USER'S MANUAL for LSMONTE A Three-dimensional, Radiative Heat Transfer Analysis Computer Code

Version 1.0

December 04, 2000

by

Charles N. Zeeb and Patrick J. Burns

Department of Mechanical Engineering

Colorado State University

Fort Collins, CO 80523

Charles.Zeeb@ColoState.EDU and Patrick.Burns@ColoState.EDU

970.491.7479 and 970.491.5778

http://www.colostate.edu/~pburns/monte.html

Livermore Software Technology Corporation

7374 Las Positas Road

Livermore, CA 94550

http://www.lstc.com

Table of Contents

Tables

Figures

1. INTRODUCTION

1.1 Background

This manual encompasses the 3-D Monte Carlo radiative exchange factor computer code known as “LSMONTE.” This version of the code was developed for the Livermore Software Technology Corporation (LSTC) from the last public domain version of the code MONT3D [Maltby et al., 1994].

MONT3D was developed in the Department of Mechanical Engineering at Colorado State University (CSU) beginning with the work of Scott Statton in 1983 [Statton, 1983], continuing with the work of James D. Maltby [Maltby, 1987; Maltby, 1990], and most recently with the work of Charles N. Zeeb [Zeeb and Burns, 1999]. The code is capable of simulating geometries modelled as assemblages of generalized quadrilaterals, which are constrained to be flat. Curved surfaces must be approximated by a sufficient number of flat surfaces to “capture” the curvature. Surfaces may absorb photons, or they may reflect or transmit them specularly and/or diffusely. All exterior surfaces must be non-transmissive (it is left to the user to ensure this) so that photons are not “lost” during tracing. All material radiative properties may be explicit functions of the incident photon angle, and be dependent upon energy through the band wavelength formulation. Mark Havstad, Charlie Landram and Donald L. Brown at the Lawrence Livermore National Laboratory (LLNL) and Katherine Bryan of Oak Ridge National Laboratory have exhaustively exercised the code, checking it for validity. The code has been used by many individuals for over a decade on many problems, including radiative heat transfer and rarefied gas dynamics.

A number of related publications exist wherein the code has been applied to various problems, including a detailed test of the Separator Development Facility (SDF) [Maltby, 1987], the calculation of radiative exchange in passive solar enclosures [Maltby et al., 1986], application to the Laser Isotope Separation (LIS) process [Burns and Pryor, 1989]. Additional publications of interest are Burns et al., 1990; Maltby and Burns, 1991; Burns et al., 1992; Schweitzer et al., 1993; Burns and Pryor, 1996; Burns and Pryor, 1999; and Zeeb and Burns, 1999.

A number of related codes are available in the public domain. These include the graphical post-processor MPLOT [Burns, 2000] which plots geometries and material properties from LSMONTE; MONT2D [Maltby et al., 1994], a Monte Carlo photon-tracing code similar to LSMONTE but for two-dimensional Cartesian and axisymmetric geometries; MONT3V, a code that simulates rarefied as dynamics with binary collisions [Dolaghan, 1991]; MONT3E [Crockett et al., 1990], a code for simulating electron transport in three dimensions in the presence of a spatially varying magnetic field; and SPUT3D [Dolaghan, 1996], a code for simulating molecular redistribution during sputtering in three-dimensional enclosures. The same technique has also been used to simulate daylighting in building geometries [McHugh et al., 1998; McHugh, 1995]. Information on these codes may be obtained from the authors of this manual.

1.2 Theoretical Formulation

The present computer codes are formulated in the Monte Carlo style where a large number of photons are emitted from each surface and “traced” until each is absorbed by another surface. Using the subscripts i and j to denote the emitting and absorbing surfaces, respectively, and the superscript k to denote the wavelength band, it can be shown that:

$$\dot{Q}_{i\rightarrow j}^{k} = \frac{\varepsilon_i^k N_{ij}^k}{N_i^k}\sigma T_i^4 f_i^k A_i \qquad (1.1)$$

where:

$\dot{Q}_{i\rightarrow j}^{k}$	= one way rate of radiative heat transfer emitted from surface *i* and absorbed by surface *j* in wavelength band *k* (W).
ε_i^k	= emittance of surface *i* in wavelength band *k.*
N_{ij}^k	= number of photons emitted in wavelength band *k* from surface *i* and absorbed by surface *j.*
N_i^k	= total number of photons emitted in wavelength band *k* from surface *i.*
σ	= Stefan-Boltzman constant, $5.669\mathrm{x}10^{-8}\left(\mathrm{W} \bullet \mathrm{m}^{-2} \bullet \mathrm{K}^{-4}\right)$.
T_i	= absolute temperature of surface *i* (K) .
f_i^k	= fraction of the blackbody energy of surface *i* in wavelength band *k.*
A_i	= area of surface *i* $\left(\mathrm{m}^2\right)$.

The ordinary definitions of the exchange factor E_{ij}^k and the exchange fraction F_{ij}^k are as follows:

$$E_{ij}^k = \varepsilon_i^k F_{ij}^k = \varepsilon_i^k \frac{N_{ij}^k}{N_i^k} \qquad (1.2)$$

The following rules apply to these quantities:

$$\sum_j E_{ij}^k = \varepsilon_i^k \qquad (1.3)$$

or:

$$\sum_j F_{ij}^k = 1 \qquad (1.4)$$

Equations (1.3) and (1.4) express conservation of energy (photons) since all photons must be absorbed by a surface. Given that there must be zero net heat flow between isothermal surfaces, the second law of thermodynamics (entropy principle) follows from equation (1.1) as:

$$E_{ij}^k A_i = E_{ji}^k A_j \qquad (1.5)$$

or:

$$\varepsilon_i^k F_{ij}^k A_i = \varepsilon_j^k F_{ji}^k A_j \qquad (1.6)$$

The net radiative exchange in wavelength band *k* from surface *i* to surface *j* is then:

$$\dot{Q}^k_{ij} = E^k_{ij} A_i \sigma \left(f^k_i T^4_i - f^k_j T^4_j \right) \qquad (1.7)$$

and

$$\dot{Q}^k_{ij} = \varepsilon^k_i F^k_{ij} A_i \sigma \left(f^k_i T^4_i - f^k_j T^4_j \right) \qquad (1.8)$$

Either of the relations expressed as equation (1.5) or (1.6) may be used to test the exchange factors or the exchange fractions for consistency (convergence). Indeed, either of these relations may be used to manipulate the values in the matrix if either of equations (1.3) or (1.4) is used as a constraint.

1.3 View Factors

The codes may be used to compute view factors, valid for diffuse reflectances independent of incident angle. To do so, all entries for curves of specular and diffuse reflectances and transmittances must be set to 0.0, resulting in an emittance of 1.0 (“black”) for all surfaces. In this limiting case, exchange factors for blackbody surfaces are equivalent to view factors.

1.4 Implementation

The codes are typically used as “preprocessors” for thermal balance studies. As such, the matrix of exchange numbers output from the programs are used as input to a thermal balance code. MONT3D is designed to be compatible with the thermal analysis codes LSDYNA, available from the Livermore Software Technology Corporation, and TOPAZ3D [Shapiro, 1985], developed at Lawrence Livermore National Laboratory. SMOOTH [Dolaghan et al., 1992] is a postprocessor designed to take advantage of reciprocity to improve the accuracy of the estimates. It operates on the output of MONT3D and produces output compatible with LSDYNA and TOPAZ3D. Dolaghan’s version of SMOOTH is not compatible with LSMONTE; a new version that will be compatible is under development. However, smoothing exchange factors is not required, its only effects are to improve the accuracy of the exchange factors and to reduce the size of the exchange factor matrix.

Fundamentally, the geometry and the material properties are the only quantities necessary to establish the exchange factors. The exchange factors result from the interaction between the geometry and the material properties in a complex fashion, and are unique to a particular geometry/material property combination. Therefore, it is not possible to extend a set of exchange factors calculated for a particular geometry/material property combination to another geometry/material property combination, even if only the geometry or only the material properties vary. After any changes, the problem must be rerun. However, if view factors are calculated (all properties black), arbitrary diffuse reflectances may be included in the thermal analysis code since view factors are dependent on geometry alone. Note that diffuse exchange with finite reflectances calculated from the present codes will yield slightly different answers than those obtained using the radiosity/irradiation approach, as the assumption in the radiosity approach is uniform radiosity/irradiation over each surface, which is not so for the present code. Specularity or material property dependence on incident angle may not be modelled using view factors.

In addition to radiation exchange, this code can also be used to simulated some special cases of other transport problems in enclosures. If the Knudsen number is much less than 1, rarefied gas dynamics problems can be simulated (a vacuum vessel). Also, some simple cases of molecular sputtering in an enclosure can be simulated.

1.5 Performance and Run Times

Monte Carlo has been termed “the method of last resort,” as the perception used to be that run times are very long. This viewpoint is no longer as relevant as it used to be. Hardware is now much faster - larger problems can be done in a reasonable amount of time. LSMONTE has been highly optimized for performance in a variety of ways. First, a very efficient photon tracing algorithm is employed. Secondly, all quantities for which it is appropriate are precomputed during the input phase to prevent repetitive computation during the solution phase. Thirdly, the grid tracing algorithm causes the total photon tracing time to scale in proportion to the number of surfaces to the power 1.5, rather than in proportion to the number of surfaces to the power 2. As the number of surfaces grows, this represents a huge savings in solution time. Significant time is also spent emitting and reflecting photons, making the overall solution time scale approximately in proportion to the number of surfaces.

The Monte Carlo method has the capability to model very sophisticated geometries and material properties - not possible with other methods. Also, the Monte Carlo method is guaranteed to converge - not so with other methods. Finally, run times must be evaluated in context against the effort required to construct and debug input files, which can take several person-months for large, complex geometries.

A theoretical development [Burns and Pryor, 1999] indicates the following approximate relation for scaling CPU time from one run (subscript 1) to another run (subscript 2):

$$\frac{\Delta t_1}{\Delta t_2} = \frac{Photons_1 \mathrm{x} Bands_1 x Surfs_1 \mathrm{x} Clock_2}{Photons_2 \mathrm{x} Bands_2 x Surfs_2 \mathrm{x} Clock_1} \times \frac{\alpha_{E,2}}{\alpha_{E,1}} \quad (1.9)$$

where

Δt	= total solution time (secs)
Photons	= constant number of photons emitted per surface
Bands	= number of wavelength bands
Surfs	= number of surfaces
Clock	= the clock rating of the CPU (Mhz)
α_E	= "enclosure" absorptance, i.e. average, area-weighted absorptance of all surfaces

The photon tracing basis for the approximate relation of eqn. (1.9) has been validated for various geometries by Zeeb and Burns [1999]. The relation of eqn. (1.9) is approximate and is only valid at the “optimal” grid. Grid tracing changes the scaling of CPU time from Surfs^2 to Surfs$^{1.5}$, providing significant savings in execution time for large geometries. Additional details on grid tracing, including guidance on selecting the “optimal” grid, are given in Section 2.5.

Factors that introduce variability into the run time include:

- Geometry - Photon tracing times are dependent upon geometry. Observations indicate that variability in run time from the “average” can vary considerably.
- Operating system - The execution speed is dependent upon the operating system (particularly the overhead introduced by the operating system). For example, LSMONTE will run under the linux operating system in about one-half the time it runs under the Windows operating system.
- Compiler - The execution speed depends upon the quality of the compiler. However, most modern compilers are mature and fairly efficient, so that the differences in execution speed among compilers may not be that significant.
- Hardware - Execution speed also depends upon hardware, especially the Floating Point Unit (FPU). Different FPU’s have different amounts of logic implemented in hardware, making them relatively more or less efficient at processing.
- Amount of Input/Output (I/O) - Execution speed also depends somewhat upon the amount of I/O, which is only significant for large files and/or small numbers of photons emitted. In particular, writing exchange fractions in ASCII format to the output file can consume significant time for very large problems, because the work of writing exchange fractions scales as $Surfs^2$, while processing time scales as *Surfs*. However, the extra work is only significant for very large problems. As an example, writing the exchange fractions for the third example in Table 1.1 below, *mtgun.in*, increases the total run time by only 548 secs, or 3.5%. The toggle for writing exchange fractions to the ASCII output file is described in Section 3.2.4.

Table 1.1 below provides some measured run-times on an IBM portable computer with a 233 MHz Pentium II chip running the Windows 2000 operating system. Note that relatively large problems can be done in reasonable run times on relatively slow hardware. In our experience, in general, run times are vastly less than the times spent preparing input files.

Table 1.1: Run times on a 233 Mhz Pentium II processor, single wavelength band, 100,000 photons per surface emitted

Input File Name	Photons per Surface	Number of Surfaces	Grid (NGX x NGY x NGZ)	Run time (secs)
etf.in	100,000	145	6x2x6	664
geo.in	100,000	1,182	22x22x22	6,263
mtgun.in	100,000	4,297	20x30x50	15,852

2. COMPUTER CODE BACKGROUND

This chapter presents the background information necessary to run the Monte Carlo computer code. The first two subsections define the geometry of a radiating enclosure in terms of nodes and radiating surfaces. The third subsection discusses surface concatenation, which is no longer available in the code, but may be done by the user in a post-processing phase. The fourth subsection describes material types, the radiative properties to be specified versus incident cone angle, and the cosine power dependence of diffuse reflection and transmittance. The fifth subsection discusses grid tracing that dramatically reduces execution time. The sixth subsection discusses accuracy versus the number of photons emitted. The seventh subsection is next on debugging information generated by the program. This is followed by the eighth subsection on the restart capability, whereby a run may proceed from a previously calculated state. The chapter concludes with a discussion of the pseudo-random number generator.

2.1 Nodes

The description of the geometry of the radiation enclosure begins with the specification of the nodes (also called nodal points) of the enclosure. A node is a point in three-dimensional space defined by three coordinates in a Cartesian coordinate system *{X, Y, Z}*, as depicted in Figure 2.1. Each node, *N*, in the enclosure is assigned a node number which must be positive and a member of the closed, full set *N* ε *{1,NUMNP}*, where *NUMNP* is the total number of nodes for the problem. Once the nodes of the enclosure have been specified, radiating surfaces can be defined by specifying assemblages of nodal points. Section 3.3 defines nodal input.

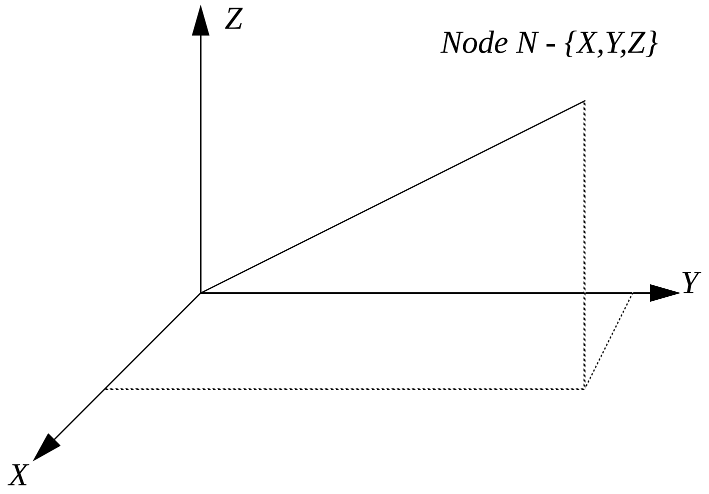


Figure 2.1 Global Coordinate System

2.2 Surfaces

Radiating surfaces defined in three dimensions consist of planar quadrilaterals or triangles and are defined by specifying 4 nodal points. If four distinct points are specified, then the surface is a generalized quadrilateral, as shown in Figure 2.2(a). If it is desired to use triangular surfaces, then the last two node numbers and only the last two node numbers must be identical as shown in Figure 2.2(b). Two caveats apply here: none of the interior angles may exceed 180° (i.e., no concave corners are permitted), and the four nodes should be coplanar (to within a small tolerance).

This tolerance can be set, and surfaces which exceed this tolerance are flagged with warnings. The code will run even if quadrilaterals are non-coplanar, as every quadrilateral is split into two triangles, which by definition are planar. However, the user is cautioned that quadrilaterals are split arbitrarily along the line from N_1 to N_3, and the geometry simulated may vary from that intended unless the quadrilaterals are indeed coplanar to within a small tolerance. See section 3.2.3 for more details.

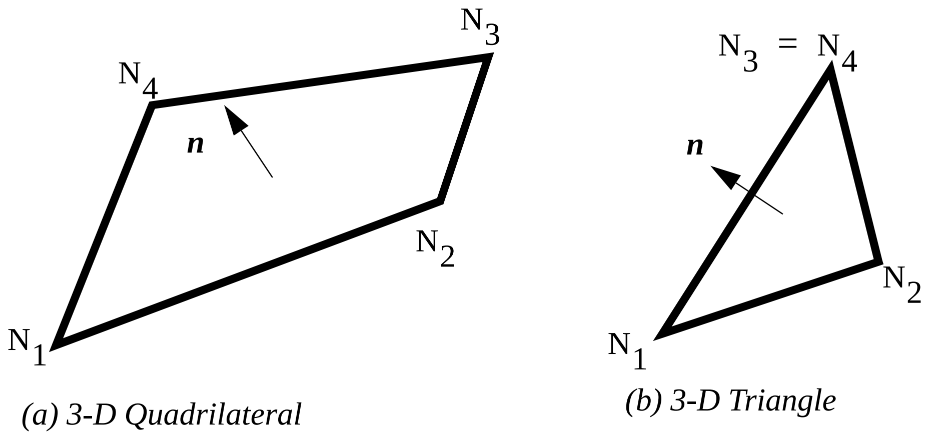


Figure 2.2 Radiating Surface Geometries

It is very important to note that the outward normal ***n*** of the radiating surface is always defined as pointing into the enclosure. The outward normal for both types of surfaces is defined consistent with the right-hand rule, i.e. if the fingers of the right hand are curled in the direction of increasing nodal point number $(N_1 \rightarrow N_4)$, then the thumb of the right hand indicates the direction of the outward normal. This results in a counterclockwise convention for nodal point numbering, when “viewing” the radiating side of the surface from above. The “back” of the surface opposite the surface normal is transparent as far as the code is concerned, and a leak will result from incorrect node numbering.

Each radiating surface is assigned a material number, and all material properties are independent of spatial position on a single surface. Referring to Figure 2.3, one observes the local (primed) coordinate system with vertex at N_1, and axes as shown. Photons are emitted randomly from points on a surface; a sufficient number of photons must be emitted from each surface to achieve “random” behavior for emission. Section 3.4 defines surface input.

2.3 Surface Concatenation

Very early versions of the code (named MONT3D) permitted surface concatenation. Current versions do not due to the limitations associated with error checking and restart (it is impossible to recover restart information from concatenated information, as some information is lost during the concatenation process). If it is desired, concatenation can be done by *a posteriori* by operating on the exchange factor matrices, but this requires manual manipulation by the user.

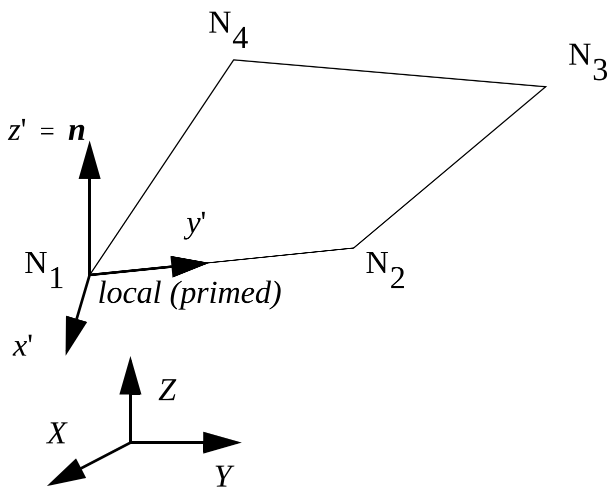


Figure 2.3 Global-Local Coordinate Systems

2.4 Material Properties

Material properties are defined and allocated to surfaces by specifying a material property for each surface. Multiple surfaces may have the same material type. When specifying material properties three items must be defined: material type, material emission, and surface interaction curves. Material types are presented in section 2.4.1, material emission is presented in section 2.4.2, and material property curves that define photon/material interactions are presented in section 2.4.3. The cosine power dependence of the outgoing directional distribution function is presented in section 2.4.4. Section 3.6 provides the details of material property input, and section 3.7 provides the details of material property curve input.

2.4.1 Material Types

Different material types are available to provide flexibility in the material model. These material types allow different combinations of emission and photon/material interactions to be defined. A separate material must be defined for each set of material emission and photon/material interactions. Material property curves for photon/material interactions may be input in tabular form, allowing great flexibility in behavior. A material is defined by a material type number that must be unique and a member of the full set such that the material type ε *{1,NUMMAT}*, where *NUMMAT* is the number of material types, and a material name that is not used in the code but provided simply for convenience.

2.4.2 Material Emission

Material emission is defined by material type and determines how photons are emitted directionally from a surface. Five possibilities exist for emission, as defined in Table 2.1. Explanation of material emission by type is provided below.

Material types -1 and -2 are intended for perfectly reflecting surfaces that do not participate in transport. No emission occurs for these material types. The only difference in these material types is that for type -1, all zeros are written into the row of the exchange number matrix, while for type -2 a single 1 is written into the diagonal position of the row of exchange number matrix and

Table 2.1 Material Type Emission Specifications

MTYPE	Material Emission
-2	No emission, F_{ii} = 1, all other F_{ij} = 0 written
-1	No emission, all F_{ij} = 0 written
0	Emission in θ according to ε(θ,IB)
1	Beam emission in direction {EX, EY, EZ}
2	Emission according to function *fcn* in *subroutine getang*

all other elements in that row are zero. To effect a plane of symmetry, a material type of -1 or -2 should be selected, and the surface interaction curves should all be constants at a value of zero, but with a specular reflectance of 1.

Material type 0 is the default material type, where emission occurs randomly in ϕ and is distributed in θ according to ε(θ) for each band. Ordinarily, material types of 0 are used for all materials except planes of symmetry.

A material type of 1 specifies "beam" or collimated emission in a constant direction. It is typically used to model "beam" solar radiation incident from the direction of the sun, or other collimated radiation such as may emanate from a laser or other optically emitting device. The emission occurs directly in global coordinates that are specified during input.

A material type of 2 causes emission to occur as specified in subroutine getang. This is a very special material type that requires coding and compilation, and should only be used by the sophisticated user when no other way of accomplishing the desired function exists.

2.4.3 Surface Interactions and Material Property Curves

The material properties curves that define a photon's interaction with a surface are defined in terms of a local spherical coordinate system. Figure 2.4 defines the cone or polar angle, θ, and the azimuthal or equatorial angle, ϕ.

Photon surface material interactions of five types may occur. Figure 2.5(a) depicts a "specular" transmission, whereby a photon passes straight through a specularly transmitting surface, with no change in direction. Figure 2.5(b) depicts a "diffuse" transmission, where the photon is transmitted, but its outgoing direction is uniformly distributed in solid angle, weighted by projected surface area. This is akin to a diffuse reflection, but from the "back" of the surface. Figure 2.5(c) depicts a specular (mirrorlike) reflection where "the angle of incidence is the angle of reflection," and Figure 2.5(d) depicts a diffuse (random) reflection, with an equal probability of reflection into any "direction." (Here again, we have exercised editorial license with this statement. In truth, a diffuse reflection has an equal probability of reflection in any solid angle, weighted by projected surface area). Finally, the photon may be absorbed.

All material property curves are independent of azimuthal angle, ϕ, but may depend upon cone angle, θ, or be constant over θ. All properties are defined as constant (gray) within a particular radiative band *k*, so that Kirchoff's law applies within each band. Explicitly:

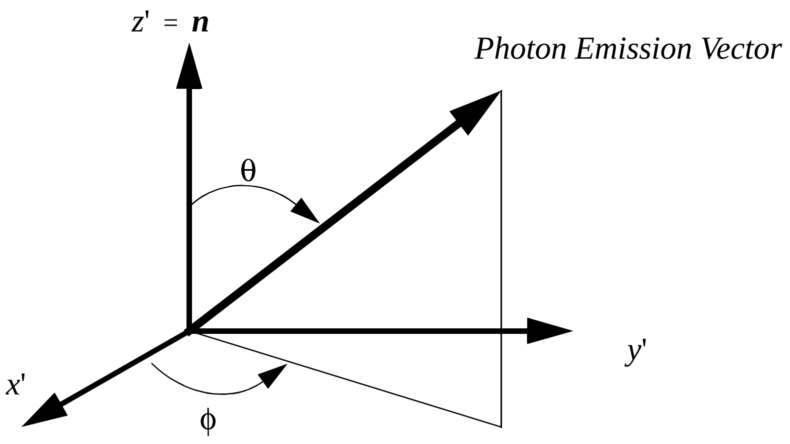


Figure 2.4 Local Material Coordinate System

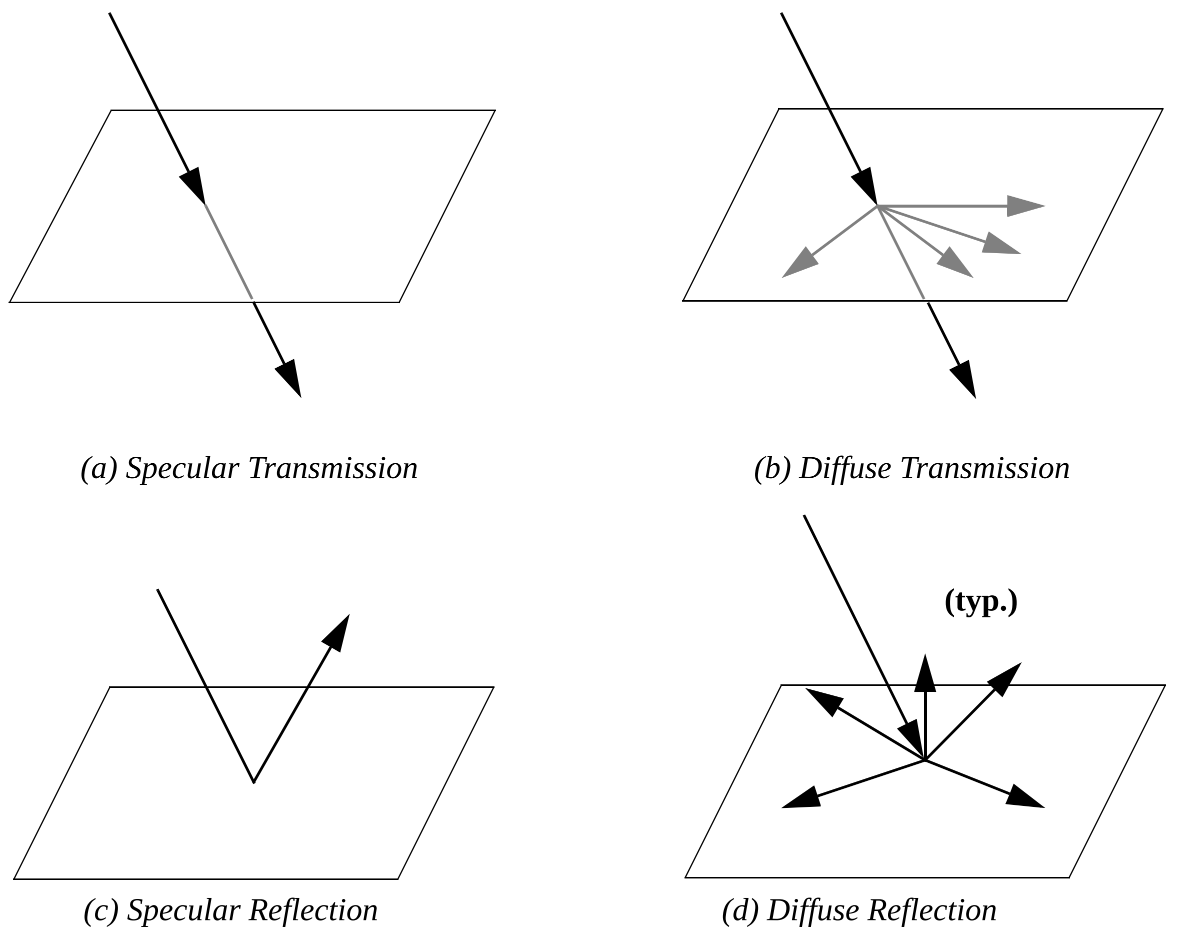


Figure 2.5 Photon-Material Interactions

$$\varepsilon^k(\theta) = \alpha^k(\theta) = 1 - \tau_s^k(\theta) - \tau_d^k(\theta) - \rho_s^k(\theta) - \rho_d^k(\theta) \quad (2.1)$$

where:

$\varepsilon^k(\theta)$	= emittance in wavelength band *k* at outgoing cone angle, θ
$\alpha^k(\theta)$	= absorbance in wavelength band *k* at incident cone angle, θ
$\tau_s^k(\theta)$	= "specular" transmittance in wavelength band *k* at incident cone angle, θ
$\tau_d^k(\theta)$	= "diffuse" transmittance in wavelength band *k* at incident cone angle, θ
$\rho_s^k(\theta)$	= specular reflectance in wavelength band *k* at incident cone angle, θ
$\rho_d^k(\theta)$	= diffuse reflectance in wavelength band *k* at incident cone angle, θ

Hereinafter, the explicit dependence upon wavelength band *k* is dropped, and an implicit dependence is carried. Note that all surface interaction properties may be considered in terms of probability, i.e. $\rho_s(\theta)$ is the probability that a photon of incident angle θ will be "specularly" reflected, $\rho_d(\theta)$is the probability that a photon of incident angle θ will be diffusely reflected, etc. As the emittance and absorptance within a given wavelength band are equal by virtue of Kirchoff's law, these properties are determined as the complement of the others. Thus, specification of the specular and diffuse reflectances and transmittances uniquely defines the surface interaction properties within one wavelength band for a material.

Figure 2.6 depicts the material properties as functions of the incident cone angle, θ_i within a wavelength band *k*. At any particular value of θ, e.g. θ = θ*, the incident photon has the following probabilities: $\tau_s(\theta)$, $\tau_d(\theta)$, $\rho_s(\theta)$, $\rho_d(\theta)$, and $1-\tau_s(\theta)-\tau_d(\theta)-\rho_s(\theta)-\rho_d(\theta)$ for specular and diffuse transmission, specular and diffuse reflection, and absorption (equal to emission), respectively. Each of these curves may be input as constant, independent of incident cone angle, or as a function of incident cone angle for every wavelength band for each material. For curves, the computer code parabolically interpolates between each 3 successive points entered. When entering curves, care must be taken to: (1) include bounding points of $\theta = 0^o$ and $\theta = 90^o$ (since no extrapolation is done), and (2) to include enough points, varying smoothly, to result in good interpolation (i.e., discontinuous jumps must be input as "steep" parabolas with 3 non-coincident points used to define the jump).

2.4.4 Diffuse Cosine Power Dependence

In the present material model, the probability for emission or re-emission (i.e. reflection) of a photon at angle θ is modeled as proportional to $\cos^r(\theta)$. Usually r is picked as 1 to model diffuse (Lambertian) behavior. Electromagnetic theory indicates that r is usually < 1 for metals, and r is usually > 1 for nonmetals. An isotropic distribution in θ in outgoing energy is obtained by setting r = 0. The effect of r on the θ distribution is shown in Figure 2.7. This figure displays $\cos^r\theta$ as a function of θ. The curves have been normalized to their areas. The normalization is such that for r = 0 (isotropic distribution), the curve has the value of 1 at all angles. It should be noted that a value of r other than 1 can cause the reciprocity relation of equations (1.5) and (1.6) to be violated; therefore use of anything other than r = 1 should be approached with extreme circumspection.

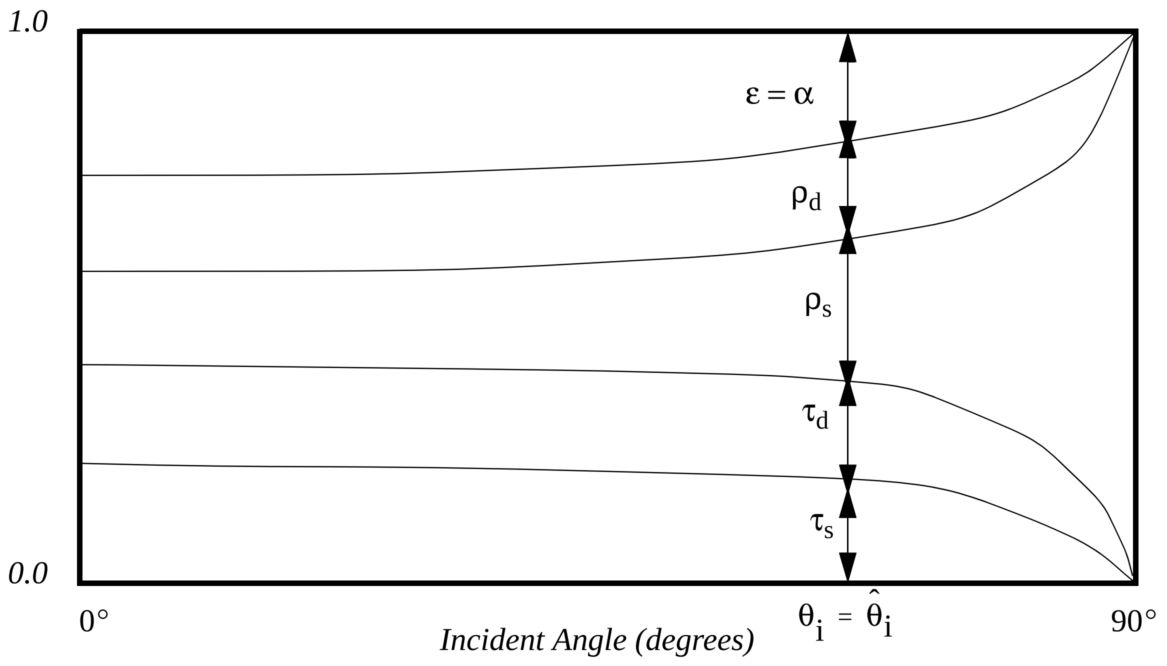


Figure 2.6 Material Properties vs. Incident Cone Angle

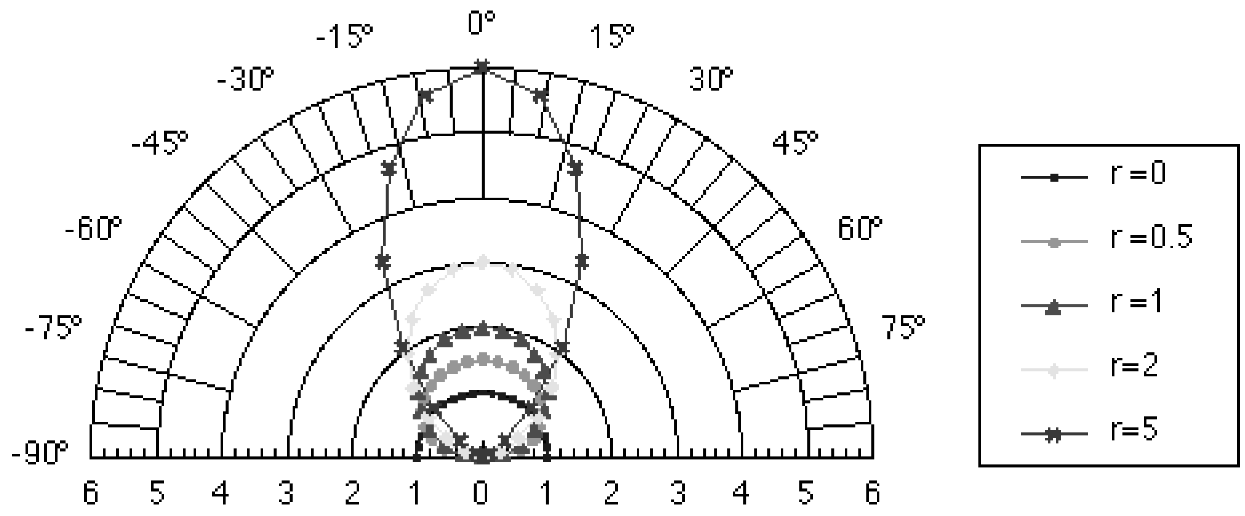


Figure 2.7 Cosine Power Dependence of Emission as a Function of r and θ

The present material model provides for the specification of separate r's for diffuse reflectance and transmittance for each material; see section 3.6.

2.5 Grid Tracing

To reduce CPU time tracing photons, a logical grid is placed over the geometry, and each photon is traced through these logical grid cells until the first intersection point is encountered. This situation is depicted in 2-D in Figure 2.8. The algorithm was suggested in 1986 by Dave Margolies of the Lawrence Livermore National Laboratory. This algorithm has resulted in significant reductions in execution time.

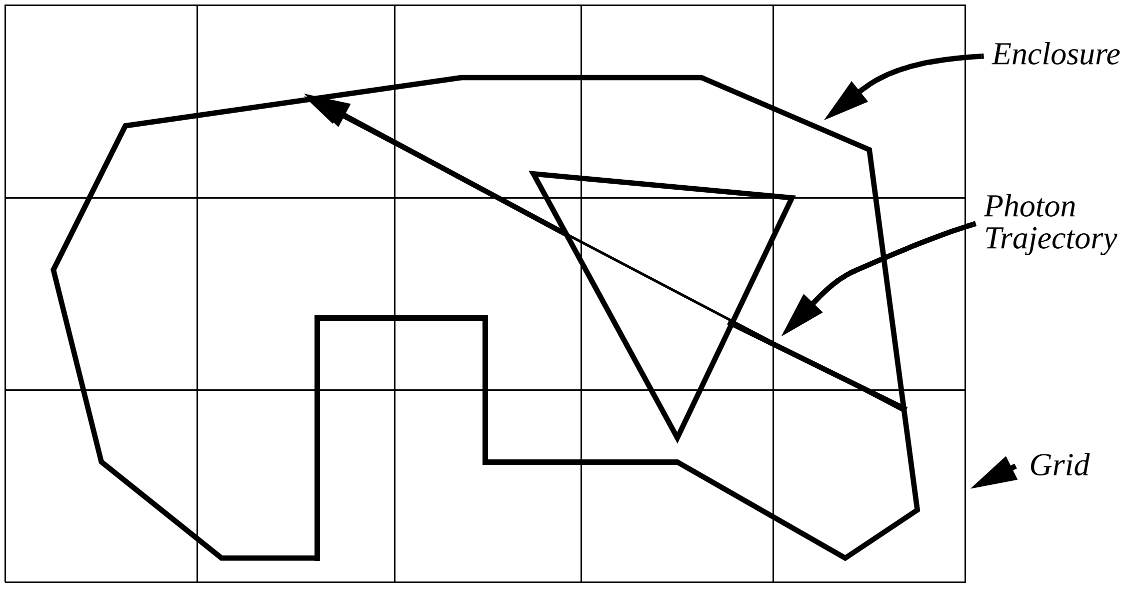


Figure 2.8 2-D Illustration of the Grid Tracing Algorithm

Previous versions of the code permitted non-uniform grid spacing in each of the coordinate directions. The current version of the code permits only uniform grid spacing in each of the coordinate directions, but the uniform spacing may vary in each of the coordinate directions. The optimum grid differs from problem to problem, and must be empirically determined by the user. Zeeb [Zeeb and Burns, 1999] recently has improved the tracing algorithm and thoroughly investigated optimal grids. A good starting guess for an optimal grid for moderate problems (several thousand surfaces) is 25 X 25 X 25. Also, the performance curve versus grid size is very flat, indicating that a wide range of grid sizes yields nearly optimal performance. The input for defining the tracing grid is described in section 3.2.5.

2.6 Number of Photons, Convergence and Accuracy

As Monte Carlo techniques are statistical in nature, “enough” photons must be emitted from each surface to yield a statistically accurate result. This number depends upon the geometry and, to some extent, upon the material properties. As a general rule, greater numbers of surfaces require greater numbers of photons emitted to achieve similar accuracy. Execution time increases linearly with number of photons. For moderate problems (about 1,000 surfaces), it has been found that on the order of 100,000 photons per surface are required to achieve exchange factors accurate to within about one percent. It is typical to observe convergence in a particular exchange factor as shown in Figure 2.9. The user is cautioned that false convergence may be indicated when comparing two values on the curve as shown. It is therefore wise to check the entire matrix of exchange factors for consistency (reciprocity) at several numbers of photons.

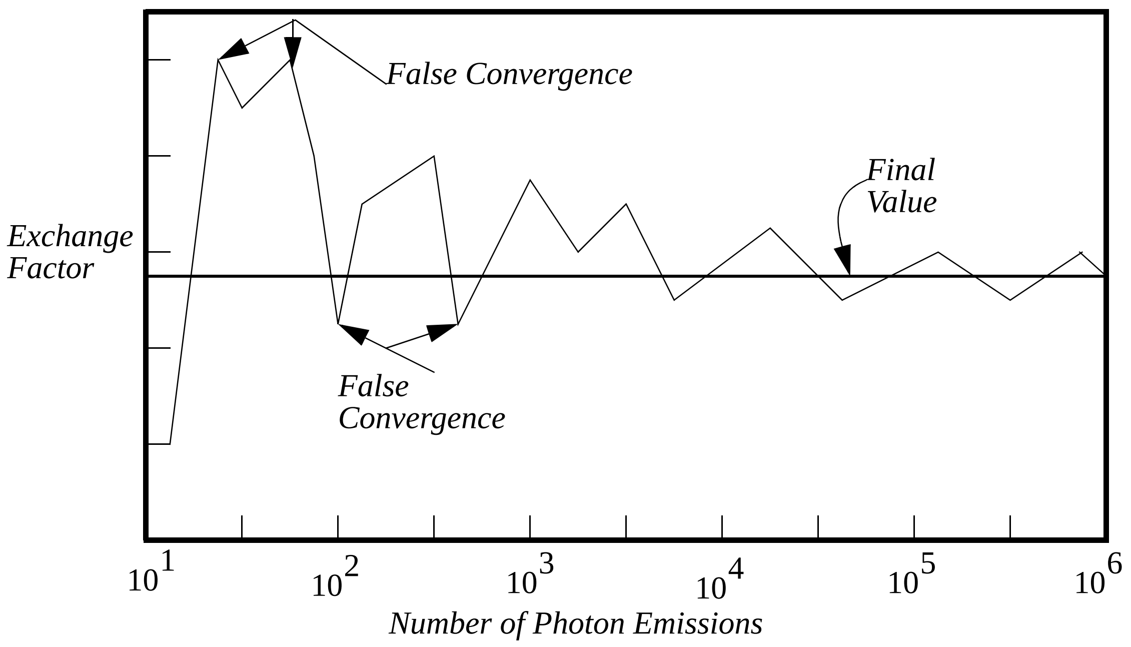


Figure 2.9 Convergence vs. Number of Photon Emissions

To estimate the number of photons required to achieve a given level of accuracy, Table 2.2 is provided. The table gives, for each exchange fraction F_{ij}^k, the number of photons, N_i^k , which

Table 2.2 Accuracy in Exchange Fractions

Exchange Fraction	*Level of Accuracy* 1%	2%	5%	10%	50%
10^{-3}	*38,377,584*	*9,594,396*	*1,535,103*	*383,776*	*15,351*
10^{-2}	*3,803,184*	*950,796*	*152,127*	*38,032*	*1,521*
10^{-1}	*345,744*	*86,436*	*13,830*	*3,457*	*138*

must be emitted from surface *i* to achieve 95% confidence that the exchange fraction is within: 1%, 2%, 5%, 10% and 50% of the exact answer. The numbers of photon emissions per surface are calculated from the formula for confidence intervals, C_{ij}^k, for the exchange fraction from surface *i* to surface *j* in wavelength band *k* (F_{ij}^k), derived by Maltby [Maltby, 1990]:

$$C_{ij}^k = Z\sqrt{\frac{1 - F_{ij}^k}{N_i^k F_{ij}^k}} \qquad (2.2)$$

where Z is taken from the standard normal tables, and is 1.96 for 95% confidence. Equation (2.2) yields the fractional accuracy in $F^k{}_{ij}$ (n.b., 100 time this value yields the percent accuracy). LSMONTE is formulated to attain a specified accuracy for each row *i* of the exchange factor matrix. The program is constructed to loop over successive emissions from each surface *i* if a preset accuracy tolerance is not met after a full surface emission. To explain this, we first note that equation (2.2) provides the confidence interval for only element *ij* of the exchange fraction matrix, when emitting additional photons actually increases the accuracy of all elements in row *i*. Equation (2.2) is modified to account for this with the rationale that exchange fractions affect the accuracy proportional to their size. Thus, we weight each confidence interval by its exchange fraction, sum and then average by dividing this amount by the total number of sides, N_s to yield the *ad hoc* row confidence factor for row *i*, C_i^k :

$$C_i^k = \frac{1}{N_s}\sum_j C_{ij}^k F_{ij}^k = \frac{Z}{N_s}\sum_j \sqrt{\frac{F_{ij}^k\left(1 - F_{ij}^k\right)}{N_i^k}} \qquad (2.3)$$

If the confidence (as a fraction) for emissions from surface *i* is not met after a full surface emission, then the program continues to perform full surface emissions until either the specified confidence is met, or a maximum number of full surface emissions have occurred. This feature can be used with the restart option to effect a specified accuracy for the surfaces, balanced against CPU usage.

A rough guess of the size of the exchange fractions is the reciprocal of the number of surfaces in the input file. This yields the “average” exchange fraction size, since the sum of any row of the exchange fraction matrix is 1. The number of photons required to be emitted to achieve an “average” level of accuracy may then be estimated from equation (2.2), or obtained from Table 2.2 through interpolation or extrapolation.

Errors in the temperatures calculated from radiative flux balances are smaller than errors in the exchange fractions due to the fourth-root dependence of temperature upon radiative flux. For small errors, one may expect the errors in temperatures to be about one-fourth of the errors in fluxes. Emitting an equal number of photons from each surface may result in a waste of computer time, since some surfaces contribute little to the radiative exchange. A better approach would be to apportion the numbers of emissions to each surface based upon its estimated power output. This is an approach which requires judgement gained through experience with specific geometries, since the power outputs are generally not known *a priori*. In any case, the above approach provides a “potentiometer” which can be used judiciously to adjust solution accuracy. Section 3.2.2 provides the details of the input for photon control and accuracy.

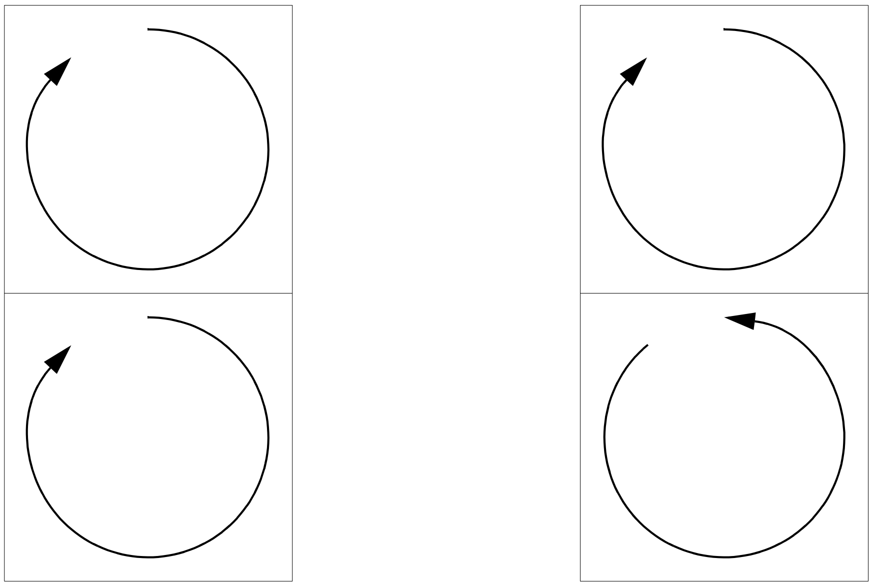
2.7 Debugging, Leak Checking, Lost Photons, and Trajectory Output Capabilities

The code is equipped to generate information useful in debugging input files. A file containing all the information in the input file (geometry and material properties) is written to disk during the input phase of extension *.plt*. This “plot” file may be used with the stand-alone graphics program MPLOT [Burns, 2000] to display the geometry and the material property curves.

An error in the specification of the geometry often results in a “leak” or hole in the enclosure, through which photons may be transmitted and “lost.” Leaks may be caused due to disjoint surfaces, missing surfaces, incorrect node numbering on a surface, misplaced nodal points, or insufficient precision in specifying coordinates. During the input phase, the geometry is checked for leaks, and results are written to the output file. An ASCII file of potential leaks, of extension *.lks*, identified by type (severity of leak), is written for 3-D geometries. This file may be used with MPLOT to highlight surfaces and sides of surfaces which have been identified as potential problems or leaks. The three types of errors identified are:

Error 1- Reversed Edge:

This error occurs when two surfaces share the same edge. If the surfaces are not defined so that both their surface normals are pointing inward or outward, a reversed edge occurs. An example is given in Figure 2.10. In Figure 2.10(a), the edge between the two surfaces is correct and both surfaces have normals pointing into the page. In Figure 2.10(b), the edge is reversed and the normals of the two surfaces are pointed in opposite directions. To see the type of problem this creates, remember that the surfaces are transparent to photons that hit the back side of the surface. These surfaces are oriented for photons coming in opposite directions. This type of error almost certainly means the enclosure will loose excessive photons resulting in an error termination.


(a) correct edge (b) reversed edge

(Arrows represent direction of increasing node number)

Figure 2.10 Example of a Reversed Edge

Error 2 - No Match Found:

This error occurs when no match was found connecting at least one side of that surface to the side of another surface. This may or may not be okay.

Error 3 - Slip Surface

This error occurs when an edge goes through a node point instead of terminating at it. Although this is sometimes a fatal error, it often is not. If the slip surface creates no “holes” in the geometry, then it should not cause the enclosure to loose photons.

When photons are lost, the endpoints of each photon ray are written to a separate file, of extension *.lst*, which may then be read by the MPLOT program. Then trajectories of the lost photons may be displayed on the geometry. Because there is no terminus of the ray, a fictitious endpoint is used.

An option is also available to write trajectory information to an output file, of extension *.trc*, which may be subsequently read and plotted by the program MPLOT. This feature is useful in obtaining a “feel” for the underlying physical processes, and to ascertain that the simulation is proceeding as planned. There is a copious amount of information written to the output file during the exercise of this option, so it is suggested that the total number of photons emitted per surface be 1,000 or fewer.

For more information, the reader is urged to consult the MPLOT documentation [Burns, 2000].

2.8 Restart Capability

The code has been designed to be restarted from a previously computed state. For example, the code may, albeit rarely, “crash” during execution (for any of a host of reasons). Alternatively, the code may run to completion, and subsequent examination of the answers indicates that they are not sufficiently accurate. In either case, it is desirable to begin a new simulation from the last available state so as to take advantage of previous work. This prevents waste of computer resources in recomputing information already available. A simulation may be restarted multiple times, until the desired level of accuracy is attained.

To effect this, the current state of the solution must periodically be written to disk, so that it will be available for a restart run. The information is written in block matrix form, a block of rows at a time, to the disk file of suffix *.nij*. A “crash” file of suffix *.crh* allowing the code to be restarted from a crash is also periodically written, and is deleted if the run completes successfully as it is then unnecessary to effect a restart. Section 3.2.3 in the following chapter provides additional detail of the control which can be exercised over writing states to the output file.

2.9 Pseudo-Random Numbers

The pseudo-random number generator used is a lagged-Fibonacci generator. The generator is a generalization of shift register sequences described by Golomb [1982]. A number of investigators have examined the properties of the lagged-Fibonacci generator, and established the conditions for it to be robust and to generate pseudorandom sequences of high quality [Anderson, 1990; Brent, 1992; Pryor et al., 1994; Marsaglia, 1985; Mascagni et al., 1995; Zeeb and Burns, 1997; Burns and Pryor, 1999].

To obtain good sequences of numbers, in particular, to pass the “birthday spacings test,” a long generator must be used [Brent, 1992]. To ensure this, a 127-long, lagged shift register sequence is used in the code, yielding uncorrelated, non-repeating random sequences with a subperiod of about 2 x 10^{38}, sufficient for virtually any application. It is also directly amenable to parallelization. The generator uses an array of seeds which are generated using a bitwise binary shift register. The initial seed can be specified by the user, generated from the current time or set to a default debugging value. See section 3.2.3 for more details on defining the initial seed.

Each initial seed creates a different sequence of random numbers so different answers are obtained. However, if “enough” photons are emitted to achieve “statistical” convergence, then the answers, whatever the initial seed, will be identical to within a statistical tolerance. It is not possible to traverse the same sequence of random numbers after a restart run involving emission from more than one surface. Thus, if a run is done with 20,000 photons emitted from each surface, the answers will be different than if an initial run is done with 10,000 photons per surface followed by a restart from this final state where 10,000 additional photons are to be emitted. However, the comments above pertaining to convergence do apply. If “enough” photons are emitted, then the answers will converge (to within a statistical tolerance) to a final state independent of the order of emissions.

3. INPUT DECK

The following pages contain the instructions necessary to enable the user to construct the input file (data deck) required by the codes. Default values are used if the input file contains blanks (read as zero by FORTRAN). Input lines are limited to 80 columns in width.

3.1 Title Card

Cols.	Format	Entry	Note
1-48	A48	Heading to appear on output.	1

Note:

1. Cards with the character "&" in column 1 can be placed anywhere in the data deck for use as comment cards or as spaces. All such cards are ignored.

3.2 Control Cards

3.2.1 Card 1 - physical parameters

Cols.	Format	Entry	Notes
1-10	I10	Number of nodal points (NUMNP).	
11-20	I10	Number of surfaces (NSURF).	
21-25	I5	Number of materials (NUMMAT).	1
26-30	I5	Number of wavelength bands (NBANDS). (DEFAULT: NBANDS = 1).	1
31-35	I5	Number of material curves (NCURVE).	2

Notes:

1. Radiative properties are defined as constant within individual wavelength bands.
2. Material properties may be input as either constant within a wavelength band, or varying by means of a curve. See section 3.7 for additional information on the input of material property curves.

3.2.2 Card 2 - photon control

Cols.	Format	Entry	Notes
1-10	I10	Number of photons emitted per band per surface per emission loop (NPHTON).	1
11-15	I5	Maximum number of reflections allowed per photon before a warning is issued (NREFS): (DEFAULT: NREFS = 100).	2
16-20	I5	Maximum number of warnings per surface before the run is aborted (NWARNS): (DEFAULT: NWARNS = 100).	2
21-25	I5	Maximum number of lost photons per surface (NLOST): (DEFAULT: NLOST = 100).	3

26-30	I5	Maximum number of photon convergence loops (NPLOOPS): (DEFAULT: NPLOOPS = 1).	4, 5
71-80	F10.0	Default convergence tolerance (ERRDEF): (DEFAULT: ERRDEF = 1 x 10^{-2}) (Note: ERRDEF is entered at the end of the line, not immediately after NPLOOPS.)	4, 5

Notes:

1. NPHTON is the "base" number of photons per surface per wavelength band per photon loop. NPHT is the photon emission number multiplier that is input for every surface. Thus, in each band, for each surface, NPHT*NPHTON number of photons are emitted per photon convergence loop. In a restart run, new photons are emitted - thus, if this is the second restart run, and NPHTON base photons were emitted in each of the previous two runs, an additional NPHTON base photons are emitted, bringing the total to 3*NPHTON.
2. The total number of reflections before a run is aborted will be 1 greater than NREFS * NWARNS on any surface.
3. Occasionally, due to precision problems, a photon is "lost" (i.e., no receiving surface is found for a given photon path). This input variable specifies the number of such occurrences on any surface before the run is aborted.
4. NPLOOPS is the maximum number of full surface photon convergence loops. In each full surface emission, for each band, NPHT*NPHOTON photons are emitted. If convergence to the specified tolerance for the surface (ERRMAX(N) - see section 3.4) is not attained within NPLOOPS full surface emissions, then a warning is printed to the screen, and execution continues with the next surface. If 0 is entered for NPLOOPS, only one photon loop will be executed.
5. ERRDEF is the default tolerance for convergence of the surface exchange fractions. This may be overridden for (a) particular surface(s) during surface input as described in section 3.4.

3.2.3 Card 3 - run control

Cols.	Format	Entry	Notes
1-5	I5	Restart write increment (NINCR): (DEFAULT: NINCR = 100)	1
6-15	I10	Initial seed for the random number generator (INSEED): INSEED < 0: Use the default, internal value for the initial seed, INSEED = 0: The initial seed is obtained from the time and date, or INSEED > 0: Use the value INSEED for the initial seed.	
16-20	I5	Number of CPU's (NCPU). (DEFAULT: NCPU = 1)	2
21-30	10X	Skip.	
31-40	E10.0	Cone angle interval for numerical integration of material properties (DELT): (DEFAULT: DELT = 0.01 degrees).	3
41-50	E10.0	Tolerance for warnings of noncoplanar surfaces (SPLITOL):	4

		(DEFAULT: SPLITOL = 0.0001).	
51-60	E10.0	Tolerance for warnings of area tolerance of split surfaces (AREATOL): (DEFAULT: AREATOL = 0.0001).	5

Notes:

1. After emission and tracing from every NINCR surfaces has completed, the block of NINCR rows of the exchange factor matrix and the restart "crash" file are written to disk. Thus, it is possible to restart only every NINCR states. Selecting a small value of NINCR will ensure that restart information is written frequently to disk. The value of 100 is appropriate for NINCR.
2. NCPU is applicable only in the parallel version, and is ignored in the serial version of the code.
3. DELT = $\Delta\theta$: the increment used in numerically integrating the cumulative distribution function for emission versus cone angle. The acceptable range is: $1E\text{-}7 \le DELT \le 0.1$. If a value outside this range is entered for DELT, DELT is set to the default value of 0.01. Note that very small values of DELT result in unnecessary consumption of excessive computer time.
4. In order to reduce the loss of photons, the four nodes of a quadrilateral surface must be coplanar. The code enforces coplanarity by dividing each quadrilateral element into two triangles. (By definition a triangle is coplanar.) This division is transparent to the user. The dot product of the surface normals of both triangles is then calculated. If the dot product differs from one by more than a specified tolerance (SPLITOL), indicating the two triangles are non-coplanar to this degree, a warning message is printed, as this may be indicative of a potential problem with the geometry. The range of acceptable values is: $1.E\text{-}20 \le SPLITOL \le 0.1$. If a value outside of this range is entered, SPLITOL is set to the default value of 0.0001.
5. All quadrilateral surfaces are split into two triangles from which emission is done separately. Surfaces may be split along the line between nodes 1 and 3 or nodes 2 and 4. In LSMONTE, surfaces are arbitrarily split along the line between nodes 1 and 3. However, a check is made to ensure that splitting along the line between nodes 2 and 4 produces a similar total area. If these areas are almost identical, then the surface possesses good geometrical properties for splitting. A poor match between the two total areas usually indicates that the surface is poorly defined geometrically (could be significantly non-coplanar or defined with three nodes colinear) and indicates that the surface should be redefined by the user, usually by partitioning it into smaller quadrilaterals or triangles. If the difference in areas exceeds a specified tolerance (AREATOL), a warning message is printed. The range of acceptable values is: $1.E\text{-}20 \le AREATOL \le 0.1$. If a value outside of this range is entered, AREATOL is set to the default value of 0.0001.

3.2.4 Card 4 - toggles

Cols.	Format	Entry	Notes
		Output print control code (IPRINT(I)) as follows:	1
1	I1	IPRINT(1) - exchange fractions written to output file.	2
2	I1	IPRINT(2) - lost photons written to output file.	3
3	I1	IPRINT(3) - grid information written to output file.	4
4	I1	IPRINT(4) - complete material property information written to output file.	5
5	I1	IPRINT(5) - unused at present.	

6	I1	Data check code (IDATA): IDATA = 0: Normal execution, exchange factors are calculated, or IDATA = 1: Data check only, exchange factors are not calculated.	6
7	I1	Trajectory control code (ITRACES): ITRACES = 1: Trajectory information written to disk file (*.trc* extension), or ITRACES = 0: Trajectory information not written to disk file, (DEFAULT: ITRACES = 0).	7

8	I1	Trajectory control code (IRESTART):
		IRESTART = 1: Restart is done.
		IRESTART = 0: No restart is done.
		(DEFAULT: IRESTART = 0).
			8

Notes:

1. In general, the information is printed if IPRINT(I) = 1 and is not if IPRINT(I) = 0.
2. If IPRINT(1) = 1, then exchange fractions are printed in the output file. If I PRINT(1) = 0, they are written only to the binary exchange matrix files. The user is cautioned that exchange factors may be extremely large, requiring a long time to write and very large disk capacity.
3. Lost photon information is written by default to the file *.lst*. If this option, IPRINT(2) = 1, then lost photon information is also written to the output file.
4. Grid information contains a list of all surfaces found partially or wholly within grids. Grid information is written to the output file if this option, IPRINT(3) = 1. The user is cautioned that this may produce a very large amount of output.
5. Material property information is by default written to the plot file (.plt). When setting this option, IPRINT(4) = 1, this information is also written to the output file.
6. It is often useful before beginning a run, especially a large one, to perform a data check and examine the output file for irregularities. To accomplish this, prepare the input file, and initiate a run with IDATA = 1. If, upon examination of the output, everything appears fine, set IDATA = 0 and proceed with the “real” run.
7. The program dumps trajectory information to the *.trc* file if this option is set. This file is used to “view” the trajectories using the MPLOT program, useful in establishing physical intuition. If chosen, a copious amount of information is printed to the file; the user is therefore advised to select this option only for very few photons per surface.
8. If IRESTART is 1 and no *.nij* file from which to initiate a restart run exists, a warning message is printed to the output file and the console, and a new run proceeds. If IRESTART is 0, no restart run is ever done - if a *.nij* file from which to initiate a restart run exists, it is overwritten by the file from a new run - the user is cautioned that this will result in the loss of information.

3.2.5 Card 5 - Grid Dimensions

Cols.	Format	Entry	Notes
1-5	I5	Number of grid cells in the X-direction (NGX).	1
6-10	I5	Number of grid cells in the Y-direction (NGY).	1
11-15	I5	Number of grid cells in the Z-direction (NGZ).	1

Notes:

1. For large geometries involving 1,000 to 5,000 surfaces, grid dimensions of NGX = NGY = NGZ = 25 are suggested as near optimal, resulting in near minimum total CPU time.

3.3 Nodal Point Data

Cols.	Format	Entry	Note
1-5	I5	Node point number (N).	
11-10	I5	Increment in number of points to be generated (INC):	1
		(DEFAULT: INC = 0, no nodes generated).	
11-30	E20.0	X-coordinate: X(N).	
31-50	E20.0	Y-coordinate: Y(N).	
51-70	E20.0	Z-coordinate: Z(N).	

Note:

1. Nodal points are generated in increments of INC from *the previous node input to the current node*. The coordinates are obtained by linearly interpolating all coordinates between the ones input on the previous card and ones input on the present card. Care must be taken such that there are exactly an integer number of generated nodes between the present node number and the one input on the previous card. Note that, as the program calculates and fills nodes, these nodes should not also be input elsewhere.

3.4 Surface Data

Cols.	Format	Entry	Notes
1-5	I5	Surface number (N).	1
6-10	I5	Node N_1: NODES (1,N).	
11-15	I5	Node N_2: NODES (2,N).	
16-20	I5	Node N_3: NODES (3,N).	
21-25	I5	Node N_4: NODES (4,N).	
26-30	5X	Skip.	
31-35	I5	Number of surfaces to be generated after current surface: NMISS.	2
36-40	I5	Increment of generation: INC.	2
41-45	5X	Skip.	
46-50	I5	Surface material number: MATNUM(N).	
51-60	10X	Skip.	
61-65	I5	Surface photon multiplier (NPHT):	3
		(DEFAULT: NPHT = 1).	
66-70	I5	Photon increment (INCP).	4
71-80	E10.0	Convergence tolerance for surface: ERRMAX(N):	5
		(DEFAULT for surface: ERRMAX(N) = ERRDEF).	5, 6

Notes:

1. Surfaces can be input in any order. The outward normal must be such that the right-hand rule as explained in section 2.3 applies. Unpredictable and erroneous results may occur if this convention is not adhered to for all surfaces.
2. NMISS surfaces are generated by successively incrementing surface numbers by 1 and by incrementing all 4 node numbers by INC.
3. NPHT is multiplied by NPHTON on control card 2, the “base” number of photons per surface, to yield the total number of photon emissions per surface per photon loop. If NPHT is negative, no photons will be emitted from the surface. If NPHT = 0, the “base” number of photons per surface, NPHOTON, will be emitted for each surface photon loop, i.e. if NPHT is read as 0, NPHT is set to 1.
4. Similar to INC above. For each missing surface, i, (i = 1 to NMISS) generated, i*INCP extra photons are added to NPHT*NPHTON.
5. This is used as explained in section 2.6 to loop over full surface emissions until either the specified number of full surface emissions have occurred, or convergence to this tolerance, C_i defined in equation (2.3), is achieved, whichever comes first.
6. If no value is input (or a value of 0 is read), then this defaults to the global ERRDEF value input as described in section 3.2.2.

3.5 Wavelength Band Data

CARDS 1 to (NBANDS-1)/8

Condition NBANDS > 1, (omit otherwise)

Cols.	Format	Entry	Note
1-10	E10.0	Wave breakpoint number 2 (micrometers).	1
11-20	E10.0	Wave breakpoint number 3 (micrometers).	1
21-30	E10.0	Wave breakpoint number 4 (micrometers).	1
		.	
		.	
71-80	E10.0	Wave breakpoint number 8 (micrometers).	1, 2

Notes:

1. The first and last wave breakpoints are assumed to be 0.0 and ∞ (1×10^{10}) in micrometers and should not be input. Only breakpoints between 0 and ∞ should be input.
2. No more than eight breakpoints should be input per card. The total number of cards should be (NBANDS - 1)/8.

3.6 Material Input

Material input cards are input by band for each material in order. For example, if 3 materials with 2 bands are used, then all material type data are read in for material 1, band 1; then material 1 band 2; then material 2, band 1; etc. MN and IB are only used as checks to make sure the values are being entered in the right order. If the information is not entered in the correct order the program will terminate with an error. All material input cards must be input before any material property curves are input (these cards control which material property curves are read).

CARD 1, material number and name

Cols.	Format	Entry	Notes
1-5	I5	Material number: NMAT.	1
6-45	A40	Material name: MNAME(NMAT).	2

Notes:

1. Data are input in order of materials by band: one card 1 input per material, and NBANDS pairs of cards 2 and 3 input for all bands, except as noted in note 2 for card 2 below. Thus, if 2 materials and 3 bands are modelled, the sequence of materials, cards and bands is

Table 3.1 Material Card Sequence

Material	Band	Card
1	All	1
1	1	2
1	1	3
1	2	2
1	2	3
1	3	2
1	3	3
2	All	1
2	1	2
2	1	3
2	2	2
2	2	3
2	3	2
2	3	3

2. Material names are not used in the code, except to name and thereby identify materials.

CARD 2, band photon emission - one card per band, grouped with card 3

Cols.	Format	Entry	Notes
1-5	I5	Band number: IB.	1, 2
6-10	I5	Emission type: MTYPE(MN,IB), domain [-2,2].	3
11-30	E20.0	Outgoing X-component for emission in global coordinates: EXE(MN,IB).	4
31-50	E20.0	Outgoing Y-component for emission in global coordinates: EYE(MN,IB).	4
51-70	E20.0	Outgoing Z-component for emission in global coordinates: EZE(MN,IB).	4

Notes:

1. Band data must be input in order, from 1 to NBANDS, except as noted below.
2. If, for the first band input, IB is set to zero, the data input on cards 2 and 3 are taken as constant across all wavelength bands.
3. Table 2.1 provides emission types for materials.
4. The beam emission components are normalized, such that a unit vector is obtained. However, for beam emission, at least one non-zero value must be input. Thus, {EX, EY, EZ} input as {1, 1, 0} is normalized to {0.7071, 0.7071, 0}.

CARD 3, band photon disposition - one per band, grouped with card 2

1-10	E10.0	Specular transmittance, ρ_s: RHOS(MN,IB,ITH).	1, 2
11-20	E10.0	Diffuse transmittance, ρ_d: RHOD(MN,IB,ITH).	1, 2
21-30	E10.0	Specular reflectance, τ_s: TAUS(MN,IB,ITH).	1, 2
31-40	E10.0	Diffuse reflectance, τ_d: TAUD(MN,IB,ITH).	1, 2
41-50	E10.0	r dependence of diffuse reflectance: RDIFFR(MN,IB): (DEFAULT: RDIFFR = 1).	3, 4, 5, 6
51-60	E10.0	r dependence of diffuse transmittance: RDIFFT(MN,IB): (DEFAULT: RDIFFT = 1).	3, 4, 5, 6

Notes:

1. Values must be between 0 and 1. In addition, all values must sum up to 1 or less, as the complement of the sum of these properties is the absorptance.
2. Negative values input are taken as the negative of curves to be read for that material property. For example, inputting -2.0 indicates that material property curve 2 will be read for this material property. Curve numbers must be between 1 and NCURVE, input on control card 1.
3. Note the special handling of the default value 0! Values greater than zero are used as is. If a value of 0 is read (or the input is blank, causing a value of 0 to be read), then the default value of 1 is used. Any negative value input causes a value of 0 to be used.
4. "Diffuse" reflection or transmission occurs distributed in cone angle θ according to the cosine power, r, in proportion to $\cos^r(\theta)\sin(\theta)\, d\theta$. Values of "r" greater than one result in biasing the distribution toward the normal, whereas values of "r" less than one result in biasing the distribution toward the grazing angle.
5. Values of "r" different from 1 have been observed to result in errors in reciprocity, so the user is strongly encouraged to consider this when selecting values of "r" different from one.
6. Lambertian behavior is achieved by using r = 1, the recommended value.

3.7 Material Property Curve Input

Material property curves are input as specified by material. Material property curves are input in sequential order for the number of material property curves specified in the material input.

REPEAT CARDS 1 to NP+1 FOR EVERY CURVE INPUT:

CARD 1, curve information

Cols.	Format	Entry	Note
1 -5	I5	Curve number: NC.	1
6-10	I5	Number of points to be input for this curve: NP.	2
11-50	A40	Name of curve (e.g., Window Specular Transmittance): CNAME.	

CARDS 2 to NP+1, angular input

Cols.	Format	Entry	Note
1-10	E10.0	Cone angle - θ.	2
11-20	E10.0	Curve value.	

Notes:

1. Curves must be input for all negative specifications in material input. NCURVE total curves must be input. The names of the curves are not used anywhere else in the code.
2. NP number of cards with angular data must be input for the curve. Angles must be input in increasing order. All curves must have values input at angles 0 and 90 degrees. All material property values must be in the domain [0, 1].

4. EXCHANGE MATRIX OUTPUT FILE

4.1 General Description

The exchange matrix files contain all of the information necessary to solve for the net radiative exchange as described in Chapter 1.

A single, binary, direct-access file, named by the user, is generated for the exchange matrices. The file is direct access with record size 4*NSURF bytes. Experience is that, for large problems, writing an ASCII file is prohibitively time-consuming. Therefore, to reduce the time it takes to write the file and to conserve disk space (binary files are much smaller than ASCII files), a binary file is written. This file is not, in general, transportable to other systems, so that the thermal balance code must be run on the same system as the exchange matrix file is resident. Finally, the information in the file is not able to be viewed by the user; if one wishes to view the exchange factors, one should set the print flag (IPRINT(1) in column 1 of control card 4, to 1 so that the exchange factors will be printed in the output file). However, the user is cautioned that the time it takes to write this information may be prohibitive for large problems (over about 5,000 surfaces and one band).

The information written to the file follows. The header contains control information and the surface areas. This is followed by the surface emittances for all wavelength bands ordered from lowest band to highest. Next are the exchange number matrices, one for each band, ordered from lowest band to highest. The file is concluded with the wavelength bands.

4.2 Output Format for Exchange Matrix Files

The file of exchange numbers is a constant record length, direct-access, binary file. All quantities are 32 bit. Integers are generally stored as binary in powers of 2, and reals are generally stored in IEEE standard 32-bit floating point format.

4.2.1 Header Card

One record of length NSURF

Format	Entry	Note
Integer*4	Geometry code (NDIM)	1
Integer*4	Number of surfaces (NSURF)	
Integer*4	Factor code (IFACT)	2
Integer*4	Number of wavelength bands (NBANDS)	
Integer*4	Number of materials (NUMMAT)	3

Notes:

1. NDIM is 3, denoting 3-dimensional results.
2. IFACT is 2, indicating that the file contains exchange numbers as opposed to view factors.
3. The number of materials is provided simply as a check - it is not used in the calculation of the radiative exchange.

4.2.2 Surface Areas

One record of length NSURF (the number of surfaces)

Format	Entry
Real*4	Surface areas from 1 to NSURF

4.2.3 Surface Emittances - Data for Each Wavelength Band

(Repeated for each wavelength band, from k = 1 to NBANDS)

For every wavelength band

One record of length NSURF

Format	Entry	Note
Real*4	Surface average emittances, $\bar{\varepsilon}_i^k$, from 1 to NSURF	1

Notes:

1. The surface emittances are averaged over direction (cone angle), i.e., the values are hemispherical emittances.

4.2.4 Photon Exchange Number Matrix - Data for Each Wavelength Band

(Repeated for each wavelength band, from k = 1 to NBANDS)

For every wavelength band

NSURF records, each of length NSURF

Numbers of absorbed photons by surface *i* for all *j*

Format	Entry	Note
Integer*4	Numbers of absorbed photons	1, 2

Notes:

1. The data are presented by row *i* for all columns *j*. That is, the first row of numbers $N_{1,(1 \rightarrow NSURF)}$ is written, then row 2 is written, etc.
2. These files contain the full exchange matrices. The program SMOOTH [Dolaghan et al., 1992] is being updated by Zeeb to process these files into an upper triangle (including the diagonal) of numbers of photons which have been smoothed to obey reciprocity.

4.2.5 Wavelength Breakpoints

For as many records as needed, each of length NSURF

Wavelength breakpoints in micrometers.

Format	Entry	Note
Real*4	Wavelength breakpoints, λ_2,..., $\lambda_{NBANDS-1}$	1, 2, 3

Notes:

1. The first point $\lambda_1 = 0$, is and the last point $\lambda_{NBANDS} = \infty$ are omitted.
2. Multiple records are used, if needed.
3. For a single band, this record is omitted.

5. SAMPLE INPUT FILES

5.1 General Description

This section contains one sample input file to demonstrate the input format. The problem is a 3-D cube (rectangular prism) with black inner surfaces. Close examination of this file should assist the user in understanding how to generate an input file.

5.2 3-D Box

Figure 5.1 shows a 3-D geometry (a cube - not to scale) for analysis. Comparison of the picture with the input file “box.in” shown below illustrates the right-hand rule for 3-D surfaces. One million photons per photon loop are emitted from each surface. The reflectance of the inner surfaces is zero. The nodes are at the vertices of a cube of edge length 10. Note that surfaces 1 through 6 are, sequentially, the bottom, top, left. right, back and front, respectively. The material is named “Black Surface,” consists of one band, is of type 0 (standard emission), has all surface properties zero, and has Lambertian diffuse properties (r = 1) for emission.

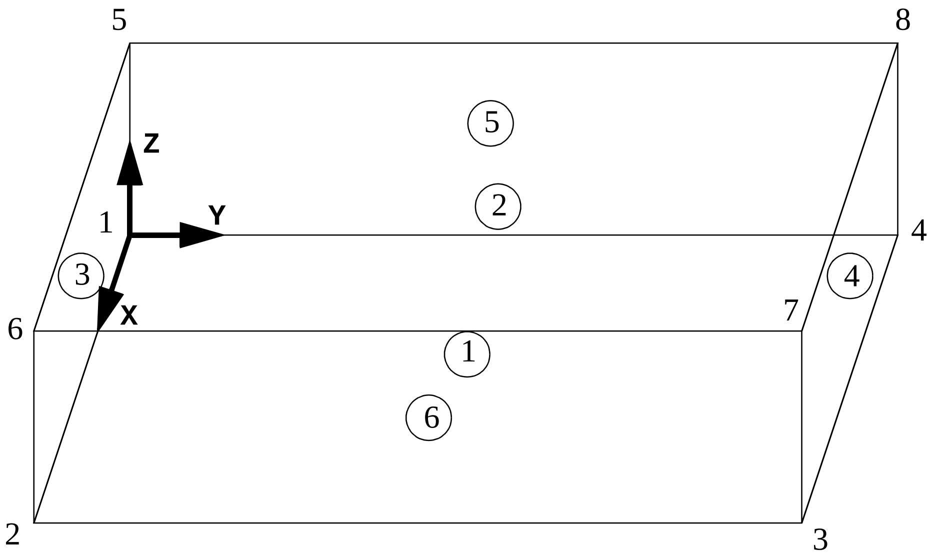


Figure 5.1 3-D Geometry of File “box.in”

```
GROUP 1 TEST 1 BLACK CUBE FILE BOX.IN
&    NUMNP        NUMMAT    NCURVE
&              NSURF    NBANDS
        8         6    1    1    0
&    NPHTN    NWARNS   NPLOOPS
&        NREFS     NLOST
&2 4 6 8(1)2 4 6 8(2)2 4 6 8(3)2 4 6 8(4)2 4 6 8(5)2 4 6 8(6)2 4 6 8(7)2 4 6 8(8
&                                                                    ERRDEF
   100000                 1                                           1.e-6
&NINCR          NCPU                DELT            AREATOL
&       INSEED                             SPLITOL
   1       -1
&IPRNT
```

```
&  ITOGGLE
10000000
& NGX  NGY  NGZ
    1    1    1
& NODES
&   N  INC                 X                   Y                   Z
    1                    0.0                 0.0                 0.0
    2                   10.0                 0.0                 0.0
    3                   10.0                10.0                 0.0
    4                    0.0                10.0                 0.0
    5                    0.0                 0.0                10.0
    6                   10.0                 0.0                10.0
    7                   10.0                10.0                10.0
    8                    0.0                10.0                10.0
& SURFACES
&   N   N1   N2   N3   N4      NMISS INC         MN          NPHT INCP    ERRMAX
    1    1    2    3    4          0    0          1             0    0      0.00
    2    5    8    7    6          0    0          1             0    0      0.00
    3    1    5    6    2          0    0          1             0    0      0.00
    4    3    7    8    4          0    0          1             0    0      0.00
    5    1    4    8    5          0    0          1             0    0      0.00
    6    2    6    7    3          0    0          1             0    0      0.00
& MATERIALS
&NMAT                           MATERIAL NAME
    1Black Surface
&    MTYPE                                        EYE
&  IB                    EXE                                         EZE
    1    0
&      RHOS      RHOD     TRANS     TRAND    RDIFFR    RDIFFT
         0.        0.       0.0       0.0       0.0       0.0
```

6. PROGRAM EXECUTION

6.1 Execution of File "box.in"

The following details the execution of the code with the file "box.in." The output to the screen has been captured and appears below in section 6.2. The command prompt is "pburns n%," where n is the command number. All input from the user is shown bold and italicized.

The executable file here is called *lsmonte* (note that most systems are case sensitive), and is invoked by typing its name - shown at the first prompt. All subsequent output is from the program. In this case, the name of the input file is entered at the prompt from *lsmonte*. Here, the user types the prefix of the input file, which MUST have the extension "in." In this case, the file is "box.in," and the user enters "box." If the program cannot find the input file, this error is trapped by the code, and a diagnostic is printed. Additional detail on the files used by the codes is given in Chapter 8.

The program proceeds with reading of the input file. The heading is printed, along with the version number (here, 1.0), the date of last modification (here, 2000/12/04 or December 04, 2000). Next, the control portion of the input is read. This establishes the numbers of nodes, surfaces (sometimes referred to as elements or surface elements), wavelength bands, materials and grids which are to be read by succeeding subroutines. Then, the input subroutines are executed one-by-one. As each subroutine is entered, information is printed concerning the processing being done, followed by the name of the subroutine in parentheses. This assists in debugging input files. For example, if the program "crashes" after the message "*Processing the node input (subroutine nodin)*" is printed, there is almost certainly a problem with the nodal input (or possibly, the card images have been entered out of order). The successful completion of the entire input phase is then indicated by the message, "i n p u t   p h a s e   c o m p l e t e."

The solution phase occurs next. After the specified number of full surface emission loops for each surface is completed, a message is printed indicating whether the exchange fractions have achieved the specified convergence tolerance. In addition, the wavelength band, surface number, total number of emitted photons (summed over all full surface emissions), and the calculated error are printed. At the end of the run, the overall program statistics are printed, including the total CPU time for the run (in seconds), the total number of photons emitted (and traced to absorption), and the execution rate in number of photons per CPU second.

Finally, upon completion of the run, the message "Stop - program terminated" is issued (PC version only). Then, the command prompt is again issued by the operating system.Then, the output file "box.out" is available; the plot file "box.plt" can be used with the MPLOT program to view the geometry and material properties, etc.

Section 6.3 is the screen output generated by a restart run of box.in. For a restart run, the only difference in the input file is that IRESTART is set to 1. The new input file must also be named the same as in the input file for the initial run, here *box.in*. Specifically, the only change in the input file is in the eighth character of Control Card 4, which is IRESTART:

```
&  ITOGGLE
10000001
```

In each run, one photon loop, each with 100,000 photons emitted per surface, is executed. Each doubling of photons causes the error to decrease by a factor of about $1/\sqrt{2}$; thus the error after the second run, with an additional 100,000 photons emitted per surface, or 200,000 totat photons emitted per surface, is $1/\sqrt{2}$ times the error from the initial run with 100,000 photons emitted per surface.

6.2 Output During First Execution of File “box.in”

```
pburns 1% lsmonte
   ****** lsmonte ******
    Enter prefix for disk files -    20 characters or less
    input file MUST have extension .in
box

Processing the control cards (subroutine ccards)

run title - GROUP 1 TEST 1 BLACK CUBE FILE BOX.IN
LSMONTE Revision: 1.0      Date: 2000/12/04 23:09:40

Processing the node input (subroutine nodin)

Reading the surface input (subroutine surfin)

Calculating surface quantities (subroutine surfcalc)

Processing the wavelength input (subroutine wavin)

Processing the material property input (subroutine matin)

Processing the material curve input (subroutine curvin)

Creating material property sets (subroutine matcalc)

Calculating the inverse CDFs (subroutine cumdis)

Setting up the .nij and .crh files (subroutine nijcrhset)

Creating a grid for the geometry (subroutine gridcalc)

Finding the surfaces in the grid cells (subroutine seging)

Creating the leaks (.lks) file (subroutine lksfile)

Creating the MPLOT (.plt) file (subroutine pltfile)

Calculating the 2D bounding planes (subroutine bplane2d)


        i n p u t   p h a s e   c o m p l e t e
```

```
                  band   surf.  iter.   npht.    error
not converged -     1      1      1    100000    .2066E-02
not converged -     1      2      1    100000    .2066E-02
not converged -     1      3      1    100000    .2066E-02
not converged -     1      4      1    100000    .2066E-02
not converged -     1      5      1    100000    .2066E-02
not converged -     1      6      1    100000    .2066E-02



normal termination




                   s o l u t i o n   t i m e   l o g

         total time for input phase      =  .10000D+01 secs
         total time for solution phase   =  .14000D+02 secs
         total run time                  =  .15000D+02 secs

         (all times measured in clock time [seconds])


    total number of photons lost           =          0
    total number of photons emitted        =  .60000D+06
    photons per cpu second (whole run)     =   40000.000 (photons/sec)
    photons per cpu second (solution pase) =   42857.143 (photons/sec)
Stop - Program terminated.
pburns 2%
```

6.3 Output During Restart Execution of File “box.in”

```
pburns 1% lsmonte
  ****** lsmonte ******
    Enter prefix for disk files -    20 characters or less
    input file MUST have extension .in
box

Processing the control cards (subroutine ccards)

run title - GROUP 1 TEST 1 BLACK CUBE FILE BOX.IN
LSMONTE Revision: 1.0      Date: 2000/12/04 23:09:40

Processing the node input (subroutine nodin)
```

```
Reading the surface input (subroutine surfin)

Calculating surface quantities (subroutine surfcalc)

Processing the wavelength input (subroutine wavin)

Processing the material property input (subroutine matin)

Processing the material curve input (subroutine curvin)

Creating material property sets (subroutine matcalc)

Calculating the inverse CDFs (subroutine cumdis)

Setting up the .nij and .crh files (subroutine nijcrhset)

Creating a grid for the geometry (subroutine gridcalc)

Finding the surfaces in the grid cells (subroutine seging)

Creating the leaks (.lks) file (subroutine lksfile)

Creating the MPLOT (.plt) file (subroutine pltfile)

Calculating the 2D bounding planes (subroutine bplane2d)


        i n p u t   p h a s e   c o m p l e t e




                  band   surf.  iter.   npht.    error
not converged -     1      1      1    200000    .1461E-02
not converged -     1      2      1    200000    .1461E-02
not converged -     1      3      1    200000    .1461E-02
not converged -     1      4      1    200000    .1461E-02
not converged -     1      5      1    200000    .1461E-02
not converged -     1      6      1    200000    .1461E-02



normal termination
```

```
                  s o l u t i o n   t i m e   l o g

         total time for input phase       =  .10000D+01 secs
         total time for solution phase    =  .14000D+02 secs
         total run time                   =  .15000D+02 secs

         (all times measured in clock time [seconds])


    total number of photons lost           =          0
    total number of photons emitted        =  .60000D+06
    photons per cpu second (whole run)     =   40000.000 (photons/sec)
    photons per cpu second (solution pase) =   42857.143 (photons/sec)
Stop - Program terminated.
```

6.4 Execution of Other Input Files

Execution of other input files produces similar output.

7. IMPLEMENTATION

The code runs on Macintosh systems, Windows systems, and various Unix systems selected with options in a Unix *make* file. The system-specific functions that exist in the code are the calls to the elapsed CPU time used only for performance metering, calls to the time and date used to initialize the random number generator (if bypassed the user may supply a seed for the random number generator), and calls to implement command line arguments. A version may be compiled without any of these options, that runs on any Unix system. The code is almost ANSI standard Fortran 77, except for the bitwise intrinsic functions used in the random number generator. We have yet to find a modern compiler without these functions

Binary executable are available via ftp from the Livermore Software Technology Corporation www.lstc.com. Versions exist for Windows, Macintosh, Sun, HP, IBM, and generic Unix systems such as some compilers that run under linux.

The first section below includes details on compilation. The second section contains information on memory usage and storage. The third and final section in this chapter contains information on precision.

7.1 Compiling

The code has been implemented with “stubs” that are subroutines containing system-specific functions, allowing various *make* files. Invoking different *make* files causes various stub files to be selected, with various options. The following modules are common to all versions:

lsmmain.f - contains the main (root) program and subroutines common to input and solution phases

lsmfile.f - contains file processing routines

lsminput.f - contains basic input routines

lsmmat.f - contains material input and initialization routines

lsmpreproc.f - contains input preprocessing (the input routines are split across several modules due to their large size)

lsmsolve.f - contains solution phase routines

lsmonte.com - contains common blocks used in the code for variable storage

lsmonte.par - contains parameters for the code, particularly some parameters set storage size for the executable (see section 7.2 for more information)

7.2 Common Blocks, Parameter Statements and the Size of Arrays

The code uses common blocks for variable storage, making the code more robust and easier to modify. A version of the code has been implemented and tested with dynamic memory allocation, however, the performance was poor and this approach was abandoned. Therefore, this implementation fixes the sizes of arrays by using specific parameters contained in the module *lsmonte.par.* These parameters limit the number of various objects that can be stored, including the number of surfaces, number of wavelength bands and number of materials. To increase the storage size, the parameters must be changed (requires editing *lsmonte.par*), and the program must be recompiled. The larger the arrays, the more memory the program uses. If the arrays become too large, the program may be too large to be stored in available physical memory and the program execution speed will severely degrade as the program is swapped to and from the disk. Or, the code may not execute, if swapping is not supported by the operating system.

In either case, it is recommended that the code be recompiled so as to fit in available physical memory. The parameters used to size arrays are:

iblk - maximum number of surfaces before writing output, used for "blocking" the matrix of exchange factors:

inod - maximum number of nodes

isrf - maximum number of surfaces

ibnd - maximum number of wavelength bands

imat - maximum number of materials

incg - maximum number of grid cells in any coordinate direction

igrd - maximum number of grid cells

iseg - maximum number of surfaces either wholly or partially within all grid cells

During execution, a check is performed to ensure that storage is not exceeded. After the input is read, the code checks that the numbers of nodes, surfaces, wavelength bands, materials, material property curves and grid cells are not exceeded. If any of the storage limits are exceeded, a message detailing the parameters for which storage is insufficient is printed together with the allowable value, and execution terminates. If storage is exceeded, various possibilities for remediation exist:

Storage exceeded for nodes, surfaces, wavelength bands, or materials. Unless the problem size can be reduced for example by removing surfaces and/or nodes that do not participate in radiation or reformulating the problem so as to reduce the numbers of nodes, surfaces and/or materials, the storage parameters must be increased, the code recompiled and the problem rerun.

Storage exceeded for grid parameters. If *iseg* or *igrd* are not sufficiently large, an easy fix is to reduce the size of the grid (see section 3.2.6) to acceptable limits, and rerun the problem - recompilation is not required. However, another possible fix is to increase either *iseg* or *igrd,* or both, recompile the code, and rerun the problem. As *iseg* represents the total number of surfaces contained in the grid, it is dependent upon the number of surfaces, the grid size and the specific geometry of the problem. The code is designed to indicate the total number of segments required, even if this is above the allotted storage.

7.3 Precision

The code is compiled using 64-bit, 8-byte floating point precision. It is desirable to use 64-bit precision, as the specification of co-planar surfaces is particularly susceptible to precision errors, and using 64-bit precision results in fewer photons being "lost."

The exchange fraction files are always written in 32-bit, binary format (note that the only real numbers written are the surface areas and emittances), as this saves storage space.

8. FILES AND FILE USAGE

This chapter briefly describes file usage. For specificity, a prefix of *fn* is used. Table 8.1 gives the files used (either pre-existing or generated during execution) by LSMONTE.

Table 8.1 LSMONTE Files

Unit	Name	Function
1	fn.scr	Input file with comment cards “stripped away.” File is generated by code and deleted before completion of run. This is the file actually read during the input phase.
2	fn.crh	Crash file in binary format, deleted upon successful termination. Contains a list of the photon blocks that have been completed.
3	fn.plt	Plot file, to be used with the program MPLOT [Burns, 2000]. Contains geometry and material property information.
4	fn.lst	Contains lost photon trajectories, if any. This files may be used by MPLOT to display trajectories of lost photons MPLOT [Burns, 2000].
5	stdin	Standard input (keyboard)
6	stdout	Standard output (screen).
7	fn.in	Input file with comment cards, in a format described in chapter 3. This file is used to generate the file of unit 1.
8	fn.nij	Exchange number matrix. These are stored in binary format as described in chapter 4.
9	fn.trc	Trajectory file (written if ITRACES $\neq$ 0, see section 3.2.2). To be used with MPLOT [Burns, 2000] to plot particle trajectories.
10	fn.out	Output file. Contains echo of input and other information as determined by IPRINT. (See section 3.2.4.)
12	fn.lks	Leaks file. To be used with MPLOT [Burns, 2000] to identify potential leaks.

Files for unit numbers 1, 3 to 7, 9, 10 and 12 are ASCII files, and may be read and printed. Files for unit numbers 2 and 8 are binary files, and must be read by other programs. For example, unit 2 is used only by the LSMONTE code. Unit 8 is written by LSMONTE, and read by LSDYNA. The UNIX utility *od* (octal dump) may be used to examine binary files (do a *man* on *od* for instructions).

8.1 Specifying File Names

LSMONTE must have the names specified for all of the files shown in Table 8.1. This section explains the naming conventions and methods of specifying these file names.

There are two methods of specifying the file names; default and command line. If no name(s) are specified on the command line, the code will query the user via the console for a base file name (*fn*). This base name is used as the prefix for all files in the run (see above). The command line method of specifying the file names allows the user more flexibility in defining file names. Each file name can be specified independently according to the conventions in Table 8.2.

Table 8.2 Command Line File Control

File name to be specified	Flag on command line
Plot file name	-p, -P, p=, or P=
Lost photon trajectory file name	-m, -M, m=, or M=
Input file name	-i, -I, i=, or I=
Absorption exchange factor file name	-e, -E, e=, or E=
Trajectory file name	-t, -T, t=, or T=
Output file name	-o, -O, o=, or O=
Leaks file name	-l, -L, l=, or L=
Family file name	-f, -F, f=, of F=

Several conventions bear emphasizing. If even one file name is specified on the command line, the user will not be queried during execution for a filename. Unless the family file name is entered, at a minimum, the names of the input file, output file and exchange factor file must be specified independently on the command line, with the other files deriving their base name from the input file name. If the *f* option is used, any file not explicitly specified will assume the naming convention indicated in Table 8.1. Those file names explicitly specified will override this default. The maximum length of any base file name is 30 characters.

To clarify the command line conventions, the following examples are offered.

Example 1: %lsmonte -i alakazam -Okaboom e= ardvark.w M=toasted

The input file named *alakazam* must exist. The following files will be created: the restart file named *alakazam.rst*, the plot file named *alakazam.plt*, the lost photon trajectory file named *toasted*, the absorption exchange factor file named *ardvark*, the trajectory file (if required) named *alakazam.trc*, the output file named *kaboom,* and the leaks file named *alakazam.lks*.

Example 2: %lsmonte -f calendar

The input file *calender.in* must exist. The following files will be created: the restart file named *calender.rst*, the plot file named *calander.plt*, the lost photon trajectory file named *calender.lst*, the absorption exchange factor file named *calendar*, the trajectory file (if required) named *calender.trc*, the output file named *calend.out*, and the leaks file named *calender.lks*,

Example 3: %lsmonte -f tri -e mi5run

The input file *tri.in* must exist. The following files will be created: the restart file named *tri.rst*, the plot file named *tri.plt*, the lost photon trajectory file named *tri.lst*, the absorption exchange factor files named *mi5run*, the trajectory file (if required) named *tri.trc*, the output file named *tri.out*, and the leaks file named *tri.lks*.

9. UNIX BATCH EXECUTION USING SCRIPTS

This section describes the use of a UNIX shell script to submit multiple runs to be executed sequentially. This has the advantage of running only one job at a time, thereby avoiding the overhead involved with swapping jobs in memory in and out of the CPU (our tests have shown that the overhead under UNIX in so doing is prohibitively large). The shell script in Figure 9.1 is an example which runs LSMONTE three times for three separate problems (input files). In this example line one executes LSMONTE and redirects standard output to the file *run1*. The second line causes the script to pause until line 1 has finished. When the first run has completed the process continues with line 3, etc.

```
lsmonte -f tc1 > run1
wait
lsmonte -f tc2 > run2
wait
lsmonte -f tc3 > run3
```

Figure 9.1 Script “submit”

The script file can be created with any ASCII editor and given any valid UNIX name (here the file is named *submit*). The files referred to in the script (here *tc1*, *run1*, etc.) can also have any valid UNIX name. The files *tc1*, *tc2* and *tc3* must be the input files (without the .in extension) to be used for the run.

Once the script is created it must have execute permission before it can be run. This is accomplished with the UNIX command *chmod*. This is accomplished as follows:

```
%chmod 744 submit
```

where % is the UNIX prompt.

The script can be run in background by typing the script name followed by “&.” Alternately, it can be submitted using the UNIX commands *at* or *batch*. The syntax of any of the UNIX commands can be obtained by referring to the man(ual) pages on your UNIX system.

REFERENCES

Anderson, Stuart L., 1990. “Random Number Generators on Vector Supercomputers and Other Advanced Architectures,” *SIAM Review, 32*, pp.221-251.

Brent, R. P., 1992. “Uniform random number generators for supercomputers,” Proceedings 5th Australian Supercomputing Conference, SASC Organizing Committee, pp. 95-104 (unpublished).

Burns, Patrick J. and Pryor, Daniel V., 1989. “Vector and Parallel Monte Carlo Radiative Heat Transfer,” *Numerical Heat Transfer, Part B: Fundamentals*, *16*, pp. 20-42.

Burns, Patrick J. and Pryor, Daniel V., 1999. “Large-scale Radiative Heat Transfer via Monte Carlo,” *Advances in Heat Transfer*, *IX*, Begell House Press, pp. 79-158.

Burns, Patrick J., Maltby, James D. and Christon, Mark A., 1990. “Large-Scale Surface to Surface Transport for Photons and Electrons via Monte Carlo,” *Computing Systems in Engineering*, *1*(1), pp. 75-99.

Burns, Patrick J., Loehrke, Richard I., Dolaghan, John S. and Maltby, James D., 1992. “Photon Tracing in Axisymmetric Enclosures,” *Developments in Radiative Heat Transfer, HTD-Vol. 203*, pp. 93-100, ASME, New York.

Burns, Patrick J., 2000. “MPLOT Computer Code,” contact the author at pburns@colostate.edu.

Crockett, David V., Maltby, James D. and Burns, Patrick J., 1990. “User’s Manual for MONT3E - A Three-Dimensional Electron-Tracing Code with Non-Uniform Magnetic Field, Release 5.0,” Department of Mechanical Engineering, Colorado State University, Fort Collins, CO 80523.

Dolaghan, John S., Loehrke, Richard I., and Burns, Patrick J., 1992. “User’s Manual for SMOOTH,” Department of Mechanical Engineering, Colorado State University, Fort Collins, CO 80523.

Dolaghan, John S., 1991. *A Monte Carlo Simulation of Molecular Redistribution in an Enclosure due to Sputtering*, M.S. Thesis, Department of Mechanical Engineering, Colorado State University, Fort Collins, CO 80523.

Dolaghan, John S., 1996. *A Monte Carlo Simulation in Rarefied Gas Dynamics with Application to Physical Vapor Deposition*, Ph.D. Dissertation, Department of Mechanical Engineering, Colorado State University, Fort Collins, CO 80523.

Golomb, S. W., 1982. *Shift Register Sequences* (Revised edition), Aegean Park Press.

Maltby, James D., 1987. *Three-Dimensional Simulation of Radiative Heat Transfer by the Monte Carlo Method*, M.S. Thesis, Department of Mechanical Engineering, Colorado State University, Fort Collins, CO 80523.

Maltby, James D., 1990. *Analysis of Electron Heat Transfer via Monte Carlo Simulation*, Ph.D. Dissertation, Department of Mechanical Engineering, Colorado State University, Fort Collins, CO 80523.

Maltby, James D. and Burns, Patrick J., 1991. “Performance, Accuracy and Convergence in a Three-Dimensional Monte Carlo Radiative Heat Transfer Simulation,” *Numerical Heat Transfer, Part B; Fundamentals*, 16, pp. 191-209.

Maltby, James D., Burns, Patrick J. and Winn, C. Byron, 1986. “Monte Carlo Simulation of Radiative Heat Transport in Passive Solar Buildings," *Proceedings of the 1986 American Solar Energy Society Conference,* Boulder, Colorado (June 9-11, 1986).

Maltby, James D., Zeeb, Charles N., Dolaghan, John. D. and Burns, Patrick J., 1994. User's Manual for MONT2D Version 2.6 and MONT3D Version 2.3, Department of Mechanical Engineering, Colorado State University, Fort Collins, CO 80523.

Marsaglia, G., 1985. "A Current View of Random Number Generators," Computing Science and Statistics: Proceedings of the XVth Symposium on the Interface, Billard (ed.), Elsevier Science Publishers, B. V. (North Holland), pp. 3-10.

Mascagni, M., Cuccaro, S., Pryor, D. and Robinson, M., 1995. "A Fast, High-quality and Reproducible Parallel Lagged-Fibonacci Pseudorandom Number Generator," *J. Computational Physics, 119,* pp. 211-219.

McHugh, J., 1995. *Daylighting Design via Monte Carlo,* M.S. Thesis, Department of Mechanical Engineering, Colorado State University, Fort Collins, CO 80523.

McHugh, J., Hittle, D. and Burns, P., May 1998. "The Energy Impact of Daylighting in an Ultra-low Energy Building," *ASHRAE Journal*, pp. 31-35.

Pryor, Daniel V. and Burns, Patrick J., July 21-25 1986. "A Parallel Monte Carlo Model for Radiative Heat Transfer," presented at the 1986 SIAM Meeting, Boston, MA.

Pryor, D. V., Cuccaro, S. A., Mascagni, M. and Robinson, M. L., 1994. "Implementation of a Portable and Reproducible Parallel Random Number Generator," Proceedings Supercomputing 94, IEEE Computer Society Press, Los Alamitos, CA, pp. 311-319.

Shapiro, Arthur B., 1985. "TOPAZ3D - A Three-Dimensional Finite Element Heat Transfer Code," Lawrence Livermore National Laboratory, UCID-20484.

Schweitzer, Roland, McHugh, Jon, Burns, Patrick J. and Zeeb, Charles N., 1993. "Daylighting design via Monte Carlo with a corresponding scientific visualization," Proceedings Supercomputing 1993, IEEE Computer Society Press, Los Alamitos, CA, pp. 250-259.

Statton, E. Scott, 1983. *MONTE - A Two-Dimensional Monte Carlo Radiative Heat Transfer Code*, M.S. Thesis, Department of Mechanical Engineering, Colorado State University, Fort Collins, CO 80523.

Zeeb, Charles N. and Burns, Patrick J., 1997. "Random Number Generator Recommendation," Report prepared for V. J. Romero of Sandia National Laboratory, available via the web at http://www.colostate.edu/~pburns/monte.html.

Zeeb, Charles N. and Burns, Patrick J., August 1999. "Performance enhancements in Monte Carlo Radiative Heat Transfer," Proceedings ASME National Heat Transfer Conference, Albuquerque, NM.