PT

Monte Carlo Surface to Surface Particle Transport

Notation Key

This section provides convenient lists of the notational conventions used throughout this chapter. Table 1 lists the notation used for various physical and geometrical quantities, Table 2 details the usage of Greek symbols, and Table 3 summarizes the conventions for superscripts and subscripts.

1 Introduction

In the following, we shall present the formulation for tracing particles from surfaces to surfaces inside an enclosure. A more complete reference is Burns et al., 1990. To illustrate the concepts and computational aspects, we deal with prismatic surfaces, which are infinite in the third dimension. Thus, they can mathematically be described in two dimensions only.

We follow particles from "birth" (emission) to "death" (absorption or extinction), through possibly many intermediate interactions (reflections) with surfaces. As we directly simulate physical objects (particles), this is often termed the Direct Simulation Monte Carlo method.

Here, between surfaces, particles travel in straight lines and do not interact. Extensions to trajectories other than straight lines are conceptually straightforward, but can be vexing in the difficulty of implementation. Where particles *do* interact, this may introduce significant additional complexity. Our approach is representative of the transport of photons, neutrons, electrons and molecules in situations where there is no participating medium—where volumetric interactions do not occur. A variety of application areas including thermal radiative transport, neutron damage and molecular sputter are well modelled under these restrictive assumptions.

Table 1: Physical and Geometrical Quantities

A	= area (m^2)
A	= absorbed photon
C_L	= Z component of the cross product of the endpoint vectors defining a surface
C_P	= Z component of the emission cross product
CDF	= cumulative distribution function
$D_{I,L}$	= distance from emission point to intersection point on surface L
D	= fractional overhead due to (useless) data motion
$\mathcal{D}$	= sum of fractional overhead due to data motion and vector start-up
d	= depth into the surface (m)
$\mathbf{E}$	= unit emission direction vector, global Cartesian coordinates
$\mathbf{e}$	= unit emission direction vector, local Cartesian coordinates
F_{ij}	= fraction of particles emitted from surface i that terminate on surface j
F	= fractional overhead due to vector start-up
f_p	= fraction of work that can be done in vector (parallel) hardware
i	= emitting surface
j	= absorbing surface
L	= surface of interest
L_L	= length of surface L
N	= number of photons
N	= node of a surface
$\mathbf{N}$	= normal vector of a surface
$NSURF$	= number of surfaces in an enclosure
$\dot{N}$	= the rate of particles emitted over a hemisphere from a differential surface area
$\dot{n}$	= the rate of particles emitted per unit solid angle per unit surface area

Table 1: Physical and Geometrical Quantities (cont.)

R	= random number, from a uniform distribution between 0 and 1
$\mathcal{R}$	= speedup ratio of scalar to vector execution times
$R_{\mathcal{D}}$	= diffusely reflected photon
$R_{\mathcal{S}}$	= specularly reflected photon
r	= radius (m)
S	= arc length (m)
S_1, S_2	= square of distance between the first or second end point of a surface and the intersection point, $\left\|\boldsymbol{X}_{(1 or 2)} - \boldsymbol{X}_I\right\|^2$ (m^2)
S_{12}	= square of distance between end points of a surface, $\|\boldsymbol{X}_1 - \boldsymbol{X}_2\|^2$ (m^2)
$\mathcal{S}$	= scalar execution rate (MFLOP's)
s	= path length (m)
T	= absolute temperature (°K)
$\mathcal{V}$	= vector execution rate (MFLOP's)
$\boldsymbol{X}$	= vector point, $\{X, Y, Z\}$ (m)
$\boldsymbol{X}_E$	= point of emission (m)
$\boldsymbol{X}_I$	= point of intersection (m)
X, Y, Z	= point in global Cartesian coordinates (m)

Table 2: Greek Symbols

α	= absorbance
$\Delta X_L, \Delta Y_L$	= difference in the X or Y direction between the end points of surface L (m)
ε	= emittance
θ	= cone angle
ρ	= reflectance
σ	= Stefan Boltzmann constant (W m^{-2} °K^{-4})
ϕ	= azimuthal angle
ω	= solid angle

Table 3: Superscripts and Subscripts

a, b	- first and second values chosen
bin	- value for a bin
c	- value at the center of a bin
d	- diffuse
E	- point of emission
H	- value over a hemisphere
I	- intersection point
i	- emitting surface
j	- absorbing surface
k	- component in the Z direction (in the direction of the k unit vector)
L	- surface L
n	- the nth component
o	- outgoing
perfect	- value for a"perfect" emitter
s	- specular
X, Y	- global Cartesian component
x', y', z'	- local Cartesian component
1,2	- nodes 1 and 2 on surface

Finally, we note that, insofar as possible, we shall bypass the physics. In so doing, we shall concern ourselves with only the matrix of the number of particles which originate from surface i and terminate upon surface j, defined as N_{ij}. The fraction of particles emitted from surface i and absorbed by surface j is then $F_{ij} = N_{ij}/N_i$, where N_i is the total number of particles emitted by surface i. Specifically, in the case of radiative transfer, the flux of photons emitted from surface i is equal to σT_i^4. Thus, the one-way rate of radiative heat transfer from surface i to surface j is $\sigma T_i^4 A_i F_{ij}$. Explicitly, we shall concern ourselves only with F_{ij}, as the remainder of the transport term is application specific and can include the emission of photons, electrons, ions or molecules, for example. Thus, our problem reduces to emitting particles from all surfaces i in an enclosure and tracing them to all surfaces j of the enclosure, and the problem can be formulated in terms of either N_{ij} or F_{ij}. "Enough" particles N_i must be emitted from every surface to achieve convergence. Typically, this is accomplished by scaling up N_i on successive runs until convergence is attained.

To achieve an accurate simulation, several features are required. First, the physical characteristics of emission must be accurately represented. This includes the outgoing distributions of: direction, energy and spatial location. Secondly, geometry must be accurately represented—both the geometry of the enclosure and the geometry of the particle trajectory. Finally, interactions of the particles with surfaces (i.e. materials) must be accurately represented.

Although it is logical to conceptualize the process chronologically from birth to death, we partition our presentation into the two areas of geometry and material properties. The material properties of the surface determine both the emission distributions and the particle/surface interactions.

2 Geometry

2.1 Introduction to Geometry

There are two aspects to be covered in this section. First, we introduce the definitions of surfaces. Then, we proceed with tracing particles which traverse straight-line trajectories. As literally millions (or even billions) of particles must be traced in a complex simulation, it is essential for the tracing algorithm to be efficiently formulated.

2.2 Surface Definitions

We begin by defining our global Cartesian coordinate system as shown in Figure 1. All surfaces and particle trajectories are defined in this coordinate system. Here, we shall use capital letters to denote quantities in the global Cartesian system and bold notation to denote vectors. Thus: $\mathbf{X} = \{X, Y, Z\}$, defining node N as shown.

Further, we shall be dealing with two-dimensional prismatic geometries which are infinite in Z. Thus, we define surfaces as shown in Figure 2, comprised of straight lines connecting nodes N_1 and N_2 (linear surfaces are used to simplify particle tracing calculations—curved

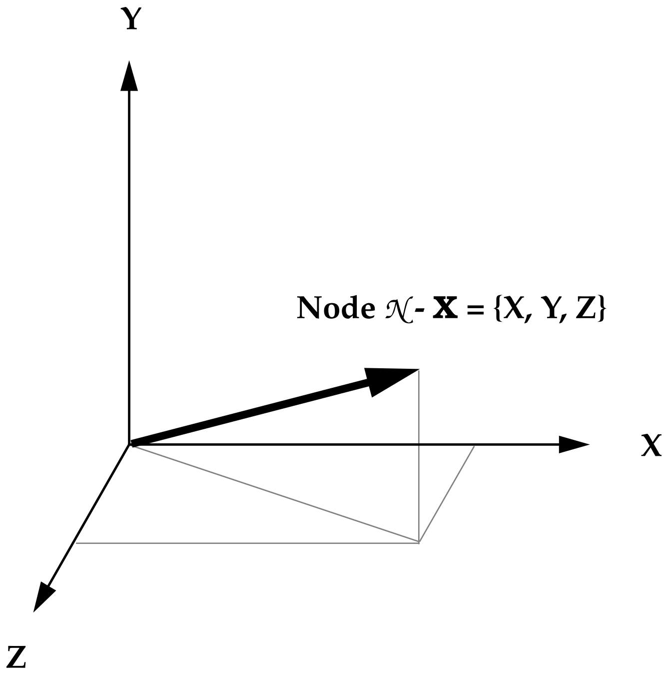


Figure 1: Global Cartesian Triad

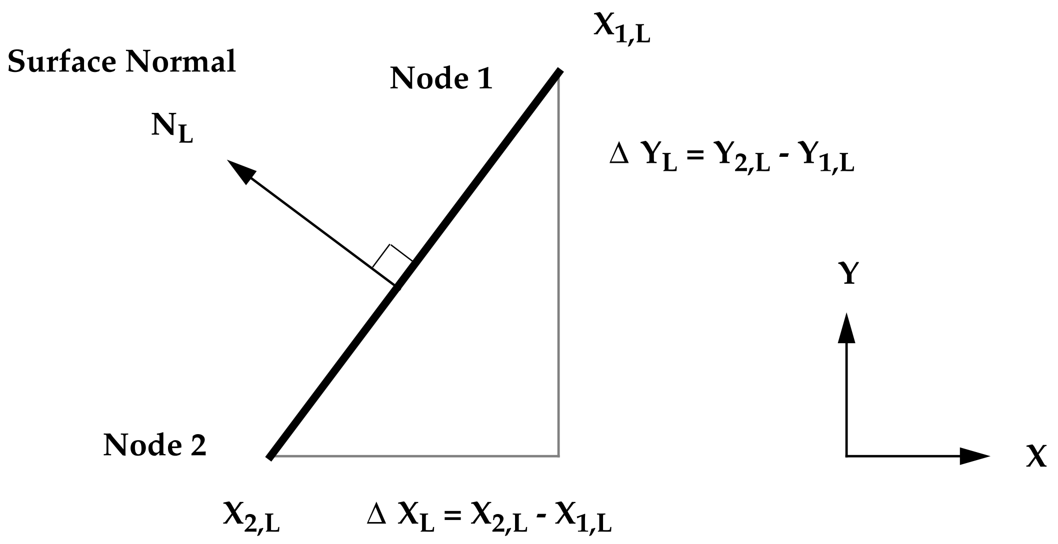


Figure 2: Details of Surface L

surfaces must be approximated by a number of linear segments). Note that, the surfaces, although mathematically described in two dimensions, are actually three-dimensional.

We define a variety of quantities for each surface L, where $L \in [1, NSURF]$. Where the context to a general surface is clear, we shall omit the subscript L. Geometries are required to form full enclosures, as all particles must be traced eventually to absorption—thus, no "holes" are permitted.

We define and store the following quantities for every surface L during the input phase of the program:

$$\begin{aligned}
\Delta X_L &= X_{2,L} - X_{1,L} \\
\Delta Y_L &= Y_{2,L} - Y_{1,L} \\
S_{12,L} &= |\boldsymbol{X}_{2,L} - \boldsymbol{X}_{1,L}|^2 = [(X_{2,L} - X_{1,L})^2 + (Y_{2,L} - Y_{1,L})^2] \\
L_L &= [S_{12,L}]^{1/2} \\
\mathbf{N}_L &= \{N_{X,L}, N_{Y,L}\} = \{\Delta Y_L / L_L, -\Delta X_L / L_L\} \\
C_L &= \{\boldsymbol{X}_{2,L} \times \boldsymbol{X}_{1,L}\}_K = X_{2,L} Y_{1,L} - X_{1,L} Y_{2,L}
\end{aligned}$$

(where $\times$ denotes the cross product).
Note that, the surface normal vector $\mathbf{N_L}$ is defined as pointing to the right when proceeding from node $N_{1,L}$ to node $N_{2,L}$. This is important because we allow particles to interact with only the *front* of the surface. Normally, the back of the surface is the solid material from whence the particles derive and which serves to absorb incident particles (recall that a surface

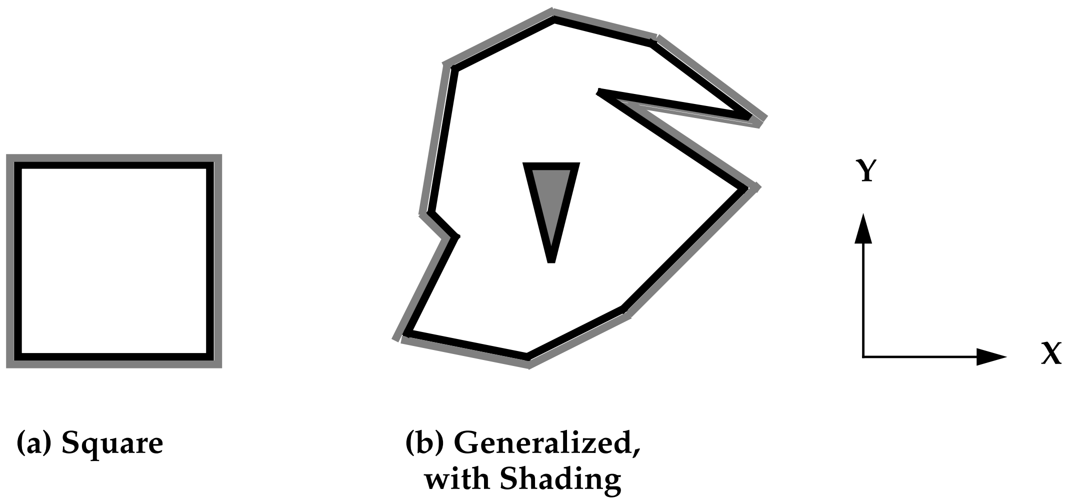


Figure 3: Cross-Sections of Prismatic Geometries

has zero thickness, and thus contains no material).

We restrict our analysis to enclosures, i.e., closed geometries such as shown in Figure 3 (frequently in practice, actual "holes" exist in geometries—these are modelled as totally absorptive surfaces). Figure 3(a) shows a square, while Figure 3(b) depicts a generalized geometry, with an interior element (which must be supported somewhere in Z—not shown) and shading, i.e., the geometry is convex.

2.3 Particle Tracing

Here, we determine efficient methods for tracing particles. In surface-to-surface transport, we trace particles from their surface of origin ("birth") to their eventual absorption ("death") through possibly many intermediate reflections. Mathematically, this reduces to tracing particles from their point of emission, $\boldsymbol{X}_{\mathbf{E}}$, to their point of intersection with a surface, $\boldsymbol{X}_I$. Given that a particle is travelling from emission (or origination) point $\boldsymbol{X}_{\mathbf{E}}$ in direction $\mathbf{E}$ ($\mathbf{E}$ is determined from the emission or reflection properties to be covered in the next section), the equation of a line in 3-space is:

$$\frac{X - X_E}{E_X} = \frac{Y - Y_E}{E_Y} = \frac{Z - Z_E}{E_Z} \tag{1}$$

and the equation describing surface L is:

$$\frac{X - X_{1,L}}{\Delta X_L} = \frac{Y - Y_{1,L}}{\Delta Y_L} \tag{2}$$

Solving equations (1) and (2) simultaneously:

$$X_{I,L} = \frac{E_X C_L + \Delta X_L C_P}{E_Y \Delta X_L - E_X \Delta Y_L} \tag{3}$$

and

$$Y_{I,L} = \frac{E_Y C_L + \Delta Y_L C_P}{E_Y \Delta X_L - E_X \Delta Y_L} \tag{4}$$

where:

$$C_P = E_Y X_E - E_X Y_E \tag{5}$$

Equations (3) and (4) yield a point that is at the intersection of the two infinite lines defined by the equations of the surface and the particle trajectory. Except where these lines are parallel (rarely, if ever, encountered in practice), there is one and only one intersection. Mathematically stated, if the denominator of equations (3) and (4) is zero, there is no intersection. Although it is possible to implement a check for parallel lines, one can usually avoid so doing (especially if your computer does hardware recovery from overflow as is common on many current RISC computers).

See exercise 1.

2.4 The Surface Search Loop

Consider the situation shown in Figure 4, where we trace a particle from $\boldsymbol{X}_{\mathbf{E}}$ to $\boldsymbol{X}_{\mathbf{I}}$ in a simple enclosure with 14 surfaces. The intersection calculations must be done over all surfaces, and the unique "valid" point of intersection identified. (Note that the straight line of the particle trajectory, in general, intersects every straight line defining every surface in the enclosure.) Specifically, Figure 4 shows emitting and intersecting surfaces as numbers 13 and 12, respectively (recall that we disallow intersections from the back of a surface—i.e., surface 3; in a well-specified geometry, there is always an intersection prior to one from behind—here, the prior intersection is that of surface 12).

The "valid" intersection is identified as the one of minimum distance from the point of emission to the point of intersection (for example, surface 12 instead of surface 2), but this comprises a daunting problem numerically, of order N_{SURF}^2. Specifically, we seek ways to limit this search to render the problem tractable. Emissions are done from all surfaces and each emission requires testing all surfaces for the unique, "valid" intersection.

Strategies to limit the search are depicted in Figure 5 for a general surface L. Among them are:

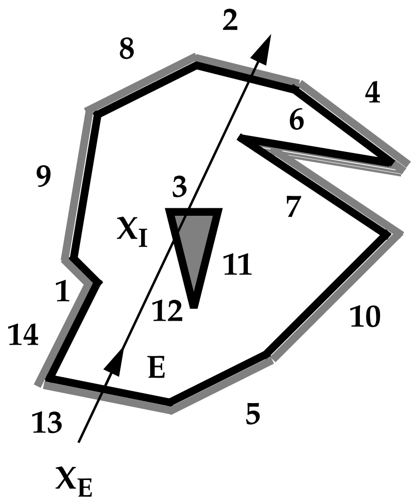


Figure 4: Particle Intersection

2.4.1 Figure 5(a)—Dot Product Test

This first test eliminates, using a simple, economical formula, up to half of the surfaces in an enclosure. The following criterion is used to *invalidate* the intersection (here, this means that the surface is not a viable candidate for intersection):

$$\mathbf{E} \cdot \mathbf{N_L} > 0 \tag{6}$$

Equation (6) involves only two multiplications, one addition and one comparison. Note that, it is not required to calculate the intersection point to eliminate a surface according to the above criterion.

2.4.2 Figure 5(b)—Intersection Lies Outside of Surface

It *is* required to compute the intersection point to perform this test. Here, we: (1) compute the intersection, (2) compute squares of distances from the point of intersection to the surface endpoints, and (3) test these versus the square of the length of the surface.

The intersection point lies outside the surface, and the intersection is *invalidated* if and only if:

$$S_1 + S_2 > S_{12} \tag{7}$$

where

$$S_i = |\boldsymbol{X}_{\mathbf{I,L}} - \boldsymbol{X}_{\mathbf{i,L}}|^2 \tag{8}$$

where $i = 1$ or 2. Recall that S_{12} has been precomputed and stored during the input phase. Note that, it is neither necessary nor desirable to take the square root in equation (8). First,

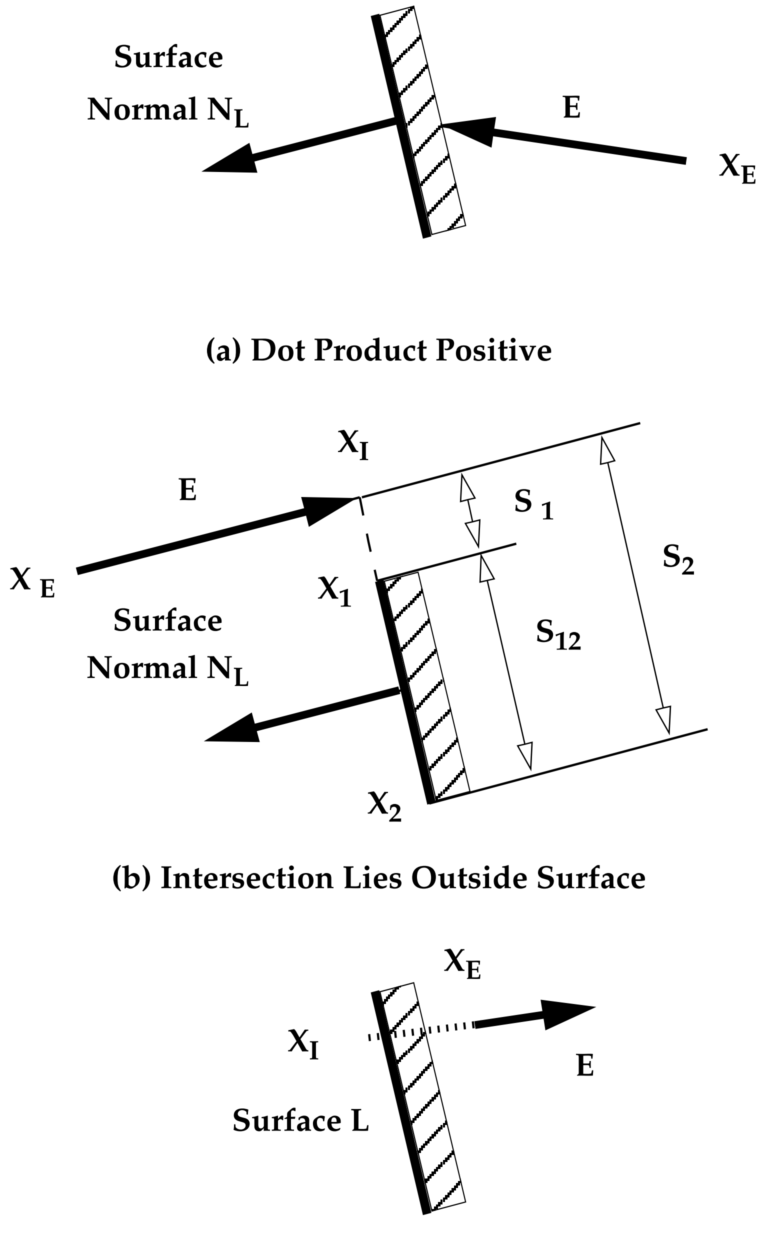


Figure 5: Conditions to Invalidate a Potential Intersection

this is an unnecessary operation. Secondly, taking the square root reduces by a factor of two any small difference in the quantities $\boldsymbol{X}_{\mathbf{I}}$ and $\boldsymbol{X}_{\mathbf{i}}$, thereby exacerbating errors which might occur due to insufficient precision.

2.4.3 Figure 5(c)—Surface of Intersection Lies Behind Emitting Point

Note that the previous two tests are independent of this test and do not include this situation. Again, it is necessary already to have calculated the intersection point. Here, the following test is used to *invalidate* the intersection:

$$(\boldsymbol{X}_{\mathbf{I,L}} - \boldsymbol{X}_{\mathbf{E}}) \cdot \mathbf{E} < 0 \tag{9}$$

2.5 Implementation

Finally, we note that, of all the intersection points that pass the above tests, shading still has not been accounted for. For example, in Figure 4, both surfaces 12 and 2 pass the above tests. Hence, it is necessary to establish all valid intersections per the above 3 tests, and then to select the unique intersection of minimum distance. Again, since we do not use the distance for anything but a comparison of length, it is not necessary to take the square root. Thus, of all the currently "valid" intersections, we design the loop to yield the single, valid intersection of minimum

$$D_{I,L} = |\boldsymbol{X}_{\mathbf{E}} - \boldsymbol{X}_{\mathbf{I,L}}|^2 = (X_E - X_{I,L})^2 + (Y_E - Y_{I,L})^2 \tag{10}$$

The above four-step sequence is implemented as an exclusive binary search over all surfaces in the enclosure. Each test can be implemented as "true" or "false" for those surfaces which pass the test (i.e., are viable candidates for intersection).

We note that the above process may be viewed from the vantage point of parallel execution. Viz., given an assemblage of many particles and focusing attention on a particular surface L, how many particles pass the tests at each level? In other words, each criterion acts as a sieve to filter out particles which cannot possibly intersect the present surface and to pass through the sieve those particles which meet the criterion. Later, we shall return to this topic, i.e., strategies for implementing the above algorithm on vector or parallel architectures.

2.6 Shading

To reduce execution time for large problems with shading, a grid tracing shading algorithm may be implemented. The geometry is divided into a series of grid cells, resulting in rectangles and rectangular parallelepipeds for the 2-D and 3-D geometries, respectively. The particle is traced from grid cell to grid cell, and the search is done within each grid cell over only those surfaces which exist either wholly or partly within that cell. The situation is depicted in Figure 6 for a 2-D geometry. This algorithm has resulted in reductions in execution time for 2-D Cartesian enclosures of factors of 2 to factors of 20, with factors of

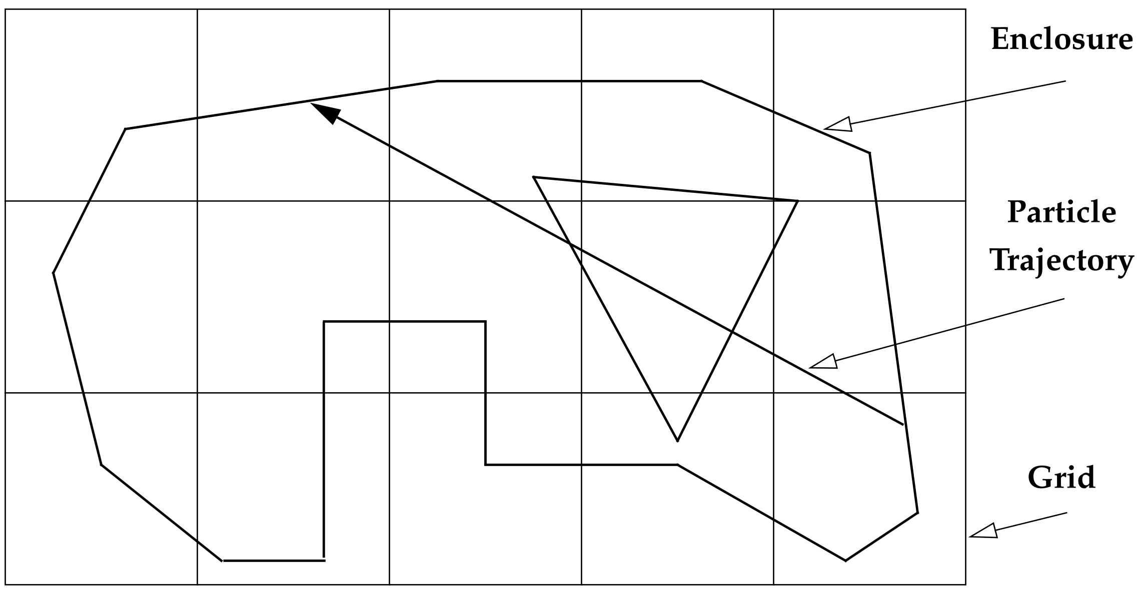


Figure 6: Illustration of the Grid Shading Algorithm

5 to 7 being typical. For 3-D enclosures, due to the higher overhead incurred in tracing through a 3-D grid, reductions in execution time of factors of 2 to factors of 4 are typical.

We have stated in section 2.4 that the surface search loop, which consumes the majority of the execution time, scales as $NSURF^2$. However, implementing grid tracing causes the surface search loop to scale as $NSURF^1.5$, thereby enabling much larger problems to be simulated. The optimum grid differs from problem to problem, and must be empirically determined by the user. However, good starting guesses are about 100 total grid cells (about 10×10 in two dimensions and $5 \times 5 \times 5$ in three dimensions).

The grid must be regular (i.e., all vertices must match up), but it need not be uniform. Our implementations have also required that the grid divisions be at constant X, Y or Z. If the surfaces of the geometry are concentrated in one region of space, then one can define a higher concentration of grids in that region. Note that, we have not vectorized this approach, as there are a variable number of surfaces within each grid, which renders the structure of the algorithm a bit more difficult to handle, as some particles will be absorbed, others will require to be moved to other cells, while still others will have surfaces in their cells remaining to be searched. However, this should be able to be well modelled on an MIMD architecture.

Table 4: Analogy of Angles to Lengths

Quantity	No Dimensions	1 Dimension	2 Dimensions	3 Dimensions
Lengths	Point	Length	Area	Volume
Angles	Line	Angle	Solid Angle	-

3 Material Properties

3.1 Introduction to Material Properties

In this section, we address the physical aspects involved in Monte Carlo particle/surface interactions. All physics involved are embodied in the material properties of the surfaces in the enclosure. There are two physical aspects to the material properties: (1) emission and (2) particle/surface interactions. Both aspects must be modelled with sufficient accuracy to achieve the fidelity required to represent the problem.

When emitting, we must replicate: (1) the directional distribution of physical emissions from a point on the surface, (2) the spatial distribution of emissions over a surface, and (3) other distributions of interest, e.g. the energy distribution. We proceed assuming that all distributions are independent, i.e., not interrelated (e.g., the directional distribution is the same, no matter where the emission point is). In fact, each *numerical* particle represents an *assemblage* of physical particles. "Enough" numerical particles must be emitted and traced to extinction in order to achieve convergence (this is similar to refining the grid in continuum calculations).

Particle/surface interactions are defined simply in terms of probabilities of certain physical behaviors upon interaction. This is fairly straightforward if the particle is absorbed by the surface. It becomes complicated if the particle survives, and an outgoing direction must be chosen. Because good data for particle/surface interactions are not typically available, it is common practice to make approximations in this regard. Fortunately, the approximations are representative of many physical situations (here, nature is in harmony with our modelling capability).

3.2 Emission

3.2.1 Local Spherical System–Directional Distribution

A particle is emitted from a solid surface into the "space" or *hemisphere* above the surface. Locations on the hemisphere are characterized by *solid angle*. We introduce the concept by analogy to linear dimensions. Recall the definition of geometrical quantities as follows:

Referring to the Table 4 and Figure 7, a line is generated by sweeping a point linearly through space, and an area is generated by sweeping a line through space. Similarly as shown in Figure 7, an angle θ is generated by sweeping a line through an arc. A solid angle is generated by sweeping two "orthogonal" angles (θ and ϕ) through arcs.

A solid angle is two-dimensional by virtue of having no radial position, i.e., the solid angle remains fixed independent of radius. This provides the key to the mathematical formulation for a solid angle. Again, we proceed by scaling up in dimensionality. Recall that a differential angle is defined as follows:

$$d\theta = \frac{dS}{r} \tag{11}$$

where dS is the differential arc length and r is the radius. A differential solid angle $d\omega$ is defined by scaling up the differential arc length to a differential area and dividing by r^2 to make the solid angle independent of radial position (distance from the origin) as follows (see Figure 7(f)):

$$d\omega = \frac{dA}{r^2} = \frac{|BD||BC|}{r^2} = \frac{(r\,d\theta)(r\sin\theta\,d\phi)}{r^2} = d\theta\,\sin\theta\,d\phi \tag{12}$$

Viz., referring to Figure 7(f), note that length $|BC|$ is generated by sweeping line $|AB| = r\sin\theta$ (not r), through angle $d\phi$ - i.e. the revolution is of a line normal to the vertical axis $|AB|$, about the vertical axis $|OA|$. Contrariwise, line $|BD|$ is generated by sweeping $|OB| = r$, through angle $d\theta$ about the origin O.

In Figure 7, θ is the "cone" angle and ϕ is the "azimuthal" angle. Emission occurs over the hemisphere defined from $\theta = 0$ to $\pi/2$ and $\phi = 0$ to 2π. While angles are measured in *radians*, solid angles are measured in *steradians*. A hemisphere contains 2π steradians, as follows:

$$\Delta\omega = \int_{\theta=0}^{\pi/2}\int_{\phi=0}^{2\pi} \sin\theta\,d\theta\,d\phi = 2\pi \tag{13}$$

The probability of emission of a particle at angle θ should properly be weighted by $\cos(\theta)$. As particles originate inside the solid, their probability of escaping from the solid and being emitted is inversely proportional to the path length s they must travel through the material to escape. In Figure 8, we depict this conceptually for particles originating a depth d beneath the surface. Thus, emission at angle θ is proportional to $1/s$ or $\cos(\theta)$. The emission from many real surfaces is well modelled in this fashion. Such surfaces are termed Lambertian or "diffuse" surfaces.

We now turn our attention to emission from a general surface. In a specific time interval, the rate of emission of particles, $\dot{N}$, emitted over the hemisphere from differential area dA may be related to the rate of particles emitted per unit solid angle per unit projected surface area, $\dot{n}$, as follows:

$$\begin{aligned}\dot{N} = \int\int \dot{n}\,d\omega\,dA_{projected} &= \int_{\theta=0}^{\pi/2}\int_{\phi=0}^{2\pi} \dot{n}(\sin\theta\,d\theta\,d\phi)(\cos\theta\,dA) \\ &= \int_{\theta=0}^{\pi/2}\int_{\phi=0}^{2\pi} \dot{n}\sin\theta\cos\theta\,d\theta\,d\phi\,dA \end{aligned} \tag{14}$$

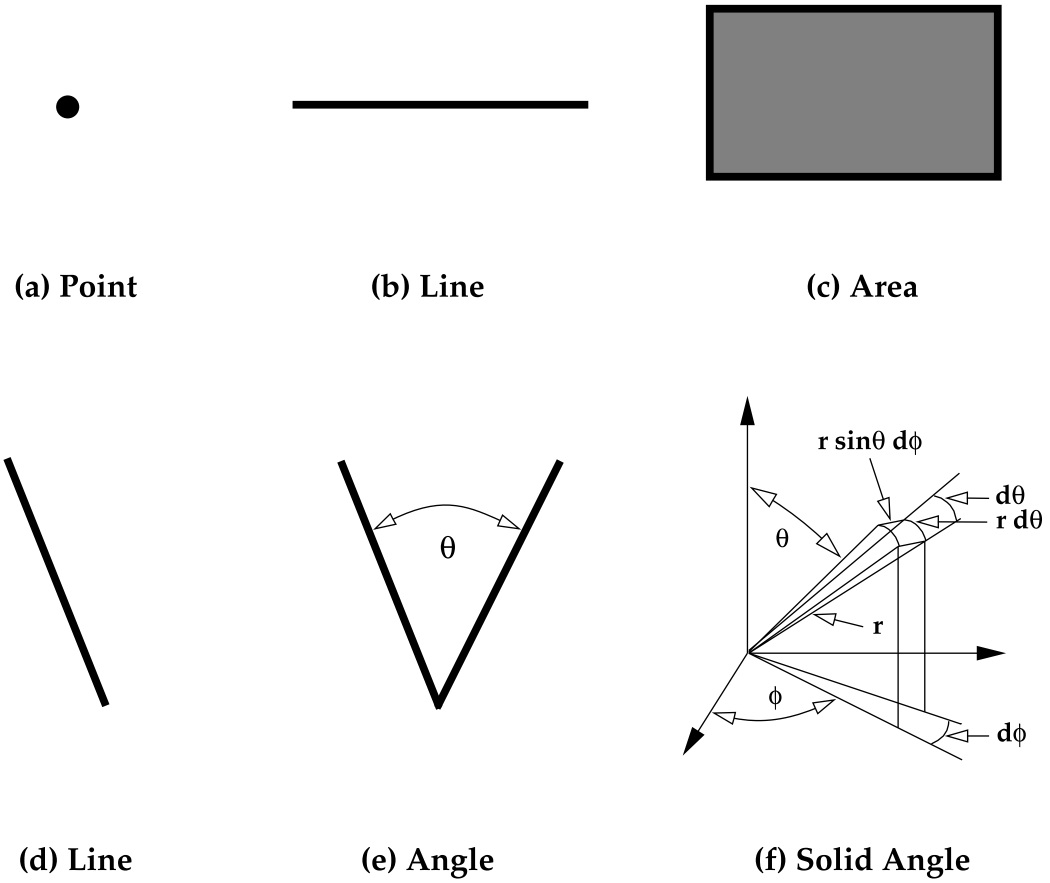


Figure 7: Generating an Angle and a Solid Angle

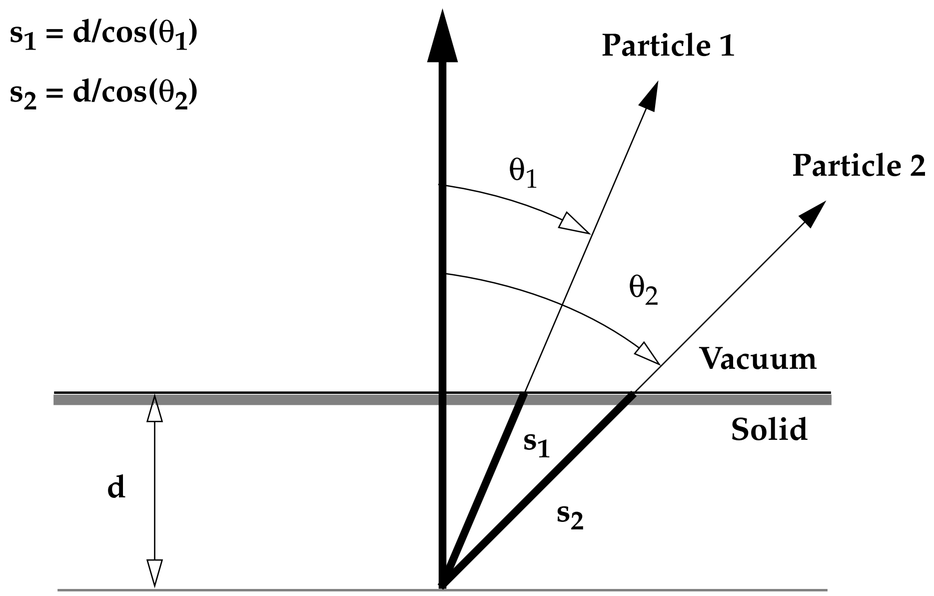


Figure 8: Emission from Within a Solid

If $\dot{n}$ is constant, i.e., independent of θ and ϕ (as it is for Lambertian emission), then

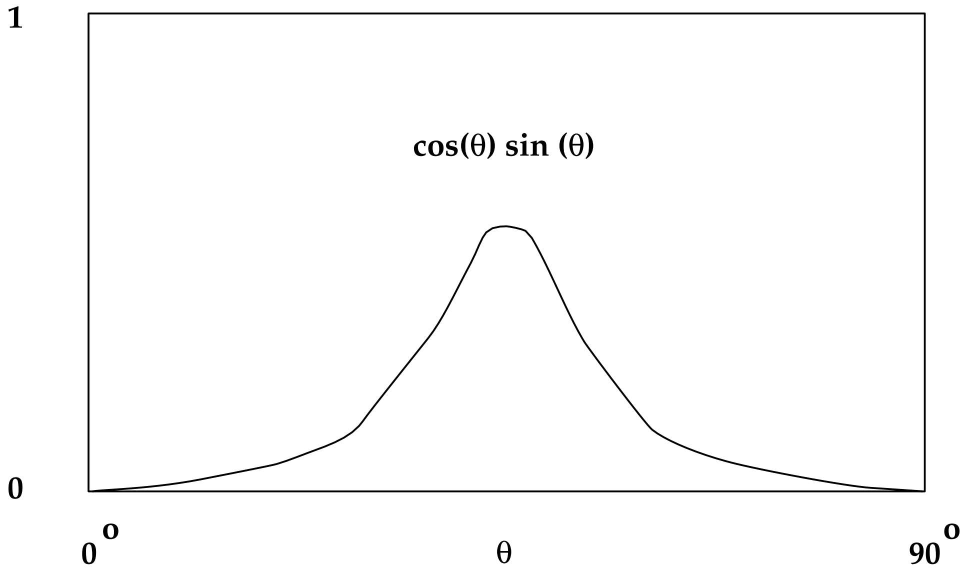
$$\dot{N} = \dot{n} \int_{\theta=0}^{\pi/2} \int_{\phi=0}^{2\pi} \sin\theta \cos\theta \, d\theta \, d\phi \, dA = \pi \dot{n} \, dA \tag{15}$$

Therefore, we obtain exactly half as many particles by weighting the solid angle with the projected area ($\cos(\theta)$) for emission (recall that there are 2π steradians in a hemisphere). In practice, we define a "perfect" emitter, which emits the maximum rate of particles over the hemisphere at each steradian. The ratio of the rates of particle emission from an actual (real) surface to that from a "perfect" emitter is termed the emittance, ε, which typically depends upon θ.

$$\varepsilon(\theta) = \frac{\dot{n}(\theta)}{\dot{n}_{perfect}} \tag{16}$$

The quantity $\varepsilon(\theta)$ is typically measured, although it may be accurately calculated in some situations. Note that, we have specified that ε may be a function of θ and not of ϕ. This is a common assumption, as only biases in surface finish (such as may be caused by specific machining practices) cause a dependence upon ϕ, and these are rarely significant. However, variations in molecular structure of emitting surfaces cause a dependence upon θ, and this must in general be accounted for.

1



θ

Figure 9: Probability of Emission in dq

Then, over the entire hemisphere,

$$\varepsilon_H = \frac{1}{\pi} \int_{\theta=0}^{\pi/2} \int_{\phi=0}^{2\pi} \varepsilon(\theta) \sin\theta \cos\theta \, d\theta \, d\phi \tag{17}$$

Since $\varepsilon(\theta)$ has a maximum of 1, ε_H has a maximum of 1. We term ε_H the total hemispherical emittance, as characteristic of the emission into the entire hemisphere above the point. Henceforth, we shall neglect all angular dependences in this introductory treatment, and proceed by examining emission from "perfect" surfaces, where $\varepsilon(\theta)$ is 1.

Thus, the directional distribution for emission is what we actually seek. This is given by the argument of the integral in equation (15), i.e., particles are emitted from perfect surfaces according to $\cos(\theta)\sin(\theta)\,d\theta$, as shown in Figure 9. As we see, per unit degree of cone angle-θ, the greatest number of particles are emitted at an angle of 45° from the surface normal (over all 360° in ϕ). As θ tends to 0° (normal to the surface), the solid angle tends to 0, and as θ tends to 90° (grazing), the projected area tends to zero. The balance between these two factors results in a maximum for emission per unit cone angle from a "perfect" surface at $\theta = 45°$.

In Monte Carlo simulation, it is often required to emit particles according to a directional distribution such as shown in Figure 9. In effect, we emit many particles to establish a directional distribution such as shown in Figure 9. One way to do this is termed the *accept/reject* method.

3.2.2 Accept/Reject Method

Here, we choose a set of random numbers uniformly distributed between 0 and $\pi/2$ to establish values for θ. We shall focus upon one of these random numbers, R_a. Thus,

$$\theta = R_a \pi/2 \tag{18}$$

Given this θ, we then choose another "independent" random number, R_b. If R_b is less than or equal to the probability of emission at angle θ, then we *accept* the emission, otherwise we *reject* the emission and repeat the process. Thus, we accept the emission if and only if:

$$R_b \leq \sin\theta\cos\theta \tag{19}$$

where $\sin(\theta)\cos(\theta)$ is proportional to the probability of emission at angle θ. We repeat the above process until we have established directions for all particles to be emitted.

The above is an <u>inefficient</u> method for achieving emissions because when the emission is *rejected* the work is wasted.

3.2.3 Inverse Cumulative Distribution Function

The eminent mathematician John von Neumann devised an <u>efficient</u> and <u>elegant</u> method to circumvent the rejections. This consists of formulating the cumulative distribution function (CDF) for emission, as follows:

$$CDF(\hat{\theta}) = \frac{\int_{\theta=0}^{\hat{\theta}} \sin\theta\cos\theta\, d\theta}{\int_{\theta=0}^{\pi/2} \sin\theta\cos\theta\, d\theta} = \sin^2(\hat{\theta}) \tag{20}$$

The CDF is the cumulative probability that an emission has occurred between $\theta = 0°$ and $\hat{\theta}$. The range of a CDF is between 0 and 1. The advantage of the CDF is that, by aggregating all emissions into a cumulative distribution, an emission is obtained in every trial (we are certain that a particle will be emitted in some direction from the surface)—thus, no work is wasted.

We proceed by replacing the CDF with a random number, R_n, and invert the function:

$$\hat{\theta} = \sin^{-1}\left(\sqrt{R_n}\right) \tag{21}$$

yielding the inverse CDF shown in Figure 10. Note that, a continuum of uniformly distributed random numbers yields the distribution of the argument in the integral, i.e.,

$$\cos(\theta)\sin(\theta) \tag{22}$$

(this is apparent because a uniform distribution of random numbers places equal numbers into uniformly spaced bins, as depicted graphically in Figure 10, where emissions are concentrated about 45°). In fact, differentiating equation (20) yields $2\sin(\theta)\cos(\theta)\, d\theta$, substantiating this argument.

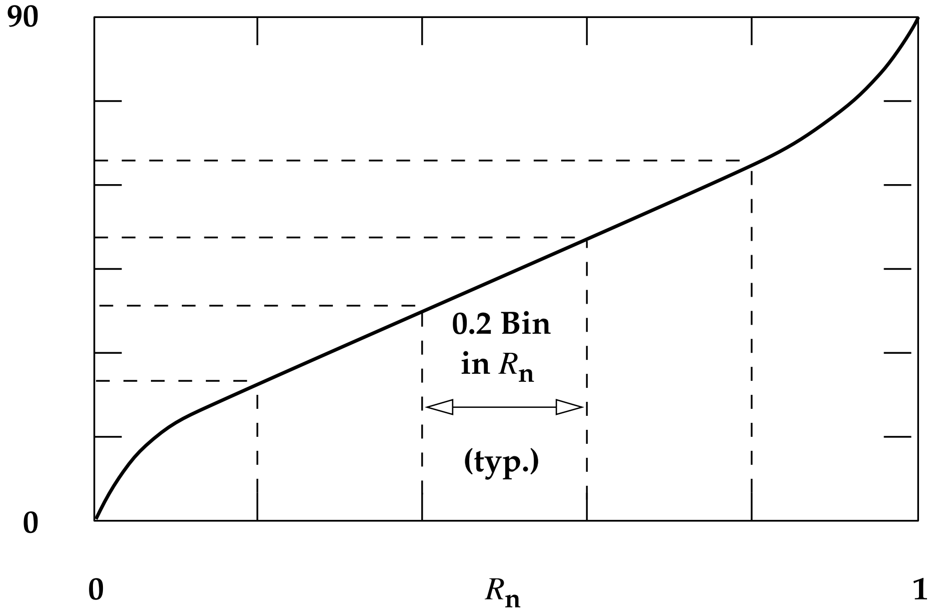


Figure 10: Inverse CDF for Emission

Thus, the final algorithm for emission is as follows:

$$\phi = 2\pi R_a \tag{23}$$

$$\theta = \sin^{-1}\left(\sqrt{R_b}\right) \tag{24}$$

(Note that, here we have dropped the hat on θ.)

See exercise 2.

3.2.4 Local and Global Cartesian System–Outgoing Directions

For a single particle, we now examine the coordinate transformations required to obtain **E**, the emission direction in the global Cartesian coordinate system. First, as shown in Figure 11, we transform the local spherical components θ and ϕ into the local Cartesian components $e_{x'}$, $e_{y'}$ and $e_{z'}$, as follows:

$$\mathbf{e} = \begin{bmatrix} e_{x'} \\ e_{y'} \\ e_{z'} \end{bmatrix} = \begin{bmatrix} \sin\theta\cos\phi \\ \cos\theta \\ -\sin\theta\sin\phi \end{bmatrix} \tag{25}$$

Next, the local Cartesian coordinates must be transformed into global coordinates, as shown in Figure 12. This transformation is a simple two-dimensional rotation, obtained using the components of the surface normal $\mathbf{N}$, by projecting the $e_{x'}$ and $e_{y'}$ directions onto the global X and Y axes:

$$\mathbf{E} = \begin{bmatrix} E_X \\ E_Y \end{bmatrix} = \begin{bmatrix} N_Y & N_X \\ -N_X & N_Y \end{bmatrix} \begin{bmatrix} e_{x'} \\ e_{y'} \end{bmatrix} = \begin{bmatrix} N_Y & N_X \\ -N_X & N_Y \end{bmatrix} \begin{bmatrix} \sin\theta\cos\phi \\ \cos\theta \end{bmatrix} \tag{26}$$

Now, we have $\mathbf{E}$—half of the information we need for an emission. We still need $\boldsymbol{X}_{\mathbf{E}}$. We proceed with this now.

3.2.5 Spatial Distribution

To render the problem tractable, we restrict our attention to surfaces of spatially constant properties. Thus, a surface will have constant values for all material properties, but a geometry can consist of any number of surfaces. Also, input parameters must be constant over the surface. For example, in radiative transfer, the surfaces are of constant temperature. When discretizing the geometry, this criterion generally is the controlling factor, i.e., how small must the surface be for the input parameters to be well modelled as constant over the surface? It is good procedure to solve the problem at more than one level of surface spatial discretization; then, the final solution is the one that obtains at the finest level of discretization where the answers are invariant (to within a small tolerance) from the previous (coarser) level of surface discretization.

This model results in a spatially constant (over each surface) number flux of emitted particles. In our implementation, if the surface is divided into small subsurfaces, each subsurface will emit a number of particles proportional to its subsurface area (this eliminates biases due to "hot spots"). In two dimensions, we subdivide each surface into a number of subsurfaces equal to the number of particle emissions, where one particle is emitted from the centroid of each subsurface. In three dimensions, we specify the number of subsurface divisions in two directions and weight the total number of emissions by subsurface area. Figure 13 depicts these strategies for two- and three-dimensional Cartesian surfaces (surfaces in other than Cartesian space are more complicated).

3.3 Particle/Surface Interactions

The material properties determine the particle/surface interaction and are defined in terms of a local spherical coordinate system. Figure 14 defines the cone angle, θ, and the azimuthal angle, ϕ.

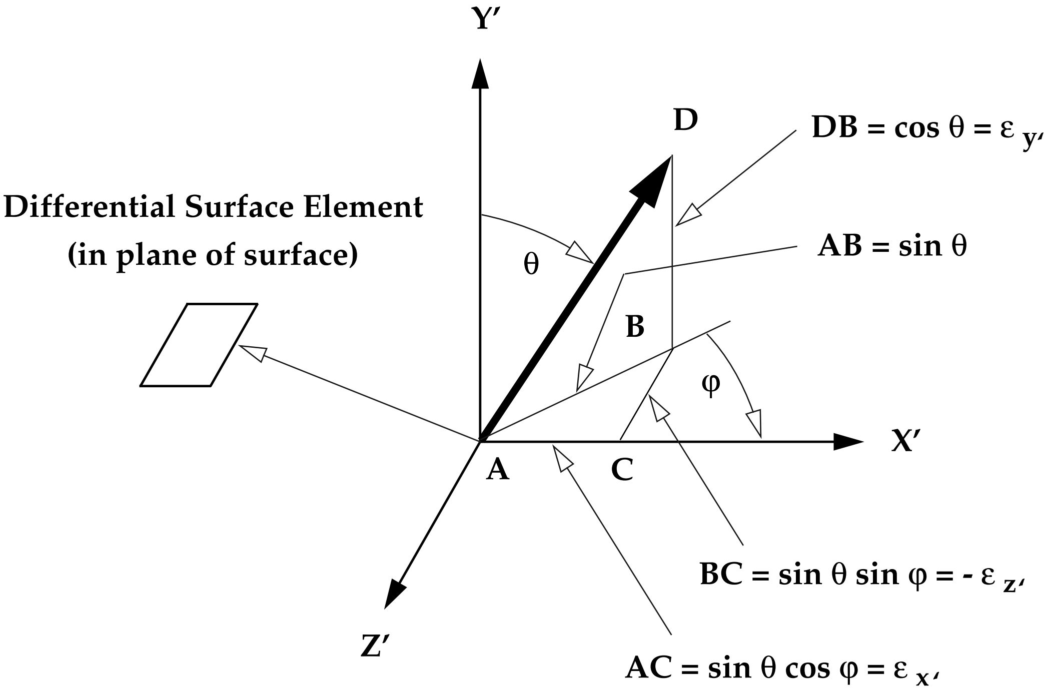


Figure 11: Local Spherical-Cartesian Transformation

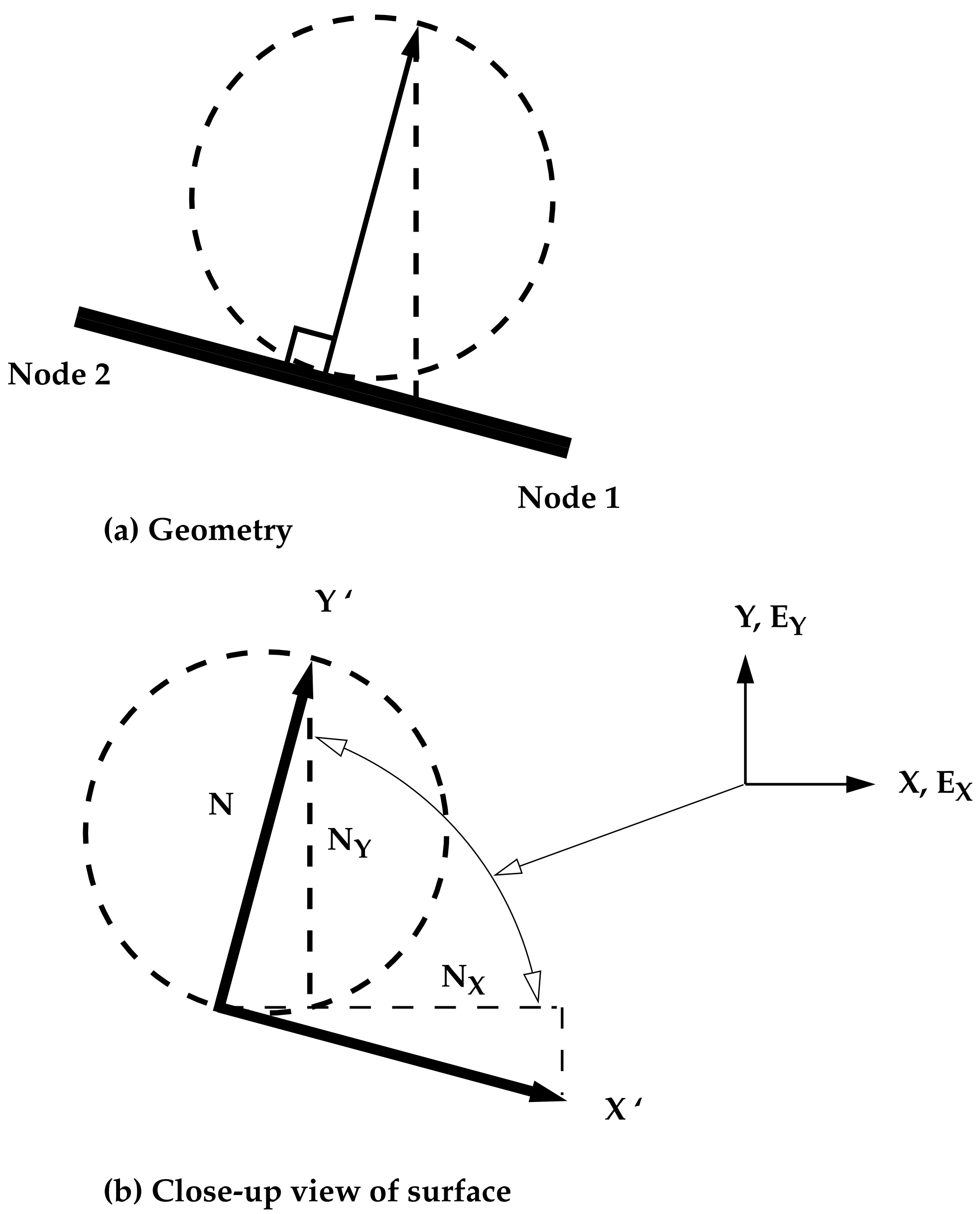
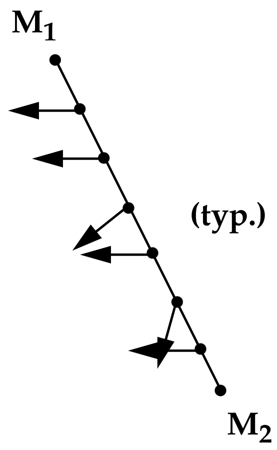
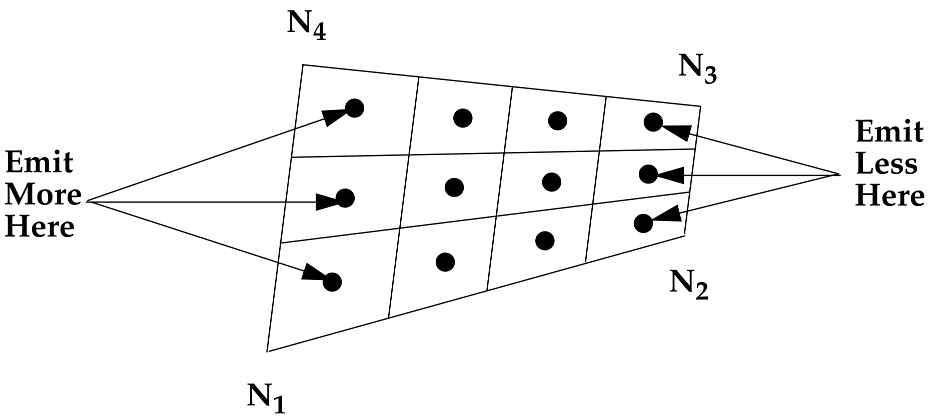


Figure 12: Local-Global Cartesian Transformation



(a) 2-D Uniformly spaced emission



(b) 3-D Emission Weighted to Subsurface Area

Figure 13: Emission in 2 and 3-Dimensions

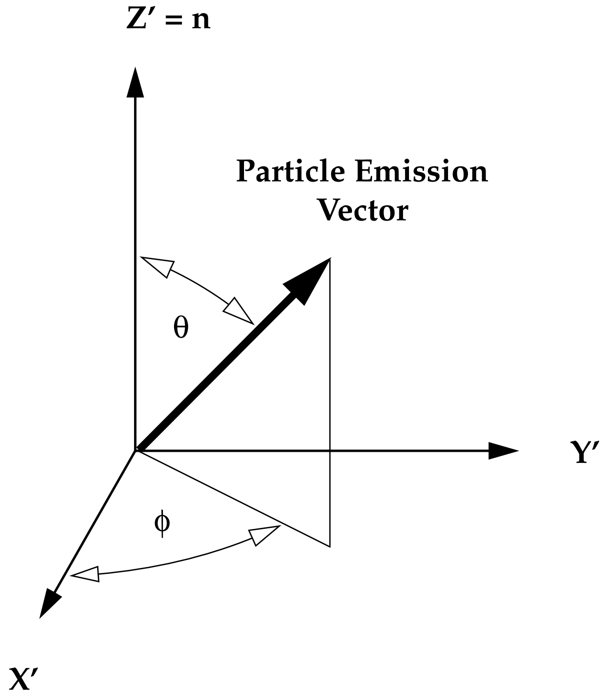


Figure 14: Local Material Coordinate System

Particle/material interactions of five types may occur as shown in Figure 15. Figure 15(a) depicts a "specular" transmission, whereby a particle passes straight through a specularly transmitting surface, with no change in direction. Figure 15(b) depicts a "diffuse" transmission, where the particle is transmitted, but its direction is "diffused" upon transmission. This is akin to a diffuse reflection, but from the "back" of the surface.

Figure 15(c) depicts a specular (mirrorlike) reflection where "the angle of incidence is the angle of reflection,"and Figure 15(d) depicts a diffuse (random) reflection, with an equal probability of reflection in any "direction." (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 particle may be absorbed. Note that, if the particle is neither transmitted nor reflected, it is absorbed—so that only 4 of the 5 material properties are required to be stored.

Explicitly:

$\tau_s(\theta_I)$ = specular transmittance at incident cone angle, θ_I

$\tau_d(\theta_I)$ = diffuse transmittance at incident cone angle, θ_I

$\rho_s(\theta_I)$ = specular reflectance at incident cone angle, θ_I

$\rho_d(\theta_I)$ = diffuse reflectance at incident cone angle, θ_I

3.3.1 Material Property Curves

Each of the above material properties is the probability of the particular interaction. Figure 16 depicts the material properties as functions of the incident cone angle, θ_I. At any particular value of θ_I, the incident particle has the following probabilities: $\tau_s(\theta_I)$, $\tau_d(\theta_I)$, $\rho_s(\theta_I)$, $\rho_d(\theta_I)$ and $1 - \tau_s(\theta_I) - \tau_d(\theta_I) - \rho_s(\theta_I) - \rho_d(\theta_I)$ for specular and diffuse transmission, specular and diffuse reflection, and absorption, respectively. Each of these curves must be input (or generated) by point value as a function of cone angle.

Material properties of much greater complexity could easily be implemented in Monte Carlo analysis (indeed, this is one of its great strengths). For example, at each incident angle, we could have specified outgoing directional distributions for transmission and reflection, in which case we would need to specify a *surface* of outgoing directions for each point on the abscissa of Figure 16 (obviously, we would have to discretize the abscissa intelligently). However, in most cases, we lack sufficient data to make this level of effort worthwhile. So, we restrict our attention to two specific outgoing directional distributions—specular and diffuse, and represent properties as shown in Figure 16. Finally, we note that we have found it sufficient to provide data every 1° in θ_I and to perform linear interpolation between points. For efficiency, we store these data together with their differences in tabular form. Thus, each material requires a storage space of $2 \times 91 \times 4 = 728$ words for particle/material interactions.

Our procedure for determining particle/surface interactions is then as follows. After the intersection is obtained, we determine the incident angle. At this incident angle, we look up (from the material property table) the probabilities of interactions. We choose a random number from a uniform distribution (between 0 and 1) of random numbers to determine the specific interaction for this particle. It is necessary to sum the probabilities for this, that is,

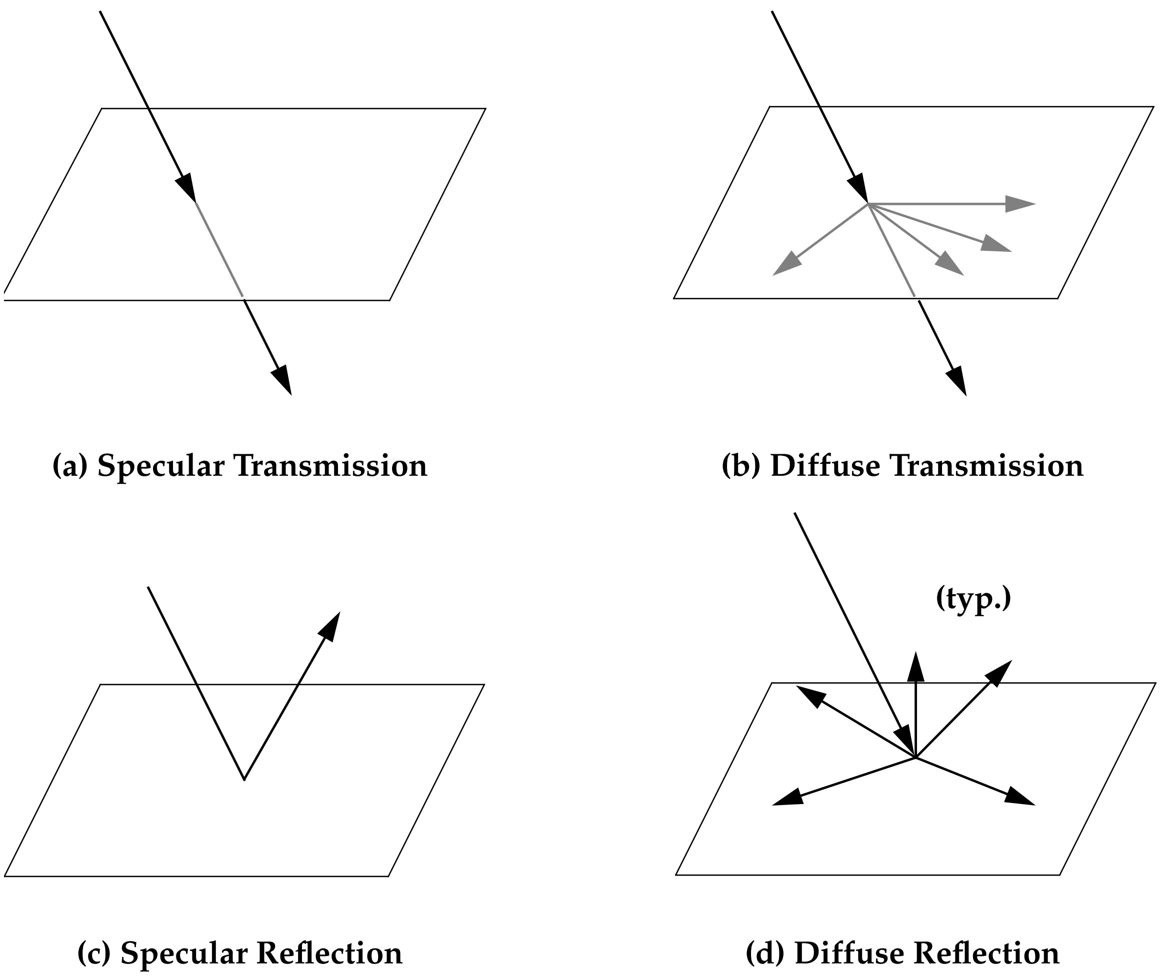


Figure 15: Particle/Material Interactions

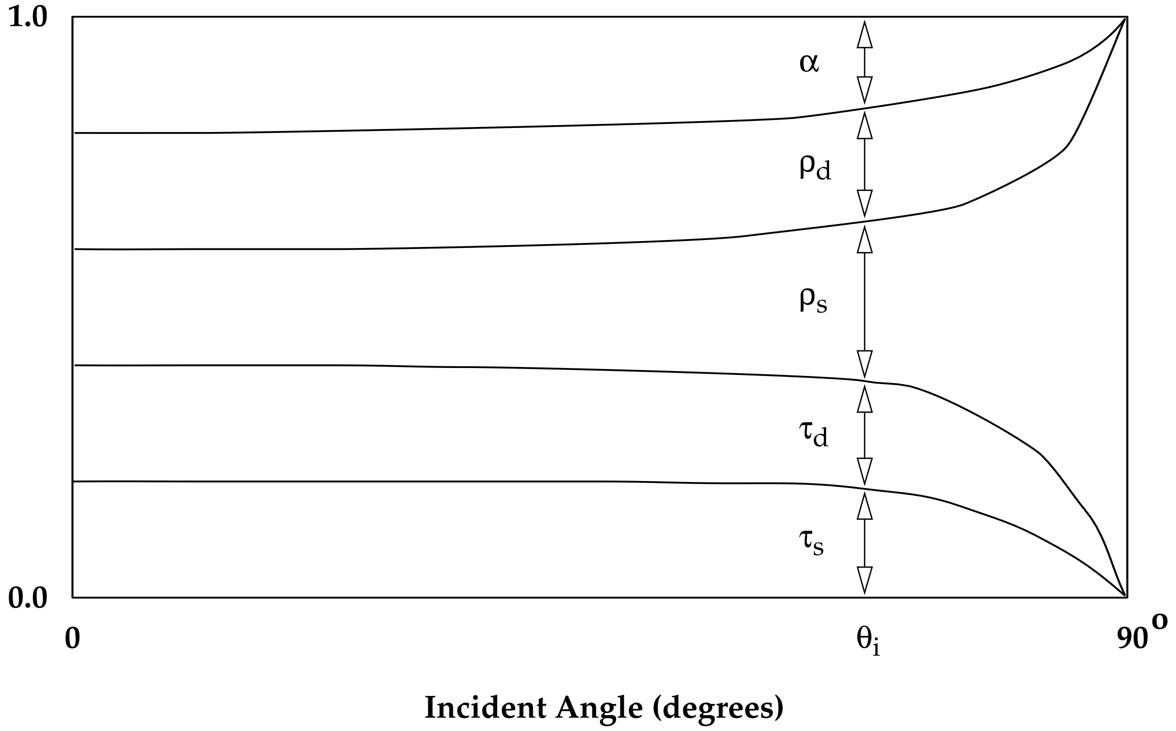


Figure 16: Material Properties vs. Incident Cone Angle

if the random number is less than τ_s, the particle is made to undergo a specular transmission; if the random number is greater than τ_s, but less than $\tau_s + \tau_d$, then the particle is diffusely transmitted; etc. For efficiency, it is wise to sum all properties during the input phase (i.e., instead of storing τ_s, τ_d, etc., we store τ_s, $\tau_s + \tau_d$, etc.) at each angle, so the sums need not be calculated during the solution phase.

4 Vectorization and Parallelization

4.1 Introduction to Vectorization and Parallelization

In this section, we begin with an example of a parallel implementation when tracing particles within a simple, two-dimensional square enclosure as shown in Figure 17. The surface numbers are enclosed in circles, and are specified in a clockwise fashion (this is by no means necessary—the surfaces may be input in unordered fashion).

For simplicity, we emit 10 particles from each surface, in parallel groups of 8. The concepts are directly extensible to larger numbers of particles per surface and larger parallel groups. We proceed with a step by step description of the process as if it were done on a vector architecture (the concepts are identical to those for a parallel architecture).

4.2 Parallel Example

Here, we use the concept identified by Martin and Brown [2] of the *event step*, where particles are traced from physical event to physical event, independent of time. For our simple problem, only 3 steps are necessary per physical event. The details for the following example were taken from an actual Monte Carlo simulation and are not contrived.

4.2.1 Step One—Emission

Here, we fill the vector of emission directions as follows:

- *Uniformly distribute in global coordinates emission positions to result in 8 emission points* $\mathbf{X_E}$
- *Compute outgoing directions in local spherical coordinates:* θ_0 *and* ϕ_0
- *Transform to local Cartesian directions* $\mathbf{e}_0$*, and*
- *Transform to global directions* $\mathbf{E}$*.*

The result is shown in Figure 18, i.e., a set of emission points $\boldsymbol{X}_E$ and outgoing global directions $\mathbf{E}$.

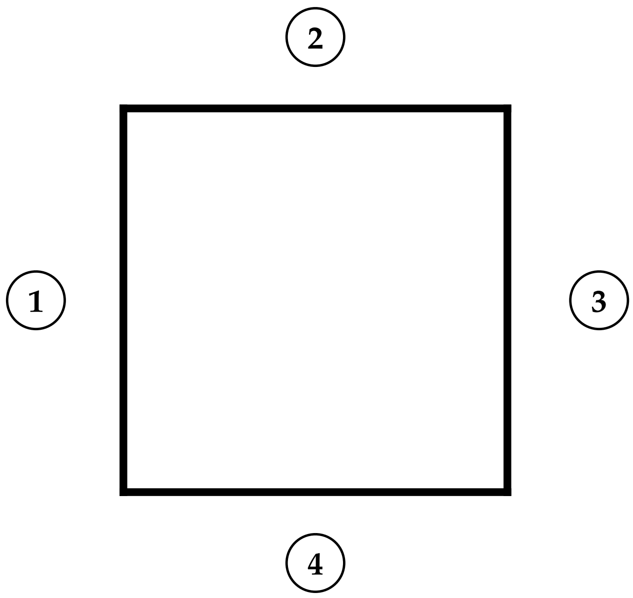


Figure 17: Geometry of Interest

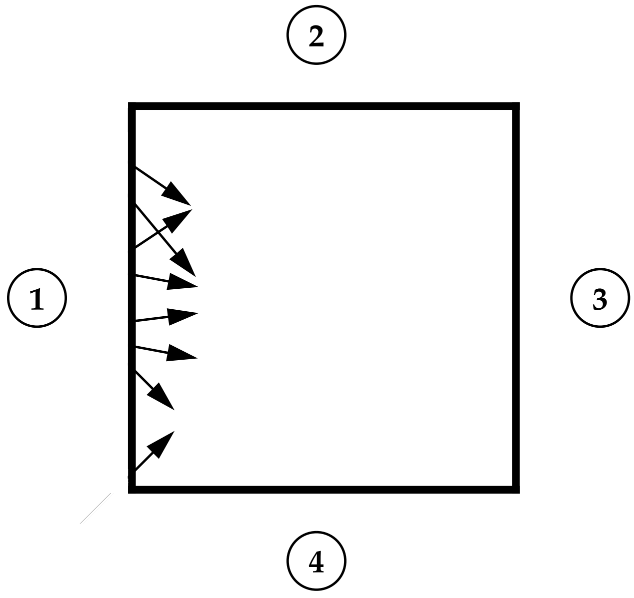


Figure 18: Initial Emission

4.2.2 Step 2—Particle Disposition

Actually, this is a two-step process which, for convenience, we "lump" into one subheading. The two steps are:

- *Trace particles to intersection points, and*
- *Determine the particle's interaction with the surface of interest.*

These steps are displayed pictorially in Figure 19. Note that, the particles are "translated" along their initial direction vectors until a surface of intersection is determined (this completely specifies the geometry of intersection, in particular the angle of incidence which determines the material properties). At these angles of incidence, and for the specific materials of the surfaces of intersection, we "look up" the material properties. We have the following possibilities, so indicated on Figure 19: absorbed (A), diffusely reflected ($R_{\mathcal{D}}$) or specularly reflected ($R_{\mathcal{S}}$) (here, we disallow transmissions for clarity). For those particles which are absorbed, we count them in the absorption matrix N_{ij}, where:

$$\begin{aligned} i &\text{ - denotes the surface of origin, and} \\ j &\text{ - denotes the surface of absorption.} \end{aligned}$$

Thus, after this event:

$$\begin{aligned} N_{12}(\text{after}) &= N_{12}(\text{before}) + 2 \\ N_{13}(\text{after}) &= N_{13}(\text{before}) + 1 \\ N_{14}(\text{after}) &= N_{14}(\text{before}) + 1 \end{aligned}$$

4.2.3 Step 3—Second Emission

We depict this in Figure 20. This also consists of two parts (which must be handled separately):

- *Disposition of "surviving" particles, and*
- *New emissions to fill the vector (note that these may extend over multiple surfaces as shown).*

Figure 20 depicts a situation close to the general case. Namely, that each particle has distinct values of: (1) *emitting* surface for this event step, (2) *emission* point for this event step, (3) originating surface, and (4) emission direction. The words "emitting" and "emission" in points (1) and (2) are italicized, as they represnt either a "birth" if this is the first event for this particle or a reflection if the event time for this particle is beyond the "birthing" event, while in point (3) "originating" represnents the "birthing" event.

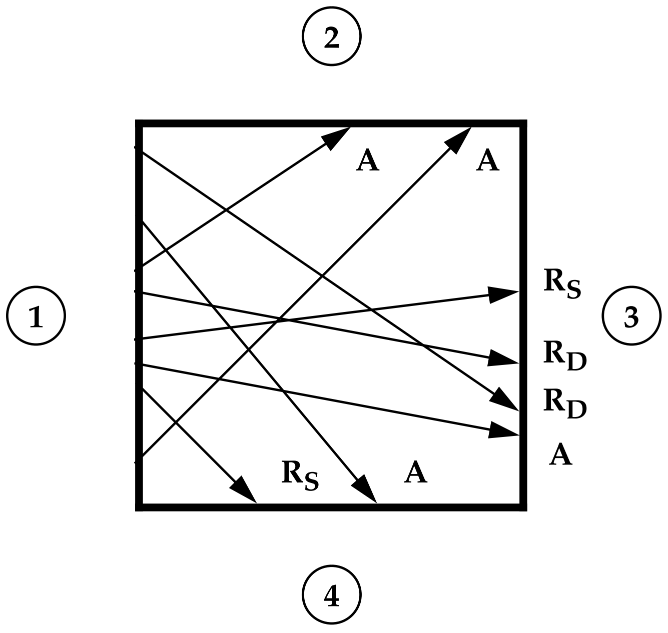


Figure 19: Step-2 Particle Disposition

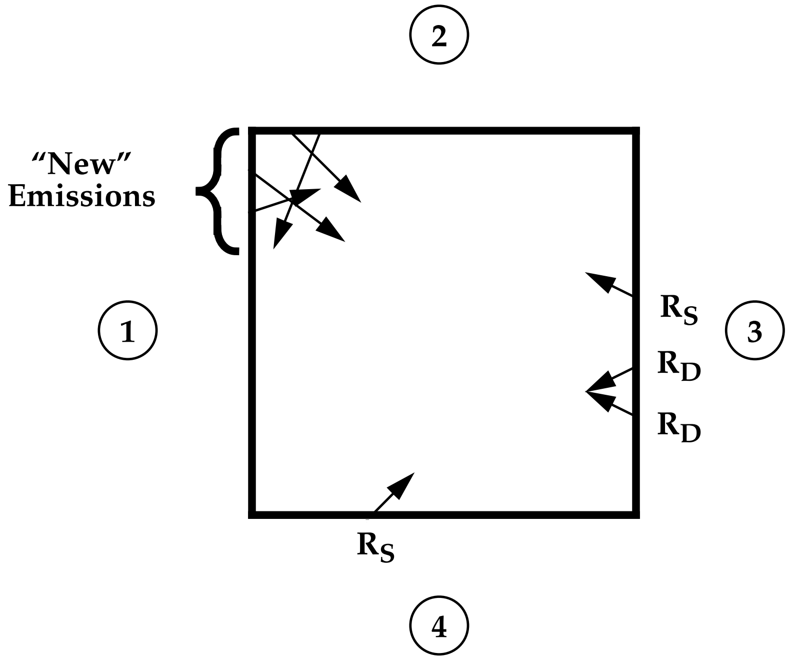


Figure 20: Step-3 Second Emission

4.2.4 Inductive Continuation

The above process repeats until all particles have been emitted and absorbed. There are several important factors to observe in order to achieve good parallel efficiency:

- *The vector (or parallel) lengths should be kept sufficiently long so as to yield good efficiency. Note that, as particles pass the hierarchical tests (i.e., traverse the "sieves"), the vector length tends to decrease. To ensure good overall efficiency, the shortest vector length should be long enough to obtain good efficiency. A suitable vector length is usually determined empirically and depends upon: (1) the input parameters (geometry and material properties) and (2) architecture.*
- *It is important to maintain long vector lengths by re-filling the vectors with new emission as particles "die off."*
- *A "tail" will inevitably exist at the end of this process, as the particles remaining after the last emission will need to be traced through enough events for them all to "die off." Zhong and Kalos [4] have identified this as the "straggler" problem. We are fortunate that it exists only at the end of our problem. In some physical situations, stragglers exist during each event step, requiring synchronization between events.*

4.3 Modified Amdahl's Law

Here, we modify Amdahl's Law [1] to describe the operations required to achieve the above vectorized events. As Monte Carlo processes are probabilistic, they result in random branching. This inhibits direct vectorization, as the particles which "pass" tests are generally located discontiguously in memory. On vector architectures, we must either: (1) pay the penalty of data motion to "gather" elements into contiguous locations in memory (and perhaps subsequently to "scatter" elements into their discontiguous original positions in memory) or (2) pay the penalty of full (100% - passed and failed) memory retrieval of elements, while masking-off elements which are not to have their values stored. On SIMD architectures, the penalty is incurred by conditionally masking-off individual processors where particles do not pass the test.

Suppose we are mapping this process to an architecture and desire to model the speedup from either: (1) scalar to vector or (2) single processor to multiprocessors. Here, the speedup is the ratio of: (1) scalar to vector or (2) single to multiprocessor execution times. Letting:

R = speedup ratio of scalar to vector execution times ()
D = fractional overhead due to (useless) data motion ()
F = fractional overhead due to vector start-up ()
f_p = fraction of work that can be done in vector (parallel) hardware, dimensionless
S =scalar execution rate (MFLOP's)
V = vector execution rate (MFLOP's)
W = useful work to be done (number of operations or OP's)

Then, the elapsed time to run the problem on a scalar processor will be:

$$\Delta t_s = \frac{W}{S} \tag{27}$$

where the elapsed time is simply the number of operations to be performed, W, divided by the scalar execution rate, S. Now, suppose that some fraction of this work, f_p, can be done in the vector unit or in multiple processors. Additionally, suppose that the overhead associated with this is composed of two parts: (1) a fractional addition, F, due to vector start-up (or due to the spreading of the problem over multiple processors) and (2) a fractional addition, D, due to vectorized data motion (or due to inter-processor communication). Note that, these fractional additions are purely overhead, as the work done by the CPU is useless work. Then, the elapsed time to execute the problem on a parallel or vector machine is:

$$\Delta t_p = \frac{(1-f_p)W}{S} + \frac{f_p W}{V}[1+F+D] \tag{28}$$

Then, dividing the above two equations yields R the speedup ratio:

$$R = \frac{1}{1 - f_p + f_p \frac{S}{V}[1+F+D]} \tag{29}$$

Note, as F and D appear in conjunction, we can combine their effects into one. Thus, hereafter, we let D represent the sum of F and D. Moreover, it is generally possible to select long enough vector lengths so that F is negligible; however, data motion, D, is always significant. In effect, we pay the overhead of data motion (useless work "gathering" data elements into contiguous memory locations), so that we can perform subsequent operations in the much faster vector hardware.

In Figure 21, we depict R versus f_p for $V/S = 12$ (representative of Cray Hardware), with D varying parametrically.

Several things are noteworthy from the graph:

- *High values of R are attainable, even if there is significant overhead in data motion,*
- *f_p must be large to obtain good speedup, and*
- *The curves are* relatively *insensitive to D*

If we fix V/S (nominally fixed by the architecture), we can consider equation (1) to contain two independent parameters, f_p and D. Therefore, if we measure R for two different architectures with known but different values of V/S, we can determine both f_p and D.

Burns et al. [3] have performed such an experiment on a Cray Y/MP and an ETA 10-G for the Monte Carlo simulation GAMTEB—one of Los Alamos' benchmark programs [2]. We measured (by turning vectorization "on" and "off" for critical loops) values of V/S of 12 and 25 for the Cray Y/MP and the ETA 10-G, respectively.

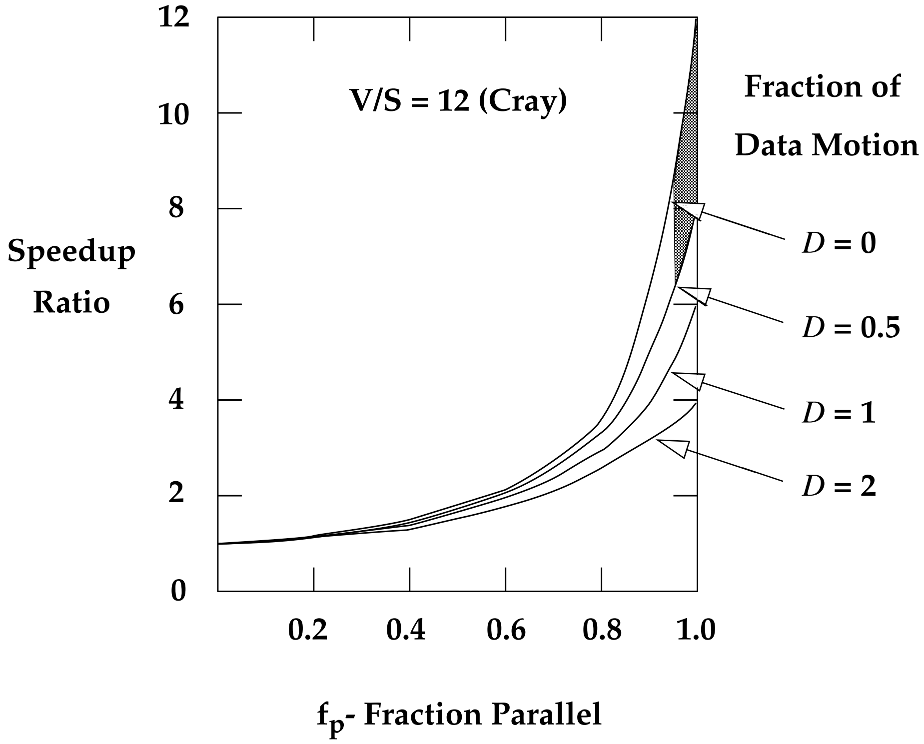


Figure 21: Modified Amdahl's Law

In particular, GAMTEB exhibits many "standard" aspects of Monte Carlo simulations, so this exercise should yield representative values for these two parameters. The results were:

$$\begin{aligned} f_p &= 0.993, \text{ and} \\ D &= 0.235, \text{ yielding} \\ R &= 9.15 \end{aligned}$$

Additionally, we independently verified these numbers using Cray's hardware performance monitor (hpm). Ergo, we conclude that Monte Carlo programs may be very highly vectorized ($> 99\%$), with about a 25% penalty due to data motion. For Cray architectures, the speedup that results exceeds 9. This region of the parameter space is shown shaded in Figure 21.

4.3.1 Perspective

In the present formulation, particles do not interact. This total independence causes the present application to fall into the realm of applications that have been described as "embarrassingly" parallel. The only difficulty is the random nature of the branching, and this is able to be handled in several ways (refer to Burns et al., 1991).

Indeed, this problem exhibits parallelism at various levels. The fine-grained level is in the independence of the particles, and the suitable scale of vectorization is the particle. An approach at a coarser level would entail vectorization over multiple surfaces. An even coarser level would be parallelizing over different energy levels of the emissions. Of course, the above approaches can be intermingled, but the implementation becomes slightly more difficult in that case, as additional detail is involved.

Choices of parallel scope should be matched to the architecture. Greater numbers of processors require finer-grained parallelism. Massively parallel architectures such as the Connection Machine Model CM-2 perform very well at the fine-grained level. One quadrant (16,384 processors) of a CM-2 performs about equally well (in total CPU time) as one CPU of a Cray 2. With the emergence of FORTRAN 90, identical code will be able to be executed on both machines.

5 Monte Carlo Code—Exercise

5.1 Introduction to Monte Carlo Exercise

Here, we examine a two-dimensional particle-tracing code which is a fully vectorized Cray implementation. We supply a FORTRAN code with "bleeps"—i.e., lines either wholly or partially incomplete, including periods in place of some FORTRAN instructions. The student is asked to supply the FORTRAN code in place of the "bleeps."

For the sake of simplicity, the code is constructed according to the following parameters:

- *Emission is perfectly diffuse, i.e., Lambertian.*

- *Surfaces are perfectly absorbing, i.e., upon impact the particle is absorbed—there is no need to deal with either reflection or transmission.*

- *The code is written to describe the radiative transfer in a polygon, of NSURF equal length sides. Here, we shall simulate a square, thus NSURF = 4.*

5.2 Locating the Code

The code *square.f* is located in the anonymous ftp area on compsci.cas.vanderbilt.edu in CSEP/codes.

5.3 Code Details

The code traces particles in a perfectly absorptive square prismatic geometry. Although simplified, this problem well illustrates the concepts of vectorizing Monte Carlo surface to surface transport. We proceed with a top-down description of the code.

Subroutines:

input - performs initialization of geometry and material properties (inverse CDF for emission)

emit - performs emissions

intsec - performs intersections, "counts" absorptions

Parameters:

np - length of vectors (arrays) for particle variables (here, np = 10,000)

npsurf - number of particle emissions per surface (here, $npsurf$ = 100,000)

ns - number of surfaces (here, $ns = 4$)

Variables:

Geometry:

x(ns,2), y(ns,2) - surface endpoint coordinates for ns surfaces

delx(ns), dely(ns) - spacing between surface nodes

s12(ns) - square of distance between surface endpoints

cross(ns) - cross product of vectors from origin to surface endpoints

x0(ns), y0(ns) - offset for particle emissions for ns surfaces (emission begins in the middle of a subsurface)

delxp(ns), delyp(ns) - differences between successive coordinates for emission

enx(ns), eny(ns) - components of surface normal vector

Material Properties:

thetar(101) - inverse CDF for emission

delthr(101) - differences between inverse CDF for emission

Particle:

xe(np), ye(np)	- particle origination positions
ex(np), ey(np)	- global components of particle direction vectors
exloc(np), eyloc(np)	- local components of particle direction vectors
det	- the denominator of both equations (3) and (4)

Output:

e(ns,ns)	- calculated matrix of exchange fractions of particle transport
etrue(ns,ns)	- analytical matrix of exchange fractions of particle transport
err(ns,ns)	- difference between calculated and analytical exchange matrices
errsumi(ns)	- row sums of *err(ns,ns)* for each row *i*
errsum	- column sum of *errsumi(ns)*

In subroutine intsec, we employ the concept of a sieve, where we wish to pass through the sieve the particles which pass the various tests. Those particles which pass the test, we "compress" into new vectors, and denote the new vector as the name of the prior vector with a "c" appended to denote a compressed quantity. We undergo two levels of compression in this simple example.

5.4 Program Execution

The code is well commented (refer to the Postscript two-column listing for line numbers). Here, we go through the process, subroutine by subroutine.

Subroutine *input*

1. First, the geometry is calculated (lines 182 through 219).

2. Next, the material properties are computed and stored (lines 220 through 232).

3. Finally, the analytical answers are calculated and stored (lines 233 through 263).

Next, the solution phase is invoked. Line 54 initiates the loop over surfaces. For each surface, we loop over *nstop* full surface emissions (line 58), each of *np* particles (here, 10,000). Within this loop, we invoke the following subroutines:

Subroutine *emit*

1. The "starting" coordinate is defined for the first particle of this vector of *np* particles (lines 292–298).

2. The emission coordinates are defined in lines 299–306. The student must supply the right sides of lines 305 and 306.

3. The outgoing local spherical angles (lines 307 through 318), local Cartesian components (lines 319 through 325) and global Cartesian components (lines 326–333) are obtained. The student must supply the right sides of lines 313, 318, 324, 325, 332 and 333.

Subroutine *intsec*

1. First, the cross products for all particles in the particle vector are obtained (lines 367–371). The student is required to supply the right-hand sides of lines 370 and 371.

2. Next, the loop over potential surfaces for intersection is executed (lines 372 through 444). The student must supply the right sides of lines 397, 413, 414, 415, 420, 421, 422, 423, 430, 431 and 432. The loop begins by calculating variables constant within the loop (for storage in registers) (lines 372–390).

3. The dot product is calculated in lines 394 through 397. The dot product test is performed by the Cray SCILIB routine *whenflt*, which returns the indices of those elements for which *dotp* is less than 0 (note, the first argument is the number of elements to be searched, the second is the input vector, the third is the increment for the search—use 1 as we wish every element to be searched, the fourth is the conditional against which the input vector is tested, the fifth is the returned vector of indices for those elements which pass the test and the last is the returned number of elements which pass the test [i.e., the length of the vector of indices]).

4. In lines 409 through 415, the elements of the original vectors for which the test was passed are compressed (gathered). Intersection points are calculated in lines 416–425 for the compressed vectors. The squares of distances form the surface endpoints to the intersection points are calculated in lines 430 and 430 and differenced to the square of the surface length in line 431. This difference is then used in line 439 to test those elements of the compressed vector which lie within the surface. In line 441, the number which pass the test (*npa*) is accumulated in the matrix *e(ns,ns)*.

See exercises 3 and 4.

---

6 Daylighting Example

Daylighting is an emerging area of Monte Carlo surface to surface transport that is gaining increased acceptance from the building design community. The essence of daylighting is to use the structure of the building to capture and distribute internally within the building the natural sunlight, in an effort to live in harmony with the environment as opposed to butting heads against it. Daylighting design involves the careful design and placement of windows to capture the sunlight and the implementation of an internal distribution system within the building. Issues which obtain are: (1) effecting the design so as to prevent excessive glare and (2) the degree to which electrical lighting energy is displaced, resulting in reduced energy

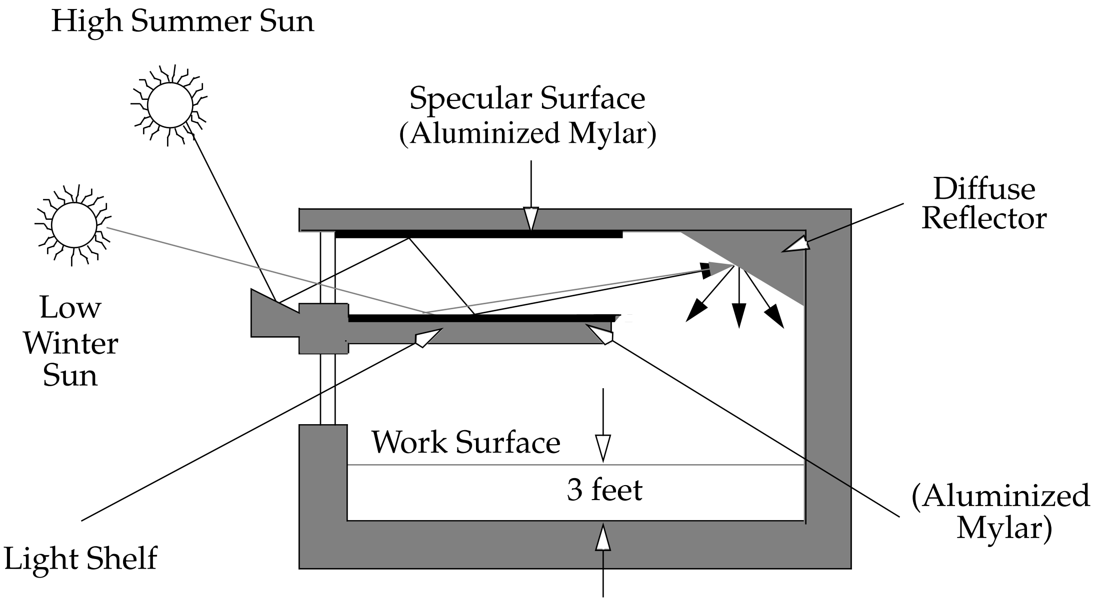


Figure 22: Cross Sectional View of Daylighting Geometry

consumption and cost savings. As we shall indicate, Monte Carlo offers some pronounced advantages over traditional design techniques, due to its ability to simulate accurately the interaction of sunlight with sophisticated building materials

To illustrate the daylighting process, we provide Figure 22, depicting a cross sectional view of a south-facing room of a typical commercial building. A lower view window 3' in height is located 4' above the floor. This view window provides light to about the front one-third of the room. Indeed, the problem with previous daylighting designs is that strategies to get the light into the rear of the building have been woefully ineffective. To address this limitation, our strategy is to use a specular light duct to light the rear two-thirds of the room. The light shelf begins 8' above the floor and is about 2" thick (a suspended ceiling). The height above the light shelf is 3', making the total height of the room 11'—suitable for running ducts for a heating/cooling system in the space (11' heights are common in commercial construction). Sunlight entering the upper light duct window encounters the aluminized mylar coating the inside of the light duct, is specularly reflected (analogous to the reflection from a mirror) and traverses the light duct until it strikes the tilted diffuse reflector at the rear of the light duct. Normally, this is a white surface, painted with a flat paint to reflect the light diffusely down and forward toward the front of the room. Monte Carlo offers the ability to simulate specular reflections, which provide just such a means to get the light into the rear of the room.

The Monte Carlo simulation process proceeds as follows. Individual photons of daylight are emitted from a plane outside the room. Each is traced from birth to death, through possibly many intermediate interactions (reflections or transmissions) with surfaces. An event proceeds as follows. First, all photons are traced to their surface of interaction. Next, the interaction occurs, resulting in the photons being absorbed (death) whence tracing ceases, reflected or transmitted. If reflected or transmitted, the outgoing direction is determined. The tracing then recommences with the next event.

The objective of each complete simulation is to compute the lluminance of the work surface. A completely transmitting "work" surface at a height of 3 ft. (1 meter) above the floor is subdivided into a 10×10 grid. The number of photons which pass through each subdivision of the work surface are tallied. A series of simulations are performed, varying the geometry and the material properties, to optimize versus cost the parameters of the design.

In the accompanying video, we illustrate this process by displaying photons as they enter the geometry of Figure 22 and interact with interior surfaces. There are multiple views of the process, from the rear of the room, and various side perspectives. Photons which are emitted from the exterior "solar" surface are initially colored yellow. If they succeed in passing through the windows into the room, their color is changed to red for those which are transmitted through the upper light duct window or green for those which are transmitted through the lower view window.

In this particular example, there are mini-blinds inside the view window, which serve to reflect the strong sunlight up onto the bottom of the light shelf, which is a diffuse white surface. This diffuses the strong daylight and distributes it to the front one-third of the room. A close-up view of this process is also shown on the video.

The video has been very effective in illustrating the process to non-technical audiences. It has received particular attention from the Architectural community.

See exercise 5.

---

7 Exercises

Exercise 1 *Verify equations (3) and (4) by hand calculations.*

Using equations (1), (2) and (5), verify equations (3) and (4). Taking the unit square geometry of Figure 3(a), assume emission points at the four centerpoints of the surfaces and normal emission. Thus, for the left-hand, vertical surface, take: $\mathbf{X_E} = (0, 0.5)$ and $\mathbf{E_X} = (1, 0)$. According to equations (3) and (4), attempt to compute the intersection points for surfaces 2–4, etc.

Exercise 2 *Explore these concepts using Cray's Random Number Generator.*

Here, we shall use the Cray random number generator, *ranf()*, to perform emissions according to Figure 10 and Equations (23) and (24). *We will do this by generating vectors (in the computer sense) of random numbers—one vector for R_a and one vector for R_b.* Using these, we'll generate one vector of ϕ values and one other vector of θ values. As we generate "many" emissions, we should expect the distribution of emissions to approach: (1) a uniform distribution in ϕ and (2) $\sin(\theta)\cos(\theta)\,d\theta$ in θ. First, generate 10,000 emissions (requiring 20,000 random numbers - 10,000 to obtain the 10,000 values of ϕ, and another 10,000 to obtain the 10,000 values of θ). Partition these emissions into groups of $N = 10$, 100, 1,000 and 10,000 (you can do this by referencing the first 10, 100, ... elements in your θ and ϕ arrays).

(a) Consider the distribution function for ϕ. Partition the domain into 36 intervals (bins) of 10° each (0°–10°, 10°–20°, ...). For each N, plot the function of emissions which fall into the bins vs. the bin centerpoint (i.e., plot N_{bin}/N vs. $\phi_c = 5°$, 15°, ...). Comment on the "convergence" of the distributions.

(b) Now perform a similar test for θ. Divide θ into 9 bins of 10° each, with $\theta_c = 5°$, 15°, ... Plot N_{bin}/N vs. θ_c for all 5 N's. Comment on the "convergence" here, too.

Notes:

1. The exact answers are:

(a) $N_{bin}/N = 1/36$

(b)

$$\frac{N_{bin}}{N} = \frac{\int_{\theta=\theta_c-5°}^{\theta_c+5°} \sin\theta\cos\theta\,d\theta}{\int_{\theta=0°}^{90°} \sin\theta\cos\theta\,d\theta} = \sin^2(\theta_c + 5°) - \sin^2(\theta_c - 5°) \tag{30}$$

2. Do the above calculations as efficiently as possible (if on the Cray, use *hpm* to assess MFLOP's).

Hand in: (1) your program, (2) the plots, and (3) discussions.

Exercise 3 *Implement a parallel Monte Carlo particle tracing algorithm on a Cray architecture.*

Obtain the code as specified in section 5.2. Supply the portions of the code "bleeped" out—these are detailed in section 5.4. Compile and run the code on a Cray architecture. Make sure your answers are correct (the errors should all be less than 10^{-2}). Hand in the code and the output, together with your comments on how well the Cray did (was it fast?). Use *hpm* to assess performance, and relate this to the photon flight rate computed and output from the code.

Exercise 4 *Implement a FORTRAN 90 version of the parallel Monte Carlo particle tracing algorithm on any architecture.*

First, finish Exercise 3 above. Then, convert the working code to FORTRAN 90. Run the same example as above, and verify that your answers are the same. (Note: Cray has FORTRAN 90 compilers available now. Many IBM RS 6000's have FORTRAN 90 compilers, and there are a variety available for PC's.) Report the results of this exercise to the author of this chapter (send electronic mail to pburns@westnet.net) to receive congratulations and be covered with glory.

Exercise 5 *Implement a daylighting version of the Monte Carlo particle tracing algorithm*

Modify the code obtained in Exercise 3 to perform a simulation of daylighting. This modification is accomplished in a series of steps as follows.

(a) First, photons must be emitted from only one surface (choose a vertical surface - say the left side), a "solar" surface which simulates photons coming from the sun. They can be emitted in a constant direction, and should be emitted with a positive E_X and a negative E_Y. Photons need not be emitted from any other surfaces. Also, if you implement a non-zero reflectance (see part (c), then all photons which get back to the "solar" surface should not be reflected (i.e. should be absorbed).

(b) Second, the photons from the "solar" surface must be traced and those which pass through the "work" surface (corresponding to a typical height of a desk) must be tallied. It is useful to capture the flux of these photons by partitioning the work surface into a grid (10 grids in the X direction is a good number). This work surface is a non-interacting surface, used only for tallying photons which pass through it. It is convenient to define this surface to be a plane at $Y = 0.3$ in the geometry given.

(c) Finally, you may wish to make the surfaces reflecting. Choosing a diffuse reflectance of 0.7 is typical of a surface painted flat white, yielding an absorptance of 0.3.

References

[1] Amdahl, 1967. "Validity of the single processor approach to achieving large-scale computing capabilities," Proceedings of the American Federation of Information Processing Societies, Washington, DC, Vol. 30, pp. 483-485.

[2] Brown, T. B. 1981. *Vectorized Monte Carlo*, Ph.D. Thesis, University of Michigan, Department of Nuclear Engineering, Ann Arbor, MI.

[3] Burns, P. J., Maltby, J. D. and Christon, M. A., 1990. "Large-scale surface to surface transport for photons and electrons via Monte Carlo," *Computing Systems in Engineering, 1*(1), pp. 75-99.

[4] Zhong, Y-Q. and Kalos, M. H., March 1983. "Monte Carlo transport calculations on the Ultracomputer," Ultracomputer Note 46, Courant Institute, New York.