum(14),iduma(10),x(20),y(20),z(20),
     *ener(100),dumrl(8),fadjij(37),fadjme(5,25,25),fadjtme(20,5,25,13)
        dimension fadje(25),fadjtx(13,25),fadjty(13,25),
     * fadjm(13,25),fadjtm(5,25,13),totadj(25),fko(70,13,13)
      dimension neta(70),etaout(10),nij(169),
     *nadj(70),ntheto(5),azi(12),beta(12),theto(10)
        dimension naadj(70)
        dimension fadjeta(5,25,5,25,13,13)
      character*8 infile(125)
c
      common /datas/ w(70),eta(70),xmu(70),energy(37),veloc(37),
     * ntheti,inrgys,nmgpa,maxalb,nazim(5),etain(5),nazimm(5)
         data neta/1,1,1,2,2,2,2,2,3,3,3,3,3,3,3,4,4,4,4,4,4,4,4,4,
     * 5,5,5,5,5,5,5,5,5,5,5,5,
     * 1,1,1,2,2,2,2,2,3,3,3,3,3,3,3,4,4,4,4,4,4,4,4,4,
     * 5,5,5,5,5,5,5,5,5,5,5,5/
c
      data naadj/1,1,2, 1,1,2,3,4, 1,1,2,3,4,5,6, 1,1,2,3,4,5,6,7,8,
     *1,1,2,3,4,5,6,7,8,9,10, 3,3,4, 5,5,6,7,8, 7,7,8,9,10,11,12,
     *9,9,10,11,12,13,14,15,16, 11,11,12,13,14,15,16,17,18,19,20/
c
         data nadj/ 3,3,4, 5,5,6,7,8, 7,7,8,9,10,11,12,
     *9,9,10,11,12,13,14,15,16, 11,11,12,13,14,15,16,17,18,19,20,
     *1,1,2, 1,1,2,3,4, 1,1,2,3,4,5,6, 1,1,2,3,4,5,6,7,8,
     *1,1,2,3,4,5,6,7,8,9,10/
        data infile/'ft2801a1','ft2802a1','ft2803a1','ft2804a1',
     *'ft2805a1','ft2806a1','ft2807a1','ft2808a1','ft2809a1','ft2810a1',
     *'ft2811a1','ft2812a1','ft2813a1','ft2814a1','ft2815a1','ft2816a1',
     *'ft2817a1','ft2818a1','ft2819a1','ft2820a1','ft2821a1','ft2822a1',
     *'ft2823a1','ft2824a1','ft2825a1',
```

```
     *'ft2801a2','ft2802a2','ft2803a2','ft2804a2','ft2805a2','ft2806a2',
     *'ft2807a2','ft2808a2','ft2809a2','ft2810a2','ft2811a2','ft2812a2',
     *'ft2813a2','ft2814a2','ft2815a2','ft2816a2','ft2817a2','ft2818a2',
     *'ft2819a2','ft2820a2','ft2821a2','ft2822a2','ft2823a2','ft2824a2',
     *'ft2825a2',
     *'ft2801a3','ft2802a3','ft2803a3','ft2804a3','ft2805a3','ft2806a3',
     *'ft2807a3','ft2808a3','ft2809a3','ft2810a3','ft2811a3','ft2812a3',
     *'ft2813a3','ft2814a3','ft2815a3','ft2816a3','ft2817a3','ft2818a3',
     *'ft2819a3','ft2820a3','ft2821a3','ft2822a3','ft2823a3','ft2824a3',
     *'ft2825a3',
     *'ft2801a4','ft2802a4','ft2803a4','ft2804a4','ft2805a4','ft2806a4',
     *'ft2807a4','ft2808a4','ft2809a4','ft2810a4','ft2811a4','ft2812a4',
     *'ft2813a4','ft2814a4','ft2815a4','ft2816a4','ft2817a4','ft2818a4',
     *'ft2819a4','ft2820a4','ft2821a4','ft2822a4','ft2823a4','ft2824a4',
     *'ft2825a4',
     *'ft2801a5','ft2802a5','ft2803a5','ft2804a5','ft2805a5','ft2806a5',
     *'ft2807a5','ft2808a5','ft2809a5','ft2810a5','ft2811a5','ft2812a5',
     *'ft2813a5','ft2814a5','ft2815a5','ft2816a5','ft2817a5','ft2818a5',
     *'ft2819a5','ft2820a5','ft2821a5','ft2822a5','ft2823a5','ft2824a5',
     *'ft2825a5'/
       data nij/8,8,8,8,8,8,8,8, 9,9,9,9,9,9, 7,8,8,8,8,8,8,8, 9,9,9,9,9,9,
     * 7,7,8,8,8,8,8,8, 9,9,9,9,9,9, 7,7,7,5,5,5,5, 6,6,6,9,9,9,
     * 7,7,7,4,5,5,5, 6,6,6,9,9,9, 7,7,7,4,4,5,5, 6,6,6,9,9,9,
     * 10,10,10,2,2,2,1,  3,3,3,11,11,11, 7,7,7,4,4,5,5, 6,6,6,9,9,9,
     * 7,7,7,4,5,5,5, 6,6,6,9,9,9, 7,7,7,5,5,5,5, 6,6,6,9,9,9,
     * 7,7,8,8,8,8,8, 9,9,9,9,9,9, 7,8,8,8,8,8,8, 9,9,9,9,9,9,
     * 8,8,8,8,8,8,8, 9,9,9,9,9,9/
c
       data ntheto/5,5,5,5,5/
c
       data im,jm/ 13,13/
c

      call link('unit6=(alb102,create,text),unit7=(new102,create) //'
     *)
      call link('unit8=terminal //')
       write(6,1001)
1001   format(5x,' THREE-DIMENSIONAL ADJOINT ALBEDO DATA CALCULATED USIN
     *G THE TORT CODE',//,17x,'25 neutron groups and 12 gamma-ray groups
     *',//,5x,'only 25 neutron energy groups are provided as incident en
     *ergies',//,10x,'only material present in the library is ordinary c
     *oncrete',//,10x,'Data generated in September,1990 by Itacil Gomes
     *at FEDC-ORNL')
      maxija = 0
      do 401 ij1=1,169
      if(nij(ij1).gt.maxija) maxija=nij(ij1)
401   continue
      write(8,636) (nij(kl),kl=1,169)
636   format(13i5)
      write(7) 'THREE-DIMENSIONAL ADJOINT ALBEDO DATA'
      write(6,1002) ntheti,inrgys,nmgpa,maxalb,im,jm
1002  format(/////,5x,'ntheti =',i3,', inrgys =',i3,', nmgp =',i3,', max
```

```
     *alb =',i3,' im = ',i4,' jm = ',i4)
      write(7) ntheti,inrgys,nmgpa,maxalb,im,jm
      write(7) (energy(i),i=1,inrgys)
      write(7) (veloc(i),i=1,inrgys)
      write(7) (etain(i),i=1,5)
      write(7) (ntheto(i),i=1,5)
c
c  now, define the limits for the polar angles
c
      k=0
      pi = 4.*atan(1.)
      dv = sqrt(pi)
      r = 1./2./dv
      theto(1) = 1.
      do 901 i=1,5
      do 902 j=1,nazim(i)
      k=k+1
      sumw=sumw+2.*w(k)
902   continue
      h = sumw/dv
      theto(i+1)=(r-h)/r
      if(i.eq.5) theto(i+1)=0.
901   continue
      do 950 kth=1,ntheti
      write(6,1003) (theto(i),i=1,ntheto(kth)+1)
1003  format(//,5x,'limits for the outgoing polar angles',//,5x,6f12.5,
     *//)
950   continue
      do 1200 kth = 1,5
      write(7)(theto(i),i=1,6)
1200  continue
      do 960 kth=1,ntheti
      write(6,1006) (i,nazim(i),i=1,ntheto(kth))
1006  format(//,5x,'number of limits for the azimuthal angles per eta le
     *vel',//,5x,' eta level',5x,'azimuthal limits',/,(i10,i18))
960   continue
      do 1210 kth=1,5
      write(7) (nazim(i),i=1,ntheto(kth))
1210  continue
c
c   now, define the limits for the azimuthal angles
c
       k = 0
       do 909 i0=1,2
       do 910 i = 1,5
       wazi = 0.
       azi(1) = -1.
       k0 = k
       do 911 j = 1,nazim(i)
       k = k + 1
       wazi = wazi + w(k)
911    continue
```

```
        azim = pi
        write(6,1004)
1004    format(5x,'theta out',2x,'  phi out',4x,'cosine phi',5x,'cosine b
     *eta')
        k = k0
        do 912 j = 1,nazim(i)
        k = k+1
        delazi = w(k) * pi / wazi
        if(i0.eq.2) delazi = -1.* delazi
        azim = azim + delazi
        azi(j) = cos(azim)
        beta(j) = sin(azim)
        write(6,1005) i,j,azi(j),beta(j)
1005    format(5x,i10,i10,f15.7,f15.7)
 912    continue
        k = k0 + nazim(i)
c
        write(7) (azi(j),beta(j),j=1,nazim(i))
c
 910    continue
 909    continue
        write(6,1102)
1102    format(//,10x,'relation between fine and super meshes numbers',
     *//,5x,'fine mesh  super mesh',5x,'fine mesh  super mesh',5x
     *,'fine mesh  super mesh')
c
        write(7) (nij(i),i=1,169)
c
        do 1101 kcol = 1,57
        if(kcol.eq.57) write(6,1008) kcol,nij(kcol)
        if(kcol.eq.57) goto 1101
        write(6,1008) kcol,nij(kcol),kcol+57,nij(kcol+57),kcol+113,
     *nij(kcol+113)
1101    continue
1008    format(5x,2i10,5x,2i10,5x,2i10)
c
c       begin loop over all incident directions
c
        do  4 mi0=1,5
c
c       define the location of the source normalization in the eta array
c
        mi01 = nazim(1) * min0( 1,(mi0-1) ) + nazim(2) * min0( 1,max0(0,
     *(mi0-2) )) + nazim(3) * min0( 1,max0(0,(mi0-3))) + nazim(4)
     * * min0( 1,max0(0,(mi0-4) )) + 1
        write(6,891) mi0,mi01,eta(mi01)
891     format(5x,'mi0,mi01,eta(mi01) = ',2i5,f15.7)
c
c       begin loop over all incident energies.
c
        do  5 ig0 = 1,20
c
```

```
c       close and delete the last file used as input angular something
c
        if(ig0.gt.1.or.mi0.gt.1) close(28,status='delete')
c
c       define the name of the new file and open it.
c
        nfile = ig0 + (mi0-1) * 25
        write(8,12345) nfile,infile(nfile)
12345   format(5x,'nfile = ',i5,'   file name = ',a8)
        open(28,file=infile(nfile),status='old',access='sequential')
c
c       record counter
c
        il=il+1
        write(6,90) il
 90     format(/,5x,'file # ',i5)
c
c       read the headers of the albedo file
c
        read(28) hname,(huse(i),i=1,2),ivers
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
        read(28) date,user,charge,case,time,(titl(i),i=1,12)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
        read(28) ifmom,ifbnd,idim, igm,neut, im,jm,km,lms,mm, nresp
     * , (idum(n),n=1,14)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
        read(28)     (iduma(i),i=1,10)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
        read(28) (x(i),i=1,im+1) , (y(j),j=1,jm+1) , (z(k),k=1,km+1)
     * , (ener(ig),ig=1,igm) , emin , eneut
     * , (dumrl(kl),kl=1,8)
        iwr = iwr + 1
        if(iwr.gt.1) goto 1702
        write(6,1013)
1013    format(/,5x,'mesh boundaries in the x-direction')
        write(6,140)(x(i),i=1,im+1)
        write(6,1014)
1014    format(/,5x,'mesh boundaries in the y-direction')
        write(6,140) (y(j),j=1,jm+1)
140     format(5e14.5)
        write(7) (x(i),i=1,im+1) , (y(j),j=1,jm+1)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       begin loop over all available energy group in the albedo file
c
 1702   continue
        do 2323 i=1,5
        do 2323 k=1,25
```

```
      do 2323 l=1,maxija
      fadjme(i,l,k) = 0.
      do 2323 j=1,20
      fadjtme(j,i,k,l) = 0.
2323  continue
c
      do 10 ig=ig0,20
c
c     read the angular data from the albedo file
c
      read(28)(((fko(m,i,j),m=1,70),i=1,13),j=1,13)
c
c     begin loop to normalize the data and to determine overall
c     summation for each distribution
c
      do 1 m01=1,70
      do 1 i01=1,13
      do 1 j01=1,13
c
c     normalize to get angular flow
c
c     source term =  area(incident)*eta(incident) = 25. * eta(mi01)
c     normalization to get angular flow from the data =
c         quadrature weight(emergent) * area(emergent) * eta(emergent) =
c       = w(m01) * (x(i01+1)-x(i01)) * (y(i01+1)-y(i01)) * eta(m01)
c
c     define the position inside the colapsing array for angular distribution
c
      ij01 = i01 + (j01-1) * 13
      ij = nij(ij01)
c
      xfko =  fko(m01,i01,j01) * w(m01)
     * * ( x(i01+1) - x(i01)) * ( y(j01+1) - y(j01))
     * * eta(m01) / 25. / abs(eta(mi01))
c
      mi1 = neta(m01)
      xadjeta = fadjeta(mi1,ig,mi0,ig0,i01,j01)
      fadjeta(mi1,ig,mi0,ig0,i01,j01) = xadjeta + xfko
c
      na = nadj(m01)
c
c     the next command represents the symmetry of the angular flux
c     to the center line in the y-direction. This is used because of
c     the spatial colapsing of the angular distribution.
c
      if(j01.gt.7) na = naadj(m01)
c
      xfadja = fadjtme(na,mi1,ig,ij)
      fadjtme(na,mi1,ig,ij) = xfadja + xfko
c
      xfadjm = fadjme(mi1,ij,ig)
      fadjme(mi1,ij,ig) = xfadjm + xfko
```

```
c
c       end loop for outgoing directions, and positions for an outgoing energy
c
  1     continue
c
c       end loop for outgoing energies
c
 10     continue
c
c       begin loop to construct cummulative probability distributions for
c       the azimuthal angle
c
        do 20 ig=ig0,20
c
        do 31 mil=1,5
c
        do 31 ij=1,maxija
c
        xfadjm = fadjme(mil,ij,ig)
c
        do 31 na = 1,2*nazimm(mil)
c
c       the next command represents the symmetry of the angular flux
c       to the center line in the y-direction. This is used because of
c       the spatial colapsing of the angular distribution.
c
        xfadja = fadjtme(na,mil,ig,ij)
c
        fadjtme(na,mil,ig,ij) = xfadja / xfadjm
         jtty = jtty + 1
       if(jtty.le.100) write(6,1736) xfadja,xfadjm,fadjtme(na,mil,ig,ij)
      * ,na,mil,ig,ij
1736     format(5x,'xadj2,xadjm,ftme = ',3e15.7,'   na,mil,ig,ij= ',4i5)
c
31     continue
20     continue
       do 4020 ig=ig0,20
       do 4031 m01 = 1,5
       do 4031 ij = 1,maxija
       xtest = 0.
       do 4032 na = 1,2*nazimm(m01)
       xtest = xtest + fadjtme(na,m01,ig,ij)
4032   continue
       ytest = 1. - xtest
       ytest = abs(ytest)
       if(ytest.le.1.e-8) goto 4031
       write(6,4035) xtest,m01,ig,ij,(fadjtme(n1,m01,ig,ij),n1=1,
      * 2*nazimm(m01))
4035   format(5x,'xtest = ',e12.7,' m01,ig,ij = ',3i8,/,(5x,6f12.8))
       stop
4031   continue
4020   continue
```

```
      write(6,184)mi0,ig0
184   format(//,5x,' forward probability distribution tables for inciden
     *t theta',i4,' incident energy group',i4)
c
      do 32 ij=1,3
      do 32 ig=ig0,ig0
      do 32 mil=1,5
      write(6,185)mil,ij,ig,(fadjtme(nazi,mil,ig,ij),nazi=1,
     *2*nazimm(mil))
32    continue
c
      do 931 mil=1,5
      write(7) (((fadjtme(nazi,mil,ig,ij),nazi=1,2*nazimm(mil)),
     *ij=1,maxija),ig=ig0,20)
 931    continue
c
 185  format(/,5x,'azimuthal distribution for eta level',i3,' region',i4
     *,' energy group',i4,/,(5e14.5))
c
c     end ig0 loop
c
  5     continue
c
c     end mi0 loop
c
  4     continue
c
      do 10001 ml=1,5
      do 10002 igl=1,20
c      clear work arrays
c
       fadjtot = 0.
c
       do 3001 igg = 1,20
       fadje(igg) = 0.
       fadjij(igg) = 0.
c
       do 3002 ik2 = 1,13
       fadjtx(ik2,igg) = 0.
       fadjty(ik2,igg) = 0.
3002    continue
c
       do 3003 ikl = 1,maxija
       fadjm(ikl,igg) = 0.
c
       do 3003 mkl = 1,5
       fadjtm(mkl,igg,ikl) = 0.
3003    continue
c
       fadjm(ikl,igg) = 0.
3001    continue
       do 925 i=1,20
```

```
925     totadj(i) = 0.
c
        do 1010 ig0=ig1,1,-1
c
c
c       begin loop to normalize the data and to determine overall
c       somation for each distribution
c
        do 2001 m01=1,5
        do 2001 i01=1,13
        do 2001 j01=1,13
c
c       add all outgoing component for a given eta, and energy incident
c
        xadj =  fadjeta(m1,ig1,m01,ig0,i01,j01)
        fadjtot = fadjtot + xadj
c
c       add all outgoing component for a given outgoing energy, and incident
c        eta and energy
c
        xadjij = fadjij(ig0)
        fadjij(ig0) = xadjij + xadj
c
c       define the position inside the colapsing array for angular distribution
c
        ij01 = i01 + (j01-1) * 13
        ij = nij(ij01)
        xadjm = fadjm(ij,ig0)
        fadjm(ij,ig0) = xadjm + xadj
c
c       end loop for outgoing directions, and positions for an outgoing energy
c
2001     continue
c
c       end loop for outgoing energies
c
1010     continue
c
c       write in the unit 6 the total albedo for incident energy and angle
c
        write(6,555)mi0,ig0,fadjtot
555     format(//,5x,'total albedo for mi0 = ',i5,' and ig0 = ',i5,' = ',
     * e20.10)
c
c       begin loop to construct cummulative probability distributions
c
        do 2020 ig0=ig1,1,-1
c
        xnormij=fadjij(ig0)
c
        do 2031 m0=1,5
        do 2031 i01=1,13
```

```
       do 2031 j01=1,13
c
       ij01 = i01 + (j01-1)*13
       ij=nij(ij01)
       xfadj0 =  fadjeta(m1,ig1,m0,ig0,i01,j01)
c
c      energy probability distribution
c
       xfadje = fadje(ig0)
       fadje(ig0) = xfadje + xfadj0 / fadjtot
c
c      position probability distribution
c
       xfadji = fadjtx(i01,ig0)
       xfadjj = fadjty(j01,ig0)
       xfadjm = fadjtm(m0,ig0,ij)
c
       fadjtx(i01,ig0) = xfadji + xfadj0 / xnormij
c
       fadjty(j01,ig0) = xfadjj + xfadj0 / xnormij
c
c
       fadjtm(m0,ig0,ij) = xfadjm + xfadj0 / fadjm(ij,ig0)
c
2031    continue
2020    continue
c
c   check if the probability distribution for each variable adds to 1.0
c
      xtest = 0.
      xtesti = 0.
      xtestj = 0.
      xtestm = 0.
      do 4001 ig=ig1,1,-1
      xtest = xtest + fadje(ig)
      xtesti = 0.
      do 4002 i=1,13
      xtesti = xtesti + fadjtx(i,ig)
4002  continue
      ytesti = xtesti - 1.0
      ytesti = abs(ytesti)
      if(ytesti.le.1.e-09) goto 4003
      write(6,4100)m1,ig1,ig,(fadjtx(i,ig),i=1,13)
4100  format(5x,'Problems with spatial distribution for x - m1,ig1,ig= '
     *,3i5,/,
     *(5x,5e12.5))
4003  continue
      xtestj = 0.
      do 4004 j=1,13
      xtestj = xtestj + fadjty(j,ig)
4004  continue
      ytestj = xtestj - 1.0
```

```
      ytestj = abs(ytestj)
      if(ytestj.le.1.e-09) goto 4005
      write(6,4101)m1,igl,ig,(fadjty(i,ig),i=1,13)
4101  format(5x,'Problems with spatial distribution for y - m1,igl,ig= '
     *,3i5,/,
     *(5x,5e12.5))
4005  continue
      do 4006 ij=1,maxij
      xtestm = 0.
      do 4007 meta=1,5
      xtestm = xtestm + fadjtm(meta,ig,ij)
4007  continue
      ytestm = xtestm - 1.0
      ytestm = abs(ytestm)
      if(ytestm.le.1.e-09) goto 4006
      write(6,4102)m1,igl,ig,ij,(fadjtm(meta,ig,ij),meta=1,5)
4102  format(5x,'Problems with spatial distribution for m - m1,igl,ig,ij
     * = '
     *,4i5,/,
     *(5x,5e12.5))
4006  continue
4001  continue
      ytest = xtesti - 1.0
      ytest = abs(ytest)
      if(ytest.le.1.e-09) goto 4008
      write(6,4103)(fadje(i),i=igl,1,-1)
4103  format(5x,'Problems with spatial distribution for m1,igl= ',2i5,/,
     *(5x,5e12.5))
4008  continue
      write(6,180)ig0,igm,(fadje(ig),ig=igl,1,-1)
c
      write(7) fadjtot, (fadje(ig),ig=igl,1,-1)
c
      do 2033 ig=igl,1,-1
      write(6,181)ig,(fadjtx(i,ig),i=1,13)
      write(6,182)ig,(fadjty(j,ig),j=1,13)
2033   continue
c
      write(7)(((fadjtx(i,ig),i=1,13),(fadjty(j,ig),j=1,13)),ig=igl,1,
     *-1)
c
      do 2032 ij=1,maxija
      do 2032 ig=igl,1,-1
      write(6,183)ij,ig,(fadjtm(m,ig,ij),m=1,5)
2032  continue
c
      write(7) (((fadjtm(leta,ig,ij),leta=1,5),ij=1,maxija),ig=igl,1,-1)
c
 180  format(/,5x,'energy distribution for group ',i5,' to group',i5,/,
     *(5e14.5))
 181  format(/,5x,'spatial distribution in x-direction for energy group'
     *,i4,/,(5e14.5,/,5e14.5,/,3e14.5))
```

```
 182  format(/,5x,'spatial distribution in y-direction for energy group'
     *,i4,/,(5e14.5,/,5e14.5,/,3e14.5))
 183  format(/,5x,'polar distribution for region',i4,' energy group'
     *,i4,/,(5e14.5))
10002 continue
10001 continue
      end
      block data
c
      common /datas/ w(70),eta(70),xmu(70),energy(37),veloc(37),
     * ntheti,inrgys,nmgpa,maxalb,nazim(5),etain(5),nazimm(5)
c
      data ntheti,inrgys,nmgpa,maxalb/ 5,20,20,1/
c
      data nazim/3,5,7,9,11/
c
      data nazimm/2,4,6,8,10/
c
      data etain/-.96225,-.83887,-.69389,-.50918,-.19245/
c
c      data etaout/.96225,.83887,.69389,.50918,.19245/
c
      data w/0.,115005.e-7,115005.e-7,0.,873399.e-8,873399.e-8,
     * 873399.e-8,873399.e-8,0.,6173.e-6,652549.e-8,6173.e-6,6173.e-6,
     * 652549.e-8,6173.e-6,0.,873399.e-8,652549.e-8,652549.e-8,
     * 873399.e-8,873399.e-8,652549.e-8,652549.e-8,873399.e-8,0.,
     * 115005.e-7,873399.e-8,6173.e-6,873399.e-8,115005.e-7,115005.e-7,
     * 873399.e-8,6173.e-6,873399.e-8,115005.e-7,
     * 0.,115005.e-7,115005.e-7,0.,873399.e-8,873399.e-8,
     * 873399.e-8,873399.e-8,0.,6173.e-6,652549.e-8,6173.e-6,6173.e-6,
     * 652549.e-8,6173.e-6,0.,873399.e-8,652549.e-8,652549.e-8,
     * 873399.e-8,873399.e-8,652549.e-8,652549.e-8,873399.e-8,0.,
     * 115005.e-7,873399.e-8,6173.e-6,873399.e-8,115005.e-7,115005.e-7,
     * 873399.e-8,6173.e-6,873399.e-8,115005.e-7/
c
       data xmu/  -27217.e-5,-19245.e-5, 19245.e-5,-54433.e-5,
     * -50918.e-5,-19245.e-5, 19245.e-5, 50918.e-5,-72008.e-5,
     * -69389.e-5,-50918.e-5,-19245.e-5, 19245.e-5, 50918.e-5,
     *  69389.e-5,-86066.e-5,-83887.e-5,-69389.e-5,-50918.e-5,
     * -19245.e-5, 19245.e-5, 50918.e-5, 69389.e-5, 83887.e-5,
     * -98131.e-5,-96225.e-5,-83887.e-5,-69389.e-5,-50918.e-5,
     * -19245.e-5, 19245.e-5, 50918.e-5, 69389.e-5, 83887.e-5,
     *  96225.e-5,
     * -27217.e-5,-19245.e-5, 19245.e-5,-54433.e-5,
     * -50918.e-5,-19245.e-5, 19245.e-5, 50918.e-5,-72008.e-5,
     * -69389.e-5,-50918.e-5,-19245.e-5, 19245.e-5, 50918.e-5,
     *  69389.e-5,-86066.e-5,-83887.e-5,-69389.e-5,-50918.e-5,
     * -19245.e-5, 19245.e-5, 50918.e-5, 69389.e-5, 83887.e-5,
     * -98131.e-5,-96225.e-5,-83887.e-5,-69389.e-5,-50918.e-5,
     * -19245.e-5, 19245.e-5, 50918.e-5, 69389.e-5, 83887.e-5,
     *  96225.e-5/
c
```

```
      data eta/ 96225.e-5, 96225.e-5, 96225.e-5,
     *  83887.e-5,  83887.e-5,  83887.e-5,  83887.e-5,  83887.e-5,
     *  69389.e-5,  69389.e-5,  69389.e-5,  69389.e-5,  69389.e-5,
     *  69389.e-5,  69389.e-5,
     *  50918.e-5,  50918.e-5,  50918.e-5,  50918.e-5,  50918.e-5,
     *  50918.e-5,  50918.e-5,  50918.e-5,  50918.e-5,
     *  19245.e-5,  19245.e-5,  19245.e-5,  19245.e-5,  19245.e-5,
     *  19245.e-5,  19245.e-5,  19245.e-5,  19245.e-5,  19245.e-5,
     *  19245.e-5,
     *  96225.e-5, 96225.e-5, 96225.e-5,
     *  83887.e-5, 83887.e-5, 83887.e-5, 83887.e-5, 83887.e-5,
     *  69389.e-5, 69389.e-5, 69389.e-5, 69389.e-5, 69389.e-5,
     *  69389.e-5, 69389.e-5,
     *  50918.e-5, 50918.e-5, 50918.e-5, 50918.e-5, 50918.e-5,
     *  50918.e-5, 50918.e-5, 50918.e-5, 50918.e-5,
     *  19245.e-5, 19245.e-5, 19245.e-5, 19245.e-5, 19245.e-5,
     *  19245.e-5, 19245.e-5, 19245.e-5, 19245.e-5, 19245.e-5,
     *  19245.e-5/
c     velocities should be calculated but they are not!
      data veloc/37*1./
c
      end
```

## THE ALBDO3D SUBROUTINE OF THE STORM PACKAGE

```
        subroutine  albdo3d(ig,u,v,w,wate,nmed,nreg,mark,x,y,z)
c
c *****
c *****
c ***** no debugging has been done on provisions for gamma-rays (as seco
c *****
c *****
c
      common /nxt/ mxalb,locpsd,lprpsd,lpepsd,ialb,dokill,un0,vn0,wn0,
     * x0,y0,z0,ncos,mregbs,iblznt,ximp,yimp,zimp,kmizxt,ndum,iiblzn,
     * locneu,locnne,locggm,locngm,ialbwt,ialbcs,ialbrg,ialbeg,jyl
     *,nduct,ndct(50),uaxis(50),vaxis(50),waxis(50),irrprl,imb,jmb,nxy
c
      common /albcon/ ntheti, inrgys, nmgpa,  nino,   ninonp, igamg,
     1 lsumenad,igalbold, locice, locenr, locvel, loccor, locthi, lntho,
     2  idnt, lsumen, lsntho, loctho, locnp,  lsnpki, lsnpi,  locphi,
     3 locnij, locmxy, loctot, loccen, locpxy, locpet, locpaz, locadj,
     4 lsumadj,maxija, ialbdo, maxalb, lenalb, medal(10),icomb,lsumenr
c
```

```
      common /user/   agstrt, wtstrt, xstrt,  ystrt,  zstrt,  dff,
     1                ebotn,  ebotg,  tcut,   i0,     i1,     iadjm,
     2                ngpqt1, ngpqt2, ngpqt3, ngpqtg, ngpqtn, nits,
     3                nlast,  nleft,  nmgp,   nmtg,   nstrt
      common/locsig/istart,isccog,inabog,igabog,ifporg,ifngp,ifspog,
     1       idsgog,iprbng,iprbgg,iscang,iscagg,isporg,isport,inpbuf,
     2       isigog,infpog,iabsog,itotsg,ngp,nds,ngg,ndsg,ingp,inds,
     3      nxxmed,nelem,nmix,ncoef,nsct,nts,ntg,ndsngp,ndsngg,iadj,
     4 nme,loc,ings,insg,iixxl,iixx0, kkk,ixtape,idel,iteml,itemg,irsg,
     5       irdsg,istr,iprin,ifmu,imom,idtf,istat,ipun
     6       ,nus,ngn,iht,inus,inusn,ingn,ingnp,innn,iggg
      common /normal/ un, vn, wn
      common bc(1)
      dimension nc(1)
      dimension biasenr(200)
      equivalence (bc(1),nc(1))
      data nnegt/0/, nnegp/0/, acons/1.e-4/
      data im,jm/13,13/
c
c * * iin = index of incident polar angle
c * *   k = index of reflected polar angle
c * *   l = index of reflected azimuthal angle
c
      igold = ig
      igalb = ig
      if(iadjm.ne.0) igalb = nmtg - ig + 1
      if (locice .ne. locenr) igalb = nc(locice + ig)
      if (locice .ne. locenr .and. iadjm .ne. 0)
     * igalb = nc(locice + nmtg - ig + 1)
      if(ig.le.13) write(6,733) igalb
733   format(5x,' entering albdo - igalb was set to = ',i5)
      ngpn=ngpqtn
      ngpg=ngpqtg
      if (nmgp.gt.ngpqtn)ngpn=ngpqtn+1
      if ((nmtg-nmgp).gt.ngpqtg .and. ngpqtg.ne.0)ngpg=ngpqtg+1
      if (locice .eq. locenr) goto 8
      igmaxn = nc(locice + ngpn)
      igmaxg = nc(locice + nmgp + ngpg)
      iggaml = nc(locice + nmgp)
      kcal = kcal + 1
      if(kcal.gt.1) goto 5
      if(loc.le.0) goto 5
      indd = loc+ntg*(nreg-1)
c
c   this procedure works only if the energy biasing values are given
c   for a single region.
c
      do 7211 ijkl = 1,ntg
      if(iadjm.eq.0) biasenr(nc(locice+ijkl)) = biasenr(nc(locice+ijkl))
     * + bc(indd+ijkl)
      if(iadjm.ne.0) biasenr(nc(locice+nmtg-ijkl+1))
     * = biasenr(nc(locice+nmtg-ijkl+1)) + bc(indd+ijkl)
```

```
7211  continue
      goto 5
c
  8   igmaxn = ngpn
      igmaxg = nmgp + ngpg
      iggaml = nmgp
c
  5   call thet0(iin,4halbo,u,v,w,dumy,scosto)
      mark=-1
c
c     now calculate the componentes of the vector in the albedo plane
c
      ul=u-scosto*un
      vl=v-scosto*vn
      wl=w-scosto*wn
      fnorm=sqrt(ul*ul+vl*vl+wl*wl)
c
c     now normalize the vector
c
      ul=ul/fnorm
      vl=vl/fnorm
      wl=wl/fnorm
c
      lsumen = 0
      lsumadj = 0
      if(igalb.eq.1) goto 442
c
c     now select the new energy group - ICG 10/15/90
c
      do 502 igen = 1,igalb-1
502   lsumen = lsumen + (inrgys - igen + 1)
      do 2502 igen = 1,igalb-1
2502  lsumadj = lsumadj + igen
442   continue
      iaenr = loccen + lsumenr * (iin - 1) + lsumen
      if(iadjm.ne.0)iaenr = loccen + lsumenr * (iin - 1) + lsumadj
      ren = fltrnf(ren)
      inr = 0
      inn = 0
      if(iadjm.eq.0) goto 1503
      if(loc.eq.0) goto 7213
      sbsig = 0.
      ix = loc + ntg*(nreg-1)
      do 7212 ijkl = igalb,1,-1
      inn = inn + 1
c
c this procedure works with a single region energy biasing parameters.
c
      bsig = bc(iaenr+inn)*biasenr(ijkl)
7212  sbsig = sbsig + bsig
      ren = ren * sbsig
7213  bx = 1.
```

```
      do 2503 igout = igalb,1,-1
      inr = inr + 1
      if(loc.ne.0) bx = biasenr(igout)
      ren = ren - bc(iaenr + inr)*bx
      if(ren.le.0) goto 504
2503  continue
      ione = 1
      write(6,3502)iaenr,igalb,ione,(bc(iaenr+inr),
     *inr=igalb,1,-1)
3502  format(5x,'problems with outgoing energy probabilities',/,5x,
     *'iaenr,igalb,inrgys =',3i10,/,5x,'bc(iaenr+inr) =',/,(2x,7f10.6))
      stop
1503  continue
      if(loc.eq.0) goto 7215
      sbsig = 0.
      ix = loc + ntg*(nreg-1)
      do 7214 ijkl = igalb,inrgys
      inn = inn + 1
c
c this procedure works with a single region energy biasing parameters.
c
      bsig = bc(iaenr+inn)*biasenr(ijkl)
7214  sbsig = sbsig + bsig
      ren = ren * sbsig
7215  bx = 1.
      do 503 igout = igalb,inrgys
      inr = inr + 1
      if(loc.ne.0) bx = biasenr(igout)
      ren = ren - bc(iaenr + inr) * bx
      if(ren.le.0) goto 504
503   continue
      write(6,1502)iaenr,igalb,inrgys,(bc(iaenr+inr),
     *inr=1,(inrgys-igalb+1))
1502  format(5x,'problems with outgoing energy probabilities',/,5x,
     *'iaenr,igalb,inrgys =',3i10,/,5x,'bc(iaenr+inr) =',/,(2x,7f10.6))
      stop
504   igalbold = igalb
      if(loc.ne.0) wate = wate * sbsig/bx
      igalb = igout
      ig = igalb
      if (locice .eq. locenr) go to 105
c
c * igalb must be converted to xsec structure as ig * * * * * * *
c
      npg = 0
      if(iadjm.eq.0) goto 1081
      do 1080 ige=igold,nmtg
      igex = nmtg - ige + 1
      if (nc(locice+igex) .ne. igalb) go to 1080
      if (npg .eq. 0) nfpg = ige
      npg = npg + 1
1080  continue
```

```
      goto 1082
1081  continue
      do 80 ige=igold,nmtg
      if (nc(locice+ige) .ne. igalb) go to 80
      if (npg .eq. 0) nfpg = ige
      npg = npg + 1
 80   continue
1082  continue
      if(npg.gt.1) goto 1083
      ig = nfpg
      goto 105
1083  continue
c
c * npg is no. of possible xsec grps corresponding to igalb based on ini
c   locenr is zero address of albedo energy array
c
      indx = locenr + igalb
c
c * select a value for the actual energy
c
      r = fltrnf(0)
      if (icomb .ne. 0  .and.  igalb .gt. nmgpa) go to 85
c
c * this selection is uniform in lethargy from the lower limit, bc(indx+
c   to the upper limit, bc(nfpg)
c
      energy = bc(indx+1)*(bc(nfpg)/bc(indx+1))**r
      if(iadjm.ne.0) energy = bc(indx+1)*(bc(nfpg+npg-1)/bc(indx+1))**r
      go to 90
 85   indx = indx + 1
c
c * * this selection is uniform in energy, same limits as above
c
      energy = bc(indx+1) + r*(bc(nfpg)-bc(indx+1))
 90   ng = nfpg + npg - 1
      if (iadjm.ne.0) ng = nfpg + npg - 1
c
c  * now which xsec energy does it correspond to?
c
      if(iadjm.eq.0) goto 1100
      do 1095 i=ng,nfpg,-1
 1095 if (energy .gt. bc(i-1)) go to 1105
1105  ig = i
      goto 105
1100  continue
      do 95 i=nfpg,ng
 95   if (energy .gt. bc(i)) go to 100
 100  ig = i - 1
 105  continue
      ial = locpxy + (im + jm)*lsumenr*(iin - 1) + (im + jm) *
     * (igalb - igalbold) + lsumen * (im + jm)
      if(iadjm.ne.0) ial = locpxy + (im + jm)*lsumenr*(iin - 1) +
```

```
     * (im + jm) * (igalbold - igalb) + lsumadj * (im + jm)
      rxy = fltrnf(rxy)
      do 505 ixx = 1,im
      rxy = rxy - bc(ial +ixx)
      if(rxy.le.0) goto 506
505   continue
      write(6,4503)ial,ixx,(bc(ial+ixy),ixy=1,im),igalb,igalbold
4503  format(5x,'problems with the spatial probability distribution',5x,
     *'ial,ixx=',2i10,' bc(ial+ixy),ixy=1,im ==',/,2x,7f10.6,2x,
     *6f10.6,2i5)
      stop
506   rxy = fltrnf(r)
      do 507 iyy = 1,jm
      rxy = rxy - bc(ial + im + iyy)
      if(rxy.le.0) goto 508
507   continue
      write(6,1504)ial,iyy,(bc(ial+iyx+im),iyx=1,jm)
1504  format(5x,'problems with the spatial probability distribution',5x,
     *'ial,iyy=',2i10,' bc(ial+iyx+im),iyx=1,im ==',/,5x,(2x,7f10.6))
      stop
 508   delx0 = bc(locmxy + ixx + 1) - bc(locmxy + ixx)
       jy1 = iyy
       nxy = nc(locnij + im*(iyy-1) + ixx )
       del1 = delx0 * fltrnf(r)
      delx = bc(locmxy + ixx) + del1 - (bc(locmxy+im+1)-bc(locmxy+1))/2.
       dely0 = bc(locmxy + iyy + im + 2) - bc(locmxy + iyy + im + 1)
       del2 = dely0 * fltrnf(r)
       dely = bc(locmxy + iyy + im + 1) + del2 - (bc(locmxy+im+jm+2)-
     *bc(locmxy+im+2))/2.
 1235  delsq = delx**2 + dely**2
       displ = sqrt (delsq)
      u2 = vn*w1 - wn*v1
      v2 = wn*u1 - un*w1
      w2 = un*v1 - vn*u1
      fnorm = sqrt(u2*u2 + v2*v2 + w2*w2)
      u2 = u2/fnorm
      v2 = v2/fnorm
      w2 = w2/fnorm
      ua = -u1 * delx - u2 * dely
      va = -v1 * delx - v2 * dely
      wa = -w1 * delx - w2 * dely
      fnorm = sqrt(ua*ua + va*va + wa*wa)
      ua = ua/fnorm
      va = va/fnorm
      wa = wa/fnorm
      mysign = 1
c
 855  call gomes(mark,ua,va,wa,displ,mysign)
c
      if(mark.eq.(-1)) goto 706
c
      if(mark.eq.1.or.mark.eq.2) goto 705
```

```
c
      if(mark.eq.3) write(6,707)x,y,z
c
c     don't forget - to implement this is only necessary to call
c     specular reflection inside gomes !!!  ICG - 10/15/90
c
707   format(5x,'*** exit called by albdo ****',/,5x,
     *'*** a specular reflective surface was encountered during the disp
     *lacement of ***',/,5x,'*** the point of emergence at an albedo eve
     *nt ***',/,5x,' x= ',e15.7,'  y= ',e15.7,'  z= ',e15.7)
c
      stop
c
705   imin = (iin-1)*nmgpa
c
      wateold = wate
      wate = wate*bc(loctot + (ialbdo-1)*lenalb + igalbold + imin)
      jmo = igalbold - 1
      ini = inrgys + 1
      nthets = nc(lsntho + iin)
      ntheto = nc(lntho + iin)
      idnt = locpet + (ialbdo - 1)*lenalb + lsumenr*nthets*maxija +
     * lsumen*maxija*ntheto + (igalb - igalbold)*maxija*ntheto +
     * (nxy - 1)*ntheto
      if(iadjm.ne.0)
     * idnt = locpet + (ialbdo - 1)*lenalb + lsumenr*nthets*maxija +
     * lsumadj*maxija*ntheto + (igalbold - igalb)*maxija*ntheto +
     * (nxy - 1)*ntheto
      ra = fltrnf(ra)
      do 10 k=1,ntheto
      ra = ra - bc(idnt+k)
      if(ra.le.0) goto 15
10    continue
      write(6,771)ra,idnt,ntheto,(bc(idnt+kk),kk=1,ntheto)
771   format(5x,'problems in the cumulative distribution for theta',/,
     *5x,'ra = ',f12.7,' idnt = ',i15,' ntheto = ',i5,' bc(idnt+kk) ='/,
     * (5x,6f10.6))
      stop
  15   indth = loctho + nthets + iin + k - 1
       coso = bc(indth) + fltrnf(ra)*(bc(indth+1)-bc(indth))
       sino = sqrt(1. - coso*coso)
       nsnpi = nc(lsnpi + iin)
       if(iadjm.ne.0) goto 6333
       ind1 = nc(lsnpki + nthets + iin + ntheto )
       ind2 = nc(lsnpki + nthets + iin + k - 1 )
       ind3 = 2*(nc(locnp + k) - 1)
       indp = locpaz+(ialbdo-1)*lenalb+lsumenr*(nsnpi)*maxija+
     * lsumen*ind1*maxija +
     * (inrgys - igalbold +1)*maxija*ind2 + (igalb - igalbold)*maxija
     * * ind3 + (nxy-1)*ind3
       nphio = nc( locnp + nthets + k ) - 1
       goto 6321
```

```
6333    lsumenad = lsumen
        if(igalb.eq.igalbold) goto 6323
        do 6322 ix = igalb,igalbold-1
        lsumenad = lsumenad - (inrgys - ix + 1)
6322    continue
6323    continue
        nsnpiadj = nc(lsnpi+k)
        nthetoad = nc(lntho+k)
        nthetsad = nc(lsntho + k)
        indladj = nc(lsnpki + nthetsad + k + nthetoad)
        ind2adj = nc(lsnpki + nthetsad + k + iin - 1)
        ind3adj = 2*(nc(locnp + iin) - 1)
        indp = locpaz + (ialbdo-1)*lenalb + lsumenr*(nsnpiadj)*maxija+
     * lsumenad*indladj*maxija +
     * (inrgys - igalb +1)*maxija*ind2adj + (igalbold - igalb)*maxija
     * * ind3adj + (nxy-1)*ind3adj
        nphio = nc( locnp + nthetsad + iin ) - 1
        do 30 l = 1,2*nphio
        ra = ra - bc(indp+l)
        if(ra.le.0) goto 35
30      continue
        write(6,772) igalb,igalbold,ig,igold,
     * ra,nphio,indp,indl,ind2,(bc(indp+ll),ll=1,2*nphio)
772     format(5x,'problems with the azimuthal probability distribution',
     * /,5x,' igalb= ',i3,' igalbold = ',i3,' ig = ',i3,' igold = ',i3,
     * /,5x,'ra= ',f12.7,' nphio= ',i5,' indp= ',i12,' indl= ',i12,
     * ' ind2= ',i12,/,5x,'bc(indp+ll),ll=1,nphio) =',/,(5x,6f11.8))
        stop
  35    kjump=0
        lll = 1
        if(l.gt.nphio .and. jyl.le.7) kjump = 70
        if(l.le.nphio .and. jyl.gt.7) kjump = 70
        if(l.gt.nphio) lll = 1 - nphio
        lp = locphi + ind2 + 2*(lll + k - 1) + kjump
        if(iadjm.ne.0) lp = locphi + ind2adj + 2*(lll + iin - 1) + kjump
        rphi = fltrnf(ra)
        cosphi = bc(lp) + rphi*(bc(lp+2) - bc(lp))
        sinphi = -sqrt(1. - cosphi*cosphi)
        if(l.gt.nphio .and. jyl.le.7) sinphi = -sinphi
        if(l.le.nphio .and. jyl.gt.7) sinphi = -sinphi
      ssint  = sqrt(1.0-scosto**2)
      if (ssint - acons) 115,110,110
 110  a = u - scosto*un
      b = v - scosto*vn
      c = w - scosto*wn
      go to 130
 115  call gtiso(a,b,c)
 120  dotn = a*un + b*vn + c*wn
      a = a - dotn*un
      b = b - dotn*vn
      c = c - dotn*wn
      if (abs(a)+abs(b)+abs(c)-acons) 115,115,125
```

```
 125  dmag = sqrt(a*a + b*b + c*c)
      a=a/dmag
      b=b/dmag
      c=c/dmag
      ssint =1.0
c
c  *  * select direction cosines and normalize
c
 130  u = coso*un + sino*cosphi*a/ssint  + sino*sinphi*(c*vn-b*wn)/ssint
      v = coso*vn + sino*cosphi*b/ssint  + sino*sinphi*(a*wn-c*un)/ssint
      w = coso*wn + sino*cosphi*c/ssint  + sino*sinphi*(b*un-a*vn)/ssint
      fnorm = sqrt(u*u+v*v+w*w)
      u= u/fnorm
      v=v/fnorm
      w= w/fnorm
706   continue
      return
        end
```

## THE ALBDOE3D SUBROUTINE OF THE STORM PACKAGE

```
       subroutine albdoe3d
c
      common /nxt/ mxalb,locpsd,lprpsd,lpepsd,ialb,dokill,un0,vn0,wn0,
     * x0,y0,z0,ncos,mregbs,iblznt,ximp,yimp,zimp,kmizxt,ndum,iiblzn,
     * locneu,locnne,locggm,locngm,ialbwt,ialbcs,ialbrg,ialbeg,jyl
     * ,nduct,ndct(50),uaxis(50),vaxis(50),waxis(50),irrprl,im,jm,nxy
c
      common /albcon/ ntheti, inrgys, nmgpa,  nino,   ninonp, igamg,
     1 lsumenad,igalbold, locice, locenr, locvel, loccor, locthi, lntho,
     2  idnt, lsumen, lsntho, loctho, locnp,  lsnpki, lsnpi,  locphi,
     3 locnij, locmxy, loctot, loccen, locpxy, locpet, locpaz, locadj,
     4 lsumadj,maxija, ialbdo, maxalb, lenalb, medal(10),icomb,lsumenr
c
      common /user/   agstrt, wtstrt, xstrt,  ystrt,  zstrt,  dff,
     1                ebotn,  ebotg,  tcut,   i0,    iread,  iadjm,
     2                ngpqt1, ngpqt2, ngpqt3, ngpqtg, ngptqn, nits,
     3                nlastu, nleft,  nmgp,   nmtg,   nstrt
c
c *** this routine for use with ALBFAC3D albedo tapes ****
c *****
c *****
c ***** no debugging has been done on provisions for gamma-rays (as
c *****        secondaries)
```

```
c *****
      common /pdet/   nd,     nne,    ne,     nt,     na,     nresp,
     1                nex,    nexnd,  nend,   ndnr,   ntnr,   ntne,
     2                nane,   ntndnr, ntnend, nanend, locrsp, locxd,
     3                locib,  locco,  loct,   locud,  locsd,  locqe,
     4                locqt,  locqte, locqae, lmax,   efirst, egtop
      common /nutron/ name,   namex,  ig,     igo,    nmed,   medold,
     1                nreg,   u,      v,      w,      uold,   vold,
     2                wold,   x,      y,      z,      xold,   yold,
     3                zold,   wate,   oldwt,  wtbc,   blznt,  blzon,
     4                age,    oldage
      common /normal/ unor,   vnor,   wnor
      common /check/igcheck
      dimension wout(70), nazim(5)
      common bc(1)
      dimension nc(1)
      equivalence (bc(1),nc(1))
      data numtg1/0/, numtl0/0/, numpg1/0/, pn/1./
c
      data nazim/2,4,6,8,10/
      data wout/0.,115005.e-7,115005.e-7,
     * 0.,873399.e-8,873399.e-8,873399.e-8,873399.e-8,
     * 0.,6173.e-6,652549.e-8,6173.e-6,6173.e-6,652549.e-8,6173.e-6,
     * 0.,873399.e-8,652549.e-8,652549.e-8,
     * 873399.e-8,873399.e-8,652549.e-8,652549.e-8,873399.e-8,
     * 0.,115005.e-7,873399.e-8,6173.e-6,873399.e-8,115005.e-7,
     * 115005.e-7,873399.e-8,6173.e-6,873399.e-8,115005.e-7,
     * 0.,115005.e-7,115005.e-7,0.,873399.e-8,873399.e-8,
     * 873399.e-8,873399.e-8,0.,6173.e-6,652549.e-8,6173.e-6,6173.e-6,
     * 652549.e-8,6173.e-6,0.,873399.e-8,652549.e-8,652549.e-8,
     * 873399.e-8,873399.e-8,652549.e-8,652549.e-8,873399.e-8,0.,
     * 115005.e-7,873399.e-8,6173.e-6,873399.e-8,115005.e-7,115005.e-7,
     * 873399.e-8,6173.e-6,873399.e-8,115005.e-7/
c
c * nex must be at least 2 to make room for an array to be filled with
c   no. of m.f.p.'s to the detector, and an array with the cdf's used to
c   select an energy for the estimate
c
       if(ialb.le.0) goto 2233
      write(6,2234) ialb
2234  format(5x,'*** ERROR in ALBDOE - ialb.gt.0 -- ialb= ',i5)
      stop
2233  iarg = locrsp + nresp*nmtg
c
c the index of cell zero of the first of the nex arrays
c
      iepr = iarg + nmtg
c
c the index of cell zero of the second array
c
      igquit = ngpqtl
      if(ig.gt.ngpqtl) igquit = ngpqt3
```

```
      igalb = ig
      iquita = igquit
      if (locice .eq. locenr) go to 10
      igalb = nc(locice + ig)
      iquita = nc(locice+igquit)
      if(iadjm.eq.0) goto 10
      igalb = nc(locice + nmtg - ig + 1)
      iquita = nc(locice+ nmtg - igquit + 1)
 10   un = - unor
      vn = - vnor
      wn = - wnor
c
c * * call thet0 to determine incoming polar angle bib (i),cosph,costho
c
      call thet0(i,4halbe,uold,vold,wold,cosph,costho)
      a = vn*wold-wn*vold
      b = wn*uold-un*wold
      c = un*vold-vn*uold
c
c * * cross product between incoming neutron direction and inward normal
c
      dl = sqrt(a*a + b*b + c*c)
      mmax = inrgys - igalb + 1
      mmaxe = igquit - ig + 1
      jmo = igalb - 1
      inid2 = (inrgys + 1)*inrgys/2
      igp = ig + 1
c
c ***
c   begin detector loop
c ***
c
      do 150 m=2,2
      if(ig.ne.46) goto 150
      id = locxd + m
      xe = bc(id)
      ye = bc(id+nd)
      ze = bc(id+2*nd)
      xl = x
      yl = y
      zl = z
      xa = xe - xl
      ya = ye - yl
      za = ze - zl
      td2 = xa*xa + ya*ya + za*za
      td = sqrt(td2)
c
c **                                  direction cosines to detector
c
      alf = xa/td
      bet = ya/td
      gam = za/td
```

```
c
c*  *
c
      costht = unor*alf + vnor*bet + wnor*gam
c
c * * dot product of vector to detector and outward normal
c * *  0 .lt. costht .lt. 1.0
c
      if (costht - 1.0) 15,15,20
 15   if (costht) 25,30,30
 20   numtgl = numtgl + 1
      if (numtgl .le. 5) write (i0,1010) costht
 1010 format (1h0,' costht .gt. 1.0 in albdoe',e15.7)
      costht = 1.0
      cosphi = 0.0
      go to 75
 25   numtl0 = numtl0 + 1
      if (numtl0 .le. 5) write (i0,1015) costht
 1015 format ('0 costht negative in albdoe',e15.7)
      costht = 0.0
      cosphi = 0.0
c
c: with present data set, cannot contribute, so skip this detector.
c
      go to 150
      if(iadjm.ne.0) bc(iarg+1) = 0
 30   do 35 j=ig,igquit
 35   bc(iarg+j) = 0.0
      mark = 1
      ntd = 1
      lastmd = nmed
      medium=nmed
      ds = td
 40   arg = 0.0
c
c * * call euclid to calculate arg
c
      igxx = ig
 45   continue
      call euclid(mark,xl,yl,zl,xe,ye,ze,td,igxx,arg,ntd,medium,blznt
     1     ,nreg)
      if (lastmd .eq. 1000) go to 55
      bc(iarg + igxx) = bc(iarg + igxx) + arg
      if (bc(iarg+igxx) .lt. -64.0) go to 150
      if (igp .gt. igquit) go to 55
 55   lastmd = medium
      markd = mark + 3
      go to (40,60,40,60), markd
 60   continue
      if(cosph)70,65,70
 65   cosphi = 0.0
      go to 75
```

```
 70   continue
      ap = vn*gam - wn*bet
      bp = wn*alf - un*gam
      cp = un*bet - vn*alf
c
c * * cross product between vector to detector and inward normal
c
      d2 = sqrt(ap*ap + bp*bp + cp*cp)
      cosphi = (a*ap+b*bp+c*cp)/(d1*d2)
c
c * * following is the cross product between the two directions
c     normal to the normal
c
      ua = alf - costht * unor
      va = bet - costht * vnor
      wa = gam - costht * wnor
      fnorm = sqrt( ua*ua + va*va + wa*wa )
      ua = ua/fnorm
      va = va/fnorm
      wa = wa/fnorm
      sinphi = (ua*a + va*b + wa*c)/d1
c
c * *  -1.0 .lt. cosphi .lt. 1.0
c
      if (abs(cosphi) .le. 1.0) go to 75
      numpgl = numpgl + 1
      if (numpgl .le. 5) write (i0,1020) cosphi
 1020 format ('0 cosphi .gt. 1.0 in albdoe',e15.7)
      cosphi = sign(1.0,cosphi)
 75   ntheto = nc(lntho+i)
      nthets = nc(lsntho+i)
      k = loctho + nthets + i
c
c * * index of theta(1,i)
c
      wtlow = 1.0
      pup = bc(k+1)
      pave = (bc(k)+pup)/2.
      if (costht - pave) 80,100,100
 80   nom = ntheto - 1
      do 85 kt = 1,nom
      k = k + 1
      pbot = pup
      pup = bc(k+1)
      poa = pave
      pave = (pbot + pup)/2.
      if (costht - pave) 85,90,90
 85   continue
      wtlow = costht/pave
      go to 100
 90   if((pave-poa).eq.0.) write(6,937) k,pave,poa,pbot,pup
937   format(5x,'(pave-poa) eq zero - k= ',i7,' pave= ',f12.7,' poa= ',
```

```
     *f12.7,' pbot= ',f12.7,' pup= ',f12.7)
      form = (costht - poa)/(pave - poa)
      if (form - fltrnf(ra)) 95,100,100
 95   k = k - 1
 100  k = k - loctho - nthets - i + 1
      nsnpi = nc(lsnpi+i)
      nthetoad = nc(lntho+k)
      nthetsad = nc(lsntho+k)
      ind1 = nc(lsnpki+nthets+ntheto+i)
      ind2 = nc(lsnpki+nthets+k+i-1)
      indladj = nc(lsnpki + nthetsad + nthetoad + k)
      ind2adj = nc(lsnpki + nthetsad + i + k - 1)
       kjump = 0
       if(sinphi .lt. 0 .and. jy1.gt.7) kjump = 70
       if(sinphi .ge. 0 .and. jy1.le.7) kjump = 70
       l = locphi + ind2 + 2*(k-1) + kjump + 1
       if(iadjm.ne.0) l = locphi + ind2adj + 2*(i-1) + kjump + 1
c
c * * index of phi(l,k,i)
c
      pbot = bc(l)
      pup = bc(l+2)
      pave = (pbot + pup)/2.
      if (pave - cosphi) 105,125,125
 105  lm = nc(locnp+nthets+k) - 1
      if(iadjm.ne.0) lm = nc(locnp+nthetsad+i) - 1
c
c * * index of nphi(k,i) - 1
c
      do 110 lt=1,2*lm
      l = l + 2
      pbot = pup
      pup = bc(l+2)
      poa = pave
      pave = (pbot + pup)/2.
      if (pave - cosphi) 110,115,115
 110  continue
      go to 125
 115  if((pave-poa).eq.0) write(6,935) l,pave,poa,pbot,pup
935   format(5x,'(pave-poa) eq zero - l= ',i8,' pave= ',f12.7,' poa= ',
     *f12.7,' pbot= ',f12.7,' pup= ',f12.7)
      form = (cosphi - poa)/(pave - poa)
      if (form - fltrnf(ra)) 120,125,125
 120  l = l - 2
 125  if(iadjm.eq.0) l = l - locphi - ind2 - 2*(k-1) - kjump - 1 + 2
      if(iadjm.ne.0) l = l - locphi - ind2adj - 2*(i-1) - kjump - 1 + 2
      indx = idnt++ k
      pi = 4. * atan(1.)
      ind2 = nc(lsnpki+nthets+k+i-1)
      ind2adj = nc(lsnpki + nthetsad + i + k - 1)
      ind3 = 2*(nc(locnp + k ) - 1)
      ind3adj = 2*(nc(locnp + i) - 1)
```

```
       inde = locpaz   (ialbdo-1)*lenalb + lsumenr*nsnpi*maxija +
      * lsumen*indl*maxija + (inrgys - igalbold + 1)*maxija*ind2 +
      * (igalb - igalbold)*maxija*ind3 + (nxy - 1)*ind3
       if(iadjm.ne.0)
      * inde = locpaz + (ialbdo-1)*lenalb + lsumenr*nsnpi*maxija +
      * lsumenad*indladj*maxija + (inrgys - igalb + 1)*maxija*ind2adj +
      * (igalbold - igalb)*maxija*ind3adj + (nxy - 1)*ind3adj
       lll = 1
        if(sinphi .lt. 0 .and. jyl.gt.7) l = l + 2*nazim(k)
        if(sinphi .ge. 0 .and. jyl.le.7) l = l + 2*nazim(k)
       if(iadjm.ne.0) iquad = ind2adj/2 + i + lll/2
       wtquad = wout(iquad)
       if(wtquad.le.0.)write(6,993) wtquad,iquad,ind2,k,lll
993    format(5x,'wtquad eq zero - wtquad = ',e14.7,' iquad= ',i7,
     *' ind2= ',i7,' k=',i5,' lll= ',i5)
       cont = wate * bc(idnt+k) * bc(inde+l/2)/td2/wtquad*wtlow/(2.*pi)
       if(bc(iarg+igxx) .lt. -64.) go to 150
       con = cont*exp(bc(iarg+igxx))
 145   call fluxst(m,igxx,con,aged,gam,0)
       mmd=m-1
       call fluxst(mmd,igxx,con,aged,gam,0)
       igck = igcheck - 40
       call fluxst(igck,igxx,con,aged,gam,0)
c
c  *  *  switch =  0  -- store in all relevant arrays except ud
c     this assumes any angle analysis is w.r.t. z-axis (arg gam to fluxst
c
 150   continue
       return
       end
```

## THE SUBROUTINE ALBIN3D OF THE STORM PACKAGE

```
       subroutine  albin3d(nlast,nlf,i0)
c *** this routine for use with ALBFAC3D albedo tapes ******
c *****
c *****
c ***** no debugging has been done on provisions for gamma-rays (as
c *****        secondaries)
c *****
       common /nxt/ mxalb,locpsd,lprpsd,lpepsd,ialb,dokill,un0,vn0,wn0,
      * x0,y0,z0,ncos,mregbs,iblznt,ximp,yimp,zimp,kmizxt,ndum,iiblzn,
      * locneu,locnne,locggm,locngm,ialbwt,ialbcs,ialbrg,ialbeg,jyl
      * ,nduct,ndct(50),uaxis(50),vaxis(50),waxis(50),irrprl,im,jm,nxy
```

```
c
      common /albcon/ ntheti, inrgys, nmgpa,  nino,   ninonp, igamg,
     1lsumenad,igalbold, locice, locenr, locvel, loccor, locthi, lntho,
     2  idnt, lsumen, lsntho, loctho, locnp,  lsnpki, lsnpi,  locphi,
     3  locnij, locmxy, loctot, loccen, locpxy, locpet, locpaz, locadj,
     4 lsumadj,maxija, ialbdo, maxalb, lenalb, medal(10),icomb,lsumenr
c
      common /user/   agstrt, wtstrt, xstrt,  ystrt,  zstrt,  dff,
     1                ebotn,  ebotg,  tcut,   i0u,    iread,  iadjm,
     2                ngpqt1, ngpqt2, ngpqt3, ngpqtg, ngptqn, nits,
     3                nlastu, nleft,  nmgp,   nmtg,   nstrt
c
      common bc(1500000)
      dimension nc(1),titl(4)
       dimension enr(26)
      equivalence (bc(1),nc(1))
      data im,jm/13,13/
c
c     The following data is the energy group structure for the
c     TORT generated albedo data
c
      data enr/14.918e+6, 13.499e+6, 12.214e+6, 11.052e+6, 10.e+6,
     * 8.1873e+6, 6.7032e+6, 4.9669e+6, 3.6788e+6, 2.7253e+6, 2.2313e+6,
     * 1.6530e+6, 1.2246e+6, 1.0026e+6, 8.2085e+5, 6.7206e+5, 5.5023e+5,
     * 3.0197e+5, 1.6573e+5, 3.1828e+4, 3.3546e+3, 3.5358e+2, 3.7267e+1,
     * 3.9279,   4.1399e-1,  1.e-11 /
      nleft=nlf
c
c * * read one card with the albedo data logical tape unit  * * * * * *
c
      read (iread,1010) itap
 1010 format (14i5)
c
c * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
c * *
c * * start of albedo medium loop
c
      ialb0 = 0
 10   ialb0 = ialb0 + 1
c * *
      read (itap) titl
c
c    144 characters of title information
c
      write (i0,1015) titl
 1015 format (1h1,3a8,a5/1x,18a4)
      if (ialb0 .ne. 1) go to 110
c
      read (itap) ntheti, inrgys, nmgpa, maxalb , im , jm
      write (i0,1020) ntheti, inrgys, nmgpa, maxalb
 1020 format ('0ntheti =',i3,', inrgys =',i3,', nmgpa =',i3,
     1        ', maxalb =',i3)
```

```
c
c     read the media numbers for albedoes   *     *     *
c
      read(iread,1010) (medal(i),i=1,maxalb)
      igamg = inrgys - nmgpa
      icomb = 0
      if (igamg .gt. 0  .and.  nmgpa .gt. 0) icomb = 1
      locice = nlast
      locenr = locice
c
c * * integer array starting at locice+1 gives the albedo energy group
c     for each cross-section group
c
      if (nmtg-inrgys) 15,25,20
 15   call help(4halbn,1,1,1,1)
      call error
      stop
 20   ia = locice + 1
      ib = ia + nmtg - 1
c
c * * read the ice array with an 14i5 format
c
      read (iread,1010) (nc(i),i=ia,ib)
      write(6,772) (nc(i),i=ia,ib)
772   format(/,5x,' the ice array ',/,(5x,14i5))
      locenr = ib
 25   locvel = locenr + inrgys  + 1
      if (icomb .ne. 0) locvel = locvel + 1
      loccor = locvel + inrgys
      locthi = loccor + nmtg
      lntho = locthi + ntheti
      lsntho = lntho + ntheti
      loctho = lsntho + ntheti + 1
c * *
c * *
      ia = locenr + 1
      ib = ia + inrgys - 1
c
      read (itap) (bc(i),i=ia,ib)
      ib = ib + 1
      do 771 ikenr = ia,locvel
      if((ikenr-locenr).gt.26) goto 771
      bc(ikenr) = enr(ikenr-locenr)
771   continue
c
c * * energies(1/inrgys+1)   (or 1/inrgys+2 for a combined problem)
c
      ebot = bc(ib)
      ia = locvel + 1
      ib = ia + inrgys - 1
c
      read (itap) (bc(i),i=ia,ib)
```

```
c
c * * velocities(1/inrgys)
c
      write (i0,1025) nmgpa
 1025 format (1h0,44hgroup parameters, group numbers greater than,i5,
     1 34h correspond to secondary particles)
      write (i0,1030)
 1030 format (1h0,3x,5hgroup,5x,10hupper edge,7x,8hvelocity,/17x,4h(ev),
     1 10x,8h(cm/sec) )
      do 30 i=1,nmgpa
      ie = locenr + i
      iv = locvel + i
 30   write (i0,1035) i, bc(ie), bc(iv)
 1035 format (i7,1p6e16.4)
      write (i0,1040) bc(ie+1)
 1040 format (7x,1pe16.4)
      if (icomb .eq. 0) go to 40
      ie = ie + 1
      do 35 i=1,igamg
      k = i + nmgpa
      ie = ie + 1
      iv = iv + 1
 35   write (i0,1035) k, bc(ie), bc(iv)
      write (i0,1040) bc(ie+1)
c
c * * fill the array corect starting in loccor + 1
c
 40   do 45 ig=1,nmtg
 45   bc(loccor + ig) = 1.0
      if (locice .eq. locenr) go to 75
      lastgp = 0
      do 70 ig=1,nmgp
      newgp = nc(locice+ig)
c
c       this is the albedo group corresponding to the xsec group ig
c
      if (newgp .eq. lastgp) go to 65
c
c * crossed into a new albedo group
c * is it the first time?
c
      if (ig .eq. 1) go to 60
c
c * process the last albedo group (group 'lastgp')
c
 50   etop = bc(nfgp)
      write(6,774) nfgp,(nfgp+ngps),etop,bc(nfgp+ngps)
774   format(5x,' nfgp,(nfgp+ngps = ',2i10,' etop,elow = ',2e20.10)
      elow = bc(nfgp + ngps)
      if (nfgp+ngps .eq. nmgp+1) elow = ebotn
      ualb = alog(etop/elow)
      do 55 jg=1,ngps
```

```
      etop = bc(nfgp + jg - 1)
      elow = bc(nfgp + jg)
      if (nfgp+jg .eq. nmgp+1) elow = ebotn
c
c       etop & elow are the energy bounds of each xsec group
c       included in albedo group 'lastgp'
c
 55   bc(loccor+nfgp+jg-1) = alog(etop/elow)/ualb
      write(6,775) etop,elow,ualb
775   format(5x,' etop,elow,ualb = ',3e20.10)
      if (ig .eq. nmgp) go to 70
 60   lastgp = newgp
      nfgp = ig
      ngps = 0
 65   ngps = ngps + 1
      if (ig .eq. nmgp) go to 50
 70   continue
 75   continue
c
c * * end of neutron portion of array corect.  if gamma rays are to be
c   processed, a comparable procedure may be performed in the following
c   subroutine.  this routine may also modify neutron portion of array
c   'corect', if additional data is available.  note that array corect
c   should sum to 1.0 over each albedo group.
c
      call corect
      ia = locthi + 1
      ib = ia + ntheti - 1
      read (itap) (bc(i),i=ia,ib)
      write(6,833)ia
833   format(5x,'reading thetin(ntheti) - ia=',i10)
      write (i0,1045) (bc(i),i=ia,ib)
 1045 format (7h0thetin      /(1p8e14.4))
c
c * * thetin(ntheti)
c * read ntheti values of theto, the numbers of outgoing polar angle bin
c
      ia = lntho + 1
      ib = lntho + ntheti
      read (itap) (nc(i),i=ia,ib)
      write(6,832)ia
832   format(5x,'reading ntheti values of theto - ia= ',i10)
      write (i0,1050) (nc(i),i=ia,ib)
 1050 format (7h0ntheto      /(8i14))
c
c * ntheto(1/ntheti)
c *
c * fill in nthets array for i=1,ntheti+1
c
      ia = lsntho + 1
      nc(ia) = 0
      do 80 i=1,ntheti
```

```
      ia = ia + 1
 80   nc(ia) = nc(ia-1) + nc(lntho+i)
c
      nino = nc(ia)
c * *
c * do pointers using nino
      locnp = loctho + nino + ntheti
      lsnpki = locnp + nino
      lsnpi = lsnpki + nino + ntheti
      locphi = lsnpi + ntheti + 1
c   *    *             *
c   *    *             *  read theta
c   *    *             *
      ib = loctho
      do 989 kth = 1,ntheti
      ia = ib + 1
      ib = ia + nc(lntho+kth)
      read (itap) (bc(i),i=ia,ib)
989   continue
      do 85 i=1,ntheti
      km = nc(lntho+i) + 1
      ia = loctho + nc(lsntho+i) + i
      ib = ia + km - 1
 85   write (i0,1055) i,(bc(j),j=ia,ib)
 1055 format (8h0theta (,i2,4h) = ,1p8e14.4,(/14x,1p8e14.4))
c
c * * theta(1/ntheto(i)+1 , 1/ntheti)
c *
c * read nphi
c
      ib = locnp
      do 987 kth = 1, ntheti
      ia = ib + 1
      ib = ia + nc(lntho+kth) - 1
      read (itap) (nc(i),i=ia,ib)
987   continue
      do 90 i=1,ntheti
      km = nc(lntho + i)
      ia = locnp + nc(lsntho+i) + 1
      ib = ia + km - 1
 90   write (i0,1060) i,(bc(j),j=ia,ib)
 1060 format (7h0nphi (,i2,4h) = ,8i14,(/13x,8i14))
c
c * * nphi(1/ntheto(i) , 1/ntheti)
c *
c  * * calculate summation of nphi
c
      insnpi = lsnpi + 1
      nc(insnpi) = 0
      ninonpx = 0
      do 100 i=1,ntheti
      kss = lsnpki + nc(lsntho+i) + i
```

```
      ksn = locnp + nc(lsntho+i)
      nc(kss) = 0
      km = nc(lntho+i)
      do 95 k=1,km
      kss = kss + 1
      ksn = ksn + 1
      ninonpx = ninonpx + (nc(ksn) - 1)*2
 95   nc(kss) = nc(kss-1) + (nc(ksn) - 1)*2
      insnpi = insnpi + 1
 100  nc(insnpi) = nc(insnpi-1) + nc(kss)
      ninonp = nc(insnpi)
      ia = locphi + 1
      ib = ia - 1
      do 710 knt=1,2
      do 710 knto=1,ntheti
      ia = ib + 1
      ib = ia + nc(locnp+knto) * 2 - 1
      read (itap) (bc(i),i=ia,ib)
 105  write (i0,1070) knto, (bc(il),il=ia,ib,2)
 1070 format (8h0knto = ,i3,14h, phi(1,k,i) = /(1p8e14.4))
710   continue
      locnij = locphi + 2 * nino + 2 * ninonp
      ia = locnij + 1
      write(6,836)ia
836   format(5x,'reading array for colapsing - ia=',i10)
      ib = ia + im * jm - 1
      read (itap) (nc(i),i=ia,ib)
c
c * * phi(1/nphi(k,i)+1 , 1/ntheto(i) , 1/ntheti)
c
c *
c * do remaining pointers
c
      lsumenr = (inrgys + 1) * inrgys / 2
      if(nmgpa.eq.inrgys) goto 791
      lsumenr = 0.
      do 700 kenr=1,nmgpa
      lsumenr = lsumenr + ( inrgys - kenr + 1 )
700   continue
791   continue
      maxija = 0
      do 701 kija = 1,im*jm
      ki = locnij + kija
      if(maxija.lt.nc(ki)) maxija = nc(ki)
701   continue
      locmxy = locnij + im * jm
      loctot = locmxy + (im + jm + 2 )
      loccen = loctot + nmgpa * ntheti
      locpxy = loccen + lsumenr * ntheti
      locpet = locpxy + lsumenr * ( im + jm ) * ntheti
      locpaz = locpet + lsumenr * nino * maxija
      nlast = locpaz + lsumenr * ninonpx * maxija
```

```
c
c * * nlast here is the last cell required for one albedo medium
c
      lenalb = nlast - loctot
      nlast = nlast + (maxalb-1)*lenalb
      if(nlast-nleft) 96,96,130
  96  continue
 110  ia = loctot + (ialb0-1)*lenalb + 1
      ial0 = locmxy + 1
      ib10 = ial0 + im + jm + 1
      read (itap) (bc(i),i=ial0,ib10)
      ib = loccen
      ib1 = locpxy
      ib2 = locpet
      ib3 = locpaz
      ib4 = locadj
      itot = ia -1
      if(iadjm.ne.0) goto 903
      do 900 m01 = 1,ntheti
      do 901 ig0 = 1,nmgpa
955   itot = itot + 1
      ia = ib + 1
      ib = ia + (inrgys - ig0)
      read (itap) bc(itot),(bc(i),i=ia,ib)
      write(6,932) m01,ig0,bc(itot),ia
932   format(5x,'total albedo for thetin #',i3,' energy in',i4,' is =',
     * f14.7,' ia=',i10)
      ial = ib1 + 1
      ib1 = ial + ( im + jm ) * (inrgys - ig0 + 1) - 1
      read (itap) (bc(i),i=ial,ib1)
      ia2 = ib2 + 1
      ib2 = ia2 + maxija * nc(lntho+m01) * (inrgys - ig0 + 1) - 1
      read (ttap) (bc(i),i=ia2,ib2)
      do 999 ibum=1,ntheti
      ia3 = ib3 + 1
      ib3 = ia3 + maxija * (nc(locnp+ibum)-1) * 2 * (inrgys - ig0 + 1)-1
      read (itap) (bc(i),i=ia3,ib3)
999   continue
 901  coniinue
 900  continue
      goto 902
903   continue
      do 1904 mi0 = 1,ntheti
      do 1905 ig0 = 1,nmgpa
      do 1999 ibum=1,ntheti
      ia3 = ib3 + 1
      ib3 = ia3 + maxija * (nc(locnp+ibum)-1) * 2 * (inrgys - ig0 + 1)-1
      read (itap) (bc(i),i=ia3,ib3)
1999   continue
1905  continue
1904  continue
      do 904 m1 = 1,ntheti
```

```
      do 905 igl = 1,nmgpa
      itot = itot + 1
      ia = ib + 1
      ib = ia + igl - 1
      read (itap) bc(itot),(bc(i),i=ia,ib)
      write(6,1932) ml,igl,bc(itot),ia
1932   format(5x,'total albedo for thetin #',i3,' energy in',i4,' is =',
     * f14.7,' ia=',i10)
      ial = ibl + 1
      ibl = ial + ( im + jm ) * igl - 1
      read (itap) (bc(i),i=ial,ibl)
      ia2 = ib2 + 1
      ib2 = ia2 + maxija * nc(lntho+ml) * igl - 1
      read (itap) (bc(i),i=ia2,ib2)
 905  continue
 904  continue
c
c  * * totalb, probability per energy group (1/inrgys )
c * * print out the total albedoes on unit i0
c
902   ic = locthi + 1
      id = ic+ntheti-1
      write (i0,1075) (bc(i),i=ic,id)
 1075 format ('0         total albedo (e grp in,polar in)'
     1 /' energy/     cosine of polar angle'/' group',f8.4,10f11.4/,
     2 (3x,11f11.4))
      ic = loctot
      do 115 j=1,nmgpa
      ic = ic + 1
      id = ic + nmgpa * (ntheti-1)
      write (i0,1080) j,(bc(i),i=ic,id,nmgpa)
 115  continue
 1080 format (i5,1p11e11.4,/(5x,1p11e11.4))
      if (ialb0 .lt. maxalb) go to 10
c
c * *
c * *
c
      ialbdo = 1
      return
 130  write(i0,1085) nlast,nleft
 1085 format( 43h0insufficient room for albedo data ,nlast = i15,3x,
     1  8hnleft =  i15)
      stop
      return
      end
```

# APPENDIX C

## ENERGY GROUP STRUCTURES AND CROSS SECTION SETS

This appendix presents the energy group structure, the concrete composition, and the cross section data set used in the solution of the example problem. The energy group structure used of the albedo data and the cross-section sets used in all the calculations performed with the modified and unmodified versions of the MORSE/BREESE code package is presented in Table C-1. The MORSE benchmark calculation also used the energy group structure presented in Table C-1. Table C-2 presents the energy group structure of the albedo data generated with the MORSE/STORM code package. The energy group structure of the cross section set used in the MORSE/STORM calculations is presented in Table C-3. All calculations during this research, with the exception of the P-8 TORT albedo information, were performed with P-3 cross-section sets derived from ENDF-IV data. The P-8 TORT albedo calculations were performed with cross-sections derived from the VITAMIN-E (ENDF-V) data. The composition of the concrete used in the example problem is presented in Table C-4.

Table C-1. Energy Group Structure Used in the MORSE/BREESE Calculations.

| Neutron | | Neutron | |
|---|---|---|---|
| Group | Energy Upper Limit (eV) | Group | Energy Upper Limit (eV) |
| 1 | 1.4918e+07 | 27 | 5.2475e+04 |
| 2 | 1.2214e+07 | 28 | 4.0868e+04 |
| 3 | 1.0000e+07 | 29 | 3.1828e+04 |
| 4 | 8.1873e+06 | 30 | 2.4788e+04 |
| 5 | 6.7032e+06 | 31 | 1.9305e+04 |
| 6 | 5.4881e+06 | 32 | 1.5034e+04 |
| 7 | 4.4933e+06 | 33 | 7.1018e+03 |
| 8 | 3.6788e+06 | 34* | 4.3074e+03 |
| 9 | 3.0119e+06 | 35 | 3.3546e+03 |
| 10 | 2.4660e+06 | 36 | 2.6126e+03 |
| 11 | 2.0190e+06 | 37 | 2.0347e+03 |
| 12 | 1.6530e+06 | 38 | 1.5846e+03 |
| 13 | 1.3534e+06 | 39 | 1.2341e+03 |
| 14 | 1.1080e+06 | 40 | 9.6112e+02 |
| 15 | 9.0718e+05 | 41 | 4.5400e+02 |
| 16 | 7.4274e+05 | 42 | 2.1445e+02 |
| 17 | 6.0810e+05 | 43 | 1.0130e+02 |
| 18 | 4.9787e+05 | 44 | 4.7851e+01 |
| 19 | 4.0762e+05 | 45 | 2.2603e+01 |
| 20 | 3.3373e+05 | 46 | 1.0677e+01 |
| 21 | 2.7324e+05 | 47 | 5.0435e+00 |
| 22 | 2.2371e+05 | 48 | 2.3824e+00 |
| 23 | 1.8316e+05 | 49 | 1.1254e+00 |
| 24 | 1.4996e+05 | 50 | 4.1400e-01 |
| 25 | 1.2277e+05 | 51 | 1.0000e-01 |
| 26 | 8.6517e+04 | | |

*Last energy group analyzed in the Monte Carlo calculations.

Table C-2. Energy Group Structure of the Spatially-Dependent Albedo Data.

| Neutron | | Neutron | |
|---|---|---|---|
| Group | Energy Upper Limit | Group | Energy Upper Limit |
| 1 | 1.4918e+07 | 14 | 1.0026e+06 |
| 2 | 1.3499e+07 | 15 | 8.2085e+05 |
| 3 | 1.2214e+07 | 16 | 6.7206e+05 |
| 4 | 1.1052e+07 | 17 | 5.5023e+05 |
| 5 | 1.0000e+07 | 18 | 3.0197e+05 |
| 6 | 8.1873e+06 | 19 | 1.6573e+05 |
| 7 | 6.7032e+06 | 20* | 3.1828e+04 |
| 8 | 4.9659e+06 | 21 | 3.3546e+03 |
| 9 | 3.6788e+06 | 22 | 3.5358e+02 |
| 10 | 2.7253e+06 | 23 | 3.7267e+01 |
| 11 | 2.2313e+06 | 24 | 3.9279e+00 |
| 12 | 1.6530e+06 | 25 | 4.1399e-01 |
| 13 | 1.2246e+06 | | |

*Last energy group analyzed in the Monte Carlo calculations.

Table C-3. Energy Group Structure of the Cross Section Set Used in the MORSE/STORM Calculations.

| Neutron | | Neutron | |
|---|---|---|---|
| Group | Energy Upper Limit | Group | Energy Upper Limit |
| 1 | 1.4918e+07 | 24 | 5.5023e+05 |
| 2 | 1.3499e+07 | 25 | 4.0762e+05 |
| 3 | 1.2214e+07 | 26 | 3.0197e+05 |
| 4 | 1.1052e+07 | 27 | 2.2371e+05 |
| 5 | 1.0000e+07 | 28 | 1.6573e+05 |
| 6 | 9.0484e+06 | 29 | 1.2277e+05 |
| 7 | 8.1873e+06 | 30 | 6.7379e+04 |
| 8 | 7.4082e+06 | 31 | 3.1828e+04 |
| 9 | 6.7032e+06 | 32 | 1.5034e+04 |
| 10 | 6.0653e+06 | 33* | 7.1017e+03 |
| 11 | 5.4881e+06 | 34 | 3.3546e+03 |
| 12 | 4.9659e+06 | 35 | 1.5846e+03 |
| 13 | 4.4933e+06 | 36 | 7.4852e+02 |
| 14 | 4.0657e+06 | 37 | 3.5358e+02 |
| 15 | 3.6788e+06 | 38 | 1.6702e+02 |
| 16 | 3.3287e+06 | 39 | 7.8893e+01 |
| 17 | 3.0119e+06 | 40 | 3.7267e+01 |
| 18 | 2.7253e+06 | 41 | 1.7603e+01 |
| 19 | 2.4660e+06 | 42 | 8.3153e+00 |
| 20 | 1.8268e+06 | 43 | 3.9379e+00 |
| 21 | 1.3534e+06 | 44 | 1.8554e+00 |
| 22 | 1.0026e+06 | 45 | 8.7643e-01 |
| 23 | 7.4274e+05 | 46 | 4.1399e-01 |

*Last energy group analyzed in the Monte Carlo calculations.

Table C-4. Concrete Composition.

| Element | Atomic Number Density (Atoms/$\mathring{A}^3$) |
|---|---|
| O | 4.386e-02 |
| Ca | 2.915e-03 |
| Al | 2.388e-03 |
| Si | 1.580e-02 |
| H | 7.768e-03 |
| K | 6.900e-04 |
| Mg | 1.487e-04 |
| Fe | 3.128e-04 |
| Na | 1.048e-03 |

## VITA

Itacil Chiari Gomes was born in São Carlos, São Paulo, Brazil on October 29, 1953, the son of Alcides Bugalho Gomes and former Lady Chiari. He attended elementary and high-school at that city. In 1972, he was accepted by the Escola Politecnica of the São Paulo University in São Paulo city through a public qualification exam to pursue the Bachelor of Science degree in Engineering. During the time spent at the University of São Paulo, he has also attended classes at Architecture and Communications Schools. In the fall of 1980 he received the Bachelor of Science degree in Civil Engineering.

In 1981, he was accepted by the Instituto Militar de Engenharia, Rio de Janeiro, Brazil to pursue the Master of Science degree in Nuclear Engineering. In 1983, he received this degree. He worked at the Instituto Militar de Engenharia from 1983 to 1986.

In March of 1987, he entered The University of Tennessee in Knoxville to work toward the degree of Doctor of Philosophy in Nuclear Engineering.

He is married to the former Luisa Maria Ribeiro Torres of Rio de Janeiro. They now have two children, Igor and Leonardo.