DEVELOPMENT OF A DAYLIGHTING PROGRAM WITH THE $\hat{F}$ METHOD

by

Jan-Uwe Kluessendorf

A thesis submitted in partial fulfillment of the

requirements for the degree of

MASTER OF SCIENCE

(Chemical Engineering)

at the

UNIVERSITY OF WISCONSIN - MADISON

1987

APPROVED:

Dec 16, 1987

Date

Prof. J.A. Duffie

ABSTRACT

Development of a daylighting program with the $\hat{F}$ method

by

Jan-Uwe Kluessendorf

under the supervision of Professors John A. Duffie and Sanford A. Klein

The daylight received at a sensor positioned at a specified point in a room can be viewed as the sum of three components: direct sun, direct diffuse, and interreflected light. It can be determined from geometrical considerations if direct beam illuminance occurs; i.e., when the sensor is directly radiated. The direct diffuse component is the daylight reaching the sensor from diffuse light sources (e.g. the window plane) without being interreflected. The window plane is diffusely illuminated from the patch of sky it views and from light that is reflected from the ground. Interreflected light reaches the sensor after diffuse and beam light are reflected from the room surfaces once or multiple times.

In the absence of direct beam illumination, the interreflected light can be a major daylight component at the sensor position. An exact calculation of the interreflected illumination depends on the spectral and specular properties of all room surfaces. Such calculations can be done with the Monte Carlo method, but with significant computational effort. At the other extreme, interreflected light can be estimated with

the split flux method [Narasimhan], in which the interior surface is considered to be spherical with suitably-averaged properties.

This paper describes a new method for calculating the interreflected daylight component which is computationally efficient, yet does not require the restrictive assumptions of the split-flux method. The method is based on the $\hat{F}$ (F-hat) thermal radiation exchange methodology developed by Beckman [Beckman]. In this case $\hat{F}_{ij}$ is defined as the fraction of the daylight leaving surface i which reaches surface j by all possible paths. Assuming the room surfaces to be diffuse reflectors, the value of $\hat{F}_{ij}$ depends only on the configuration factors of surface i to surface j and the reflectances of the room surfaces. Matrix algebra must be done only once at the start of the calculations. The effect of reflected radiation from room surface patches which are directly illuminated is accounted for by treating these patches as additional windows.

A program has been developed to calculate the direct and reflected daylight at a specified position in a rectangular room with one or more windows. Using the $\hat{F}$ method, the program requires minimal computational effort and can be used for hour-by-hour daylight calculations for yearly periods.

ACKNOWLEDGEMENTS

The German Academic Exchange Service (DAAD) provided me with the possibility of attending graduate school at the University of Wisconsin - Madison. This scholarship supports participants of the program "Integriertes Auslandsstudium" for two semesters. For my third semester, I was granted a research assistantship by the Department of Chemical Engineering. My parents improved my financial situation throughout my stay, and without their help, this thesis would not have been possible.

At least as important as the financial aid, was the technical knowledge and engineering expertise of Professors J.A. Duffie and S.A. Klein. Their encouragement and ideas were essential.

I want to thank Karla for taking care of another component of my life in the United States that needed support. Without her, I would not have survived my "zombie periods". I appreciate her patience, especially during the last several weeks, and admire her ability to give me moral support as my personal back-up system. Finally, her proofreading eliminated my German punctuation in this work.

Besides supporting me financially, my parents were as understanding as they could be from 3000 miles away, and they helped me with many encouraging letters and care-packages.

Going to school in Madison was an unforgettable experience. The Solar Energy Laboratory was a great place to work, and I will recommend it to all future German exchange students.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

The engineering objective of calculating the daylight inside a room is to predict the electric lighting load that is needed to obtain a specific workplane illuminance. With this information, lighting loads may be reduced resulting in lower air conditioning costs. The computer program described here, FLITE, furnishes the user with the required data for these calculations, i.e. the illuminances at any specified point on a plane parallel to the floor of a rectangular room.

The nomenclature of used parameters is given after each formula. For recurring parameters, the definition is noted previously in the same chapter or following a referenced expression.

1.1 Overview

A computer program has been developed that calculates the daylight in a rectangular room with any combination of windows in the four wall surfaces. The program is based, in part, on the daylighting program DALITE [Gillette] of the National Bureau of Standards (NBS), although the approach to solving the internal reflection of daylight is entirely different.

It was not possible to use DALITE for the generation of illuminance data due to apparent errors in that program. For instance, for certain geometries, if the point for

which the illuminance had to be calculated did not view the sky, the total daylight turned out to be zero. This cannot be true, because there should still be light reaching the point which was reflected from the room surfaces. Considerable time has been invested trying to debug the program. The author of DALITE was very cooperative but it was ultimately decided that it was impractical to base this work on DALITE. The consequence was that a new program had to be designed, since other daylighting programs in source code form and suitable for incorporation into a general building simulation program were not available.

DALITE calculates the illuminance at a specific point in the room by determining the fraction of light coming from the patch of sky viewed by this point through the window. The contributions to the total illuminance by direct beam and interreflected light are then added. The interreflected light is calculated by applying the split flux method [Narasimhan] in which the actual internal space is approximated by a sphere with an average reflection factor representing the reflectances of all of the room surfaces.

1.2 Objective

The objective of this work was to develop a computer program which could accurately estimate the daylight at a prescribed point in a room with minimal computational effort. DALITE was used to provide the ideas for the sky models used and a basic structure of the program. In the new approach developed in this thesis, the interreflected component is calculated in a new way which does not involve the restrictive assumptions of the split flux method. The new method takes into account that there may be beam sunlight reaching one or more of the room

surfaces. The beam radiation on these surfaces is assumed to be diffusely reflected; and the beam images are treated as additional windows.

One of the goals of this study was that the new program could use the TRNSYS program [Solar Energy Laboratory] as a source of input data. TRNSYS is a simulation program which can be used for the investigation of the energy performance of buildings. A subroutine of TRNSYS (Type16) generates hourly radiation data, which are needed for daylighting calculations. Compatibility had to be provided in order to use the new program in conjunction with TRNSYS.

FLITE allows calculations with variable accuracy, where the dependence of the accuracy on the computation time can be investigated.

1.3 Units

Daylight is the visible portion of the radiation emitted by the sun between wavelengths of 0.38 μm and 0.78 μm that reaches the surface of the earth. It is usually expressed in terms of luminance L or illuminance E.

A uniform light source with the luminous intensity of one candela (cd) induces a luminous flux density of one lux (lx) or one lumen per square meter (lm/m^2), called illuminance, on each point of an imaginary sphere with a radius of one meter around this source. Luminous intensity is the same as luminous flux per unit solid angle (steradian) in a given direction. It is measured in candela (cd) or lumen per steradian (lm/sr). Luminance is the luminous flux density per unit solid angle in a given direction, measured in candela per square meter (cd/m^2). For further information see [Kaufman, pp.1-1 - 1-3].

Illuminance is the luminous flux density *reaching* a surface. The luminous flux density *leaving* a surface is referred to as exitance M, where

$$M = \rho * E \qquad (1.1)$$

with ρ being the surface reflectance. Luminance can be converted into exitance by multiplication with π:

$$M = \pi * L \qquad (1.2)$$

Conversion tables from SI units to English units are given in the IES Handbook [Kaufman, pp.1-31 - 1-36].

CHAPTER 2

Sky model

Values of the sky and sun luminance must be obtained from data that are generally available, i.e. total horizontal solar radiation, diffuse horizontal solar radiation, solar azimuth angle and solar altitude angle. The methods described in this chapter are recommended by the Illuminating Engineering Society (IES) and/or taken from DALITE.

2.1 Intermediate sky conditions

To account for the intermediate states between completely overcast and completely clear skies, DALITE [Gillette, p.14] uses the cloud ratio, also called diffuse fraction [Duffie, Beckman, pp.70 - 72], as a measurement of the clearness of the day.

$$CR = I_D / I_T \qquad (2.1)$$

CR = cloud ratio

I_D = diffuse horizontal solar radiation

I_T = total horizontal radiation

DALITE suggests that the actual intermediate sky is a unique function of the cloud ratio. For an almost overcast sky the clear sky component is negligible and for an almost perfectly clear sky the diffuse radiation has virtually no influence on the average luminance.

DALITE uses a clear sky fraction, based on the cloud ratio that calculates an intermediate value that is biased towards the extremes.

$$f = (1+\cos(CR*\pi))/2 \qquad (2.2)$$

f = clear sky fraction

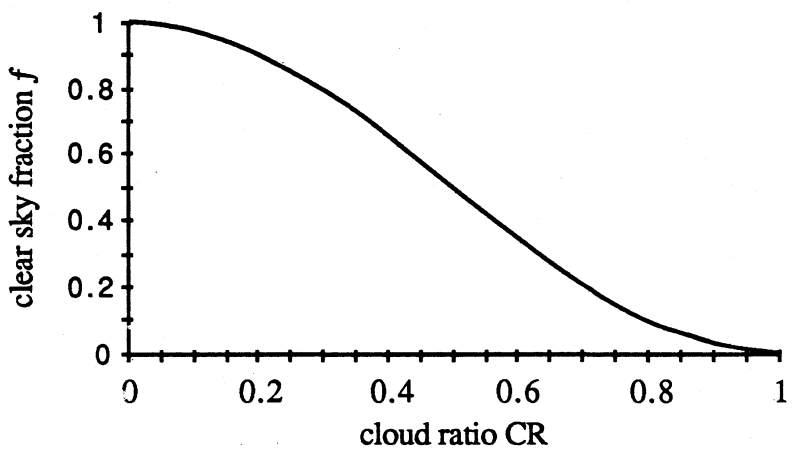


Figure (2.1): Clear sky fraction

The intermediate sky luminance is found from

$$L = f*L_{clr} + (1-f)*L_{ovr} \qquad (2.3)$$

L = intermediate sky luminance

L_{clr} = clear sky luminance at specified point of the sky dome

L_{ovr} = overcast sky luminance at specified point of the sky dome

2.2 Luminous efficacy

Luminous efficacy is defined as the ratio of luminous flux to radiant flux, where luminous flux is measured in lm and radiant flux in W. This quotient can be used to estimate illuminance data directly from radiation data:

$$\Phi = K * S \text{ [lm/W]} \qquad (2.4)$$

Φ = luminous flux

K = luminous efficacy

S = radiant flux

The luminous efficacy varies somewhat for different sky conditions (overcast and clear sky) and surface orientations. For these sky conditions and for a horizontal plane, the values for the diffuse luminous efficacy (which does not take into account direct sunlight) generally range from 105 lm/W to 120 lm/W and for the global luminous efficacy from 100 lm/W to 115 lm/W. Littlefair found that some

researchers measured or calculated values that lie considerably above or below these numbers [Littlefair, p.172]. The values used in DALITE (111 lm/W for diffuse sky luminous efficacy and 93 lm/W for total luminous efficacy) [Gillette,p. 20] represent average values and have been adopted in this program for calculating the total (global) horizontal illuminance on the ground:

$$E_T = (93 + 18\ CR)I_T \ [lux] \qquad (2.5)$$

E_T = total horizontal illuminance

Equation (2.5) requires that the total horizontal radiation I_T be inserted in W/m^2. The cloud ratio CR is used to find the intermediate luminous efficacy value between those for completely diffuse and perfectly clear skies.

2.3 Overcast sky luminance distribution

The IES Handbook recommends the following formula to calculate the nonuniform overcast sky luminance distribution [Kaufman, p.9-83]:

$$L_{ovr} = 3/7 * E_T/\pi * (1 + 2 \sin(\alpha)) \qquad (2.6)$$

α = altitude angle of luminance point on sky dome

The total horizontal illuminance E_T can be determined using equation (2.5).

Equation (2.6) shows that the overcast sky luminance distribution is symmetric about an axis drawn through the center of the sky dome. A point with a luminance value can be specified with the altitude angle α of its location. The normalized overcast sky luminance distribution is plotted in fig.(2.2).

2.4 Clear sky luminance distribution

The nonuniform clear sky luminance distribution was derived by Kittler and is recommended by the IES [Kaufman, p.9-84]:

$$L_{clr} = L_{Z,clr} \frac{(1 - e^{-0.32/\sin(\alpha)})(0.91 + 10e^{-3\psi} + 0.45\cos^2(\psi))}{0.274(0.91 + 10e^{-3\theta} + 0.45\sin^2(h))} \qquad (2.7)$$

$L_{Z,clr}$ = zenith clear sky luminance

ψ = great circle angle between sun and luminance point

h = solar altitude angle

θ = solar zenith angle

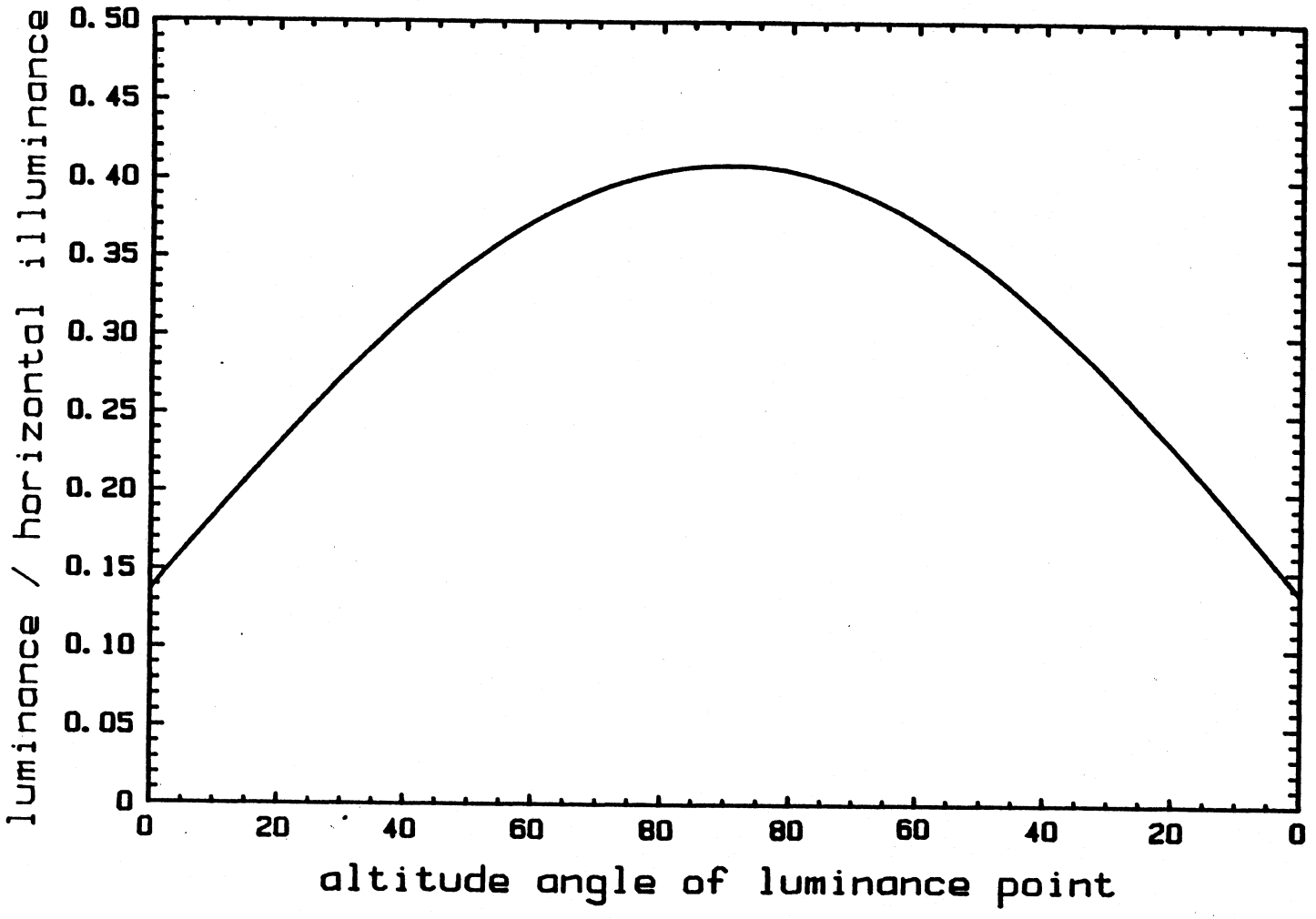


Figure (2.2): Overcast sky luminance distribution

ψ is also called scattering angle and is given by several authors [Karayel et al., p.4; Dogniaux, p.18]:

$$\psi = \arccos[\sin(\theta)\sin(\alpha_Z)\cos(\gamma) + \cos(\theta)\cos(\alpha_Z)] \qquad (2.8)$$

α_Z = zenith angle of luminance point

γ = angle between sun azimuth and azimuth of luminance point

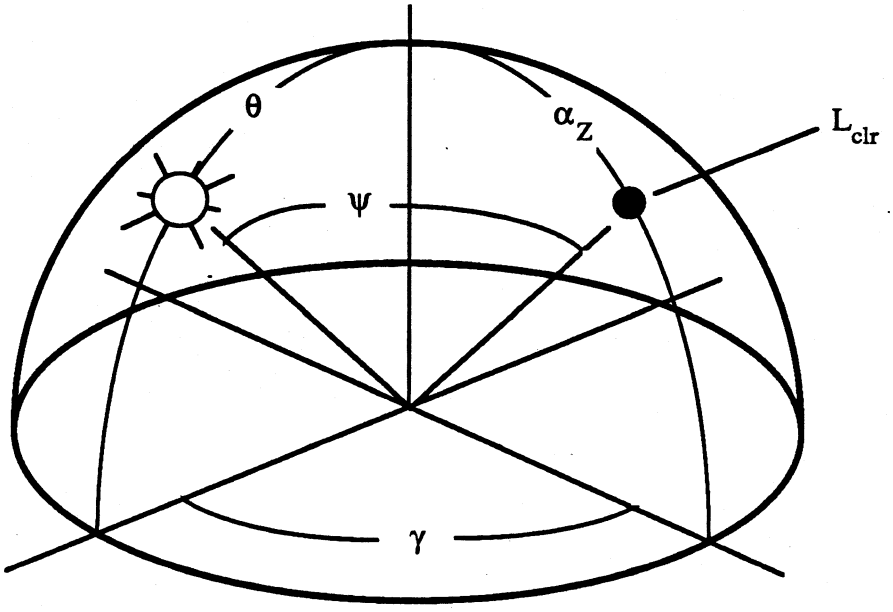


Figure (2.3): Sky dome showing angles in equation (2.8)

The zenith (air mass = 1) clear sky luminance $L_{Z,clr}$ is

$$L_{Z,clr} = a_0 + a_1 h^2 + a_2 h^3 \ [cd/m^2] \qquad (2.9)$$

Dogniaux found that the coefficients a_0, a_1 and a_2 depend on the climate of the considered region, which is characterized by the precipitable water vapor content of the atmosphere (ω) and a turbidity coefficient (β), which expresses the radiation absorbing effect of particles other than water. He developed a table showing the parameters a_0, a_1 and a_2 for the different water vapor contents and turbidity coefficients [Dogniaux, p.22]. The program requires ω and beta as inputs, where beta identifies the turbidity coefficient β. The constants a_0, a_1 and a_2 are stored in the program and automatically found according to beta and ω. These can be determined from table 2.1, which also shows the turbidity coefficients that correspond to the different choices of beta.

location	beta	β	climate	ω [cm of water]
rural	1	0.05	dry	0.5, 1
urban	2	0.1	moderate	2, 3, 4
industrial	3	0.2	humid	5

Table (2.1): Climatic coefficients

Unlike L_{ovr}, the location of a luminance point with a specific clear sky luminance L_{clr} must be defined by two parameters, the altitude *and* the azimuth angle of the point. For a solar altitude angle of 60° and a solar azimuth angle of 0°, the normalized clear sky luminance distribution is plotted across the sky dome in fig.(2.4). The azimuth angles of the luminance point are 0° in the left half (increasing altitude angle of the luminance point) of the plot and 180° in the right half (decreasing altitude angle).

2.5 Direct beam

In addition to the diffuse light from the sky, the case of direct beam must also be considered. DALITE suggests [Gillette, p.20]:

$$E_B = E_{SC}\left(1 + 0.033\cos\left(\frac{360*J}{365}\right)\right) e^{-a/\sin(h)} \qquad (2.10)$$

E_B = direct normal beam illuminance

E_{SC} = solar illuminance constant

J = Julian date (day of the year)

a = atmospheric extinction coefficient

This relationship resembles the equation for the extraterrestrial solar radiation [Duffie, Beckman, p.22]. It substitutes the solar constant by the solar illuminance constant of 127.5 klx, and includes a term which accounts for the attenuation of the atmosphere with an extinction coefficient a=0.21.

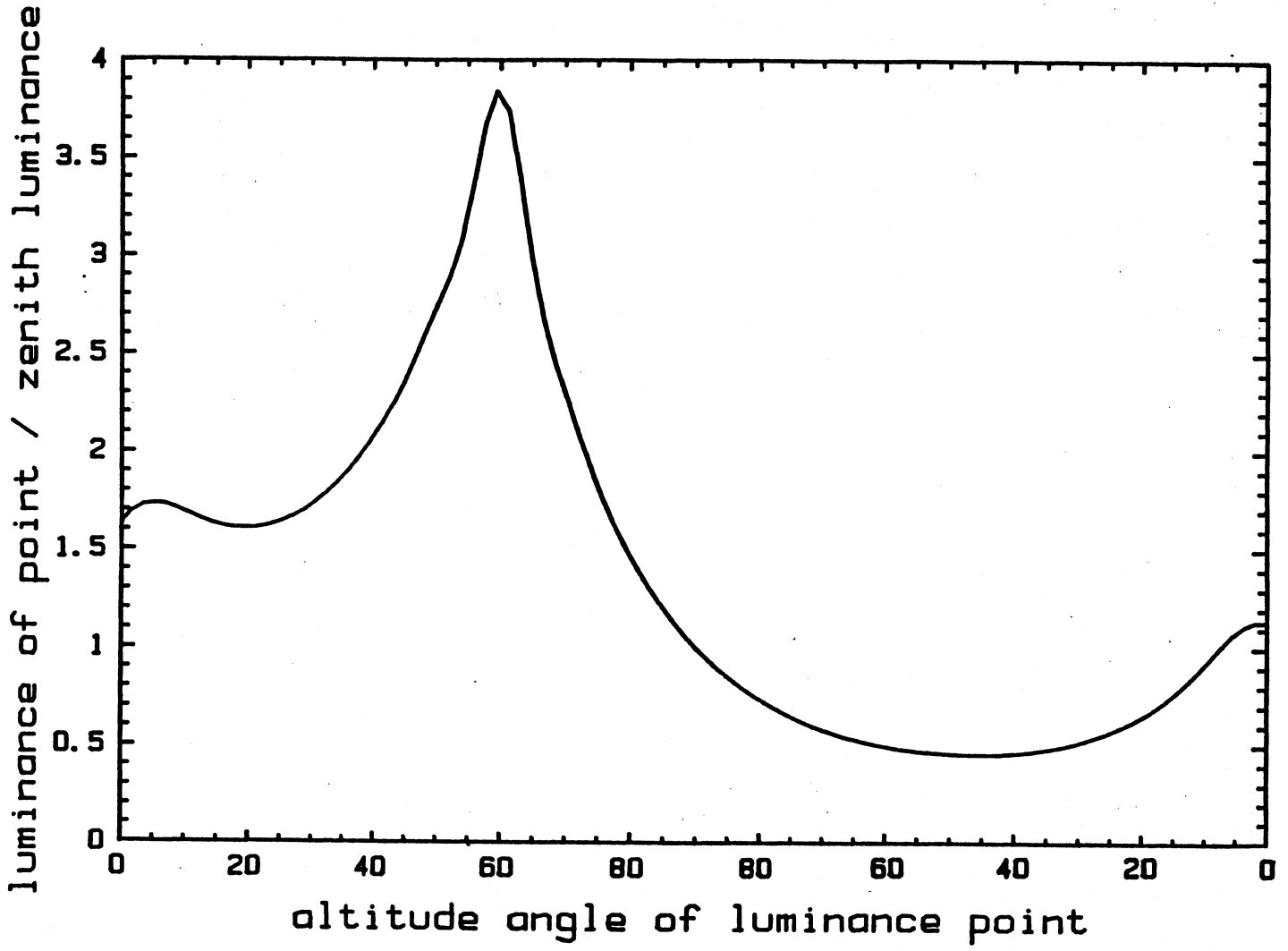


Figure (2.4): Clear sky luminance distribution

2.6 Transmittance

In the diffuse case, a constant transmittance of the window can be used, also called hemispherical transmittance. This value can be found in tables in various sources, e.g. the IES Handbook [Kaufman, p.7-10]. For direct beam, the transmittance is dependent on the incidence angle, which is the angle between the normal to the window plane and the beam itself.

For vertical windows:

$$\sigma = \arccos(h) \qquad (2.11)$$

σ = incidence angle

The transmittance also depends on the number of glazings. The formula used in DALITE [18] that has been adopted is:

$$t_\sigma = 1.018\, t_0 \cos(\sigma)(1 + \sin^3(\sigma)) \qquad (2.12)$$

t_σ = angular dependent transmittance

t_0 = hemispherical transmittance

It is valid for single glazings. Equation (2.12) is plotted in fig.(2.5) for t_0 = 0.85. Duffie and Beckman suggest a different relationship that accounts for the number of glazings and is based on the Fresnel equations [Duffie, Beckman, pp.171 - 174]. The respective statement function in this program (TRANSMIT) can easily

be substituted.

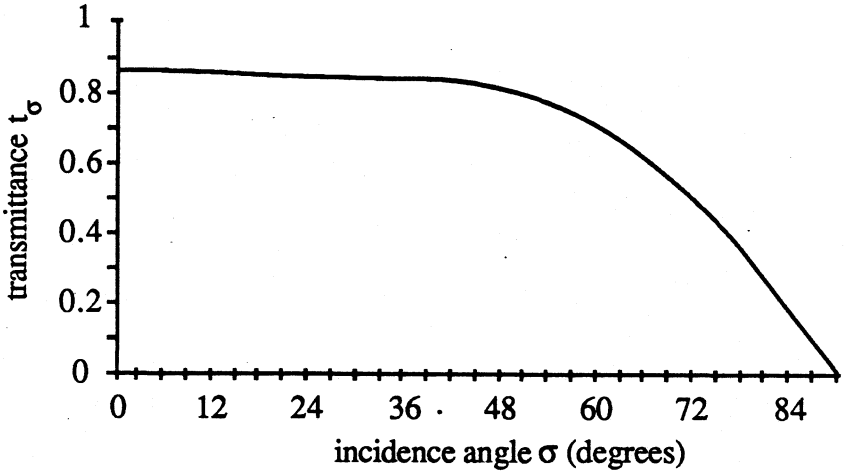


Figure (2.5): Angular dependent transmittance

CHAPTER 3

Interreflected light

In order to find the interreflected component of daylight that enters a room and eventually reaches the specified point, the $\widehat{F}$ method [Beckman] has been applied. This method is generally used for heat flux calculations and has not been chosen for daylighting calculations previously. The $\widehat{F}$ method is exact under the following assumptions:

1) The room is totally empty.

2) The point (sensor) for which the illuminance is calculated, has a transmittance of one. In other words, it interferes in no way with the light fluxes reflecting between the room surfaces.

3) All surfaces are perfectly diffuse, including the window(s). This assumption refers only to the light interreflected within the room and not to beam sunlight, which is treated separately.

4) The surfaces involved in the calculations form an enclosure and are opaque.

3.1 Theory

The fraction of the energy leaving surface i that impinges on surface j directly without any interreflections is the configuration factor F_{ij}, also called view or shape factor. $\widehat{F}_{ij}$ is the fraction of the total energy emitted by surface i that reaches j by all

possible paths. It includes that portion of energy that impinges on surface j after being reflected from all surfaces, including surface i itself, and the direct component, expressed by F_{ij}. The general equation for a problem with n surfaces reads:

$$\hat{F}_{ij} = F_{ij} + (\rho_1 F_{i1})\hat{F}_{1j} + (\rho_2 F_{i2})\hat{F}_{2j} + \ldots \qquad (3.1)$$

F_i = fraction going directly from i to j without being reflected

F_{i1} = fraction initially going from i to 1

ρ_1, ρ_2 = surface reflectances

$\rho_1 F_{i1}$ = fraction of F_{i1} which is reflected from i

$(\rho_1 F_{i1})\hat{F}_{1j}$ = fraction of $\hat{F}_{1j}$ eventually reaching j

For the two infinitely large parallel plates in fig.(3.1), eqn.(3.1) gives:

$$\hat{F}_{11} = F_{11} + (\rho_1 F_{11})\hat{F}_{11} + (\rho_2 F_{12})\hat{F}_{21} \qquad (3.2)$$

$$\hat{F}_{21} = F_{21} + (\rho_1 F_{21})\hat{F}_{11} + (\rho_2 F_{22})\hat{F}_{21} \qquad (3.3)$$

$$\hat{F}_{12} = F_{12} + (\rho_1 F_{11})\hat{F}_{12} + (\rho_2 F_{12})\hat{F}_{22} \qquad (3.4)$$

$$\hat{F}_{22} = F_{22} + (\rho_1 F_{21})\hat{F}_{12} + (\rho_2 F_{22})\hat{F}_{22} \qquad (3.5)$$

Equations (3.2) through (3.5) represent a linear system of four equations for four unknowns. They could be solved by hand, but an increasing number of surfaces involved in the problem makes the use of a computer essential.

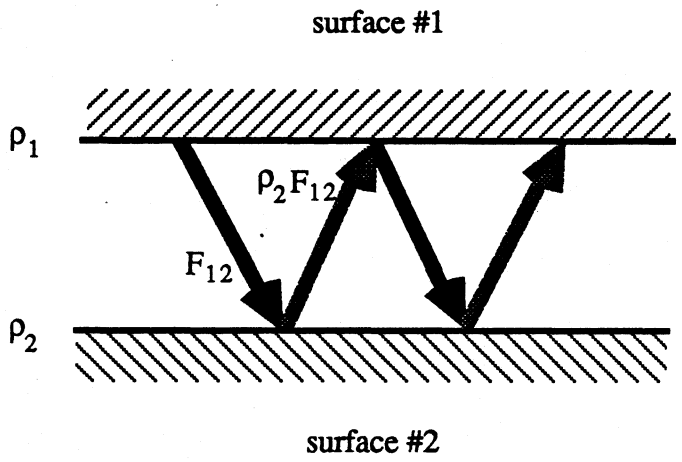


Figure (3.1): Energy flux between parallel plates

Eqn.(3.1) can systematically be written in matrix form, which provides the adaptability to the computer [Klein]. The general matrices read:

$$\underline{\hat{F}} = \begin{vmatrix} \hat{F}_{11} & \hat{F}_{12} & \dots & \hat{F}_{1n} \\ \hat{F}_{21} & \hat{F}_{22} & \dots & \hat{F}_{2n} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \hat{F}_{n1} & \hat{F}_{n2} & \dots & \hat{F}_{nn} \end{vmatrix} \qquad (3.6)$$

$$\underline{F} = \begin{vmatrix} F_{11} & F_{12} & \cdots & F_{1n} \\ F_{21} & F_{22} & \cdots & F_{2n} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ F_{n1} & F_{n2} & \cdots & F_{nn} \end{vmatrix} \tag{3.7}$$

$$\underline{R} = \begin{vmatrix} (1-\rho_1 F_{11}) & -\rho_2 F_{12} & \cdots & -\rho_n F_{1n} \\ -\rho_1 F_{21} & (1-\rho_2 F_{22}) & \cdots & -\rho_n F_{2n} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ -\rho_1 F_{n1} & -\rho_2 F_{n2} & \cdots & (1-\rho_n F_{nn}) \end{vmatrix} \tag{3.8}$$

$\underline{\hat{F}} = \hat{F}$ matrix

$\underline{F}$ = configuration factor matrix

$\underline{R}$ = reflectivity matrix

The matrices represent a linear system of n equations for n unknowns.

The unknown $\hat{F}_{ij}$ can be found by solving the matrix equation:

$$\underline{\hat{F}} = \underline{R}^{-1} \times \underline{F} \qquad (3.9)$$

$\underline{R}^{-1}$ = inverse of reflectivity matrix $\underline{R}$

This program uses the subroutines SGEFA and SGESL from the library LINPACK [Dongarra et al.] to solve equation (3.9).

The $\hat{F}$ method assumes that the surfaces involved form an enclosure and that there is no energy leaving the system by transmittance. All energy is either absorbed or reflected:

$$\rho_i + \alpha_i = 1 \qquad (3.10)$$

ρ_i = surface reflectances

α_i = surface absorptances

The $\hat{F}_{ij}$ are only zero in the cases of zero reflectances of the surfaces They can be greater than one. That happens if the reflectances are high with low absorptances. Light may be reflected from other surfaces several times onto the same surface without having lost much energy. This does not mean that the gain is larger than the energy input, because small absorptances coincide with high reflectances (see eqn.(3.10)).

With the knowledge of the $\hat{F}_{ij}$ the energy leaving surface i that reaches surface j

by all possible paths can be determined. Here the energy is light and the surfaces involved in the $\hat{F}$ method are the walls, the ceiling, the floor and the windows of the room. Hence, the illuminance of each room surface due to interreflection can be calculated. The illuminance at the point of interest can be derived from this information with configuration factors between the point and the portion of the surfaces viewed.

3.2 Configuration factors

The $\hat{F}$ method requires the calculation of the configuration factors between all surfaces involved in the model. The following must be considered:

1) In order to obtain the luminance distribution on the room surfaces, it is necessary to break them up into elements. If an entire room surface is used rather than smaller elements, only a single luminance value for the surface can be calculated, which is the equivalent to treating the surface as a uniformly lit light source.

2) Each window must be treated as an additional surface in the model, because the energy input must be assigned to a surface.

3) There are no simple equations available that express the configuration factor between two areas in terms of a variable location of these surfaces with respect to each other. The area-area configuration factor equations are solved for specific cases, for instance for two surfaces with a common edge, or for two surfaces that are directly opposite of each other [Hamilton, Morgan]. These equations apply only, if a luminance distribution, as stated in 1), is not desired.

Pierpoint and Hopkins developed a point-area configuration factor [Pierpoint]

that provides the desired flexibility. The point can be located arbitrarily with respect to a surface, which may be tilted towards the plane on which the point is located at an angle ϕ:

$$F_{ij} = \frac{1}{2\pi}\left(\frac{x}{\sqrt{x^2+y^y}} \arctan\left[\frac{r\sqrt{x^2+z^2}}{x^2+y^2+z^2-yr}\right] \right.$$

$$+ \frac{t\cos(\phi)-x}{\sqrt{x^2+z^2+t^2+2tg}} \arctan\left[\frac{r\sqrt{x^2+z^2+t^2+2tg}}{x^2+y^2+z^2-yr+t^2+2tg}\right]$$

$$+ \frac{y\cos(\phi)}{\sqrt{y^2+h^2}} \arctan\left[\frac{t\sqrt{y^2+h^2}}{y^2+h^2+g^2+tg}\right]$$

$$\left. - \frac{(y-r)\cos(\phi)}{\sqrt{(y-r)^2+h^2}} \arctan\left[\frac{t\sqrt{(y-r)^2+h^2}}{(y-r)^2+h^2+g^2+tg}\right] \right) \qquad (3.11)$$

F_{ij} = point-area configuration factor

$g = z\sin(\phi) - x\cos(\phi)$

$h = z\cos(\phi) + x\sin(\phi)$

The nomenclature of eqn.(3.11) is illustrated in figure (3.2).

Pierpoint and Hopkins called their expression an area-source equation, which may be misleading, because the actual configuration factor is from point to area.

The term area-source refers to the energy source and not to direction of the configuration factor.

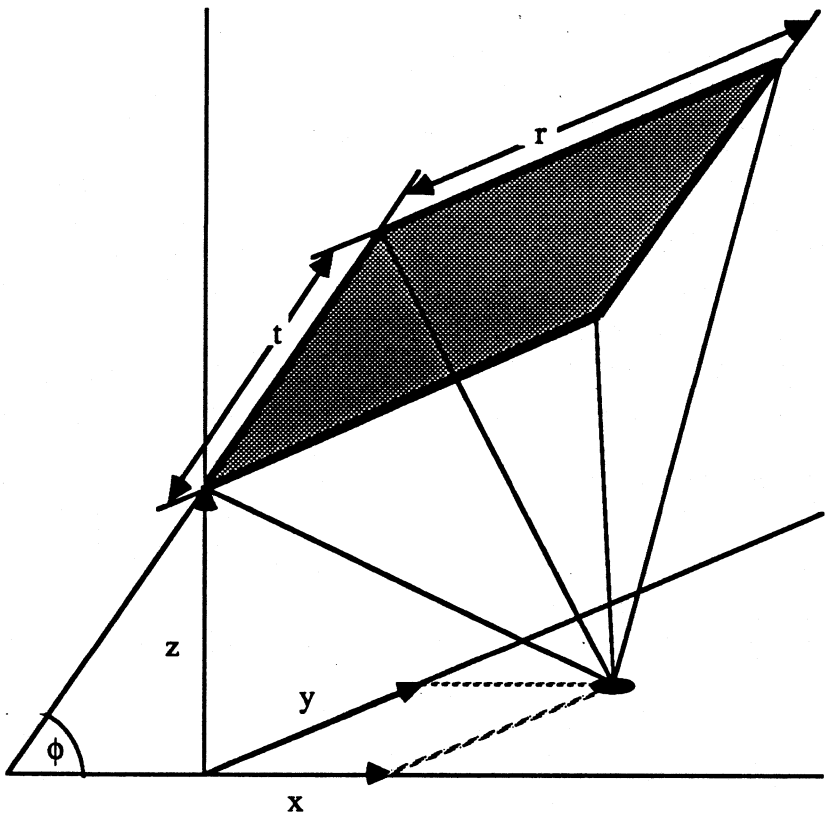


Figure (3.2): Nomenclature in equation (3.11)

Equation (3.11) has been derived with the reciprocity theorem:

$$A_i F_{ij} = A_j F_{ji} \qquad (3.12)$$

A_i, A_j = areas of energy sources

For point-area configuration factors, the reciprocity theorem reads:

$$d(A_i)F_{ij} = A_j d(F_{ji}) \tag{3.13}$$

$d(A_i)$ = infinitesimal area of the point

$d(F_{ji})$ = infinitesimal configuration factor between source and point

The configuration factor from area-source to point F_{ji} is infinitesimally small, because the energy impinging on a point of infinitesimal area approaches zero.

For the total illuminance reaching a point from an area source:

$$E_i d(A_i) = M_j A_j d(F_{ji}) \tag{3.14}$$

E_i = illuminance of the point

M_j = exitance of the area-source

A_j = area of the source

Reciprocity can be applied and yields with equation (3.13):

$$E_i = M_j F_{ij} \tag{3.15}$$

The infinitesimal terms have been eliminated and the illuminance reaching the point can be expressed with the finite configuration factor between point and area-source F_{ij}.

Using point-area configuration factors rather than area-area configuration factors requires that a numerical integration be performed. This can be done by dividing the room up into small elements. Eventually, as the number of elements increases, the point-area configuration factors will approach area-area configuration factors.

3.3 Room model

The surfaces of a rectangular room with variable geometry had to be divided into a sufficient number of elements. Since the sufficient number had to be determined, a variable input n, defining the number of elements, was used. In addition to n, the variables are the room geometry in terms of height, width, and length, and the inputs of the configuration factor function (eqn.(3.11)):

x,y and z for the coordinates of the point with respect to the area-source;

r and t for the dimensions of the area-source; and

ϕ for the tilt angle of the area,

where ϕ is 0° for elements on opposite room surfaces and 90° for elements on surfaces with a common edge. See fig.(3.2) for illustration.

Figure (3.3) shows how the elements divide up the room surfaces, viewed from inside the room and facing surface #1. The convention of numbering the room surfaces has been adopted from DALITE. The walls are surface #1 to #4, the ceiling is #5 and the floor is #6. The number of elements along one edge of each surface is n, so the total number of elements is 6*n*n. Facing the respective surface from inside the room, the count starts at (sn-1)*n*n+1, with sn being the surface number, and the largest element number is sn*n*n in the lower right corner.

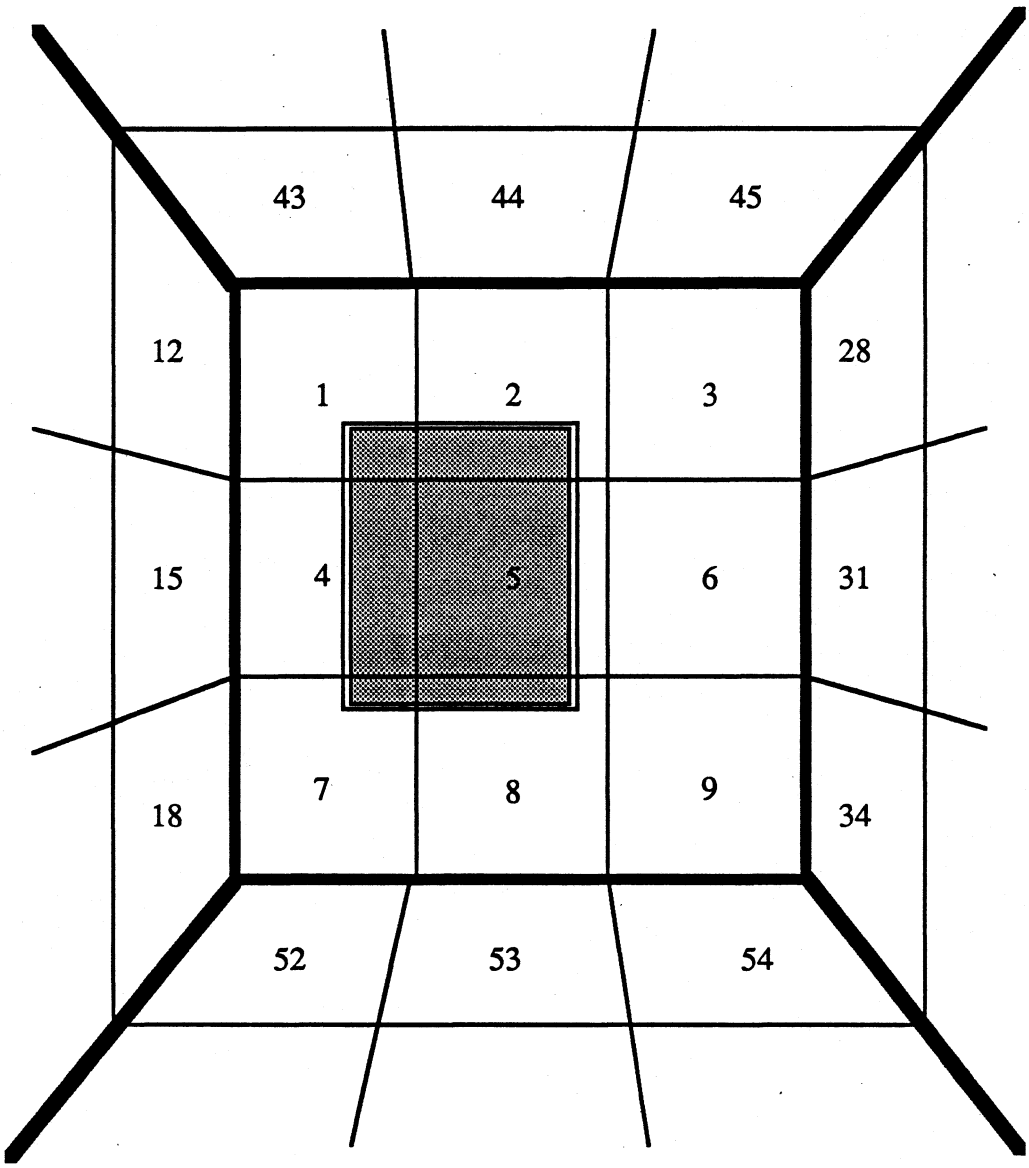


Figure (3.3): Numbering convention of surface elements for n = 3

This convention is also applied to the ceiling, by facing surface #1 as in fig.(3.3) and looking upwards. The floor element numbers are found from adding n*n to the element number directly opposite on the ceiling.

The algorithm performed in subroutine CONFIG calculates the configuration factors according to equation (3.11) between all points in the center of each element and all elements, thus generating $\underline{F}$ with $(6*n*n)^2$ elements.

The reciprocity theorem has been applied to determine how many elements must be considered to give reasonable accuracy in approximating area-area with point-area configuration factors. For point area configuration factors the reciprocity theorem is given in eqn.(3.13).

In order to perform the step from finite elements to differential elements a numerical integration must be performed. Then, by increasing the number of elements, eqn.(3.13) approaches eqn.(3.12). This is shown in figure (3.4) for two wall surfaces with a common edge. The walls have different dimensions, because otherwise the reciprocity theorem gives no new information, since the configuration factors F_{ij} and F_{ji} are the same. The following has been plotted:

$$\sum_{j=1}^{n^2}\sum_{i=1}^{n^2}(A_i F_{ij}) \text{ and } \sum_{i=1}^{n^2}\sum_{j=1}^{n^2}(A_j F_{ji}) \text{ against } n,$$

for elements i on one and j on the other surface.

From fig.(3.4), n=5 has been chosen as an approximation, thus the total number of elements in each $\underline{F}$, $\underline{\hat{F}}$ and $\underline{R}$ is $(6*5^2) = 22500$.

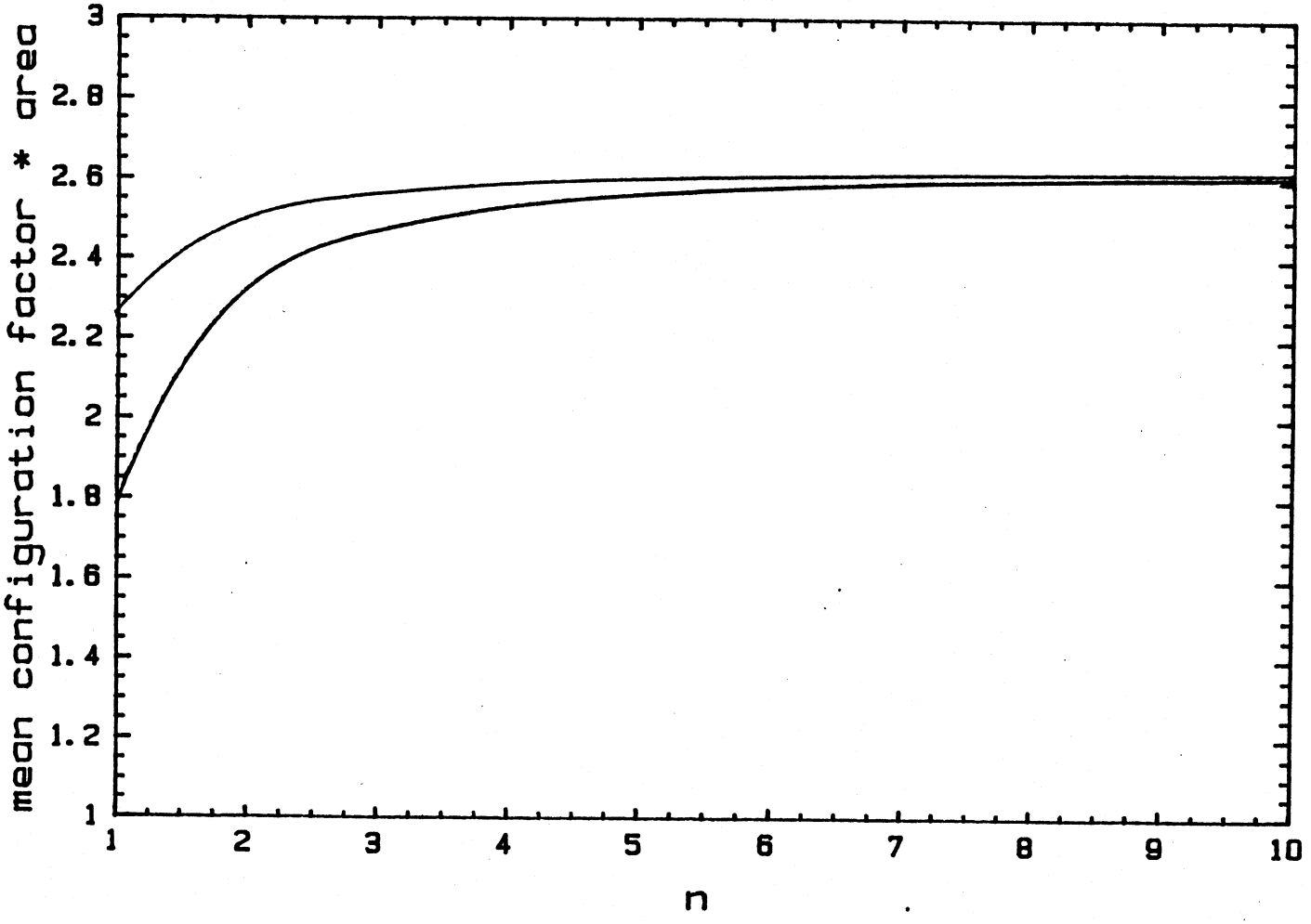


Figure (3.4): Numerical integration of point-area configuration factors

The time and memory space needed to solve eqn.(3.9) with LINPACK on a MicroVAX II is shown in figure (3.5). The user of this program may change n, adapting it to needs, available hardware, and CPU time. Along with n, the dimensions of the arrays in the program must also be adapted by setting the parameter lda = 6*n*n.

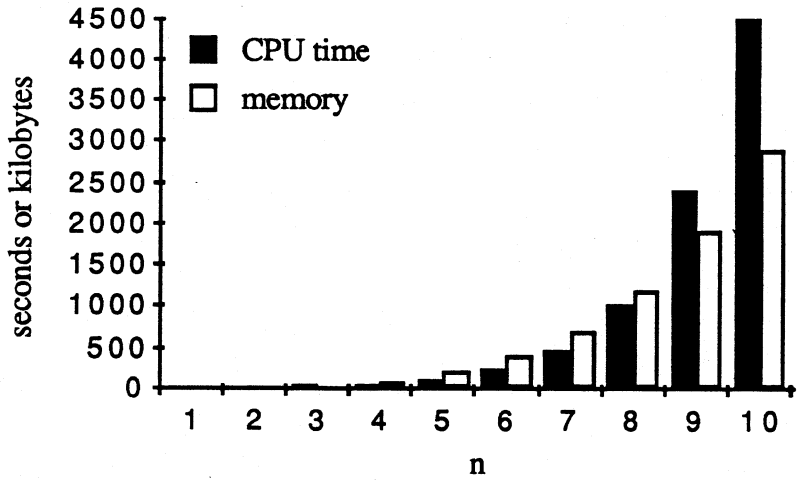


Figure (3.5): CPU time and memory space needed

As can be seen from fig.(3.3), the area of a window in a wall does not necessarily match the area of the elements that lie in the window area. Thus, if the luminance of the window plane is the input of the interreflection routine and the window area is larger than the area of the elements representing it, the illuminance at the sensor position will be too small. On the other hand, if the window is smaller than the area of the respective elements, the illuminance will be too high.

Rather than choosing a large number of elements to make the error small, the input luminance has been area-weighted:

$$\bar{L}_w = \bar{L} * \frac{A_w}{\sum_i A_{w,i}} \qquad (3.16)$$

$\bar{L}_w$ = area-weighted window luminance

$\bar{L}$ = luminance inside of window plane

A_w = area of window

$A_{w,i}$ = area of one element that lies in window area

The effect of area-weighting the input luminance on the total daylight at a specific point inside the room is shown in fig.(3.6). The normalized illuminance is plotted against n. If the window is larger than the area of the respective surface elements that represent the window, the result is too small. On the other hand, if the window is smaller the illuminance is higher, which is illustrated by the wavy form of the non-weighted curve. For n = 5, the window area matches the area of the respective elements exactly for this geometry. Thus, the curves intercept at this point. The area-weighted curve is already close to the correct value for small n, i.e. 2 - 3. Therefore, it is not necessary to use large values of n to represent the window with surface elements.

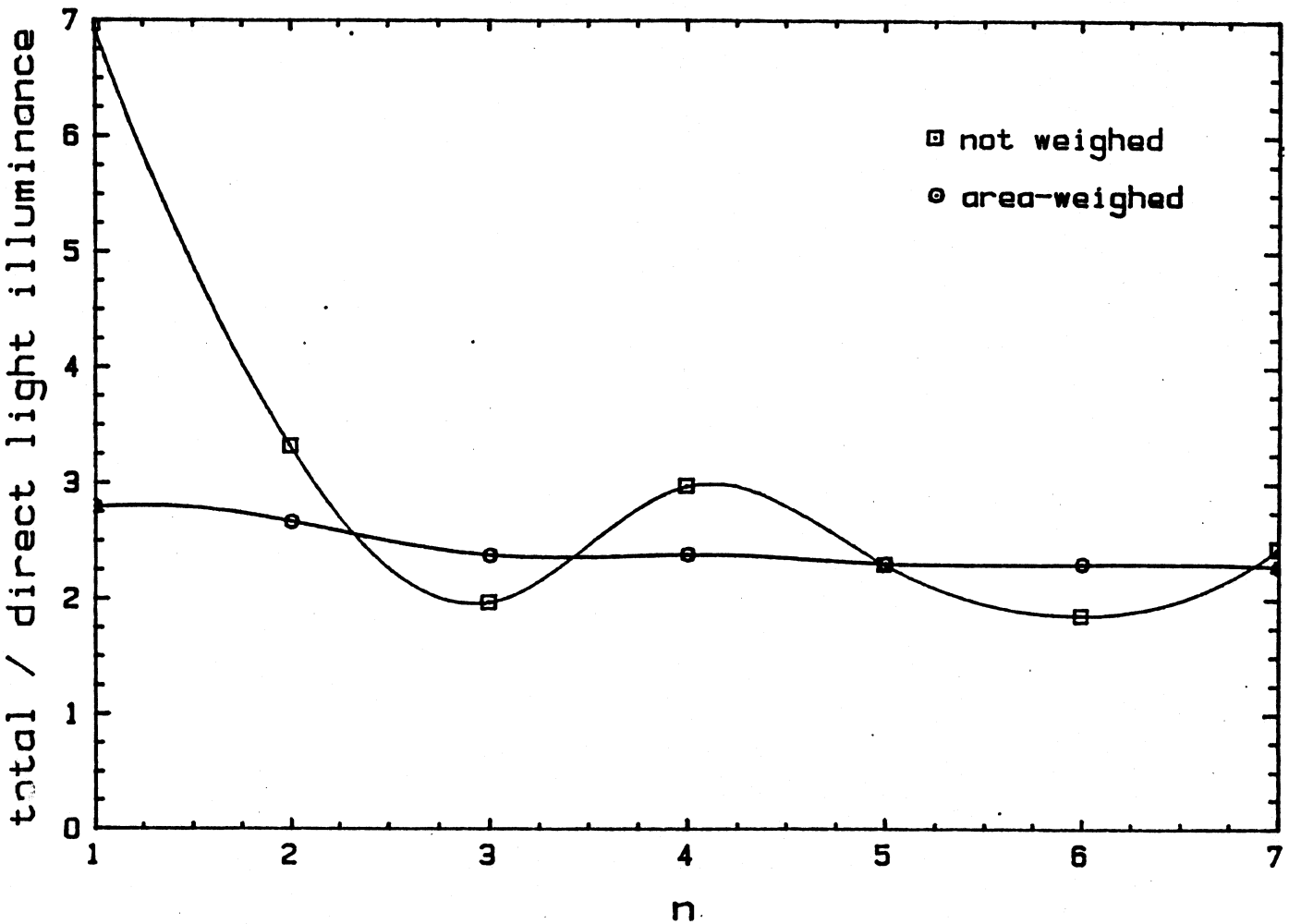


Figure (3.6): Area weighted sky luminance

3.4 Checks

Several checks may be employed to test the accuracy of the calculations:

1) The reciprocity (eqn.(3.12)) must hold, which has been shown in fig.(3.5) for the F matrix. Reciprocity must hold for the $\hat{F}$ as well:

$$A_i\hat{F}_{ij} = A_j\hat{F}_{ji} \tag{3.17}$$

2) The sum of the configuration factors from one point to all elements of the enclosure must be one:

$$\sum_{j=1}^{6*n^2} F_{ij} = 1 \tag{3.18}$$

For an enclosure, all energy leaving one point reaches all surfaces it views. In $\underline{F}$, the sum of each row must be one. This check is built into the program and an error message will be displayed, if eqn.(3.18) is not satisfied.

3) Eventually, all energy leaving a point must be absorbed, so that

$$\sum_{j=1}^{6n^2}(\hat{F}_{ij}\,\alpha_j) = 1 \tag{3.19}$$

For the $\hat{F}$ method, the surfaces are opaque (see eqn.(3.10)). This check is made by the program and an error message will be displayed, if eqn.(3.19) is not satisfied.

4) For blackened surfaces (ρ_i, ρ_j = 0), the interreflected component becomes zero and $\underline{\hat{F}}$ becomes $\underline{F}$.

5) For mirrored surfaces (ρ_i, ρ_j = 1) and a sensor position on the floor, the interreflected component plus the direct diffuse component from the light source(s) equals the sum of the $\hat{F}_{ip}$ from the source(s) to the sensor directly. The $\hat{F}_{ip}$ already include the direct diffuse component in this special case. This check will be clear after reading the next chapter, in which the equations for the interreflected component at the sensor position are derived.

6) The values of F and $\hat{F}$ must obey symmetry considerations. For the rectangular room in fig.(3.3) this symmetry requires that the configuration factors between the element pairs #12/#4 and #28/#6 or for #44/#2 and #53/#8 be identical.

CHAPTER 4

Components

Daylight at a point in a room consists of direct beam, direct diffuse and interreflected components. A daylighting program must determine how these components contribute to the total illuminance at the point of interest. Some details concerning the daylight at the window surface have not been accounted for in this study. Examples are trees obstructing the fenestration, reflection from other buildings, and light reflected from overhangs and wingwalls. The contributing factors that have been considered in this program are described in this chapter.

4.1 Window luminance

The interreflected light algorithm and the direct diffuse component from the window both require the luminance of the window plane as input. DALITE [1] calculates the window luminance by determining the portion of the sky dome viewed by the point of interest through the window. The interreflection routine must also take into account the portion of the sky dome that is viewed by the surface elements. There are portions of the sky viewed by the elements, i.e. other points in the room, that are obstructed for the point of interest. Thus, the luminance as seen by the window itself is calculated in order to cover the range seen by all elements and the specified point. The view of the window plane may be restricted by an overhang

and wingwalls and a reduced view is taken into account.

To account for the nonuniformity of the sky luminance distribution models, more than a single sky luminance value is calculated. For the point in the center of the window, 25 sky luminance values are determined, with these 25 locations being evenly distributed over the patch of sky viewed. The sky luminance at a specific point is calculated with eqn.(2.3) and the average luminance inside the window plane becomes

$$\bar{L}_d = \frac{\sum_{k=1}^{25} L_k}{25} t_0 \qquad (4.1)$$

$\bar{L}_d$ = average luminance inside of window plane due to direct diffuse light from sky dome

L_k = sky luminance at point k

t_0 = hemispherical transmittance

Another component contributing to the light entering a room is the light reflected from the ground. A simple method to obtain a value for the ground illuminance is the luminous efficacy technique.

Assuming the ground is an infinitely extending plane, the area-area configuration factor between the ground and a vertical window is 0.5. This does not take into account that there may be buildings obstructing the view through the window. The following expression is recommended by the IES [Kaufman, p.9-91]:

$$L_{gr} = 0.5/\pi \, E_T \, \rho_{gr} t_0 \qquad (4.2)$$

E_T = given in eqn.(2.5)

L_{gr} = luminance inside the window plane due to ground reflected light

ρ_{gr} = ground reflectance

The factor π converts the ground exitance, found with eqn.(2.5) multiplied by the ground reflectance, into the ground luminance and results from integration over the hemisphere (see eqn.(1.2)).

This adds to the average window luminance due to direct diffuse light from the sky:

$$\overline{L} = \overline{L}_d + L_{gr} \qquad (4.3)$$

$\overline{L}$ = total luminance inside the window plane

At this point the light reflected from overhangs and wingwalls onto the window plane may be added. If an algorithm for these calculations should be developed, the following must be considered:

1) There may be beam sunlight reaching the wingwalls for some hours of the day.

2) The accuracy of the model depends on the number of interreflected fluxes used. It may be sufficient not to account for light reflected from overhang on the wingwalls and between the wingwalls.

4.2 Interreflected component

With the knowledge of the $\hat{F}$ matrix as derived in the previous chapter, the luminance of each surface element due to the luminous flux leaving a light source, reaching the element by all possible paths, can be derived.

For the daylight reaching element i:

$$A_i E_i = \sum_s (A_s \hat{F}_{si})\, M_l \qquad (4.4)$$

A_i = area of surface element

E_i = illuminance of surface element i

A_s = area of source element

$\hat{F}_{si}$ = fraction of M_l leaving the source that reaches the surface element i by all possible paths

M_l = uniform exitance of the light source

Eqn.(4.4) describes the summation over all elements that represent a light source. M_l is the same for all elements of one light source, thus subscribed with l and not with s.

The area terms can be eliminated with the reciprocity theorem (eqn.(3.17)), and the new expression reads:

$$E_i = \sum_s \hat{F}_{is} M_1 \qquad (4.5)$$

$\hat{F}_{is} = \hat{F}$ from surface element to source element

In order to find the luminance of the point of interest, the configuration factors between the point and all surface elements viewed must be determined, which are all elements above the workplane, including the ceiling elements.

For the energy reaching the point of interest:

$$d(A_p)E_p = \sum_i (A_i\, d(F_{ip})\, M_i) \qquad (4.6)$$

$d(A_p)$ = infinitesimal area of the point

E_p = illuminance of the point

$d(F_{ip})$ = infinitesimal configuration factor between element i and point p

M_i = exitance of surface element i

where

$$M_i = \rho_i E_i \qquad (4.7)$$

ρ_i = reflectance of surface element i

from equation (1.1).

Again by applying eqn.(3.12), the reciprocity theorem, the infinitesimal terms can be eliminated:

$$E_p = \sum_i (F_{pi} \rho_i E_i) \tag{4.8}$$

F_{pi} = configuration factor between point of interest and surface element

Combining equations (4.5) and (4.8) gives the illuminance of the point of interest due to the source luminance by interreflection:

$$E_{p,i} = \sum_i (F_{pi} \rho_i \sum_s \hat{F}_{is}) M_l \tag{4.9}$$

$E_{p,i}$ = illuminance of point of interest due to one light source by interreflection

The illuminance of the point of interest found with eqn.(4.9) does *not* include the direct diffuse component from the luminance source. The illuminance of each surface element, including those representing the source, was found with the $\hat{F}$ method. This means that the illuminance of an element representing a light source is the luminous flux reaching it from the source itself by all possible paths, including the direct component. However, the direct component from source element to source element is zero, because the sources in this problem are plane and the configuration factors are zero between source elements. The illuminance of a source element, determined with the $\hat{F}$ method, is the light reflected back from all other elements only. Thus, the illuminance at the point of interest due to the exitance of

the source elements, as in eqn.(4.9), does not include the direct diffuse component from source to point and must still be added.

For certain configurations, there may be patches of beam sunlight on the walls and/or on the floor of the room. It is assumed here that the beam reflected from the walls or floor is perfectly diffuse. After determining the elements that represent the patch, it is treated as an additional window and the illuminance due to interreflection from the patch at the point of interest is found from eqn.(4.9). Again, this does *not* include the direct diffuse component from the source to the point. For a beam patch on the floor, the latter is zero, but for patches on the walls, that part of the patch contributes a direct diffuse component that is viewed by the point of interest, which is the part that lies above the workplane.

4.3 Direct diffuse component

In addition to the interreflected component also the light reaching the point of interest directly without being interreflected must also be accounted for. With the knowledge of the configuration factors between the point of interest and the elements that represent a light source, eqn.(4.8) applies, where $\rho_i E_i$ is substituted by the uniform source exitance M_l:

$$E_{p,d} = \sum_i F_{pi} M_l \qquad (4.10)$$

$E_{p,d}$ = illuminance at point of interest due to direct diffuse light

4.4 Direct beam component

The internal illuminance at the specified point due to beam sunlight is found from eqn.(2.10) and (2.12):

$$E_{B,i} = E_B * t_\sigma * \sin(h) \qquad (4.11)$$

$E_{B,i}$ = beam illuminance at point of interest

t_σ = found from eqn.(2.12)

h = solar altitude angle

where sin(h) expresses the normal component of the illuminance reaching the workplane.

In the case of beam patches that may occur on the walls of the room, eqn.(4.11) must be modified and sin(h) is exchanged by the a term that accounts for the component of the beam sunlight that is normal to the respective surface. The new term is dependent on the room azimuth angle, the surface that the beam patch is on, solar azimuth and altitude angles, and the surface that includes the window. As an example, for the room in fig.(4.1) with beam sunlight on surface #2, eqn.(4.11) becomes

$$E_{B,i} = E_B * t_\sigma * \cos(h) * \sin(\omega_s - \gamma_r) \qquad (4.12)$$

ω_s = solar azimuth angle

γ_r = room azimuth angle

The term that must substitute sin(h) in eqn.(4.11) is calculated by the program for windows in all surfaces and beam patches on all wall surfaces.

For the floor, eqn.(4.11) is directly applicable. The exitance of the beam patches is found with eqn.(1.1) and for the illuminance due to the beam patches the uniform source exitance M_l becomes:

$$M_l = \rho\, E_{B,i} \qquad (4.13)$$

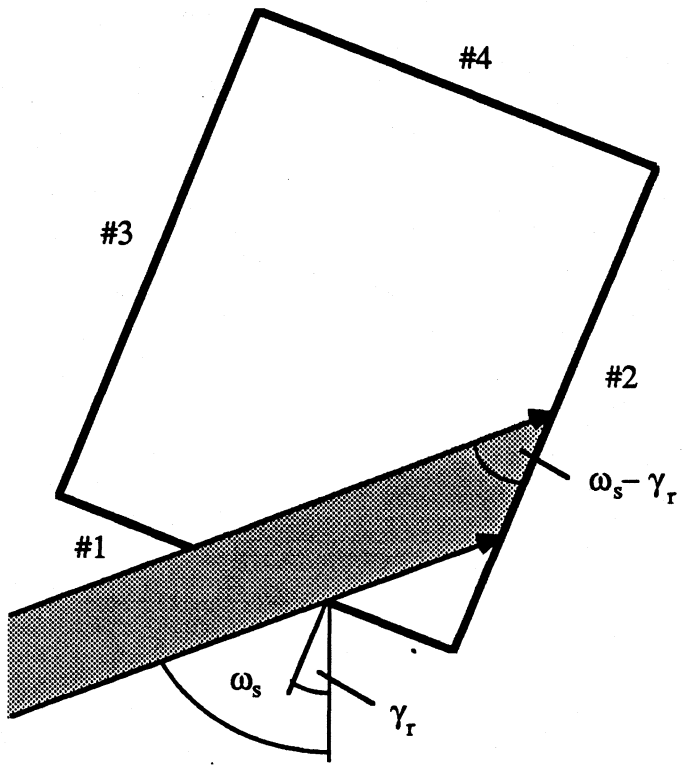


Figure (4.1): Beam patch on surface #2

4.5 Total daylight

For each light source, the direct diffuse and the direct beam components must be added to the interreflected component, where light sources for the direct diffuse component are windows and that portion of beam patches on the walls that lie above the workplane. The final equation for the total daylight at the specified point is obtained by adding eqn.(4.9), (4.10) and (4.11):

$$E_{p,j} = \left[\sum_i (F_{pi}\rho_i \sum_s \hat{F}_{is}) + \sum_s F_{ps}\right] M_l + E_{B,i} \qquad (4.14)$$

$E_{p,j}$ = total illuminance due to one light source

For the exitance of a window , with the area weighted luminance inside the window plane from eqn.(3.16), M_l becomes:

$$M_l = \pi \bar{L}_w \qquad (4.15)$$

Equation (4.11) must be applied for each light source. The daylight at the point of interest due to all light sources in the room is found from:

$$E_{p,t} = \sum_j E_{p,j} \qquad (4.16)$$

$E_{p,t}$ = total illuminance at specific point

CHAPTER 5

Program description

The computer program developed (FLITE) is written in FORTRAN 77. Chapter 5.1 demonstrates how the problem of calculating the daylight inside a room is split up into subproblems. Naturally, correct results depend on a thorough understanding of the input variables. The interpretation of the input and output parameters is given in this chapter. Finally, the limitations of FLITE are discussed.

5.1 Data flow

The subproblems that must be solved with this program are shown in fig.(5.1). The shaded rectangles show those subproblems that deal with the $\widehat{F}$ method. These parts must only be solved initially for a room geometry. The program is set up to process hourly radiation data and jumps over the time-consuming $\widehat{F}$ calculations after the first time step. Some of the subproblems had to be divided up into even smaller parts that are not shown in the chart. For instance, in order to calculate the numbers of the elements that lie in a beam patch area, it is necessary to determine the dimensions of the patch, which involved the development of a separate algorithm.

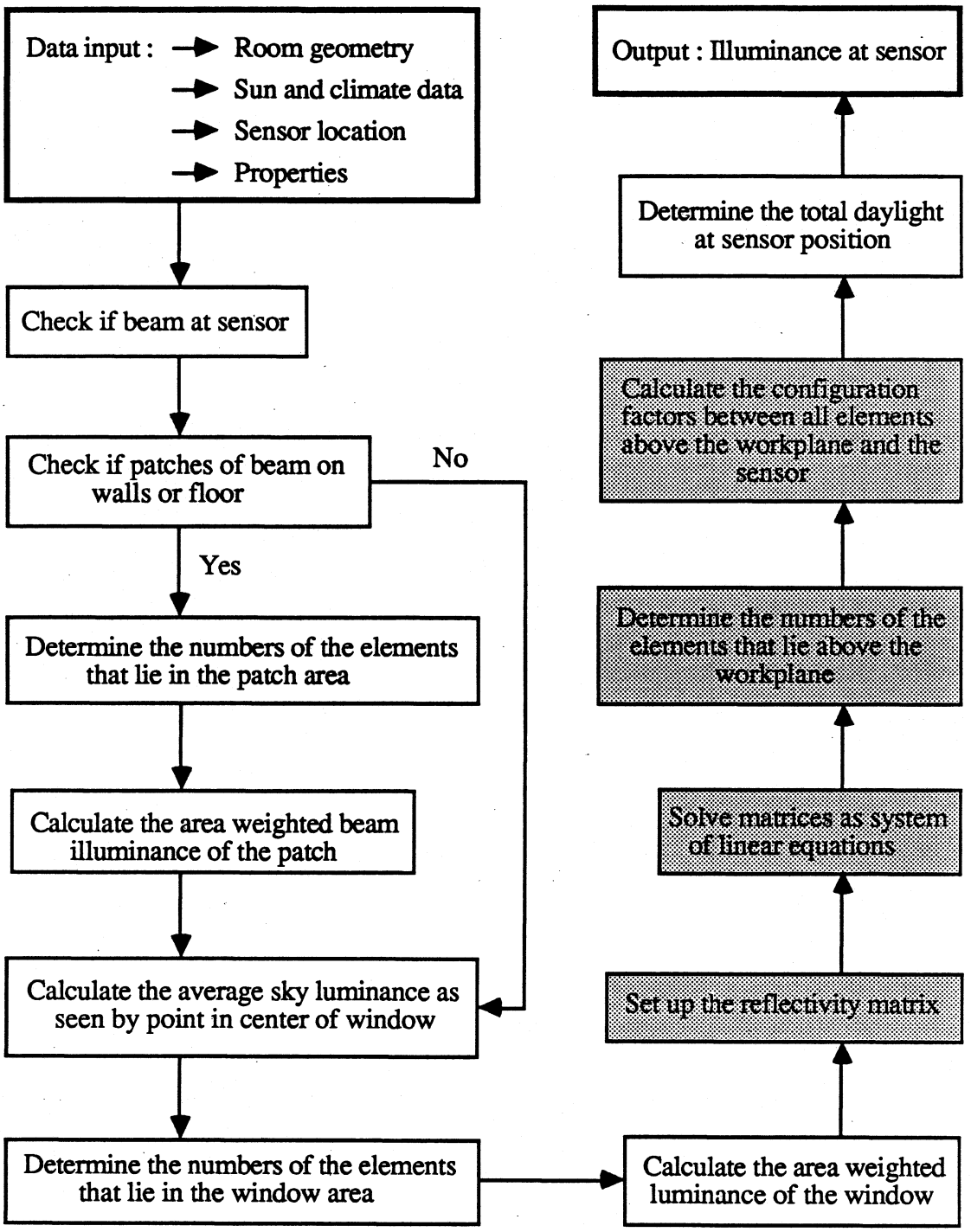


Figure (5.1): Flowchart of FLITE

5.2 Room geometry

A rectangular room is defined by its height, length and width. The room position is defined by the room azimuth angle. It is zero due south with east being negative and west positive, following the sign convention of the solar azimuth angle (Duffie, Beckman, p.10). Surface numbers are used for two reasons:

1) defining which is the length and which is the width of the room, and

2) defining in which walls the windows are located.

The wall facing south is always surface #1. Then, the room may be turned by an azimuth angle, to obtain an angular displacement from south. This angle must be between 90° and -90°. An angle larger than 90° and smaller than -90° is not needed, since the surface numbers would be redefined. If, for instance, a room is turned by 100°, this is the same as assigning #1 to the wall that was facing south after 90° of the turn were completed, and then turning it by another 10°.

The width of the room is measured along surfaces #1 and #3 and the length along surfaces #2 and #4. The sensor position inside the room is defined by the elevation above the floor or workplane height and the length and width distances from the room reference point. The room reference point is in the right lower corner of surface #1, viewed from inside the room. The width and length distances of the sensor from the reference point are consistent with the definition of length and width of the room walls.

The location of a window is defined by assigning a surface number, which is surface #1 - #4. Viewed from inside the room, the distance from the left edge of the window to the left edge of the respective wall is required as input along with the distance between the bottom of the window and the floor. The window

dimensions must be defined as width and height.

An overhang may be assigned to the window by giving it a length. The overhang is perpendicular extending from the top of the respective wall and the length is measured away from the wall. An infinite width of the overhang is assumed. Wingwalls, also called sidefins, may be placed on either side of the window, where the distance from the left or right window edge, respectively, do not have to be the same. Their lengths, perpendicular to the window, may be different as well.

A mullion correction factor is used to account for mullions, which may obstruct the window and reduce its effective size, as defined with the dimensions above. The mullion correction factor is:

$$cm = 1 - \frac{A_m}{A_w} \qquad (5.1)$$

cm = mullion correction factor

A_m = area of the mullions

A_w = gross area of the window

The luminance of the window plane from eqn.(4.3) is multiplied by the mullion correction factor to account for these obstructions by reducing the input luminance.

5.3 Input and output

FLITE requires a file with the name 'FLITE.in' as one input file:

rw	rl	rh	raz							
sw	sl	wkpln	nw							
reflw	reflc	reflf	reflg	t0						
beta	omega	reflgr								
wl	wb	wh	ww	ov	finl	dfl	finr	dfr	cm	sn

Table (5.1): File 'FLITE.in'

The abbreviations in table (5.1) and fig.(5.2) are the same as those in the source code of the program, where

rw = room width [m]

rl = room length [m]

rh = room height [m]

raz = room azimuth angle

sw = width distance of sensor from room reference point [m]

sl = length distance of sensor from room reference point [m]

wkpln = workplane height [m]

nw = number of windows, the maximum number is 5

reflw = reflectance of the walls [0...1]

reflc = reflectance of the ceiling [0...1]

reflf = reflectance of the floor [0...1]

reflg = reflectance of the windows [0...1]

t0 = transmittance of the windows [0...1]

beta = climate indicator, found from table (2.1)

omega = water vapor content of the atmosphere [cm of water], from table (2.1)

reflgr = ground reflectance [0...1]

wl = distance from left edge of window to left edge of respective wall [m]

wb = distance from bottom of window to floor [m]

wh = window height [m]

ww = window width [m]

ov = length of overhang [m]

finl = length of left wingwall [m]

dfl = distance from left edge of window to left wingwall [m]

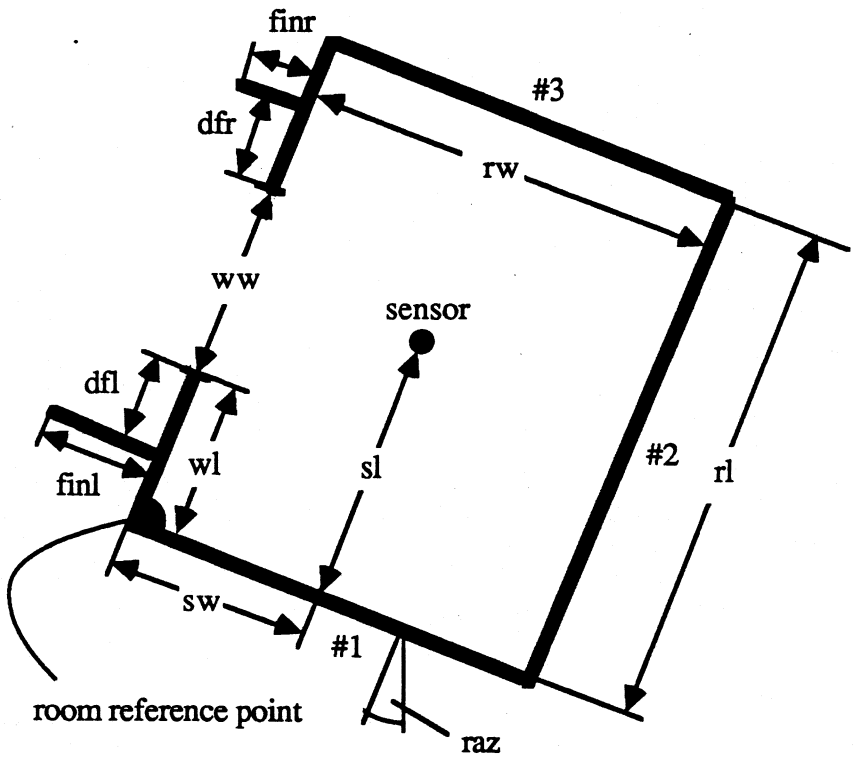
finr = length of right wingwall [m]

dfr = distance from right edge of window to right wingwall [m]

cm = mullion correction factor [0...1]

sn = surface number of the wall with the window

These parameters can be varied independently. Some cases of impossible configurations, like a sensor position outside the room or a window that does not fit in a wall, are indicated by the program. The room geometry is illustrated in fig.(5.2):



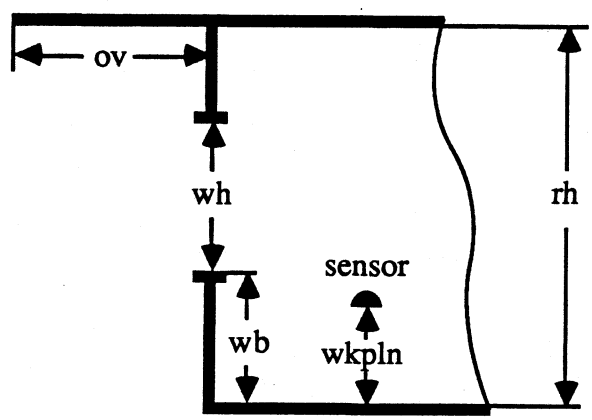


Figure (5.2): Room geometry

The radiation and sun data are read from file 'TRNSYS.out', where the FLITE read statement is compatible with the output generated by TRNSYS Type16, a radiation processor. The inputs required from this file are:

hour of the year	solar zenith angle	solar azimuth angle	I_D	I_T
...	...	...	...	...
...	...	...	...	...

Table (5.2): File 'TRNSYS.out'

where I_D and I_T, the diffuse and total horizontal radiation, are measured in kJ/m^2.

The output from FLITE is written on the file 'FLITE.out':

hour of the year	interreflected light	direct diffuse light	patch light	total daylight
...	...	...	...	...
...	...	...	...	...

Table (5.3): File 'FLITE.out'

The interreflected and direct diffuse light contain the contributions from all light sources, i.e. windows and patches. The column 'patch light' summarizes the contribution to the total daylight, including interreflected and direct diffuse light from the beam patches. The total daylight is the sum of the interreflected and the direct diffuse light. The light output is given in lux.

5.4 Limitations

The assumptions that were made when the program was developed must be considered when interpreting the results for a certain application. The limitations of the $\hat{F}$ method have already been stated in chapter 3, i.e. all surfaces of the room are assumed to behave like lambertian sources with the same luminance regardless of the viewing angle. This is certainly not realistic, since most surfaces reflect light in part in a specular way. Especially in the case of beam sunlight, which is assumed to be perfectly diffuse after the first reflection, this assumption causes some inaccuracy. It is possible to account for specular reflecting surfaces with the $\hat{F}$ method, but the simpler model was expected to be sufficient. An exact model of the internal daylight distribution also requires an exact model of the sky luminance distribution and other components that contribute to the daylight entering the room in order to be justified. Furthermore, the luminance of the window plane was assumed to be uniform. An exact solution would require to determine the luminance of the patch of sky as seen by each surface element.

A totally empty room, as modeled here, has no light absorption by furniture or pictures on the walls. The influence on the daylight distribution by objects in the room cannot be modeled, other than to estimate their effect on the surface reflectances.

If a room with more than one window is modeled, the transmittances of the windows are the same. This is sufficient for most cases, but, for instance, a combination of windows with normal and frosted glass cannot be simulated.

Reflection from overhangs and wingwalls onto the window plane has not been accounted for. This may be a serious limitation for windows with wingwalls and

overhangs that obstruct the patch of sky viewed by the center of the window to a great extent. The daylight calculated inside the room will be conservatively low.

The sensor always is assumed to be on a horizontal plane. The daylight reaching points on tilted surfaces cannot be calculated, but in most cases the workplane illuminance is desired, which refers to the illuminance of a horizontal plane.

While the reflectances of the ceiling and the floor can be independently varied, the reflectances of the four wall surfaces are assumed to be the same. Thus, including the window reflectance, there are four reflectance values that characterize the internal room surfaces. A window absorbs radiation, so the reflectance of the window cannot be obtained from the knowledge of its transmittance.

CHAPTER 6

Results

Various parameters can be varied in the input of FLITE. This chapter shows the influence of some of these variations on the daylight availability in a room.

6.1 Ideal day

The radiant energy reaching the surface of the earth does not vary uniformly over a day. Even for very clear or perfectly overcast days, the data calculated with TRNSYS Type16 is not symmetric over noon. For testing purposes of this program it was desired that the radiant energy distribution over a day follows a perfect pattern. Then, any irregularities in the results of FLITE would not be due to the radiation data. TRNSYS Type16 was used to generate hourly sun and radiation data, i.e. the contents of file 'TRNSYS.out' (see fig.(5.4)). The radiation data was calculated from the direct normal and total horizontal solar radiation of a typical meteorological year (TMY) for Madison, Wisconsin. The 9th of March has been randomly picked to represent a clear day with approximate highs of the total and diffuse horizontal radiation of 800 W/m^2 and 150 W/m^2, respectively. A sine curve has been applied to generate symmetric data over the day. The radiation distribution of this ideal day is plotted in fig.(6.1).

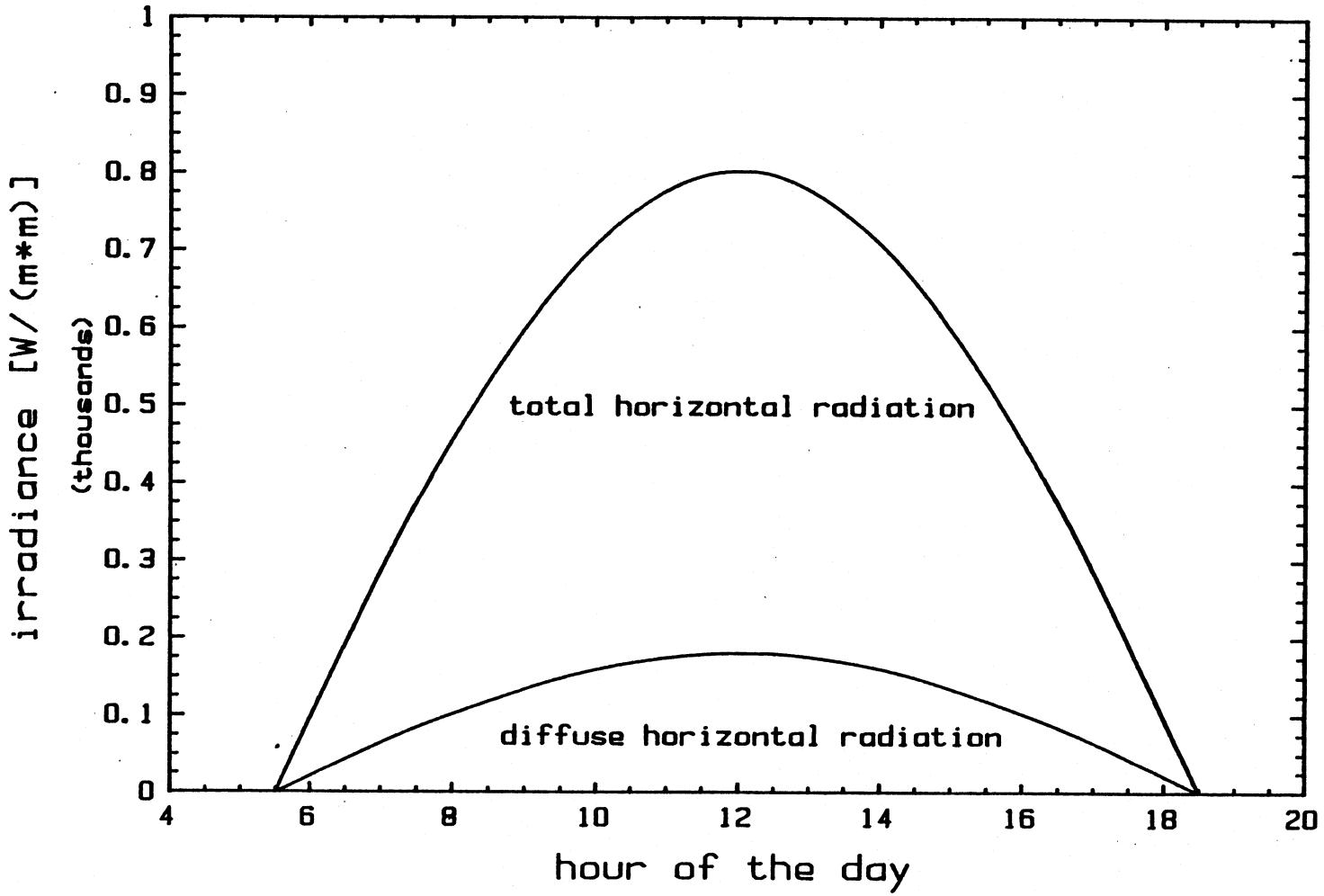


Figure (6.1): Ideal day radiation distribution

The TMY data is integrated over an hour, identified by the end of the hour, and given in kJ/m^2. The input required by FLITE must be in W/m^2, which is computed by dividing the TRNSYS output by 3.6, thus switching from hourly radiant energy to energy per second or instantaneous radiant energy. Daylighting calculations only make sense for instantaneous data, because there is no storage medium for light. Integrated input data would result in the calculation of accumulated daylight. In order to obtain a symmetric distribution of the instantaneous radiant energy over the day, the new values have been assigned to a time half an hour before the time of the integrated values.

6.2 Contributions

Figures (6.3), (6.4),(6.5) and (6.6) show the contributions of four components to the total daylight at the sensor position. These four components are written on the file 'FLITE.out', which was illustrated in table (5.3):

1) interreflected light from all light sources, i.e. beam patches and windows;

2) direct diffuse light from all light sources;

3) total contribution of the beam patches, i.e. direct diffuse and interreflected component; and

4) total daylight, i.e. the sum of 1) and 2).

The total contribution of the beam patches is shown in a separate curve to illustrate their significant contribution to the daylight illuminance at a specific point.

The room used to generate fig.(6.3) has a window facing south and the geometry allows beam patches to occur on surface #4 in the morning, on the floor during the day , and on surface #2 in the evening. The room is illustrated in

fig.(6.2):

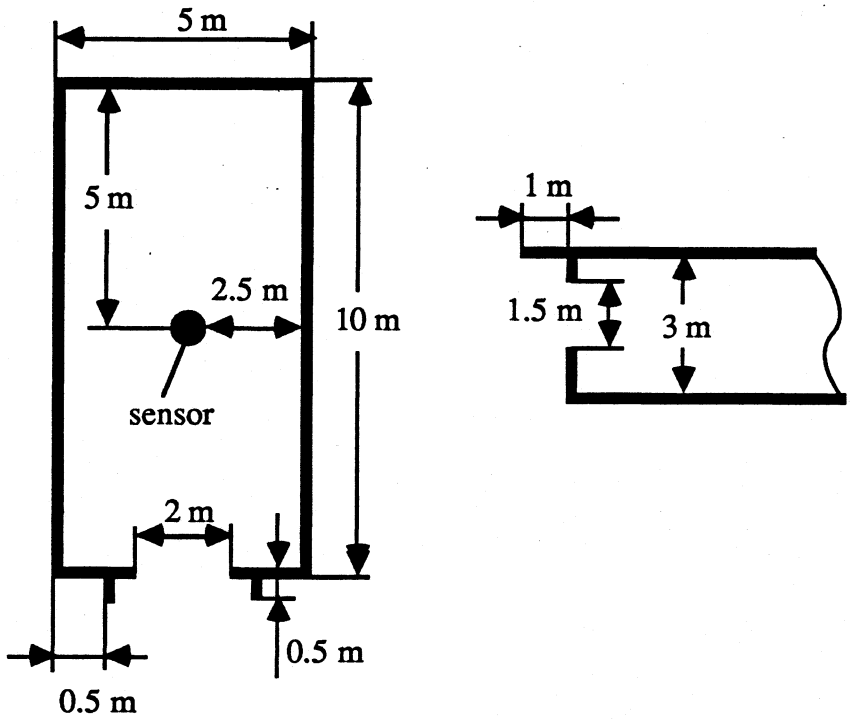


Figure (6.2): Room geometry for fig.(6.4)

The data not shown in fig.(6.2) can be taken from the input file 'FLITE.in' for this simulation, shown in table (6.1). For the meaning of the values refer to table (5.1).

5.0	10.0	3.0	0.0							
2.5	5.0	0.76	1							
0.5	0.7	0.3	0.1	0.85						
2	2.0	0.2								
1.5	1.0	1.5	2.0	1.0	0.5	0.5	0.5	0.5	1.0	1

Table (6.1): File 'FLITE.in' for fig.(6.3)

At noon, about 75% of the total daylight is due to the interreflected component, while only approximately 24% is due to the combined direct diffuse component from beam patches and window. The direct diffuse component of the patches is negligible in most cases, because only that part of the beam patch that is above the workplane contributes, i.e it must be viewed by the sensor. In addition to this, the configuration factor between sensor and a beam patch element is usually very small. This means that the total beam patch component shown in fig.(6.3) consists mostly of the interreflected light from the patch. This is exactly true for the hours between 10:30 am and 1:30 pm, when there are no beam patches on the walls, but merely on the floor. This can be derived from the geometry of the room and the solar zenith and azimuth angle, given in appendix A for the perfect day. The shape of the curve showing the beam patch contribution indicates the changing sizes of the patches. For 9:30 am and 2:30 pm there are patches both on walls and on the floor, which increases the illuminance at the sensor position due to the patches.

The same room geometry was used for fig.(6.4), but the window was facing north. The input file shows the data used:

5.0	10.0	3.0	0.0							
2.5	5.0	0.76	1							
0.5	0.7	0.3	0.1	0.85						
2	2.0	0.2								
1.5	1.0	1.5	2.0	1.0	0.5	0.5	0.5	0.5	1.0	3

Table (6.2): File 'FLITE.in' for fig.(6.4)

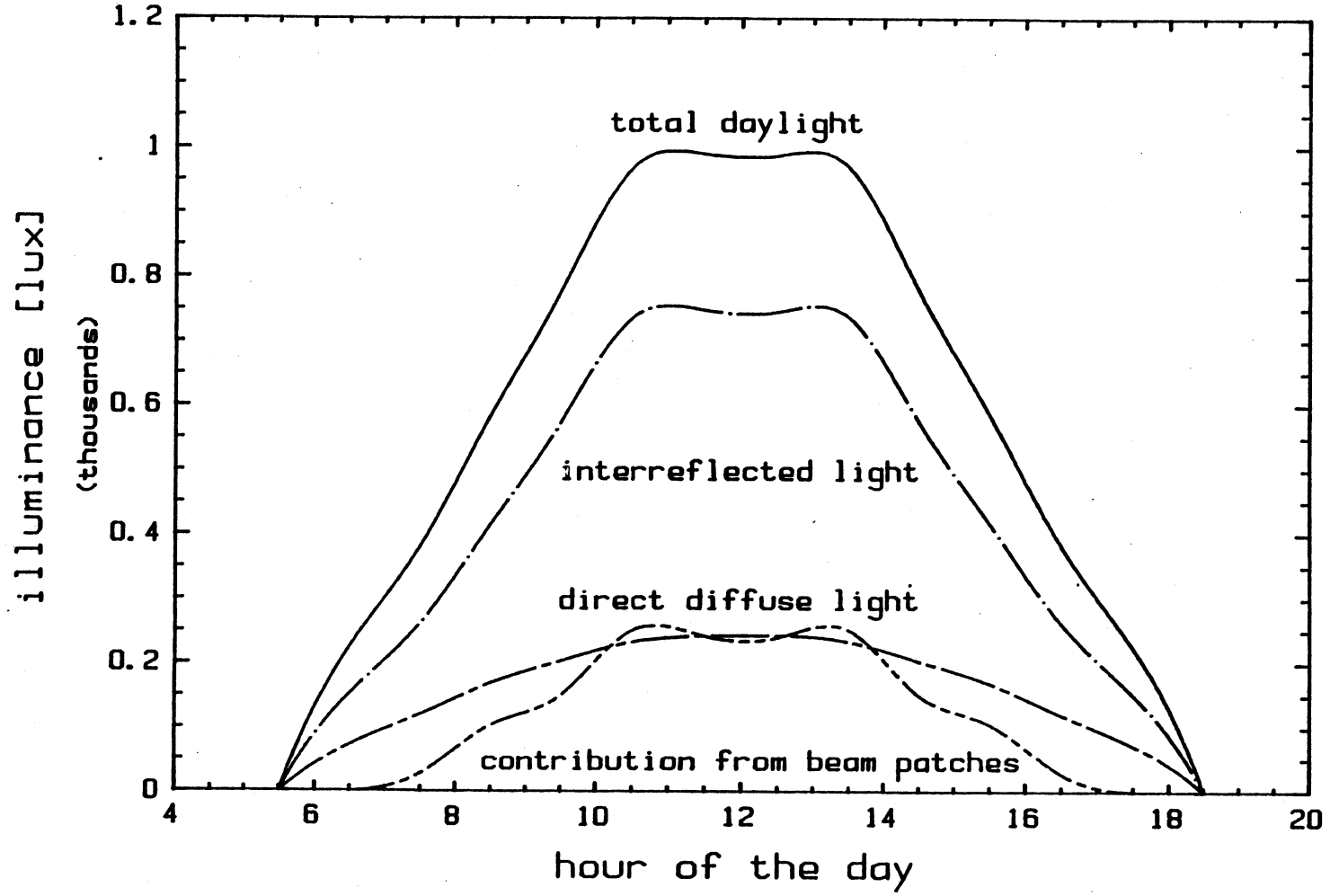


Figure (6.3): Daylight contributions for one window facing south

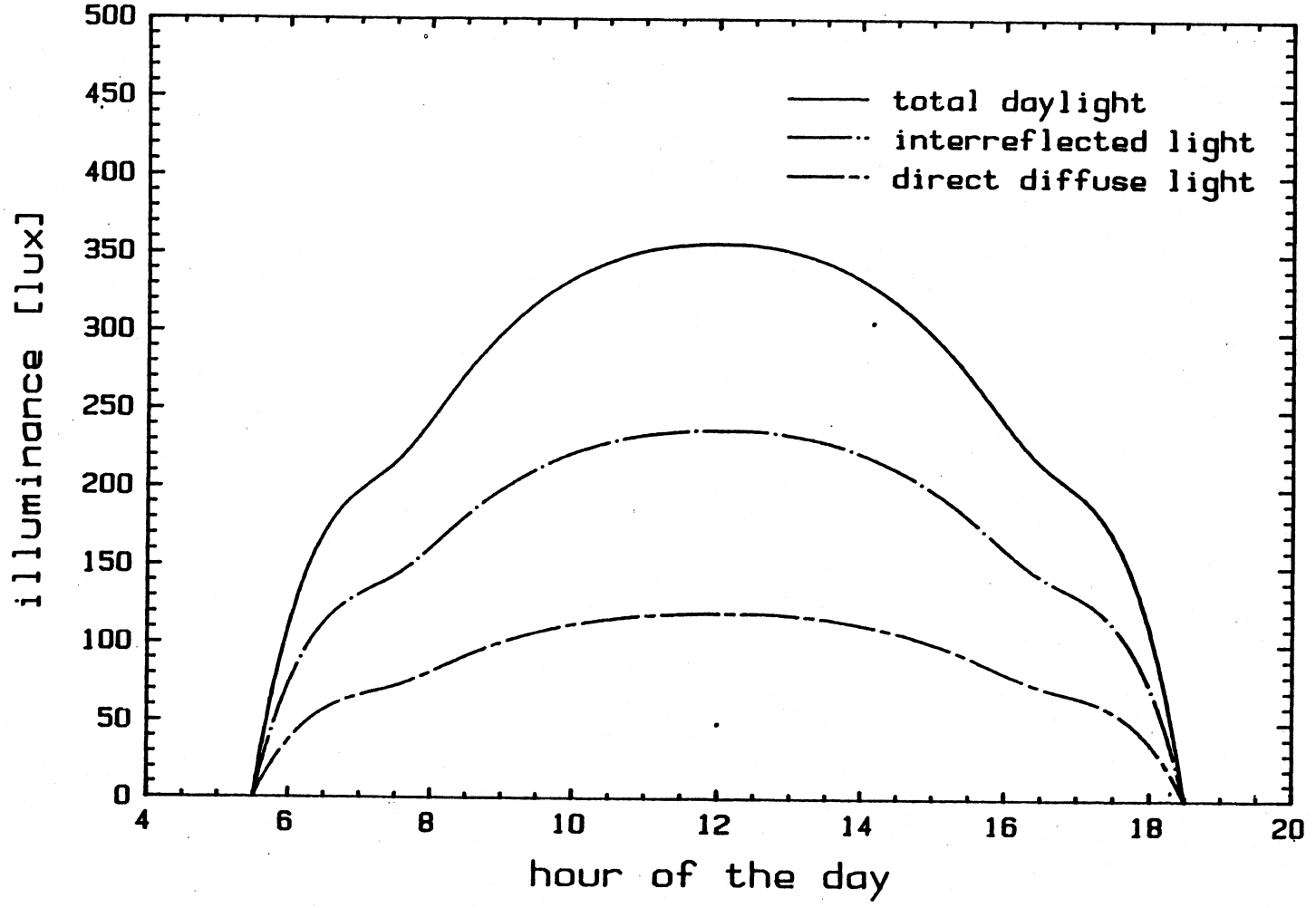


Figure (6.4): Daylight contributions for one window facing north

As expected, the total daylight is much lower, which is about 36% of the total in fig.(6.3) in this case. The curve indicating the contribution of the beam patches is not on the plot, because there is no beam sunlight entering a window facing north in march in the northern hemisphere. The interreflected light accounts for approximately 66% and the direct diffuse component for about 33% of the total daylight at the sensor position. The irregularities at 7:00 am and 5:00 pm are due to the sky luminance distribution models used. The averaging of the sky luminance distribution over 25 points for the window plane instead of assigning discrete luminance values to each element in the room may be a contributing factor as well (see chapter 5.4).

Fig.(6.5) and (6.6) illustrate a case where beam sunlight reaches the point of interest directly. The room geometry from the previous examples was used, but instead of one, the room had two windows. For fig.(6.5), the windows were in surface #1 facing south and surface #2 facing east, where the window facing east was in the center of the wall. The input data was as follows:

5.0	10.0	3.0	0.0							
2.5	5.0	0.76	2							
0.5	0.7	0.3	0.1	0.85						
2	2.0	0.2								
1.5	1.0	1.5	2.0	1.0	0.5	0.5	0.5	0.5	1.0	1
4.0	1.0	1.5	2.0	1.0	0.5	0.5	0.5	0.5	1.0	2

Table (6.3): File 'FLITE.input for fig.(6.5)

At 7:30 am, beam sunlight reaches the sensor and overwhelms the other components. The sensor illuminance is more than three times as high as at 10:30 am, when the illuminance due to direct diffuse and interreflected light only is highest. This high is not at noon, but shifted towards the morning. The window luminance of the window facing east is higher in the morning and lower in the afternoon, while the luminance of the window facing south is symmetrical over the day, as seen in fig.(6.3). Considering the path of the sun in the sky, this is a reasonable result.

The direct diffuse component increases by a factor 5 compared to the case of the window facing south only (fig.(6.3)). This can be explained with the sensor position. It is 5 meters away from the window facing south, but only 2.5 meters from the window facing east. The direct diffuse component is calculated with the configuration factors between the elements that represent the window and the point of interest. For the window facing east these are greater than for the window facing south. The configuration factor is not a linear function of the distance from the source, as can be seen from eqn.(3.11). The interreflected component is only twice as high as opposed to the case of the single window facing south. This indicates that the $\widehat{F}$ method does not take into account the direct components between the sources and the point of interest, as stated in section 4.2. It processes the configuration factors between the surface elements, which are independent of the sensor position. Another interpretation of the doubled interreflected component is, that the surface elements viewed by the sensor are illuminated twice as much due to interreflection for the two windows than in the case of the single window facing south.

The illuminance at the point of interest due to beam patches is the same in the afternoon as in fig.(6.3) for the window facing south only, but in the morning there are additional beam patches due to the window facing east.

In fig.(6.6), the results for a symmetrical window configuration are shown. The room geometry was kept the same as in fig.(6.5), but instead of windows facing south and east, windows facing south and west were simulated. The input is shown in table (6.4):

5.0	10.0	3.0	0.0							
2.5	5.0	0.76	2							
0.5	0.7	0.3	0.1	0.85						
2	2.0	0.2								
1.5	1.0	1.5	2.0	1.0	0.5	0.5	0.5	0.5	1.0	1
4.0	1.0	1.5	2.0	1.0	0.5	0.5	0.5	0.5	1.0	4

Table (6.4): File 'FLITE.in' for fig.(6.6)

For a perfectly symmetrical day with respect to the radiation data and a symmetrical window configuration, fig.(6.5) is symmetrical to fig.(6.6).

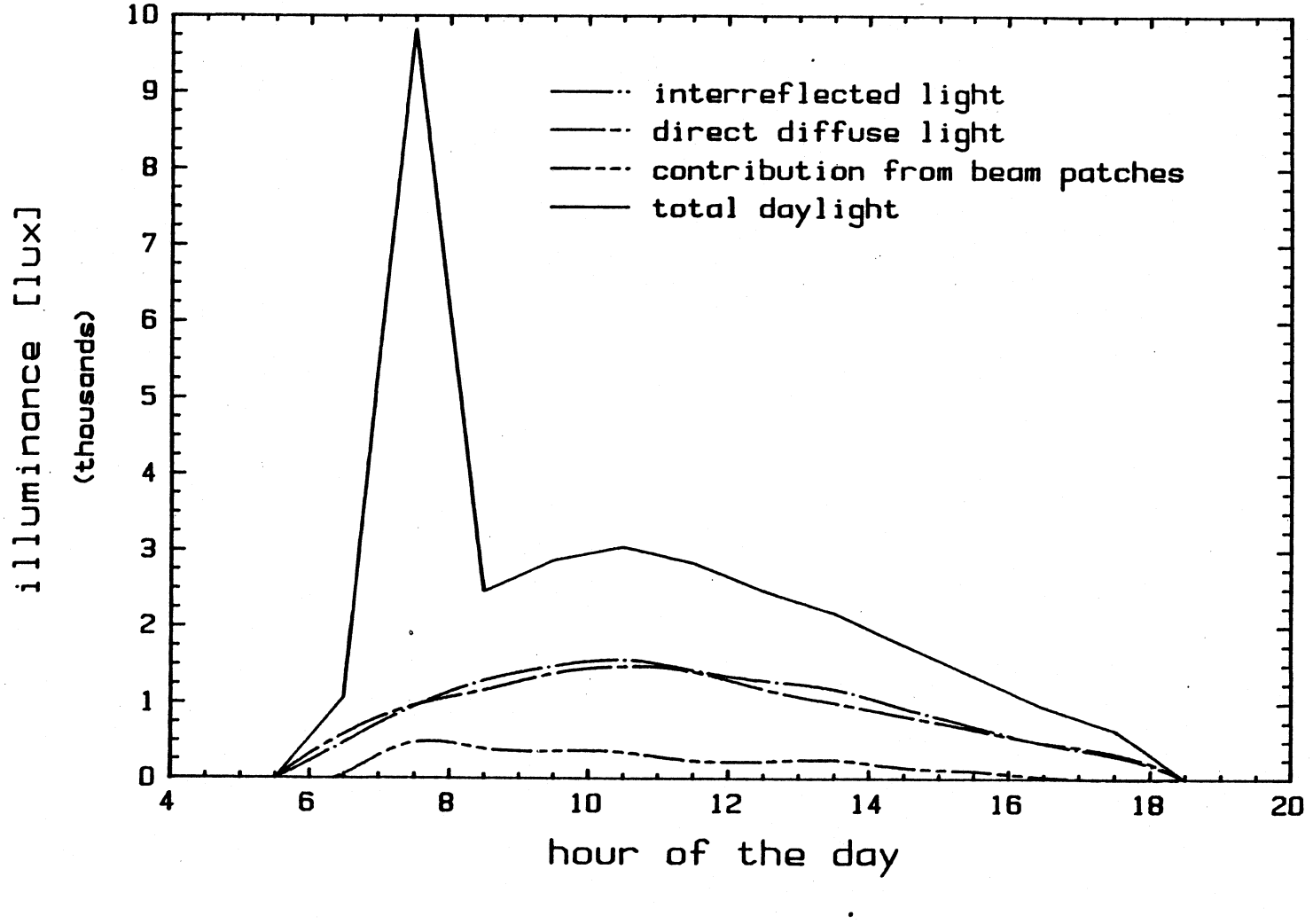


Figure (6.5): Daylight contributions for windows facing south and east

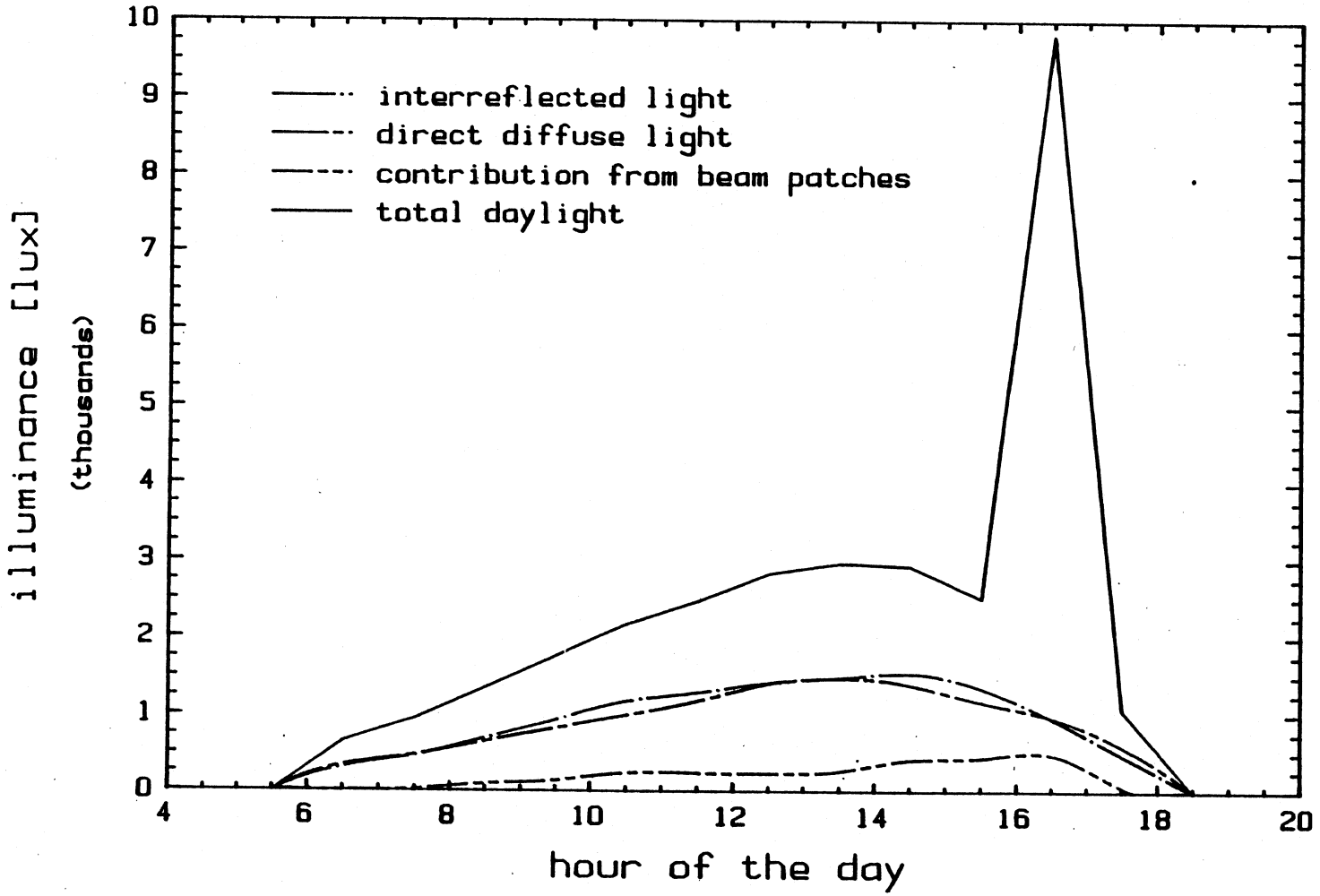


Figure (6.6): Daylight contributions for windows facing south and west

6.3 Reflectances

The influence of the surface reflectances on the total daylight at the sensor position is investigated in this chapter. The advantage of calculating the interreflected light with the $\widehat{F}$ method is that the reflectances can be precisely accounted for. FLITE requires the reflectances of the four walls, the ceiling, the floor, and the window as input.

For the room geometry already used as a model in the previous chapter, and a single window facing north, an input file as in table (6.2) had been used for fig.(6.7), (6.9) and (6.10). All data are exactly the same, only the reflectances for the respective surfaces have been varied.

The reflectance of the walls was expected to have the greatest influence on the total daylight availability in the room, because the surface area of the walls is greater than the area of the ceiling or the floor. For wall reflectances between 0.1 and 0.9, the results have been plotted in fig.(6.7). Increasing the wall reflectance in increments of 0.2 results in increasing the total illuminance at the sensor position by a factor of 1.5 for each increment.

In order to show the influence of the room geometry, the room dimensions have been changed to 6 by 5 by 3 meters for the results shown in fig.(6.8). The input is shown in table (6.5), where the x indicates that the wall reflectance was varied:

5.0	6.0	3.0	0.0							
2.5	1.0	0.76	1							
x	0.7	0.3	0.1	0.85						
2	2.0	0.2								
1.5	1.0	1.5	2.0	1.0	0.5	0.5	0.5	0.5	1.0	3

Table (6.5): File 'FLITE.in' for fig.(6.8)

The total daylight at the sensor position is higher than in the case of the larger room. The distance of the sensor from the window was kept the same, and the different result is not due to a higher direct diffuse component from the window. A smaller area of the room surfaces is causing a higher interreflected component, because a smaller surface also absorbs less energy. Furthermore, the distance between the sensor and the wall facing south is only 1 meter as opposed to 5 meter in fig.(6.7), which will increase the direct diffuse component from the surface elements (see eqn.(4.10)).

The floor and ceiling reflectances were varied in increments of 0.4 between 0.1 and 0.9. The room configuration is again the same as for fig.(6.7) with the input data shown in table (6.2). For the variation of the floor reflectance, the total daylight at the sensor position increases by approximately 1.3 for each increment.(fig.(6.9)). The smaller influence on the total daylight, compared to varying the wall reflectance, is due to the smaller area of the floor.

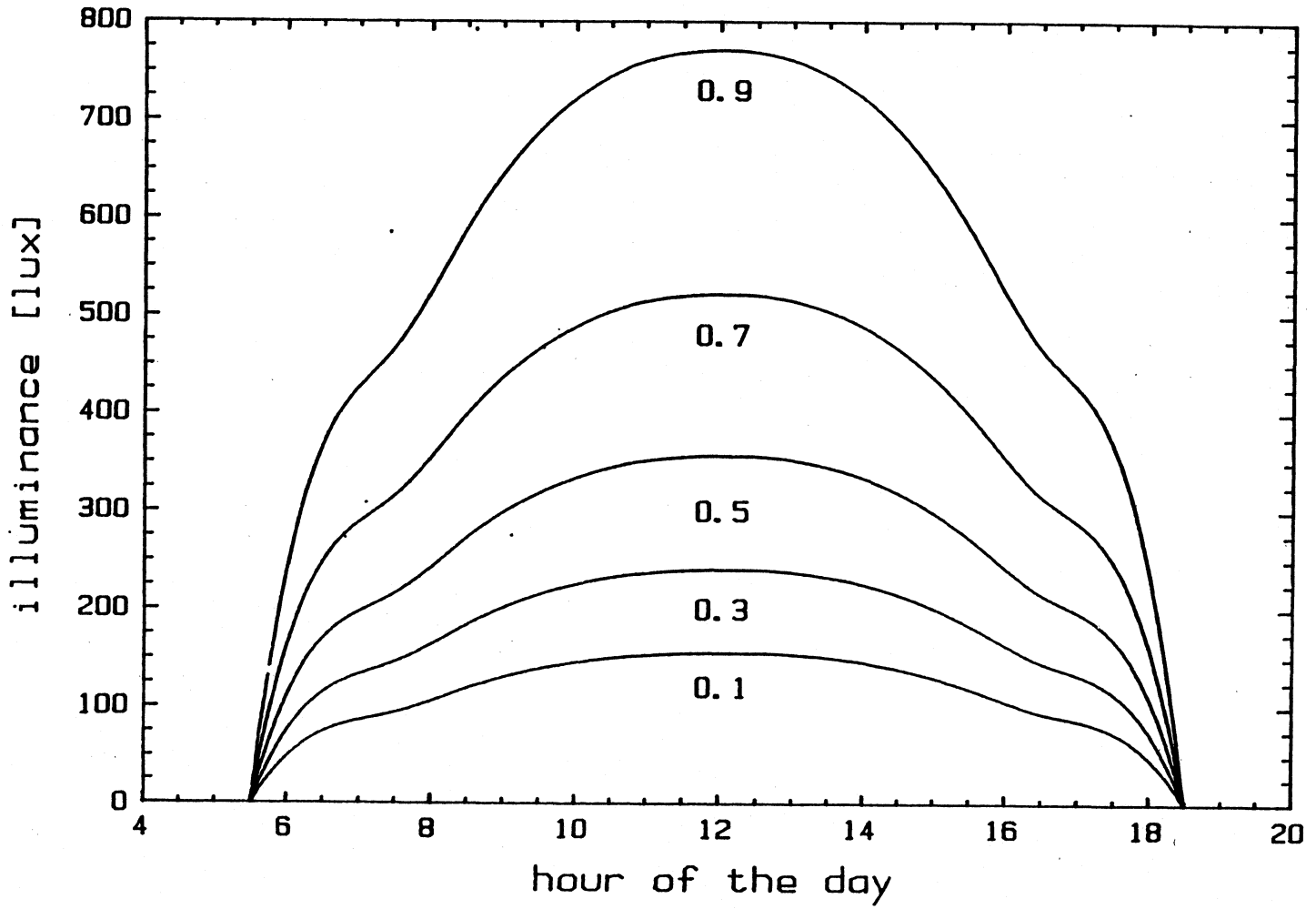


Figure (6.7): Variation of the wall reflectances

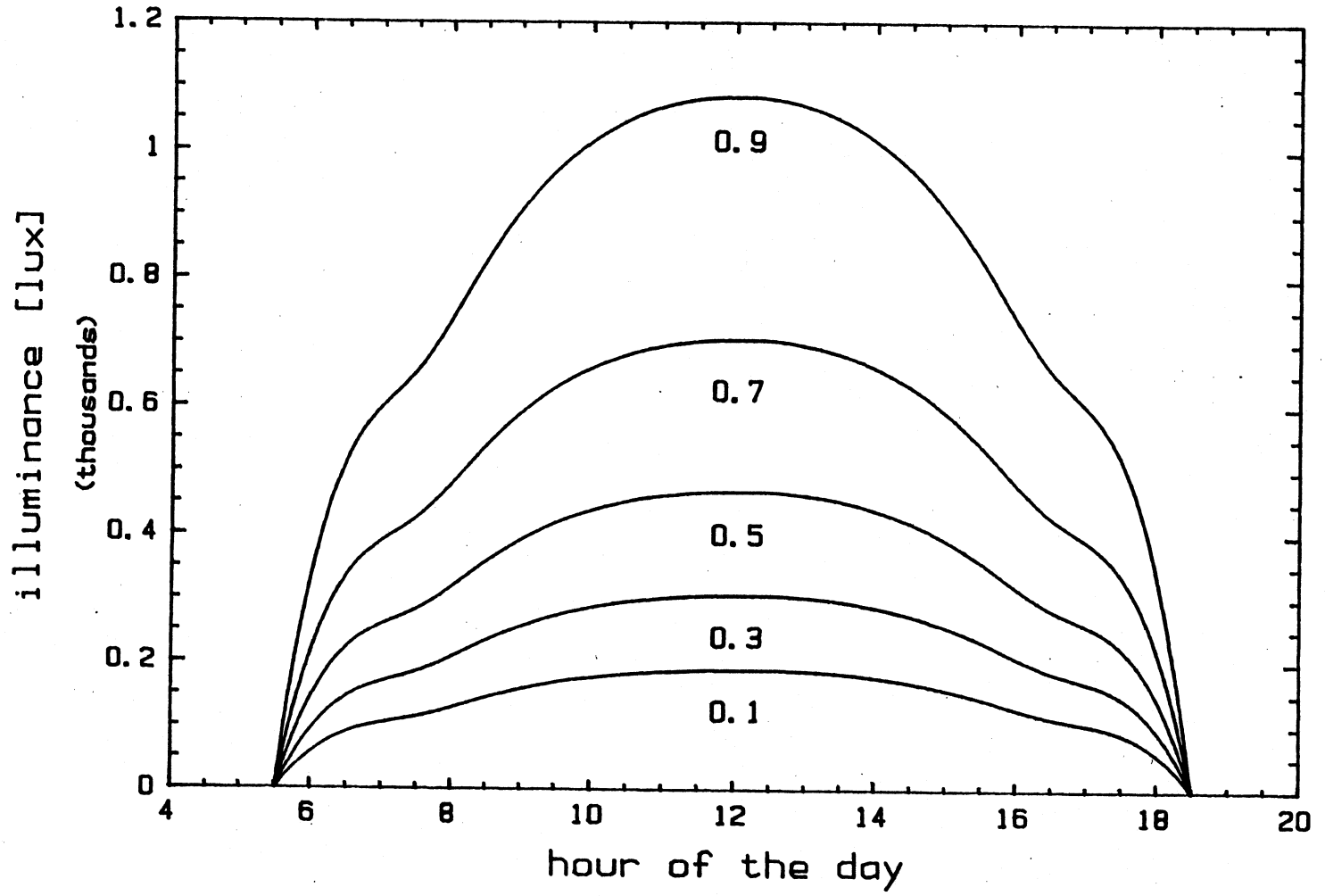


Figure (6.8): Variation of the wall reflectances for a smaller room

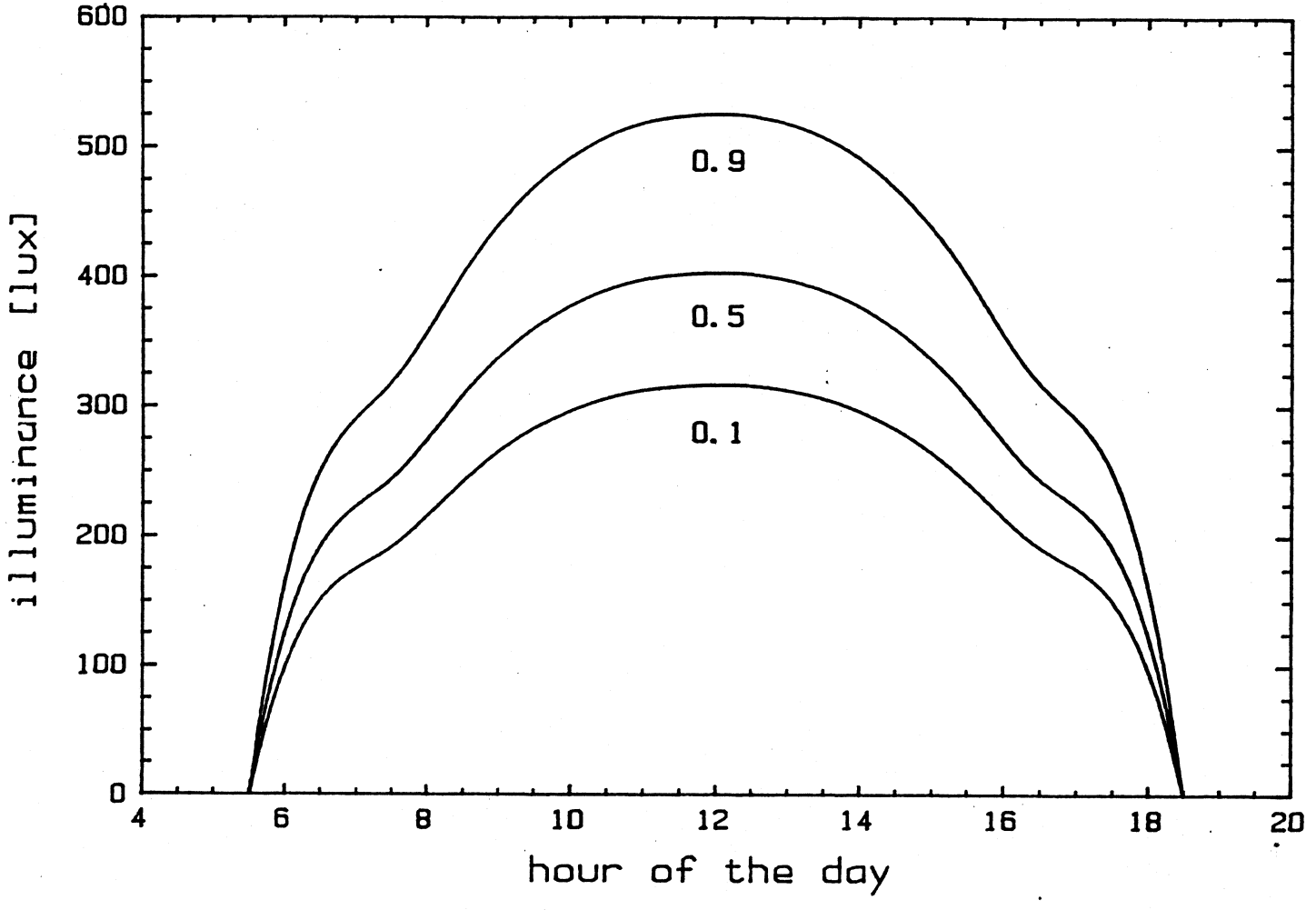


Figure (6.9): Variation of the floor reflectance

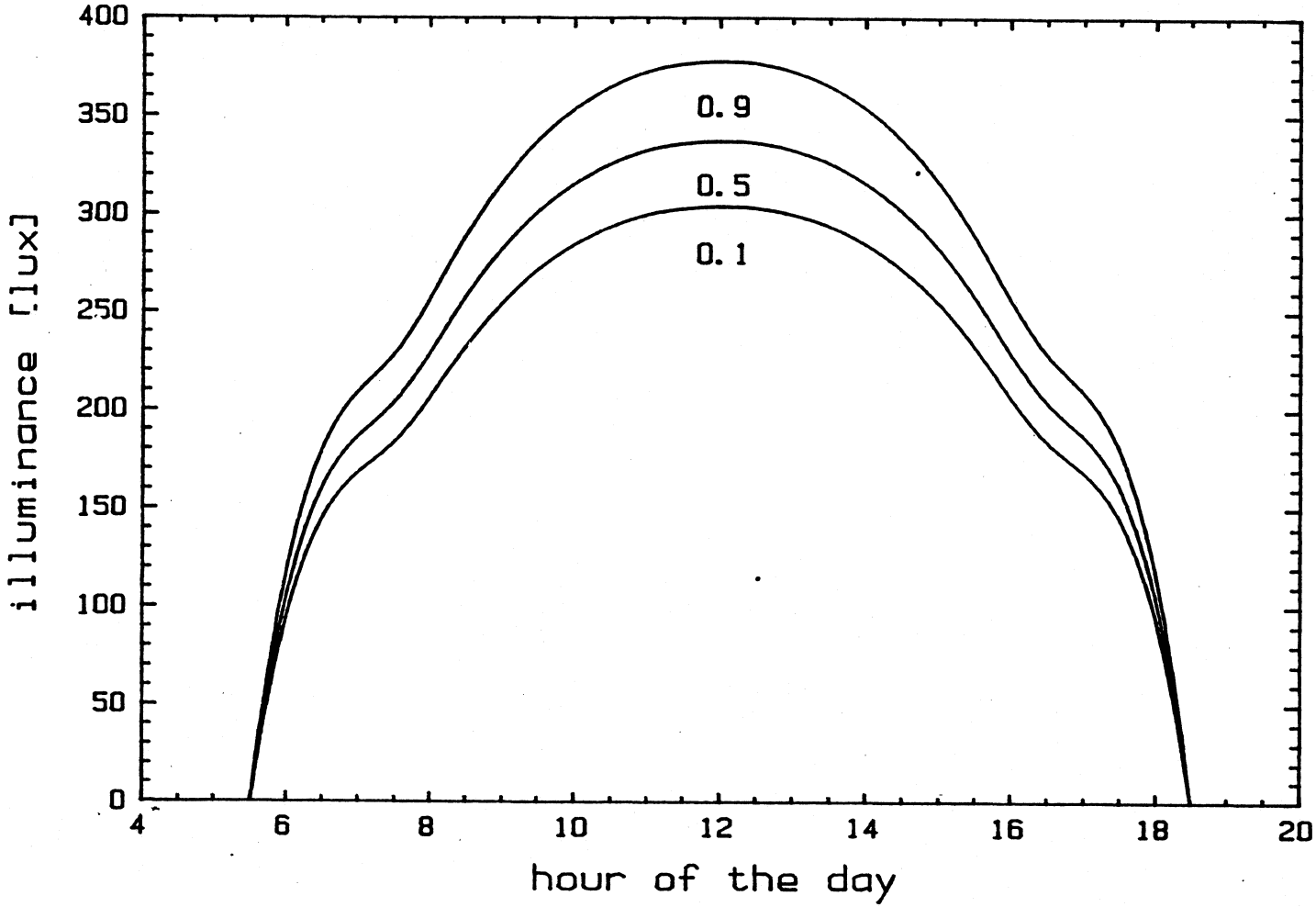


Figure (6.10): Variation of the ceiling reflectance

The influence of the ceiling reflectance on the total illuminance of the point of interest is shown in fig.(6.10). Increasing the ceiling reflectance by 0.4 between 0.1 and 0.9 results in an illuminance value that is greater by 1.1 for each step. Not only the areas of the surfaces, but also their location (with respect to the sensor) contributes to the total daylight, as stated before. The floor does not contribute to the total illuminance with a direct diffuse component, because the sensor cannot view the floor. Considering the ceiling, this direct diffuse component has to be taken into account. The higher illuminance value at the the sensor for fig.(6.9) is due to the ceiling reflectance of 0.7. The floor reflectance in fig.(6.10) is only 0.3, thus resulting in lower illuminances. The factors by which the daylight increases are not the same for the variation of the floor and ceiling reflectances. This is due to the high ceiling reflectance in fig.(6.9) and the low floor reflectance in fig.(6.10). The $\hat{F}$ method accounts for multiple interreflections between the room surfaces and the exitance of a surface depends not only on the reflectance of this surface, but also on the properties of the other surfaces involved in the model.

6.4 Sensor position

The room shown in fig.(6.2) has been modeled to simulate different sensor positions in the room. The sensor has been moved on the centerline between the wall facing east and the wall facing west from the surface facing north to the window facing south. The input data is consistent with table (6.1), except that the length distance of the sensor from the room reference point has been varied. The result of this simulation is shown in fig.(6.11).

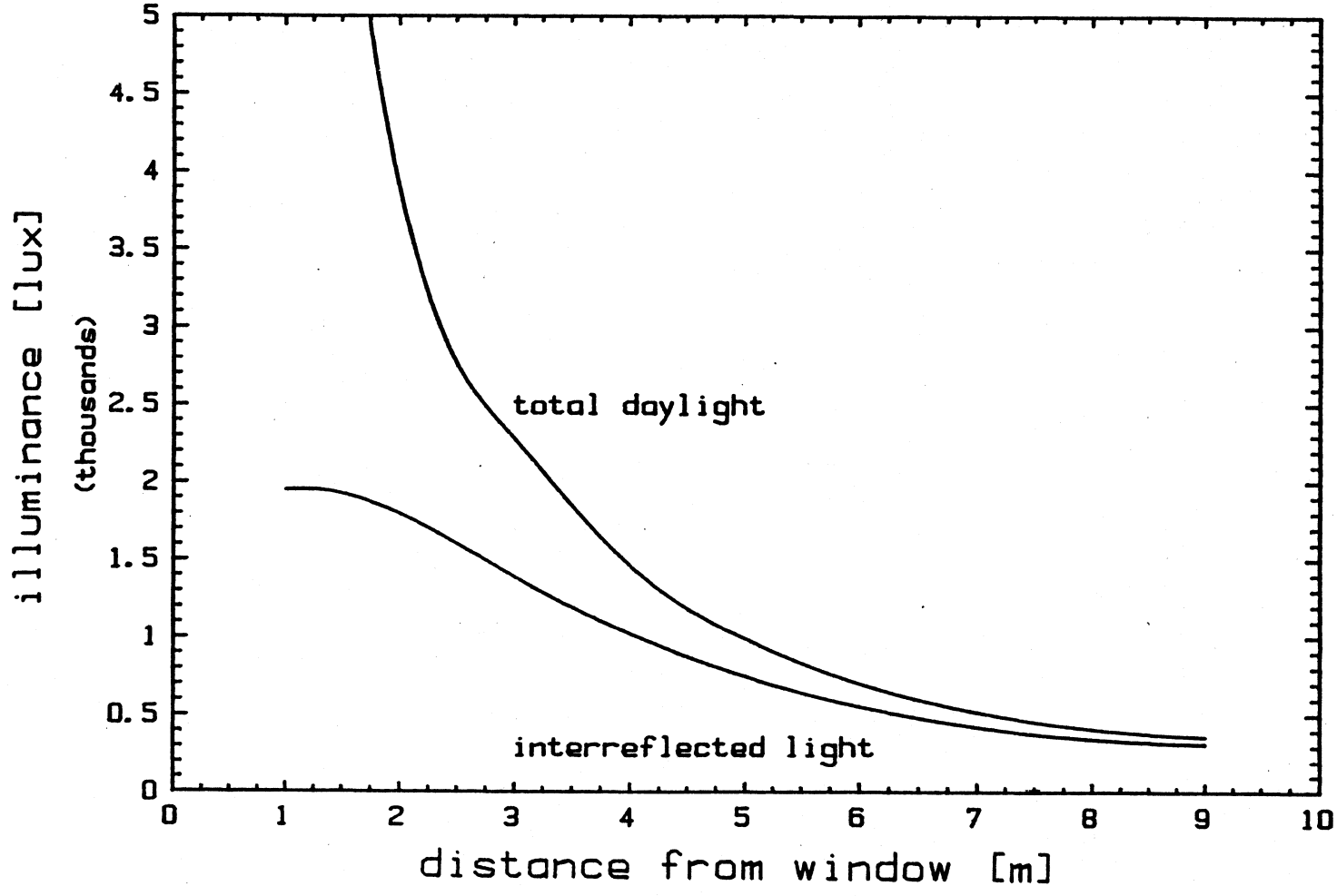


Figure (6.11): Variation of the sensor position

The interreflected component increases as the distance between sensor and window decreases. The interreflected light is calculated by determining the illuminance of the elements that the room surfaces have been divided into, and using these elements as light sources that contribute to the illuminantion of the point of interest. These elements will have a higher illuminance near the window, which causes the interreflected component near the window to be higher than in the back of the room. As the sensor gets close to the window, the total daylight increases rapidly due to beam sunlight on the point of interest.

CHAPTER 7

Conclusion

FLITE provides the option to vary parameters that characterize a rectangular room and its properties. The $\hat{F}$ method introduces the possibility of studying the effect of different reflectances of the room surfaces without using average values. Thus, the effect of changing the properties of a specific surface on the daylight at a specific point in the room can be studied.

FLITE can be used to develop daylight utilizability charts by simulating the changes of room parameters and researching their dependence on each other. Utilizability charts would illustrate the useful daylight over a period of time in terms of room parameters.

The program is set up to process hourly radiation data. Thus, longterm simulations can be done with minimal computational effort. The geometry of the room is taken into account with the $\hat{F}$ method before the first time step and the matrix equations involved do not have to be solved repeatedly.

The significant contribution of beam patches on the room surfaces can be simulated with FLITE. By treating them as additional windows it was assumed that the beam impinging on the respective surface is reflected in a perfectly diffuse manner. This assumption made it possible to apply algorithms that were already developed for the windows.

The time limit given for developing FLITE made it impossible to compare its results with measurements or the results of other daylighting programs. Some

research has been done in this respect, and working with illuminance values for one and half years provided the experience to decide, if calculation results were within a reasonable range. Suitable daylighting data for comparisons is hard to find, since most researchers do not provide all the information necessary for this task. In addition to the illuminance data, the sun position, climate and radiation data, the room geometry and the surface properties are needed.

This thesis intends to provide a clear picture of what has been done and how the program should be used. It is hoped that this study is useful for another daylighting researcher.

APPENDIX A

Ideal day data

hour of the year	solar zenith angle	solar azimuth angle	diffuse horizontal radiation	total horizontal radiation
1609.0000	9.900E+01	0.000E+00	0.000E+00	0.000E+00
1610.0000	9.900E+01	0.000E+00	0.000E+00	0.000E+00
1611.0000	9.900E+01	0.000E+00	0.000E+00	0.000E+00
1612.0000	9.900E+01	0.000E+00	0.000E+00	0.000E+00
1613.0000	9.900E+01	0.000E+00	0.000E+00	0.000E+00
1614.0000	9.900E+01	0.000E+00	0.000E+00	0.000E+00
1615.0000	8.639E+01	-7.929E+01	1.548E+02	6.876E+02
1616.0000	7.759E+01	-7.038E+01	3.024E+02	1.343E+03
1617.0000	6.771E+01	-5.862E+01	4.284E+02	1.904E+03
1618.0000	5.911E+01	-4.494E+01	5.328E+02	2.369E+03
1619.0000	5.254E+01	-2.869E+01	6.048E+02	2.689E+03
1620.0000	4.890E+01	-9.932E+00	6.444E+02	2.866E+03
1621.0000	4.890E+01	9.932E+00	6.444E+02	2.866E+03
1622.0000	5.254E+01	2.869E+01	6.048E+02	2.689E+03
1623.0000	5.911E+01	4.494E+01	5.328E+02	2.369E+03
1624.0000	6.771E+01	5.862E+01	4.284E+02	1.904E+03
1625.0000	7.759E+01	7.038E+01	3.024E+02	1.343E+03
1626.0000	8.639E+01	7.929E+01	1.548E+02	6.876E+02
1627.0000	9.900E+01	0.000E+00	0.000E+00	0.000E+00
1628.0000	9.900E+01	0.000E+00	0.000E+00	0.000E+00
1629.0000	9.900E+01	0.000E+00	0.000E+00	0.000E+00
1630.0000	9.900E+01	0.000E+00	0.000E+00	0.000E+00
1631.0000	9.900E+01	0.000E+00	0.000E+00	0.000E+00
1632.0000	9.900E+01	0.000E+00	0.000E+00	0.000E+00

The diffuse horizontal and total horizontal radiation are given in kJ/m^2. The solar azimuth angle is zero due south with east being negative and west positive.

APPENDIX B

FLITE listing

```
                              PROGRAM FLITE
C ************************************************************************
C ***                  A dayligting program developed by                ***
C ***                                                                   ***
C ***                        Jan-Uwe Kluessendorf                       ***
C ***                                                                   ***
C ***                      SOLAR ENERGY LABORATORY                      ***
C ***                  UNIVERSITY OF WISCONSIN - MADISON                ***
C ***                              1987                                 ***
C ************************************************************************
C
C  FLITE has to be linked to the subroutine-package LINPACK for
C  subroutines SGEFA and SGESL that are used for the matrix operations.
C
C  Input ==> File 'FLITE.in' :
C
C     rw   | rl     | rh     | raz
C     ---------------------------
C     sw   | sl     | wkpln | nw
C     ---------------------------------------
C     reflw | reflc  | reflf  | reflg  | t0
C     ---------------------------------------
C     beta  | omega | reflgr  |
C     ---------------------------------------------------------------
C     wl   | wb    | wh     | ww    | ov | finl | dfl | finr | dfr | cm | sn
C
C  where :
C
C  rw   : room width [m]
C  rl   : room length [m]
```

```
C rh    : room height [m]
C raz   : Room azimuth angle (exact south=0., east negative, west positive)
C ov    : Overhang length (distance away from wall)
C sl    : sensor location from reference point in length direction
C sw    : sensor location from reference point in width direction
C wkpln : Workplane height [m]
C nw    : number of windows, maximum = 5
C reflw : reflectance of walls [0...1]
C reflc : reflectance of ceiling [0...1]
C reflf : reflectance of floor [0...1]
C reflg : reflectance of glass used as fenestration [0...1]
C t0    : hemispherical transmittance of fenestration [0...1]
C beta  : climate indicator (rural - 1, urban - 2, industrial - 3)
C omega : water vapor content of the atmosphere (0.5, 1, 2, 3, 4 or 5) in
C         cm of water
C reflgr: reflectance of ground [0...1]
C wl    : distance from left edge of window to left edge of respective wall [m]

C wb    : distance from bottom of window to floor [m]
C wh    : height of window [m]
C ww    : width of window
C ov    : length of overhang [m]
C finl  : length of fin left of window (as seen from inside room) [m]
C dfl   : distance of fin left of window to left window edge along wall
C         with window (as seen from inside room) [m]
C finr  : length of fin right of window (as seen from inside room) [m]
C dfr   : distance of fin right of window to right window edge along
C         wall with window (as seen from inside room) [m]
C cm    : mullion correction factor [0...1]
C sn    : surface number of the wall with the window [1, 2, 3 or 4], south
C         facing surface is 1, counting counter-clockwise.
C
C Input ==> File 'TRNSYS.out' : Sun and radiation data fron TRNSYS Type16
C
C  hour of the year | solar zenith angle | solar azimuth angle | Id  | It
C ---------------------------------------------------------------------
C          ...          |          ...          |     ...                | ... | ...
```

```
C
C solar zenith angle : zero due south, west positive, east negative
C Id              : diffuse horizontal radiation [kJ/(m*m)]
C It              : total horizontal radiation   [kJ/(m*m)]
C
C Output ==> File 'FLITE.out'
C
C  hj - 0.5 | interr. light | dir. diff. light |  patche light | total light
C ---------------------------------------------------------------
C     ... |    ...        |    ...            |        ...    |  ...
C
C where :
C
C hj              : hour of the year
C interr. light   : total interreflected light [lux]
C dir. diff. light : direct diffuse light [lux]
C patch light     : total contribution from beam patches [lux]
C total light     : interr. light + dir. diff. light [lux]
C
C Parameters and arrays :
C
C F(i,j)      : configuration factor matrix (= F-hat matrix after being
C               processed in subroutine SGESL)
C R(i,j)      : reflectivity matrix
C winelem(i,j) : array containing the numbers of the elements that lie in a
C               window area
C b(i)        : working array in subroutine SGESL
C floorbeam(i) : array containing the numbers of the elements that lie in the
C               area of a beam patch on the floor.
C FS(i)       : array containing the configuration factors from the sensor
C               to all elements above the workplane.
C ipvt(i)     : working array in subroutine SGEFA
C skylum(i)   : array containing 25 luminance points evenly distributed over
C               the patch of sky as seen by a point in the center of the
C               window.
C surfelem(i) : array containing the numbers of the elements that lie above
C               the workplane.
C wallbeam(i) : array containing the numbers of the elements that lie in the
```

```
C              area of a beam patch on a wall.
C  xr(i)      : working array subroutine CONFIG
C  yr(i)      : working array in subroutine CONFIG
C  zr(i)      : working array in subroutine CONFIG
C  lda        : leading dimension of array (=6*n*n)
C  lumwin     : luminance of window plane
C  n          : number of elements along one edge of the surfaces of the room.


    Parameter (lda=150, n=5)

    real skylum(25), lumwin, F(lda,lda),R(lda,lda)
    real b(lda), xr(lda), yr(lda), zr(lda), FS(lda)
    integer ipvt(lda), winelem(5,lda/6), surfelem(lda)
    integer sn, beta, wallbeam(lda/2), floorbeam(lda/2)

    nf = 0
    pi = 4.*atan(1.)

    open(1,file='FLITE.in',status='old')
    read(1,*) rw, rl, rh, raz
    read(1,*) sw,sl,wkpln,nw
    read(1,*) reflw, reflc, reflf, reflg, t0
    read(1,*) beta, omega, reflgr
    read(1,*,end=2000) wl,wb,wh,ww,ov,finl,dfl,finr,dfr,cm,sn
    read(1,*,end=2000) wl,wb,wh,ww,ov,finl,dfl,finr,dfr,cm,sn
    read(1,*,end=2000) wl,wb,wh,ww,ov,finl,dfl,finr,dfr,cm,sn
    read(1,*,end=2000) wl,wb,wh,ww,ov,finl,dfl,finr,dfr,cm,sn
    read(1,*,end=2000) wl,wb,wh,ww,ov,finl,dfl,finr,dfr,cm,sn

2000 continue

    if (wkpln.ge.rh) then
      write(*,*) ' *** Error : Workplane is above ceiling : wkpln'
      write(*,*) '             has to be smaller than room height !'
      stop
    end if
```

```
      if ((raz.ge.90.) .or. (raz.le.(-90.))) then
        write(*,*) ' *** Error : Room azimuth angle raz should not be'
        write(*,*) '             equal or greater than 90 degr. and'
        write(*,*) '             not be equal or less than -90 degr..'
        write(*,*) '             Choose the new south facing surface'
        write(*,*) '             as surface # 1 and the left or most'
        write(*,*) '             southern roomcorner as room ref. pt.'
        write(*,*) '             to avoid this message !'
        stop

      end if

      open(2,file='TRNSYS.out',status='old')
      do 999 mm = 1,1000
        read(2,'(1x,f9.4,4(e11.7))',end=1000) hj,zenith, sunaz,
     @                                        radd, radt

        if (radt.eq.0.) goto 999
        jul = int((hj-0.5)/24.)+1
        sunalt = 90.-zenith
       diffuse = radd/3.6
        total = radt/3.6

      if ((sw.ge.rw) .or. (sl.ge.rl) .or. (sw.le.0) .or. (sl.le.0)) then
        write(*,*) ' *** Error : Sensor is outside room or in wall!'
        stop
      end if


C *** input of window data
      nnw = 0
      allight = 0.
      beamlight = 0.
      patchlight = 0.
      rlight = 0.
      dirlight = 0.
```

```
rewind 1

do 888 jj = 1,nw+4

   if (jj.lt.5) then
     read(1,*)
      goto 888
   end if

   read(1,*) wl,wb,wh,ww,ov,finl,dfl,finr,dfr,cm,sn

   nnw = nnw+1


if (wh+wb.gt.rh) then
   write(*,*) ' *** Error : wb = ',wb,'wh = ',wh,' rh = ',rh
   write(*,*) '     ==> Window does not fit in wall !'
    stop
end if


if (wb.gt.wkpln) then
   z = wb-wkpln

else
   z = 0.

end if

if (sn.eq.1) then
   dsw = sl
   dsr = sw
   wwl = rw

else if (sn.eq.2) then
   dsw = rw - sw
   dsr = sl
```

```
      wwl = rl

    else if (sn.eq.3) then
      dsw = rl-sl
      dsr = rw-sw
      wwl = rw

    else if (sn.eq.4) then
      dsw = sw
      dsr = rl-sl
      wwl = rl

    else
      write(*,*)
      write(*,*) ' ***  Error : surface # sn =', sn,' is not allowed'
      stop

    end if

C *** Determine elements that lie in window area
    call elements1(winelem,n,sn,nnw,rh,wb,wh,ww,wwl,wl)

C *** Determine the ground reflected light
    call reflight(diffuse,total,reflgr,t0,outrefl)

    if (sn.eq.1) then
      azl = -anglefl+raz
      azr = anglefr+raz

    else if (sn.eq.2) then
      azl = -anglefl+raz-90.
      azr = anglefr+raz-90.

    else if (sn.eq.3) then
      azl = -anglefl+raz-180.
      azr = anglefr+raz-180.
```

```
      else if (sn.eq.4) then
        azl = -anglefl+raz+90.
        azr = anglefr+raz+90.

      end if

      if (azl.lt.(-180.)) then
        azl = 360.+azl
      end if

      if (azr.lt.(-180)) then
        azr = 360.+azr
      end if

      if (azl.gt.180.) then
        azl = azl-360.
      end if

      if (azr.gt.180.) then
        azr = azr-360.
      end if

C *** Check if sun is visible by sensor through the window.
      if ((sunaz.gt.azl) .and. (sunaz.lt.azr)
     @                   .and. (sunalt.lt.angleov)
     @                   .and. (sunalt.gt.angle3)) then
        sunbeam = beam(sunalt,jul)*transmit(raz,sn,sunalt,sunaz,t0)
     @            *sind(sunalt)*phase                    ! [lux]
        beamlight = sunbeam
      else
        beamlight = 0.

      end if


C *** Check if there are patches of beam on the surfaces of the room.
C     If so, determine their location and size ,the numbers of the
C     elements that lie in the respective patch areas and the
```

```
C     illuminance of these points.
      call patches(floorbeam,wallbeam,n,sn,rw,rl,rh,wl,wb,ww,wh,
     @             finl,finr,dfl,dfr,ov,sunalt,sunaz,raz,jul,t0,phase,
     @             beam1,beam2,beam3,beam4,beam6,sw,sl,wkpln)


C *** Calculate the average sky luminance as seen by a point in the center
C     of the window.
      call winlum(sn,finl,dfl,finr,dfr,ww,ov,rh,wh,wb,lumwin,t0,
     @            sunaz,sunalt,diffuse,total,beta,omega,phase)


C *** weigh the sky luminance as seen by the window with the fraction
C     of actual window area and area of the elements that model the
C     window.
      Fwin = ww*wh

      k = 0
      do i = 1,n*n
        if (winelem(nnw,i).eq.0) then
          goto 22
        else
          k = k+1
        end if
      end do

 22   if ((sn.eq.1) .or. (sn.eq.3)) then
        elemw = rw/n
      else
        elemw = rl/n
      end if
      elemh = rh/n

      Felem = k*elemw*elemh

      weigh = Fwin/Felem
```

```
      lumwin = lumwin*weigh                                  ! [cd/(m*m)]

C *** Add ground reflected light to luminance of window plane and take
C     mullion correction factor into account
      lumwin = (lumwin+outrefl)*cm


C *** Check if F-hats were already calculated !
      if (nf.eq.1) goto 199
      nf = 1


C *** Define configuration factors for interreflection model
      call config(F,xr,yr,zr,lda,n,rw,rl,rh)


C *** Check configuration factors for accuracy (The sum of all factors
C     from one point to all elements has to be one !
      testsum = 0.
      indicator = 0
      eps = 1.e-05
      do j=1,6*n*n

        testsum = 0.
        do i=1,6*n*n
          testsum = testsum + F(j,i)
        end do

        if (abs(testsum-1.).gt.eps) then
          write(*,*) '   The sum of the configuraion factors in each
     @ row of the'
          write(*,*) '   configuration factor matrix F has to be one!'
          write(*,'(a,f8.6,a,i3,a)')' Sum =', testsum,' in row'
     @                              ,j,' of matrix'
          indicator = 1
        end if

      end do
```

```
      testsum1 = 0.
      testsum2 = 0.
      do i = 1,n*n
        do j = n*n+1, 2*n*n
          testsum1 = testsum1+F(i,j)
          testsum2 = testsum2+F(j,i)
        end do
      end do

C *** Define reflectivity matrix
      call reflec(R,F,winelem,lda,n,nw,reflg,reflw,reflc,reflf)

C *** Apply matrix factorization to reflectivity matrix with the LINPACK
C     subroutine SGEFA. The contents of R will be changed after use !
      call sgefa(R,lda,6*n*n,ipvt,info)

      if (info.ne.0) then
        write(*,*) ' *** Warning : Error may occur when using'
        write(*,*) '     subrout. SGESL to solve the system of linear'
        write(*,*) '    equations in the interreflection algorythm !'
        write(*,*) '    Check matrices F and R for senseless values,'
        write(*,*) '    f.i. zeros in the diagonal of F.'
      end if

C *** Solve system of linear equations with LINPACK subroutine SGESL.
C     After use, array F will contain the solution, the F-hats.
      do i=1,6*n*n
        do j=1,6*n*n
          b(j) = F(j,i)
        end do

        call sgesl(R,lda,6*n*n,ipvt,b,0)

        do j=1,6*n*n
          F(j,i) = b(j)
        end do
```

```
      end do


C *** Test F-hats matrix for accuracy : The sum of all F-hats from one
C     point to all other surfaces muliplied by their absorptances has
C     to be one.
      ind = 1
      do i = 1,6*n*n

        testsum = 0.
        do j = 1,6*n*n

          do k = 1,n*n
            do kk = 1,nw
              if (winelem(kk,k).eq.j) then
                absorb = 1.-reflg
                goto 11
              end if
            end do
          end do

          if (int(real(j-.5)/(n*n))+1.lt.5) then
            absorb = 1.-reflw
          else if (int(real(j-.5)/(n*n))+1.eq.5) then
            absorb = 1.-reflc
          else
            absorb = 1.-reflf
          end if

 11       testsum = testsum + F(i,j)*absorb

        end do

        if (abs(1.-testsum).gt.1.e-05) then
          write(*,*)
          write(*,*) '    Error : Sum of F-hats times absorptance'
          write(*,*) '            has to be one for each row.'
```

```
        write(*,*) '          row =',i,' and sum =',testsum
        write(*,*)
        ind = 0
      end if
    end do

C *** Determine the numbers of the elements that lie above the workplane
    call elements2(surfelem,n,rh,wkpln)

C *** Calculate the configuration factors for the elements that lie above
C     the workplane.
    call consens(FS,surfelem,n,sw,sl,rh,rw,rl,wkpln)

    testsum = 0.
    do i = 1,5*n*n
      testsum = testsum+FS(i)
    end do

    if (abs(1.-testsum).gt.1.e-05) then
      write(*,*) '  The sum of all configuraion factors from the'
      write(*,*) '  sensor to all elements above the workplane'
      write(*,*) '  in matrix FS has to be one :'
      write(*,*) '  Error : sum =',testsum
    end if


199 continue


C *** Determine the illuminance at the sensor with the F-hats method.
    call light(F,FS,winelem,surfelem,floorbeam,wallbeam,lda,n,lumwin,
   @          reflg,reflw,reflf,reflc,beam1,beam2,beam3,beam4,beam6,
   @           daylight,reflect,plight,dlight,nw,nnw)

    allight = allight+daylight+beamlight
    rlight = rlight+reflect
    patchlight = patchlight+plight
```

```
      dirlight = dirlight+dlight

  888 continue


C *** Write results on file 'FLITE.out':
      open(3,file='FLITE.out',status='new')
      write(3,*) hj-0.5,rlight,dirlight,patchlight,allight

  999 continue


 1000 stop
      end


C *** Subroutine PATCHES determines the location of beam patches on the
C     floor (surface #6), the numbers of the surface elements that lie
C     in the respective area of the floor and calculates the area-weighed
C     illuminance of the patch.

      Subroutine patches(floorbeam,wallbeam,n,sn,rw,rl,rh,wl,wb,ww,wh,
     @                   finl,finr,dfl,dfr,ov,sunalt,sunaz,raz,jul,t0,phase,
     @                   beam1,beam2,beam3,beam4,beam6,sw,sl,wkpln)

      integer floorbeam(n*n), wallbeam(3*n*n), sn

      beam1 = 0.
      beam2 = 0.
      beam3 = 0.
      beam4 = 0.
      beam6 = 0.

      S1 = 0.
      S2 = 0.
      S3 = 0.
      S4 = 0.
```

```
patchl = 0.
R1 = 0.
R2 = 0.
R3 = 0.
R4 = 0.
patchr = 0.
B1 = 0.
B2 = 0.
B3 = 0.
B4 = 0.
patchb = 0.
T1 = 0.
T2 = 0.
T3 = 0.
T4 = 0.
patcht = 0.

do i = 1,3*n*n
  wallbeam(i) = 0.
end do
do i = 1,n*n
  floorbeam(i) = 0.
end do

if ((ov.eq.0.).or.(finr.eq.0.).or.(finl.eq.0)) then
  eps = 1.e-05
else
  eps = 0.
end if

sub = raz-sunaz
if (sub.eq.0.) sub = 1.e-05
if (sub.eq.90.) sub = sub+1.e-05
if (sub.eq.180.) sub = sub+1.e-05
if (sub.eq.-90.) sub = sub+1.e-05

sub1 = sunaz-raz
if (sub1.eq.0.) sub1 = 1.e-05
```

```
      if (sub1.eq.90.) sub1 = sub+1.e-05
      if (sub1.eq.180.) sub1 = sub+1.e-05
      if (sub1.eq.-90.) sub1 = sub+1.e-05

      if (sn.eq.1) then

C *** Check if beam sunlight reaches the floor for window in surface #1
        angler = atand((dfr+ww)/(finr+eps))+raz
        anglel = -atand((dfl+ww)/(finl+eps))+raz

        if (sunalt.lt.(atand((rh-wb)/(ov+eps))) .and.
     @      (sunalt.gt.atand(wb/rl))          .and.
     @      (anglel.lt.sunaz)                 .and.
     @      (angler.gt.sunaz))                then

          if (sunalt.gt.atand((rh-wb-wh)/(ov+eps))) then
            patcht = (rh-ov*tand(sunalt))/tand(sunalt)
          else
            patcht = (wh+wb)/tand(sunalt)
          end if
          if (patcht.gt.rl) patcht = rl

          patchb = wb/tand(sunalt)
          if (patchb.ge.rl) patchb = rl

          patchh = patcht-patchb

          if (sunaz.lt.-atand(dfl/(finl+eps))+raz) then
            patchl = wl-dfl+tand(raz-sunaz)*(finl+(patchb+patcht)/2.)
          else
            patchl = wl+tand(raz-sunaz)*((patcht+patchb)/2.)
          end if
          if (patchl.ge.rw) patchl = rw
          if (patchl.le.0.) patchl = 0.

          if (sunaz.gt.atand(dfr/(finr+eps))+raz) then
            patchr = wl+ww-tand(sunaz-raz)*(finr+(patcht+patchb)/2.)
```

```
          else
           patchr = wl+ww-tand(sunaz-raz)*((patcht+patchb)/2.)
          end if
          if (patchr.gt.rw) patchr = rw
          if (patchr.le.0.) patchr = 0.

          patchw = patchr-patchl

C *** Determine the beam illuminance on the floor due to the window in
C     surface #1.
          beam6 = beam(sunalt,jul)*transmit(raz,sn,sunalt,sunaz,t0)
     @              *sind(sunalt)*phase                          ! [lux]

          if ((patchh.eq.0.) .or. (patchw.eq.0.)) then
            beam6 = 0.
          end if

        end if

C *** Check if beam sunlight reaches suface #2 for window in surface #1
        if ((sunaz.gt.atand(wl/rl)+raz)        .and.
     @      (sunaz.lt.angler)                  .and.
     @      (sunalt.lt.atand((rh-wb)/(ov+eps)))) then

          if (sunaz.lt.atand(dfr/(finr+eps))+raz) then
            S2 = rl-(wl+ww)/tand(sub1)
          else
           S2 = rl-(wl+ww+dfr-finr*tand(sunaz-raz))
     @              /tand(sub1)
          end if
          if (S2.lt.0.) S2 = 0.

          R2 = rl-wl/tand(sub1)

          W2 = R2-S2

          X2 = tand(sunaz-raz)*(rl-(S2+R2)/2.)
          Y2 = sqrt(X2*X2+(rl-(S2+R2)/2.)**2)
```

```
      B2 = wb-tand(sunalt)*Y2
       if (B2.lt.0.) B2 = 0.

      if (sunalt.gt.atand((rh-wb-wh)/(ov+eps))) then
        Z2 = tand(sunalt)*(ov+eps)
        T2 = rh-Z2-tand(sunalt)*Y2
       else
        T2 = wb+wh-tand(sunalt)*Y2
      end if
       if (T2.lt.0.) T2 = 0.

      H2 = T2-B2

C *** Determine the beam illuminance on surface #2 due to the window in
C    surface #1.
       beam2 = beam(sunalt,jul)*transmit(raz,sn,sunalt,sunaz,t0)
   @           *sind(abs(sunaz-raz))*cosd(sunalt)*phase        ! [lux]

       if ((H2.eq.0.) .or. (W2.eq.0.)) then
         beam2 = 0.
       end if

      end if

C *** Check if beam reaches surface #3 from window in surface #1
      X31 = (wl+ww+dfr)/tand(abs(sub))-finr
      Y31a = (wl+ww)/tand(abs(sub))
      Y31b = (rw-wl-ww)/tand(abs(sub))

       if ((X31.gt.rl)                .and.
   @    (sunaz.gt.atand(dfr/(finr+eps))-raz)) then

        S3 = rw-tand(abs(sunaz-raz))*(X31-rl)

      else if ((Y31a.gt.rl) .and. (sunaz-raz.gt.0.)) then
        S3 = rw-(Y31a-rl)*tand(abs(sunaz-raz))
```

```
      else if ((Y31b.gt.rl) .and. (sunaz-raz.lt.0.)) then
        S3 = (Y31b-rl)*tand(abs(sunaz-raz))
      else if ((Y31b.lt.rl) .and. (sunaz-raz.lt.0.)) then
         S3 = 0.
      else if ((Y31a.lt.rl) .and. (sunaz-raz.gt.0.)) then
         S3 = rw
      end if

       if (S3.lt.0.) S3 = 0.
       if (S3.gt.rw) S3 = rw

      X32 = (rw-wl+dfl)/tand(abs(sub))-finl
      Y32a = (rw-wl)/tand(abs(sub))
      Y32b = wl/tand(abs(sub))

       if ((X32.gt.rl)                .and.
     @    (sunaz.lt.-atand(dfl/(finl+eps))-raz)) then

        R3 = tand(abs(raz-sunaz))*(X32-rl)

      else if ((Y32a.gt.rl) .and. (sunaz-raz.lt.0.)) then
        R3 = tand(abs(raz-sunaz))*(Y32a-rl)
      else if ((Y32b.gt.rl) .and. (sunaz-raz.gt.0.)) then
        R3 = rw-tand(abs(raz-sunaz))*(Y32b-rl)
      else if ((Y32b.lt.rl) .and. (sunaz-raz.gt.0.)) then
         R3 = rw
      else if ((Y32a.lt.rl) .and. (sunaz-raz.lt.0.)) then
         R3 = 0.
      end if

       if (R3.lt.0.) R3 = 0.
       if (R3.gt.rw) R3 = rw

      if ((sunaz-raz).eq.0.) then
         S3 = rw-wl-ww
         R3 = rw-wl
      end if
```

```
      W3 = R3-S3

      Z31 = (rl+ov)*tand(sunalt)
      Z32 = rl*tand(sunalt)

      if ((Z31.lt.rh) .and.
     @    (sunalt.gt.atand((rh-wb-wh)/ov))) then

        T3 = rh-Z31

      else if (Z32.lt.wh+wb) then
        T3 = wb+wh-Z32

      end if

      if (T3.lt.0.) T3 = 0.

      B3 = wb-Z32
      if (B3.lt.0.) B3 = 0.

      H3 = T3-B3

C *** Determine the beam illuminance on surface #3 due to the window
C     in surface #1.
      beam3 = beam(sunalt,jul)*transmit(raz,sn,sunalt,sunaz,t0)
     @          *cosd(abs(sunaz-raz))*cosd(sunalt)*phase          ! [lux]

      if ((H3.eq.0.) .or. (W3.eq.0.)) then
        beam3 = 0.
      end if

C *** Check if beam reaches surface #4 from window in surface #1
      if ((sunaz.lt.-atand((rw-wl-ww)/rl)+raz) .and.
     @    (sunaz.gt.anglel)                    .and.
     @    (sunalt.lt.atand((rh-wb)/(ov+eps)))) then

        if (sunaz.gt.-atand(dfl/(finl+eps))+raz) then
```

```
      R4 = (rw-wl)/tand(sub)
     else
      R4 = (rw-wl+dfl-finl*tand(raz-sunaz))/tand(sub)
     end if
     if (R4.gt.rl) R4 = rl

     S4 = (rw-wl-ww)/tand(sub)
     if (S4.gt.rl) S4 = rl

     W4 = R4-S4

     X4 = tand(raz-sunaz)*(S4+R4)/2.
     Y4 = sqrt(X4*X4+((S4+R4)/2.)**2)

     if (sunalt.gt.atand((rh-wb-wh)/(ov+eps))) then
       Z4 = tand(sunalt)*(ov+eps)
       T4 = rh-Z4-tand(sunalt)*Y4
     else
       T4 = wb+wh-tand(sunalt)*Y4
     end if
     if (T4.lt.0.) T4 = 0.

     B4 = wb-tand(sunalt)*Y4
     if (B4.lt.0.) B4 = 0.

     H4 = T4-B4

C *** Determine the beam illuminance on surface #4 due to the window in
C    surface #1.
       beam4 = beam(sunalt,jul)*transmit(raz,sn,sunalt,sunaz,t0)
   @           *sind(abs(sunaz-raz))*cosd(sunalt)*phase          ! [lux]

       if ((H4.eq.0.) .or. (W4.eq.0.)) then
         beam4 = 0.
       end if

     end if
```

```
      else if (sn.eq.2) then

C *** Check if beam sunlight reaches the floor for window in surface #2
       anglel = -atand((dfl+ww)/(finl+eps))+raz-90.
        if (anglel.lt.-180.) then
          anglel = 360.+anglel
        end if

       angler = atand((dfr+ww)/(finr+eps))+raz-90.
        if (angler.lt.-180.) then
          angler = 360.+angler
        end if

       if (sunalt.lt.(atand((rh-wb)/(ov+eps))) .and.
     @      (sunalt.gt.atand(wb/rl))          .and.
     @      (anglel.lt.sunaz)                 .and.
     @      (angler.gt.sunaz))                then

         if (sunalt.gt.atand((rh-wb-wh)/(ov+eps))) then
           patchr = (rh-ov*tand(sunalt))/tand(sunalt)
         else
           patchr = (wh+wb)/tand(sunalt)
         end if
         if (patchr.gt.rw) patchr = rw

         patchl = wb/tand(sunalt)
         if (patchl.ge.rw) patchl = rw

         patchw = patchr-patchl

         angleaz1 = -atand(dfl/(finl+eps))+raz-90.
          if (angleaz1.lt.-180.) then
            anglaz1 = 360.+angleaz1
          end if

          if (sunaz.lt.angleaz1) then
           patcht = rl-wl+dfl-tand(raz-90.-sunaz)*
```

```
@               (finl+(patchl+patchr)/2.)
      else
       patcht = rl-wl-tand(raz-90.-sunaz)*((patchl+patchr)/2.)
      end if
      if (patcht.ge.rl) patcht = rl
      if (patcht.le.0.) patcht = 0.

     angleaz2 = atand(dfr/(finr+eps))+raz-90.
      if (angleaz2.lt.-180.) then
        angleaz2 = 360.+angleaz2
      end if

      if (sunaz.gt.angleaz2) then
       patchb = rl-wl-ww+tand(sunaz-raz+90.)*(finr+
@               (patchr+patchl)/2.)
      else
       patchb = rl-wl-ww+tand(sunaz-raz+90.)*((patchr+patchl)
@               /2.)
      end if
      if (patchb.gt.rl) patchb = rl
      if (patchb.le.0.) patchb = 0.

      patchh = patcht-patchb

C *** Determine the beam illuminance on the floor due to the window in
C    surface #2.
       beam6 = beam(sunalt,jul)*transmit(raz,sn,sunalt,sunaz,t0)
   @           *sind(sunalt)*phase                          ! [lux]

       if ((patchh.eq.0.) .or. (patchw.eq.0.)) then
         beam6 = 0.
       end if

     end if

C *** Check if beam sunlight reaches suface #3 for window in surface #2
      if ((sunaz.gt.atand(wl/rw)+raz-90.)    .and.
```

```
@     (sunaz.lt.angler)                .and.
@     (sunalt.lt.atand((rh-wb)/(ov+eps)))) then

      if (sunaz.lt.atand(dfr/(finr+eps))+raz-90.) then
        S3 = rw-(wl+ww)/tand(sub1+90.)
      else
       S3 = rw-(wl+ww+dfr-finr*tand(sunaz-raz+90.))
@              /tand(sub1+90.)
      end if
      if (S3.lt.0.) S3 = 0.

      R3 = rw-wl/tand(sub1+90.)

      W3 = R3-S3

      X3 = tand(sunaz-raz+90.)*(rw-(S3+R3)/2.)
      Y3 = sqrt(X3*X3+(rw-(S3+R3)/2.)**2)

      B3 = wb-tand(sunalt)*Y3
      if (B3.lt.0.) B3 = 0.

      if (sunalt.gt.atand((rh-wb-wh)/(ov+eps))) then
        Z3 = tand(sunalt)*(ov+eps)
        T3 = rh-Z3-tand(sunalt)*Y3
      else
        T3 = wb+wh-tand(sunalt)*Y3
      end if
      if (T3.lt.0.) T3 = 0.

      H3 = T3-B3

C *** Determine the beam illuminance on surface #3 due to the window in
C     surface #2.
      beam3 = beam(sunalt,jul)*transmit(raz,sn,sunalt,sunaz,t0)
@           *sind(abs(sunaz-raz+90.))*cosd(sunalt)*phase       ! [lux]

      if ((H3.eq.0.) .or. (W3.eq.0.)) then
        beam3 = 0.
```

```
        end if

      end if

C *** Check if beam reaches surface #4 from window in surface #2
      X41 = (wl+ww+dfr)/tand(abs(sub1+90.))-finr
      Y41a = (wl+ww)/tand(abs(sub1+90.))
      Y41b = (rl-wl-ww)/tand(abs(sub1+90.))

      if ((X41.gt.rw)                    .and.
     @    (sunaz.gt.atand(dfr/(finr+eps))-raz+90.)) then

        S4 = rl-tand(abs(sunaz-raz+90.))*(X41-rw)

      else if ((Y41a.gt.rw) .and. (sunaz-raz+90..gt.0.)) then
        S4 = rl-(Y41a-rw)*tand(abs(sunaz-raz+90.))
      else if ((Y41b.gt.rw) .and. (sunaz-raz+90..lt.0.)) then
        S4 = (Y41b-rw)*tand(abs(sunaz-raz+90.))
      else if ((Y41b.lt.rw) .and. (sunaz-raz+90..lt.0.)) then
        S4 = 0.
      else if ((Y41a.lt.rw) .and. (sunaz-raz+90..gt.0.)) then
        S4 = rl
      end if

      if (S4.lt.0.) S4 = 0.
      if (S4.gt.rl) S4 = rl

      X42 = (rl-wl+dfl)/tand(abs(sub1+90.))-finl
      Y42a = (rl-wl)/tand(abs(sub1+90.))
      Y42b = wl/tand(abs(sub1+90.))

      if ((X42.gt.rw)                    .and.
     @    (sunaz.lt.-atand(dfl/(finl+eps))-raz-90.)) then

        R4 = tand(abs(sunaz-raz+90.))*(X42-rw)

      else if ((Y42a.gt.rw) .and. (sunaz-raz+90..lt.0.)) then
```

```
      R4 = tand(abs(sunaz-raz+90.))*(Y42a-rw)
    else if ((Y42b.gt.rw) .and. (sunaz-raz+90..gt.0.)) then
      R4 = rl-tand(abs(sunaz-raz+90.))*(Y42b-rw)
    else if ((Y42b.lt.rw) .and. (sunaz-raz+90..gt.0.)) then
       R4 = rl
    else if ((Y42a.lt.rw) .and. (sunaz-raz+90..lt.0.)) then
       R4 = 0.
    end if

     if (R4.lt.0.) R4 = 0.
     if (R4.gt.rl) R4 = rl

    if ((sunaz-raz+90.).eq.0.) then
       S4 = rl-wl-ww
       R4 = rl-wl
    end if

     W4 = R4-S4

    Z41 = (rw+ov)*tand(sunalt)
    Z42 = rw*tand(sunalt)

     if ((Z41.lt.rh) .and.
@       (sunalt.gt.atand((rh-wb-wh)/ov))) then

       T4 = rh-Z41

     else if (Z42.lt.wh+wb) then
       T4 = wb+wh-Z42
     end if

     if (T4.lt.0.) T4 = 0.

     B4 = wb-Z42
     if (B4.lt.0.) B4 = 0.

     H4 = T4-B4
```

```
C *** Determine the beam illuminance on surface #4 due to the window
C    in surface #2.
      beam4 = beam(sunalt,jul)*transmit(raz,sn,sunalt,sunaz,t0)
     @          *cosd(abs(sunaz-raz+90.))*cosd(sunalt)*phase          ! [lux]

      if ((H4.eq.0.) .or. (W4.eq.0.)) then
        beam4 = 0.
      end if

C *** Check if beam reaches surface #1 from window in surface #2
      if ((sunaz.lt.-atand((rl-wl-ww)/rw)+raz-90.) .and.
     @     (sunaz.gt.anglel)                       .and.
     @     (sunalt.lt.atand((rh-wb)/(ov+eps))))     then

        if (sunaz.gt.-atand(dfl/(finl+eps))+raz-90.) then
          R1 = (rl-wl)/tand(sub-90.)
        else
          R1 = (rl-wl+dfl-finl*tand(raz-sunaz-90.))
     @           /tand(sub-90.)
        end if
        if (R1.gt.rw) R1 = rw

        S1 = (rl-wl-ww)/tand(sub-90.)
        if (S1.gt.rw) S1 = rw

        W1 = R1-S1

        X1 = tand(raz-sunaz-90.)*(S1+R1)/2.
        Y1 = sqrt(X1*X1+((S1+R1)/2.)**2)

        if (sunalt.gt.atand((rh-wb-wh)/(ov+eps))) then
          Z1 = tand(sunalt)*(ov+eps)
          T1 = rh-Z1-tand(sunalt)*Y1
        else
          T1 = wb+wh-tand(sunalt)*Y1
        end if
        if (T1.lt.0.) T1 = 0.
```

```
      B1 = wb-tand(sunalt)*Y1
       if (B1.lt.0.) B1 = 0.

      H1 = T1-B1

C *** Determine the beam illuminance on surface #1 due to the window in
C     surface #2.
        beam1 = beam(sunalt,jul)*transmit(raz,sn,sunalt,sunaz,t0)
     @          *sind(abs(sunaz-raz+90.))*cosd(sunalt)*phase       ! [lux]

        if ((H1.eq.0.) .or. (W1.eq.0.)) then
          beam1 = 0.
        end if

      end if


C *** Check if beam sunlight reaches the floor for window in surface #3
    else if (sn.eq.3) then

      anglel = -atand((dfl+ww)/(finl+eps))+raz-180.
       if (anglel.lt.-180.) then
         anglel = 360.+anglel
       end if

      angler = atand((dfr+ww)/(finr+eps))+raz-180.
       if (angler.lt.-180.) then
         angler = 360.+angler
       end if

       if (sunaz.le.0.) then

        if (sunalt.lt.(atand((rh-wb)/(ov+eps))) .and.
     @      (sunalt.gt.atand(wb/rl))          .and.
     @      (angler.gt.sunaz))                then

          ind = 1
```

```
      else
        ind = 0
      end if

    else

      if (sunalt.lt.(atand((rh-wb)/(ov+eps))) .and.
     @      (sunalt.gt.atand(wb/rl))          .and.
     @      (anglel.lt.sunaz))                then

        ind = 1
      else
        ind = 0
      end if

    end if

    if (ind.eq.1) then

      if (sunalt.gt.atand((rh-wb-wh)/(ov+eps))) then
        patchb = rl-(rh-ov*tand(sunalt))/tand(sunalt)
      else
        patchb = rl-(wh+wb)/tand(sunalt)
      end if
      if (patchb.lt.0.) patchb = 0.

      patcht = rl-wb/tand(sunalt)
      if (patcht.le.0.) patcht = 0.

      patchh = patcht-patchb

      angleaz1 = -atand(dfl/(finl+eps))+raz-180.
      if (angleaz1.lt.-180.) then
        anglaz1 = 360.+angleaz1
      end if

      if (sunaz.lt.angleaz1) then
```

```
          patchr = rw-wl+dfl-tand(raz-180.-sunaz)*
     @              (finl+(patcht+patchb)/2.)
          else
          patchr = rw-wl-tand(raz-180.-sunaz)*((patcht+patchb)/2.)
          end if
          if (patchr.ge.rw) patcht = rw
          if (patchr.le.0.) patcht = 0.

          angleaz2 = atand(dfr/(finr+eps))+raz-180.
          if (angleaz2.lt.-180.) then
            angleaz2 = 360.+angleaz2
          end if

          if (sunaz.gt.angleaz2) then
           patchl = rw-wl-ww+tand(sunaz-raz+180.)*(finr+
     @              (patcht+patchb)/2.)
          else
           patchl = rw-wl-ww+tand(sunaz-raz+180.)*((patcht+patchb)
     @              /2.)
          end if
          if (patchl.gt.rw) patchl = rw
          if (patchl.le.0.) patchl = 0.

          patchw = patchr-patchl

C *** Determine the beam illuminance on the floor due to the window in
C     surface #3.
          beam6 = beam(sunalt,jul)*transmit(raz,sn,sunalt,sunaz,t0)
     @            *sind(sunalt)*phase                          ! [lux]

          if ((patchh.eq.0.) .or. (patchw.eq.0.)) then
            beam6 = 0.
          end if

        end if

C *** Check if beam sunlight reaches suface #4 for window in surface #3
```

```
      if ((sunaz.gt.atand(wl/rl)+raz-180.)    .and.
     @     (sunaz.lt.angler)                  .and.
     @     (sunaz.lt.0.)                      .and.
     @     (sunalt.lt.atand((rh-wb)/(ov+eps)))) then

        if (sunaz.lt.atand(dfr/(finr+eps))+raz-180.) then
          S4 = rl-(wl+ww)/tand(sub1+180.)
        else
          S4 = rl-(wl+ww+dfr-finr*tand(sunaz-raz+180.))
     @              /tand(sub1+180.)
        end if
        if (S4.lt.0.) S4 = 0.
        if (S4.gt.rl) S4 = rl

        R4 = rl-wl/tand(sub1+180.)
        if (R4.lt.0.) R4 = 0.
        if (R4.gt.rl) R4 = rl

        W4 = R4-S4

        X4 = tand(sub1+180.)*(rl-(S4+R4)/2.)
        Y4 = sqrt(X4*X4+(rl-(S4+R4)/2.)**2)

        B4 = wb-tand(sunalt)*Y4
        if (B4.lt.0.) B4 = 0.

        if (sunalt.gt.atand((rh-wb-wh)/(ov+eps))) then
          Z4 = tand(sunalt)*(ov+eps)
          T4 = rh-Z4-tand(sunalt)*Y4
        else
          T4 = wb+wh-tand(sunalt)*Y4
        end if
        if (T4.lt.0.) T4 = 0.

        H4 = T4-B4

C *** Determine the beam illuminance on surface #4 due to the window in
C     surface #3.
```

```
      beam4 = beam(sunalt,jul)*transmit(raz,sn,sunalt,sunaz,t0)
     @          *sind(abs(sunaz-raz+180.))*cosd(sunalt)*phase      ! [lux]

      if ((H4.eq.0.) .or. (W4.eq.0.)) then
        beam4 = 0.
      end if

     end if

C *** Check if beam reaches surface #1 from window in surface #3
      if (sunaz.lt.0.) then
        zmul = 1.
      else
        zmul = -1.
      end if

     X11 = (wl+ww+dfr)/tand(abs(sub1+(180.*zmul)))-finr
     Y11a = (wl+ww)/tand(abs(sub1+(180.*zmul)))
     Y11b = (rw-wl-ww)/tand(abs(sub1+(180.*zmul)))


      if (sunaz.lt.0.) then
        if ((X11.gt.rl)                    .and.
     @     (sunaz.gt.atand(dfr/(finr+eps))-raz+180.)) then

          S1 = rw-tand(abs(sunaz-raz+180.))*(X11-rw)
        end if
      end if

     if ((Y11a.gt.rl) .and. (sunaz-raz+(180.*zmul).gt.0.)) then
       S1 = rw-(Y11a-rl)*tand(abs(sunaz-raz+(180.*zmul)))
     else if ((Y11b.gt.rl) .and. (sunaz-raz+(180.*zmul).lt.0.)) then
       S1 = (Y11b-rl)*tand(abs(sunaz-raz+(180.*zmul)))
     else if ((Y11b.lt.rl) .and. (sunaz-raz+(180.*zmul).lt.0.)) then
        S1 = 0.
     else if ((Y11a.lt.rl) .and. (sunaz-raz+(180.*zmul).gt.0.)) then
        S1 = rw
```

```
 end if

 if (S1.lt.0.) S1 = 0.
 if (S1.gt.rw) S1 = rw

X12 = (rw-wl+dfl)/tand(abs(sub1+(180.*zmul)))-finl
Y12a = (rw-wl)/tand(abs(sub1+(180.*zmul)))
Y12b = wl/tand(abs(sub1+(180.*zmul)))

 if (sunaz.gt.0.) then
   if ((X12.gt.rl)                    .and.
@      (sunaz.lt.-atand(dfl/(finl+eps))-raz+180.)) then

    R1 = tand(abs(sunaz-raz+180.))*(X12-rl)
   end if
 end if

 if ((Y12a.gt.rl) .and. (sunaz-raz+(180.*zmul).lt.0.)) then
  R1 = tand(abs(sunaz-raz+(180.*zmul)))*(Y12a-rl)
 else if ((Y12b.gt.rl) .and. (sunaz-raz+(180.*zmul).gt.0.)) then
  R1 = rw-tand(abs(sunaz-raz+(180.*zmul)))*(Y12b-rl)
 else if ((Y12b.lt.rl) .and. (sunaz-raz+(180.*zmul).gt.0.)) then
   R1 = rw
 else if ((Y12a.lt.rl) .and. (sunaz-raz+(180.*zmul).lt.0.)) then
   R1 = 0.
 end if

 if (R1.lt.0.) R1 = 0.
 if (R1.gt.rw) R1 = rw

if ((sunaz-raz+(180.*zmul)).eq.0.) then
   S1 = rw-wl-ww
   R1 = rw-wl
 end if

 W1 = R1-S1

Z11 = (rl+ov)*tand(sunalt)
```

```
      Z12 = rl*tand(sunalt)

      if ((Z11.lt.rh) .and.
     @    (sunalt.gt.atand((rh-wb-wh)/ov))) then

        T1 = rh-Z11

      else if (Z12.lt.wh+wb) then
        T1 = wb+wh-Z12
      end if

      if (T1.lt.0.) T1 = 0.

      B1 = wb-Z12
      if (B1.lt.0.) B1 = 0.

      H1 = T1-B1

C *** Determine the beam illuminance on surface #1 due to the window
C     in surface #3.
      beam1 = beam(sunalt,jul)*transmit(raz,sn,sunalt,sunaz,t0)
     @        *cosd(abs(sunaz-raz+180.))*cosd(sunalt)*phase        ! [lux]

      if ((H1.eq.0.) .or. (W1.eq.0.)) then
        beam1 = 0.
      end if

C *** Check if beam reaches surface #2 from window in surface #3
      if ((sunaz.gt.-atand((rw-wl-ww)/rl)+raz-180.) .and.
     @    (sunaz.gt.0.)                             .and.
     @    (sunaz.gt.anglel)                         .and.
     @    (sunalt.lt.atand((rh-wb)/(ov+eps))))      then

        anglefin = -atand(dfl/(finl+eps))+raz-180.
        if (anglefin.lt.-180.) anglefin = 360.+anglefin

        if (sunaz.gt.anglefin) then
```

```
        R2 = (rw-wl)/tand(sub-180.)
       else
        R2 = (rw-wl+dfl-finl*tand(raz-sunaz-180.))
     @          /tand(sub-180.)
       end if
       if (R2.gt.rl) R2 = rl
       if (R2.lt.0.) R2 = 0.

      S2 = (rw-wl-ww)/tand(sub-180.)
       if (S2.gt.rl) S2 = rl
       if (S2.lt.0.) S2 = 0.

       W2 = R2-S2

      X2 = tand(raz-sunaz-180.)*(S2+R2)/2.
      Y2 = sqrt(X2*X2+((S2+R2)/2.)**2)

      if (sunalt.gt.atand((rh-wb-wh)/(ov+eps))) then
        Z2 = tand(sunalt)*(ov+eps)
        T2 = rh-Z2-tand(sunalt)*Y2
       else
        T2 = wb+wh-tand(sunalt)*Y2
       end if
       if (T2.lt.0.) T2 = 0.

      B2 = wb-tand(sunalt)*Y2
       if (B2.lt.0.) B2 = 0.

      H2 = T2-B2

C *** Determine the beam illuminance on surface #2 due to the window in
C    surface #3
       beam2 = beam(sunalt,jul)*transmit(raz,sn,sunalt,sunaz,t0)
   @          *sind(abs(sunaz-raz-180.))*cosd(sunalt)*phase      ! [lux]

      if ((H2.eq.0.) .or. (W2.eq.0.)) then
        beam2 = 0.
      end if
```

```
      end if


C *** Check if beam sunlight reaches the floor for window in surface #4
      else if(sn.eq.4) then

       anglel = -atand((dfl+ww)/(finl+eps))+raz+90.
        if (anglel.gt.180.) then
          anglel = anglel-360.
        end if

       angler = atand((dfr+ww)/(finr+eps))+raz+90.
        if (angler.gt.180.) then
          angler = angler-360.
        end if

       if (sunalt.lt.(atand((rh-wb)/(ov+eps))) .and.
     @      (sunalt.gt.atand(wb/rl))          .and.
     @      (anglel.lt.sunaz)                 .and.
     @      (angler.gt.sunaz))                then

         if (sunalt.gt.atand((rh-wb-wh)/(ov+eps))) then
           patchl = rw-(rh-ov*tand(sunalt))/tand(sunalt)
         else
           patchl = rw-((wh+wb)/tand(sunalt))
         end if
         if (patchl.lt.0.) patchl = 0.

        patchr = rw-wb/tand(sunalt)
         if (patchr.le.0.) patchr = 0.

        patchw = patchr-patchl

        angleaz1 = -atand(dfl/(finl+eps))+raz+90.
         if (angleaz1.gt.180.) then
           anglaz1 = angleaz1-360.
```

```
        end if

        if (sunaz.lt.angleaz1) then
          patchb = wl-dfl+tand(raz+90.-sunaz)*
     @              (finl+(patchl+patchr)/2.)
        else
          patchb = wl+tand(raz+90.-sunaz)*((patchl+patchr)/2.)
        end if
        if (patchb.ge.rl) patchb = rl
        if (patchb.le.0.) patchb = 0.

       angleaz2 = atand(dfr/(finr+eps))+raz+90.
        if (angleaz2.gt.180.) then
          angleaz2 = angleaz2-360.
        end if

        if (sunaz.gt.angleaz2) then
          patcht = wl+ww-tand(sunaz-raz-90.)*(finr+
     @              (patchr+patchl)/2.)
        else
          patcht = wl+ww-tand(sunaz-raz-90.)*((patchr+patchl)
     @              /2.)
        end if
        if (patcht.gt.rl) patcht = rl
        if (patcht.le.0.) patcht = 0.

        patchh = patcht-patchb

C *** Determine the beam illuminance on the floor due to the window in
C     surface #4.
        beam6 = beam(sunalt,jul)*transmit(raz,sn,sunalt,sunaz,t0)
     @          *sind(sunalt)*phase                    ! [lux]

        if ((patchh.eq.0.) .or. (patchw.eq.0.)) then
          beam6 = 0.
        end if

      end if
```

```
C *** Check if beam sunlight reaches suface #1 for window in surface #4
      if ((sunaz.gt.atand(wl/rw)+raz+90.)     .and.
     @     (sunaz.lt.angler)                  .and.
     @     (sunalt.lt.atand((rh-wb)/(ov+eps)))) then

        if (sunaz.lt.atand(dfr/(finr+eps))+raz+90.) then
          S1 = rw-(wl+ww)/tand(sub1-90.)
        else
          S1 = rw-(wl+ww+dfr-finr*tand(sunaz-raz-90.))
     @               /tand(sub1-90.)
        end if
        if (S1.lt.0.) S1 = 0.

        R1 = rw-wl/tand(sub1-90.)

        W1 = R1-S1

        X1 = tand(sunaz-raz-90.)*(rw-(S1+R1)/2.)
        Y1 = sqrt(X1*X1+(rw-(S1+R1)/2.)**2)

        B1 = wb-tand(sunalt)*Y1
        if (B1.lt.0.) B1 = 0.

        if (sunalt.gt.atand((rh-wb-wh)/(ov+eps))) then
          Z1 = tand(sunalt)*(ov+eps)
          T1 = rh-Z1-tand(sunalt)*Y1
        else
          T1 = wb+wh-tand(sunalt)*Y1
        end if
        if (T1.lt.0.) T1 = 0.

        H1 = T1-B1

C *** Determine the beam illuminance on surface #1 due to the window in
C     surface #4.
        beam1 = beam(sunalt,jul)*transmit(raz,sn,sunalt,sunaz,t0)
```

```
@           *sind(abs(sunaz-raz-90.))*cosd(sunalt)*phase      ! [lux]

      if ((H1.eq.0.) .or. (W1.eq.0.)) then
        beam1 = 0.
      end if

    end if


C *** Check if beam reaches surface #2 from window in surface #4
    X21 = (wl+ww+dfr)/tand(abs(sub1-90.))-finr
    Y21a = (wl+ww)/tand(abs(sub1-90.))
    Y21b = (rl-wl-ww)/tand(abs(sub1-90.))

    if ((X21.gt.rw)                    .and.
   @    (sunaz.gt.atand(dfr/(finr+eps))-raz+90.)) then

     S2 = rl-tand(abs(sunaz-raz-90.))*(X21-rw)

    else if ((Y21a.gt.rw) .and. (sunaz-raz-90..gt.0.)) then
     S2 = rl-(Y21a-rw)*tand(abs(sunaz-raz-90.))
    else if ((Y21b.gt.rw) .and. (sunaz-raz-90..lt.0.)) then
     S2 = (Y21b-rw)*tand(abs(sunaz-raz-90.))
    else if ((Y21b.lt.rw) .and. (sunaz-raz-90..lt.0.)) then
      S2 = 0.
    else if ((Y21a.lt.rw) .and. (sunaz-raz-90..gt.0.)) then
      S2 = rl
    end if

    if (S2.lt.0.) S2 = 0.
    if (S2.gt.rl) S2 = rl

   X22 = (rl-wl+dfl)/tand(abs(sub1-90.))-finl
   Y22a = (rl-wl)/tand(abs(sub1-90.))
   Y22b = wl/tand(abs(sub1-90.))

    if ((X22.gt.rw)                    .and.
```

```
@     (sunaz.lt.-atand(dfl/(finl+eps))-raz-90.)) then

      R2 = tand(abs(sunaz-raz-90.))*(X22-rw)

    else if ((Y22a.gt.rw) .and. (sunaz-raz-90..lt.0.)) then
      R2 = tand(abs(sunaz-raz-90.))*(Y22a-rw)
    else if ((Y22b.gt.rw) .and. (sunaz-raz-90..gt.0.)) then
      R2 = rl-tand(abs(sunaz-raz-90.))*(Y22b-rw)
    else if ((Y22b.lt.rw) .and. (sunaz-raz-90..gt.0.)) then
       R2 = rl
    else if ((Y22a.lt.rw) .and. (sunaz-raz-90..lt.0.)) then
       R2 = 0.
    end if

     if (R2.lt.0.) R2 = 0.
     if (R2.gt.rl) R2 = rl

    if ((sunaz-raz-90.).eq.0.) then
       S2 = rl-wl-ww
       R2 = rl-wl
    end if

     W2 = R2-S2

    Z21 = (rw+ov)*tand(sunalt)
    Z22 = rw*tand(sunalt)

    if ((Z21.lt.rh) .and.
@     (sunalt.gt.atand((rh-wb-wh)/ov))) then

      T2 = rh-Z21

    else if (Z22.lt.wh+wb) then
      T2 = wb+wh-Z22
    end if

    if (T2.lt.0.) T2 = 0.
```

```
      B2 = wb-Z22
      if (B2.lt.0.) B2 = 0.

      H2 = T2-B2

C *** Determine the beam illuminance on surface #2 due to the window
C     in surface #4.
      beam2 = beam(sunalt,jul)*transmit(raz,sn,sunalt,sunaz,t0)
     @          *cosd(abs(sunaz-raz-90.))*cosd(sunalt)*phase          ! [lux]

      if ((H2.eq.0.) .or. (W2.eq.0.)) then
        beam2 = 0.
      end if

C *** Check if beam reaches surface #3 from window in surface #4
      if ((sunaz.lt.-atand((rl-wl-ww)/rw)+raz+90.) .and.
     @     (sunaz.gt.anglel)                       .and.
     @     (sunalt.lt.atand((rh-wb)/(ov+eps))))     then

        if (sunaz.gt.-atand(dfl/(finl+eps))+raz+90.) then
          R3 = (rl-wl)/tand(sub+90.)
        else
          R3 = (rl-wl+dfl-finl*tand(raz-sunaz+90.))
     @            /tand(sub+90.)
        end if
        if (R3.gt.rw) R3 = rw

        S3 = (rl-wl-ww)/tand(sub+90.)
        if (S3.gt.rw) S3 = rw

        W3 = R3-S3

        X3 = tand(raz-sunaz+90.)*(S3+R3)/2.
        Y3 = sqrt(X3*X3+((S3+R3)/2.)**2)

        if (sunalt.gt.atand((rh-wb-wh)/(ov+eps))) then
          Z3 = tand(sunalt)*(ov+eps)
```

```
          T3 = rh-Z3-tand(sunalt)*Y3
         else
          T3 = wb+wh-tand(sunalt)*Y3
         end if
         if (T3.lt.0.) T3 = 0.

        B3 = wb-tand(sunalt)*Y3
         if (B3.lt.0.) B3 = 0.

        H3 = T3-B3
C *** Determine the beam illuminance on surface #3 due to the window in
C    surface #4.
        beam3 = beam(sunalt,jul)*transmit(raz,sn,sunalt,sunaz,t0)
   @           *sind(abs(sunaz-raz-90.))*cosd(sunalt)*phase      ! [lux]

        if ((H3.eq.0.) .or. (W3.eq.0.)) then
          beam3 = 0.
        end if

       end if

     end if


C *** Determine the numbers of the elements that lie in the beam patch
C     on surface #1.
     if (beam1.ne.0.) then
       call elements3(wallbeam,n,l,rh,B1,H1,W1,rw,S1)

C *** Calculate the weighted illuminance of the patch of beam on surface #1
C    to correct for differences in the areas of the actual patch and
C    the area of the elements that lie in the patch location.
       k = 0
       do i=1,n*n
        if(wallbeam(i).eq.0) then
           goto 11
        else
           k = k+1
```

```
        end if
      end do

11    if (k.eq.0) then
        pweigh = 0.
      else
        pweigh = (W1*H1)/(k*rh/n*rw/n)
      end if

      beam1 = beam1*pweigh

    end if

C *** Determine the numbers of the elements that lie in the beam patch
C    on surface #2.
    if (beam2.ne.0.) then
      call elements3(wallbeam,n,2,rh,B2,H2,W2,rl,S2)

C *** Calculate the weighed illuminance of the patch of beam on surface #2
C    to correct for differences in the areas of the actual patch and
C    the area of the elements that lie in the patch location.
      k = 0
      do i=1,n*n
        if(wallbeam(i).eq.0) then
          goto 12
        else
          k = k+1
        end if
      end do

12    if (k.eq.0) then
        pweigh = 0.
      else
        pweigh = (W2*H2)/(k*rh/n*rl/n)
      end if

      beam2 = beam2*pweigh
```

```
      end if


C *** Determine the numbers of the elements that lie in the beam patch
C     on surface #3.
      if (beam3.ne.0.) then
        call elements3(wallbeam,n,3,rh,B3,H3,W3,rw,S3)

C *** Calculate the weighed illuminance of the patch of beam on surface #3
C     to correct for differences in the areas of the actual patch and
C     the area of the elements that lie in the patch location.
        k = 0
        do i=1,n*n
          if(wallbeam(i).eq.0) then
             goto 13
          else
             k = k+1
          end if
        end do

 13     if (k.eq.0) then
          pweigh = 0.
        else
          pweigh = (W3*H3)/(k*rh/n*rw/n)
        end if

        beam3 = beam3*pweigh

      end if

C *** Determine the numbers of the elements that lie in the beam patch
C     on surface #4.
      if (beam4.ne.0.) then
        call elements3(wallbeam,n,4,rh,B4,H4,W4,rl,S4)

C *** Calculate the weighed illuminance of the patch of beam on surface #4
C     to correct for differences in the areas of the actual patch and
```

```
C     the area of the elements that lie in the patch location.
      k = 0
      do i=1,n*n
        if(wallbeam(i).eq.0) then
           goto 14
        else
           k = k+1
        end if
      end do

 14    if (k.eq.0) then
        pweigh = 0.
      else
        pweigh = (W4*H4)/(k*rh/n*rl/n)
      end if

      beam4 = beam4*pweigh

    end if

C *** Determine the numbers of the elements that lie in the beam patch
C     on the floor.
    if (beam6.ne.0.) then
     call elements3(floorbeam,n,6,rl,patchb,patchh,patchw,rw,patchl)

C *** Calculate the weighed illuminance of the patch of beam on the floor
C     to correct for differences in the areas of the actual patch and
C     the areas of the elements that lie in the patch location.
      k = 0
      do i=1,n*n
        if(floorbeam(i).eq.0) then
           goto 16
        else
           k = k+1
        end if
      end do
```

```
16    if (k.eq.0) then
        pweigh = 0.
      else
        pweigh = (patchw*patchh)/(k*rw/n*rl/n)
      end if

      beam6 = beam6*pweigh

    end if

    return
    end


C *** Subroutine CONFIG determines the coordinates for function CFW
C     and calculates the configuration factors by using function CFW
C     and taking advantage of symmetric configurations

    Subroutine config(F,xr,yr,zr,lda,n,rw,rl,rh)

    real xr(6*n*n), yr(6*n*n), zr(6*n*n), F(lda,6*n*n)

C *** The configuration factors between points and areas on the same
C     wall (= surface) are zero.
    do k=1,6
      do i=(k-1)*n*n+1,k*n*n
        do j=(k-1)*n*n+1,k*n*n
          F(i,j) = 0.
        end do
      end do
    end do

C *** determine the coordinates for function CFW and calculate the
C     configuration factors from n*n points on surface #1 to each of
C     the n*n surface elements on surface #2, from the points on surface
C     #2 to elements on surface #3, from #3 to #4 and from #4 to #1.
C
```

```
C    define height of elements and room dimension in y-direction
     rr = rh/n
     yy = rh

C *** me = # of emitting surface, which is the surface with the point sources.
C    define width of elements and room dimensions in x- and z-direction.
     do 10 me = 1,4

       if ((me.eq.1) .or. (me.eq.3)) then
         xx = rw
         zz = rl
         tt = rl/n
       else
         xx = rl
         zz = rw
         tt = rw/n
       end if

C *** calculate x- and z-distance from left lower corner of each element on
C    receiving surface to each point source (the n*n point sources are
C    assumed to lie in the center of the n*n elements on the emitting
C    surface). The arrangement of source to element is according to the
C    illustration in : 'The Derivation of a New Area-Source Equation' by
C    W. Pierpoint and J. Hopkins, journal of the IES, 4/1984.
       do i=1,n
         xr(i) = (i*2.-1.)*xx/(2*n)
         zr(i) = (i-1.)*zz/n
       end do

       i  = 0
       jj = 0

       mm = me
       if (me.eq.4) then
         mm = 0
       end if

C *** do 11 : counter for elements on emitting surface. The elements are
```

```
C    numbered with the smallest number in the upper left corner of the
C    wall (as seen from the inside of the room) and have increasing numbers
C    by one with respect to the rows. The smallest number of a surface is
C    (sn-1)*n*n+1, where sn is the surface number. Below is an example for
C    surface #3 and n=3 :
C                        ---------------
C                         | 19 | 20 | 21 |
C                        ---------------
C                         | 22 | 23 | 24 |
C                        ---------------
C                         | 25 | 26 | 27 |
C                        ---------------
      do 11 l = (me-1)*n*n+1,me*n*n
        i = i+1

        if (i.eq.(n+1)) then
          i=1
        end if

C *** do 12 : counter for elements on receiving surface
        k = 0
        m = n+1
        do 12 j=mm*n*n+1,(mm+1)*n*n
          m = m-1

          if (j.eq.(k*n+mm*n*n+1)) then
            k = k+1
          end if

          if (l.eq.(jj*n+(me-1)*n*n+1)) then
            jj = jj+1

C *** determine distance from lower left corner of element on receiving
C    surface to each point in one row on emitting surface in y-direction.
            do kk=1,n
              yr(kk) = (2.*kk-1.)*yy/(2.*n)-(jj-1.)*yy/n
            end do
```

```
            end if

            if (m.eq.0) then
              m = n
            end if

C *** calculate the configuration factor
            F(l,j) = cfw(xr(i),yr(k),zr(m),rr,tt,90.)

 12       continue
 11     continue
 10  continue

C *** Configuration factors from surface #1 to surface #2 are the same for
C    the respective elements as from surface #1 to surface #4. The same holds
C    for sn #2/#3 and #2/#1, sn #3/#4 and  #3/#2 and sn #4/#1 and #4/#3.
C
C    do 20 : counter for # of emitting surface
     do 20 me = 1,4

        mr1 = me
        if(mr1.eq.4) then
          mr1 = 0
        end if

        mr2 = me+2
        if(mr2.ge.4) then
          mr2= mr2-4
        end if

C *** do 21 : counter for elements on emitting surface.
C    k : counter representing decrement to be deducted from
C    the largest (=most right) element # of each row on 'old'
C    receiving surface.
C    kk : counter representing decrement to be deducted from the
C    largest element of each row of emitting surface.
        kk = 0
        mm = 1
```

```
      do 21 j = (me-1)*n*n+1,me*n*n
        k = 0
        m = 1

        if (kk.eq.n) then
          kk = 0
          mm = mm+1
        end if

C *** i : counter representing increment to be added to smallest element #
C    on 'new'receiving surface
        do i=0,n*n-1

          if (k.eq.n) then
            k = 0
            m = m+1
          end if

          F(mm*n+(me-1)*n*n-kk,mr2*n*n+1+i) =  F(j,m*n+mr1*n*n-k)

          k = k+1
        end do

        kk = kk+1
 21    continue
 20  continue

C *** Calculate configuration factors between surface #1 (surface with point
C    sources in the center of each of the n*n elements) and surface #5
C    (= ceiling devided in n*n elements). Imagine you stand in the center
C    of the room facing surface #1 and look upward : The elemnt with the
C    lowest number, which is 4*n*n+1, will be in the upper left corner of
C    the ceiling. The numbers increase by one going in the direcion of rows
C    parallel to those on surface #1, so the element with the largest number,
C    which is 5*n*n, will be in the lower right corner, having a common edge
C    with element #n on surface #1.
C    define coordinates in x- and z-direction
```

```
      do i=1,n
        xr(i) = (i*2.-1.)*rh/(2.*n)
      end do

      do i=1,n
        zr(i) = (i-1.)*rl/n
      end do

C *** do 30 : counter for elements on surface #1.
C     nx indicates, where the appropriate x-dimension in array
C     xr(n) is stored.
      nx = 1
      ml = n
      do 30 l = 1,n*n

C *** The x-dimension changes with each row on surface #1
        if (l.eq.nx*n+1) then
          nx = nx+1
        end if

        if (ml.eq.0) then
          ml = n
        end if

C *** nz indicates, where the appropriate z-dimension in array
C     zr(n) is stored.
        nz = n
        nk = 1
        mk = n+1

C *** counter for elements on surface #5 (= ceiling).
C     The z-dimension changes with each row on surface #5.
        do k=4*n*n+1,5*n*n

          if (k.eq.4*n*n+nk*n+1) then
            nz = nz-1
            nk = nk+1
          end if
```

```
C *** calculate coordinate in y-direction and configuration factor F(l,k)
        y5 = (2.*mk-1.)*rw/(2.*n)-ml*rw/n

        mk = mk-1
        if (mk.eq.1) then
          mk = n+1
        end if

        F(l,k) = cfw(xr(nx),y5,zr(nz),rw/n,rl/n,90.)

      end do

      ml = ml-1
 30  continue

C *** Determine the configuration factors between the elements
C    on surface #2 (point sources) and surface #5 (= ceiling).
C    The numbering of the elements is described above.
C    do 40 : counter for elements on surface #2. The array xr(n)
C    from above does not change. zr(n) has to be defined again.
C    nx indicates, where the appropriate x-dimension in array
C    xr(n) is stored.
     do i=1,n
       zr(i) = (i-1.)*rw/n
     end do

     nx = 1
     ml = n
     do 40 l = n*n+1,2*n*n

C *** The x-dimension changes with each row on surface #2
       if (l.eq.nx*n+n*n+1) then
         nx = nx+1
       end if

       if (ml.eq.0) then
```

```
          ml = n
        end if

C *** nz indicates, where the appropriate z-dimension in array
C     zr(n) is stored.
C     i : Counter for rows on surface #5 (= ceiling) with rows parallel
C     to rows on surface #2.
        mk = n+1
        nz = n
        do i=0,(n-1)
          ii = i

C *** counter for elements on surface #5 (= ceiling).
C     The z-dimension changes with each row on surface #5.The rows on
C     surface #5 are parallel to the rows on surface #2. Calculate
C     coordinate in y-direction and configuration factor F(l,k).
          do k=4*n*n+n-ii,4*n*n+n-ii+(n-1)*n,n

            y5 = (2.*mk-1.)*rl/(2.*n)-ml*rl/n

            mk = mk-1
            if (mk.eq.1) then
              mk = n+1
            end if

            F(l,k) = cfw(xr(nx),y5,zr(nz),rl/n,rw/n,90.)

          end do

          nz = nz-1
        end do

        ml = ml-1
 40   continue

C *** Determine configuration factors between points on surface #5 and
C     elements on surface #1.
C     Calculate x- and z-distance from left lower corner of each element on
```

```
C    surface #1 to each point source on surface #5.
     do i=1,n
       xr(i) = (i*2.-1.)*rl/(2*n)
       zr(i) = (i-1.)*rh/n
     end do

     i = 0
     m = 0

C *** do 50 : counter for elements on emitting surface.
     do 50 l = 4*n*n+1,5*n*n
       i = i+1
       if (i.eq.(n+1)) then
         i = 1
         m = m+1
       end if

C *** determine distance from lower left corner of element on receiving
C    surface to each point in one row on emitting surface in y-direction.
       do kk=1,n
         yr(kk) = (2.*kk-1.)*rw/(2.*n)-(i-1.)*rw/n
       end do

C *** do 51 : Counter for elements on receiving surface
       k = 0
       mm = 0
       do 51 j = 1,n*n

         mm = mm+1
         if (j.eq.(k*n+1)) then
           k = k+1
           mm = 1
         end if

         F(l,j) = cfw(xr(n-m),yr(mm),zr(k),rw/n,rh/n,90.)

 51    continue
```

```
 50 continue

C *** Determine configuration factors between points on surface #5 and
C     elements on surface #2.
C     Calculate x- and z-distance from left lower corner of each element on
C     surface #2 to each point source on surface #5. Array zr(i) is the same
C     as above.
      do i=1,n
        xr(i) = (i*2.-1.)*rw/(2*n)
      end do

      i = 0
      m = 1

C *** do 60 : counter for elements on emitting surface.
      do 60 l = 4*n*n+1,5*n*n
        i = i+1
        if (i.eq.(n+1)) then
          i = 1
          m = m+1
        end if

C *** determine distance from lower left corner of element on receiving
C     surface to each point in one row on emitting surface in y-direction.
        do kk=1,n
          yr(kk) = (2.*kk-1.)*rl/(2.*n)-(m-1.)*rl/n
        end do

C *** do 61 : Counter for elements on receiving surface
        k  = 0
        mm = 0
        do 61 j = n*n+1,2*n*n

          mm = mm+1
          if (j.eq.(k*n+n*n+1)) then
            k  = k+1
            mm = 1
```

```
        end if

        F(l,j) = cfw(xr(i),yr(mm),zr(k),rl/n,rh/n,90.)

 61   continue
 60 continue

C *** Symmetry : The configuration factors between surface #1 and #5
C    are the same for the respective elements as between surface #3
C    and #5 (= ceiling). The same holds for the factors between
C    surfaces #5/#1 and #5/#3. The factors between surface #1 and #5
C    are not necessarily the same as between #5 and #1 !
C    m  : Counter for rows of elements on surface #3
C    k  : Counter for elements on surface #3
C    kk : Counter for elements on surface #5 as viewed from surface #3
C    i  : Counter for elements on surface #1
C    ii : Counter for elements on surface #5 as viewed from surface #1
     i = 0
     do  70 m = 1,n

       l = 0
       do 71 k = 2*n*n+m*n-l, 2*n*n+(m-1)*n+1, -1
         i  = i+1
         l  = l+1

         ii = 4*n*n
         do mm = (n-1), 0, -1
           do kk = 4*n*n+mm*n+1,4*n*n+mm*n+n
             ii = ii+1

             F(k,kk) = F(i,ii)
             F(kk,k) = F(ii,i)

           end do
         end do
 71    continue
 70 continue
```

```
C *** Symmetry : The configuration factors between surface #2 and #5
C     are the same for the respective elements as between surface #4
C     and #5 (= ceiling). The same holds for the factors between
C     surfaces #5/#2 and #5/#4. The factors between surface #2 and #5
C     are not necessarily the same as between #5 and #2 !
C     m  : Counter for rows of elements on surface #4
C     k  : Counter for elements on surface #4
C     kk : Counter for elements on surface #5 as viewed from surface #4
C     i  : Counter for elements on surface #2
C     ii : Counter for elements on surface #5 as viewed from surface #2
      i = n*n
      do  80 m = 1,n

        l = 0
        do 81 k = 3*n*n+m*n-l, 3*n*n+(m-1)*n+1, -1
          i  = i+1
          l  = l+1

          ii = 4*n*n
          do mm = 1,n
            do kk = 4*n*n+mm*n,4*n*n+(mm-1)*n+1,-1
              ii = ii+1

              F(k,kk) = F(i,ii)
              F(kk,k) = F(ii,i)

            end do
          end do
 81     continue
 80   continue

C *** Symmetry : The configuration factors between surface #1(#2,#3,#4) and
C     #5  are the same for the respective elements as between surface
C     #1(#2,#3,#4) and #6 (= floor). The same holds for the factors between
C     #5/#1(#2,#3,#4) and #6/#1(#2,#3,#4) but they are not necessarily the same
C     as those above. The elements on surface #6 are defined by adding n*n to
C     the element number that is located directly on the opposite side on
```

```
C    surface #5 (= ceiling).
C    l : loop for surface #1 (0), #2 (1), #3 (2), and #4 (3)
C    i : Counter for elements on surface #1(#2)
C    ii : Counter for respective elements on surface #1(#2)
C    j : Counter for elements on surface #5
C    jj : Counter for elements on surface #6
C    k : Counter for rows on surface #1(#2)
     do 90 l= 0,3

       m = n-1
       do 91 k = 1,n

        do i = (k-1)*n+l*n*n+1,k*n+l*n*n
           ii = i+m*n

          do j = 4*n*n+1,5*n*n
            jj = j+n*n

            F(ii,jj) = F(i,j)
            F(jj,ii) = F(j,i)

          end do

        end do
        m = m-2

 91    continue
 90  continue

C *** Determine configuration factors between surfaces #1/#3 and #2/#4, where
C    the point sources are on #1 and #2 and the elements on #3 and #4. All
C    previously calculated factors involved surfaces at an angle of 90
C    degrees. The following surface pairs form an angle of 0 degrees: They
C    are parallel. Since the surfaces that are opposite of each other have
C    the same dimensions, their elements have the same areas and it can be
C    concluded that the configuration factors from points i to elements j are
C    identical to those from points j to elements i.
C    do 100 : Indicator for surface pairs #1/#3 and #2/#4
```

```
C    do 101 : Counter for surface #1 and #2
C    do 102 : Counter for surface #3 and #4
     do 100 ii = 0,1

      if (ii.eq.0) then
        zd = rl
        yy = rw
      else
        zd = rw
        yy = rl
      end if

      m  = 0
      k  = 1
      km = 0
      do 101  i = ii*n*n+1,(ii+1)*n*n

        km = km+1
        if (km.eq.k*n+1) then
          k = k+1
        end if

        do  l=1,n
          xr(l) = (2.*l-1.)*rh/(2.*n)-(k-1.)*rh/n
        end do

        m = m+1
        if (m.eq.n+1) then
          m = 1
        end if

        do l=1,n
          yr(l) = (2.*l-1.)*yy/(2.*n)-(m-1.)*yy/n
        end do

        kk = n+1
        mm = 1
```

```
      ll = 0
      do 102 j = (ii+2)*n*n+1,(ii+3)*n*n

        kk = kk-1
        if (kk.eq.0) then
          kk = n
        end if

        ll = ll+1
        if (ll.eq.mm*n+1) then
          mm = mm+1
        end if

        F(i,j) = cfw(xr(mm),yr(kk),zd,yy/n,rh/n,0.)
        F(j,i) = F(i,j)

102     continue
101   continue
100 continue

C *** The same as above holds for surfaces #5 and #6.
C    do 110 : Counter for points on surface #5
C    do 111 : Counter for elements on surface #6
     m  = 0
     k  = 1
     km = 0
     do 110  i = 4*n*n+1,5*n*n

       km = km+1
       if (km.eq.k*n+1) then
         k = k+1
       end if

       do  l=1,n
         xr(l) = (2.*l-1.)*rl/(2.*n)-(k-1.)*rl/n
       end do

       m = m+1
```

```
      if (m.eq.n+1) then
        m = 1
      end if

      do l=1,n
        yr(l) = (2.*l-1.)*rw/(2.*n)-(m-1.)*rw/n
      end do

      kk = n+1
      mm = 1
      ll = 0
      do 111 j = 5*n*n+1,6*n*n

        kk = kk-1
        if (kk.eq.0) then
          kk = n
        end if

        ll = ll+1
        if (ll.eq.mm*n+1) then
          mm = mm+1
        end if

        F(i,j) = cfw(xr(mm),yr(kk),rh,rw/n,rl/n,0.)
        F(j,i) = F(i,j)

111   continue
110  continue

     return
     end


C *** Subroutine ELEMENTS1 determines the elements the room surfaces have been
C     devided into that lie in a window area

     Subroutine elements1(winelem,n,sn,nnw,rh,wb,wh,ww,wwl,wl)
```

```
integer winelem(5,n*n), sn

if (n.eq.1) then
   winelem(nnw,1) = sn
   goto 11
end if

wfrac = wwl/n
hfrac = rh/n

ipos1 = nint(wl/wfrac)
do i = 1,n
   if (ipos1.eq.i*n) then
      ipos1 = ipos1-1
   end if
end do

ipos2 = nint((rh-wb-wh)/hfrac)
do i = 1,n
   if (ipos2.eq.(n-1)*n+i) then
      ipos2 = ipos2-1
   end if
end do

ipos3 = nint((wl+ww)/wfrac)-1
do i = 1,n
   if (ipos3.eq.i*n) then
      ipos3 = ipos3-1
   end if
end do

ipos4 = nint((rh-wb)/hfrac)-1
do i = 1,n
   if (ipos4.eq.(n-1)*n+i) then
      ipos4 = ipos4-1
   end if
```

```
      end do

      m = 0
      do j = ipos2,ipos4
        do k = ipos1,ipos3

          m = m+1
          winelem(nnw,m) = (sn-1)*n*n+j*n+k+1

        end do
      end do

   11 return
      end


C *** Subroutine ELEMENTS2 determines the numbers of the elements that
C     lie above the workplane including the ceiling elements.

      Subroutine elements2(surfelem,n,rh,wkpln)

      integer surfelem(5*n*n), plnfrac

      hfrac   = rh/n
      plnfrac = nint(wkpln/hfrac)

C *** Elements on surface #1 to #4 :
      m = 0
      do k = 1,4
        do i = 0,n-plnfrac-1
          do j = 1,n

            m = m+1
            surfelem(m) = (k-1)*n*n+i*n+j

          end do
        end do
      end do
```

```
C *** Ceiling elements (surface #5) :
      do i = 4*n*n+1,5*n*n
        m = m+1
        surfelem(m) = i
      end do

      return
      end


C *** Subroutine ELEMENTS3 determines the elements the room surfaces have been
C     devided into that lie in a beam patch area

      Subroutine elements3(beamelem,n,sn,rh,wb,wh,ww,wwl,wl)

      integer beamelem(3*n*n), sn

      if (n.eq.1) then
        beamelem(1) = sn
        goto 11
      end if

      wfrac = wwl/n
      hfrac = rh/n

      ipos1 = int(wl/wfrac)+1
      if (ipos1.gt.n) ipos1 = n

      iw = int(ww/wfrac)
      if (iw.gt.n) iw = n


      isum1 = ipos1+iw
      if (isum1.gt.n) isum1 = n

      ipos2 = int((rh-wb-wh)/hfrac)
```

```
      if (ipos2.ge.n-1) ipos2 = n-1

      ih = nint(wh/hfrac)
      if (ih.gt.n-1) ih = n-1

      isum2 = ipos2+ih
      if (isum2.gt.n-1) isum2 = n-1

      m = 0
      do i = ipos2,isum2
        do j = ipos1,isum1

          m = m+1
          beamelem(m) = (sn-1)*n*n+i*n+j

        end do
      end do

      if (beamelem(1).eq.0) then

        write(*,*) ' no elements for the beam patch can be assigned !'
        write(*,*) ' ==> Calculate with more elements n.'

      end if

   11 return
      end


C *** Subroutine REFLEC determines the reflectivity matrix R by assiging
C     the appropriate reflectances to the configuration factors.

      Subroutine reflec(R,F,winelem,lda,n,nw,reflg,reflw,reflc,reflf)

      real R(lda,6*n*n),F(lda,6*n*n)
      integer winelem(5,n*n)

      do 10 i = 1,6*n*n
```

```
do 20 j = 1,6*n*n

if (i.eq.j) then

  do k = 1,nw

    do kk = 1,lda/6

      if (j.eq.winelem(k,kk)) then
        R(i,j) = 1.-reflg*F(i,j)
        goto 20
      end if

    end do

  end do

  if(j.le.4*n*n) then
    R(i,j) = 1.-reflw*F(i,j)
    goto 20
  else if((j.gt.4*n*n) .and. (j.le.5*n*n)) then
    R(i,j) = 1.-reflc*F(i,j)
    goto 20
  else if(j.gt.5*n*n) then
    R(i,j) = 1.-reflf*F(i,j)
    goto 20
  end if

else

  do k = 1,nw

    do kk = 1,lda/6

      if (j.eq.winelem(k,kk)) then
        R(i,j) = -reflg*F(i,j)
        goto 20
```

```
            end if

          end do

        end do

        if(j.le.4*n*n) then
          R(i,j) = -reflw*F(i,j)
          goto 20
        else if((j.gt.4*n*n) .and. (j.le.5*n*n)) then
          R(i,j) = -reflc*F(i,j)
          goto 20
        else if(j.gt.5*n*n) then
          R(i,j) = -reflf*F(i,j)
          goto 20
        end if

      end if

 20   continue
 10 continue

    return
    end


C *** CONSENS determines the configuration factors between those
C     elements that lie above the workplane and the sensor. The
C     conf. factor is stored in FS(i), where i is the number of
C     the surface element above the workplane it refers to, in other
C     words, the conf. factor from a surface element to the sensor
C     is stored in location FS(surfelem. #).

    Subroutine consens(FS,surfelem,n,sw,sl,rh,rw,rl,wkpln)

    real FS(5*n*n)
    integer surfelem(5*n*n)
```

```
hfrac = rh/n

k  = 1
ll = 0
do 10 i= 1,5*n*n
  if (surfelem(k).eq.i) then

     if (ll.eq.n) ll = 0

     ns = int(real(i-.5)/(n*n))+1

     if (ns.eq.1) then
        rr   = rw/n
        tt   = rh/n
        xx   = sl
        yy   = rw-sw-ll*rr
     else if (ns.eq.2) then
        rr   = rl/n
        tt   = rh/n
        xx   = rw-sw
        yy   = rl-sl-ll*rr
     else if (ns.eq.3) then
        rr   = rw/n
        tt   = rh/n
        xx   = rl-sl
        yy   = sw-ll*rr
     else if (ns.eq.4) then
        rr   = rl/n
        tt   = rh/n
        xx   = sw
        yy   = sl-ll*rr
     else if (ns.eq.5) then
        rr   = rw/n
        tt   = rl/n
        yy   = sw-rw+(ll+1)*rr
        zz   = rh-wkpln
     end if
```

```
      do 20 j = n,n*n,n
        if (i.le.(ns-1)*n*n+j) then

          if (ns.le.4) then
           zz = rh-wkpln-real(j/n)*hfrac
            else
           xx = rl-sl-(real(j/n)-1.)*rl/n

            if (xx.le.0.) then
              xx = sl-(n-real(j/n))*rl/n
              yy = rw-sw-ll*rr
            end if

            FS(i) = cfw(xx,yy,zz,rr,tt,0.)
            goto 11
          end if

          if (j/n.eq.n-nint(wkpln/(rh/n))) then
             zz = 0.
           frac = (wkpln/hfrac)-int(wkpln/hfrac)

             if (frac.lt.0.5) then
               tt = hfrac-frac*hfrac
             else
               tt = hfrac+(1.-frac)*hfrac
             end if

          end if
          goto 12

        end if
20      continue

12      FS(i) = cfw(xx,yy,zz,rr,tt,90.)

11      ll = ll+1
      k  = k+1
```

```
      end if
   10 continue

      return
      end


C *** Subroutine LIGHT determines the illuminance at the sensor including
C     the interreflected component.

      Subroutine light(F,FS,winelem,surfelem,floorbeam,wallbeam,lda,n,
     @                 lumwin,reflg,reflw,reflf,reflc,beam1,beam2,beam3,
     @                 beam4,beam6,daylight,reflect,plight,
     @                 dlight,nw,nnw)

      real F(lda,6*n*n), FS(5*n*n), lumwin
      integer winelem(5,n*n), surfelem(5*n*n), floorbeam(3*n*n)
      integer wallbeam(3*n*n)

      pi = 4.*atan(1.)

      reflect = 0.
      plight = 0.
      dlight = 0.
      daylight = 0.
      dayl1 = 0.
      dayls1 = 0.
      dayls2 = 0.
      dayls3 = 0.
      dayls4 = 0.
      dayls6 = 0.
      do 10 l = 1,5*n*n

        if (int(real(l-.5)/(n*n))+1.lt.5) then
          rho = reflw
        else
```

```
      rho = reflc
    end if

    do kk = 1,nw
      do ll = 1,n*n
        if (winelem(kk,ll).eq.surfelem(l)) then
          rho = reflg
        end if
      end do
    end do

    Fsum1 = 0.
    Fsum2 = 0.
    Fsum31 = 0.
    Fsum32 = 0.
    Fsum33 = 0.
    Fsum34 = 0.

    do 20 j = 1,n*n

      if ((winelem(nnw,j).eq.0)  .and.
     @      (floorbeam(j).eq.0)    .and.
     @      (wallbeam(j).eq.0))    then
        goto 11
      end if

      if (winelem(nnw,j).ne.0) then
        Fsum1 = Fsum1+F(surfelem(l),winelem(nnw,j))
      end if

      if (floorbeam(j).ne.0) then
        Fsum2 = Fsum2+F(surfelem(l),floorbeam(j))
      end if

      if (wallbeam(j).ne.0) then

        if (wallbeam(j).le.n*n) then
```

```
            Fsum31 = Fsum31+F(surfelem(l),wallbeam(j))
          else if (int((wallbeam(j)-0.5)/(n*n))+1.eq.2) then
            Fsum32 = Fsum32+F(surfelem(l),wallbeam(j))
          else if (int((wallbeam(j)-0.5)/(n*n))+1.eq.3) then
            Fsum33 = Fsum33+F(surfelem(l),wallbeam(j))
          else if (wallbeam(j).gt.3*n*n) then
            Fsum34 = Fsum34+F(surfelem(l),wallbeam(j))
          end if

        end if

 20   continue

C *** reflected components due to the window and beam patches on surfaces
C     #6, #1, #2, #3, #4
 11   dayl1 = dayl1+rho*Fsum1*FS(surfelem(l))*pi*lumwin
      dayls6 = dayls6+rho*Fsum2*FS(surfelem(l))*reflf*beam6
      dayls1 = dayls1+rho*Fsum31*FS(surfelem(l))*reflw*beam1
      dayls2 = dayls2+rho*Fsum32*FS(surfelem(l))*reflw*beam2
      dayls3 = dayls3+rho*Fsum33*FS(surfelem(l))*reflw*beam3
      dayls4 = dayls4+rho*Fsum34*FS(surfelem(l))*reflw*beam4

      daylight = daylight+rho*Fsum1*FS(surfelem(l))*pi*lumwin
     @                    +rho*Fsum2*FS(surfelem(l))*reflf*beam6
     @                    +rho*Fsum31*FS(surfelem(l))*reflw*beam1
     @                    +rho*Fsum32*FS(surfelem(l))*reflw*beam2
     @                    +rho*Fsum33*FS(surfelem(l))*reflw*beam3
     @                    +rho*Fsum34*FS(surfelem(l))*reflw*beam4

 10 continue

C *** light at sensor position due to interreflection from all sources
     reflect = daylight


C *** Add direct diffuse component from windows and beam patches
C     to interreflected light
     dirwin = 0.
```

```
      dirbeam1 = 0.
      dirbeam2 = 0.
      dirbeam3 = 0.
      dirbeam4 = 0.

      do i = 1,3*n*n

        if (i.gt.n*n) goto 13
        if (winelem(nnw,i).ne.0) then
          dirwin = dirwin + FS(winelem(nnw,i))
        end if

   13   if (wallbeam(i).ne.0) then

          if (wallbeam(i).le.n*n) then
            dirbeam1 = dirbeam1 + FS(wallbeam(i))
          else if (int((wallbeam(i)-0.5)/(n*n))+1.eq.2) then
            dirbeam2 = dirbeam2 + FS(wallbeam(i))
          else if (int((wallbeam(i)-0.5)/(n*n))+1.eq.3) then
            dirbeam3 = dirbeam3 + FS(wallbeam(i))
          else if (wallbeam(i).gt.3*n*n) then
            dirbeam4 = dirbeam4 + FS(wallbeam(i))
          end if

        end if

      end do

      dlight = dirwin*pi*lumwin + dirbeam1*beam1*reflw
     @                          + dirbeam2*beam2*reflw
     @                          + dirbeam3*beam3*reflw
     @                          + dirbeam4*beam4*reflw

      daylight = daylight+dlight

C *** reflected plus direct light due to beam patches.
      plight = dayls1+dayls2+dayls3+dayls4+dayls6+dirbeam1*beam1*reflw
```

```
     @                          +dirbeam2*beam2*reflw
     @                          +dirbeam3*beam3*reflw
     @                          +dirbeam4*beam4*reflw

      return
      end


C *** Function CFW calculates the Configuration Factor for vertical Windows
C     according to Pierpoint and Hopkins.
C     note: phi in degrees!

      Function cfw(x,y,z,r,t,phi)

      pi = 4.*atan(1.)

      g = z*sind(phi)-x*cosd(phi)
      h = z*cosd(phi)+x*sind(phi)
      a = x*x+y*y+z*z-y*r
      b = a+t*t+2.*t*g
      c = y*y+h*h+g*g+t*g
      d = (y-r)*(y-r)+h*h+g*g+t*g

C  Devide check
      if (a.eq.0) then
        a = 1.e-10
      end if

      if (b.eq.0) then
        b = 1.e-10
      end if

      if (c.eq.0) then
        c = 1.e-10
      end if

      if (d.eq.0) then
        d = 1.e-10
```

```
      end if

      arg1 = r*sqrt(x*x+z*z)/a
      arg2 = r*sqrt(x*x+z*z+t*t+2.*t*g)/b
      arg3 = t*sqrt(y*y+h*h)/c
      arg4 = t*sqrt((y-r)*(y-r)+h*h)/d
      targ1 = atan(arg1)
      targ2 = atan(arg2)
      targ3 = atan(arg3)
      targ4 = atan(arg4)

      if (arg1.lt.0) then
        targ1 = targ1 + pi
      end if

      if (arg2.lt.0) then
        targ2 = targ2 + pi
      end if

      if (arg3.lt.0) then
        targ3 = targ3 + pi
      end if

      if (arg4.lt.0) then
        targ4 = targ4 + pi
      end if

      cfw = x/sqrt(x*x+z*z)*targ1+(t*cosd(phi)-x)/sqrt(x*x+z*z+t*t+
     @     2.*t*g)*targ2+y*cosd(phi)/sqrt(y*y+h*h)*targ3-
     @     (y-r)*cosd(phi)/sqrt((y-r)*(y-r)+h*h)*targ4
      cfw = 0.5/pi*cfw

      return
      end


C *** REFLIGHT calculates the ground reflected light on the window plane
```

```
C     taking into acount the window transmittance.

      subroutine reflight(diffuse,total,reflgr,t0,outrefl)

      pi = 4.*atan(1.)

      groundr = 0.5/pi*reflgr*(93.+18.*diffuse/total)*total*t0          ! [lm]
      outrefl = groundr

      return
      end


C *** WINLUM calculates the luminance of the window plane as seen by a point
C     in the center of the window and averaged over 25 evenly distributed
C     points.

      subroutine winlum(sn,finl,dfl,finr,dfr,ww,ov,rh,wh,wb,lumwin,t0,
     @                  sunaz,sunalt,diffuse,total,beta,omega,phase)

      real azangle(5), altangle(5), skylum(25), lumwin
      integer sn, beta

      pi = 4.*atan(1.)

      if (ov.eq.0.) then
        eps = 1.e-05
      else
        eps = 0.
      end if

      do i = 1,5
        altangle(i) = atand((rh-wb-wh/2.)/(ov+eps))/5. * real(i)
      end do

      azangle(1) = atand(finl/(dfl+ww/2.)) - 90.
      azangle(5) = 90. - atand(finr/(dfr+ww/2.))
```

```
azfrac = (azangle(5)-azangle(1))/4.

azangle(2) = azangle(1) + azfrac
azangle(3) = azangle(1) + 2.*azfrac
azangle(4) = azangle(1) + 3.*azfrac

if (sn.eq.1) then

  do i = 1,5
    azangle(i) = azangle(i) + raz
  end do

else if (sn.eq.2) then

  do i = 1,5
    azangle(i) = azangle(i) +raz - 90.
  end do

else if (sn.eq.3) then

  do i = 1,5
    azangle(i) = azangle(i) + raz - 180.
  end do

else if(sn.eq.4) then

  do i = 1,5
    azangle(i) = azangle(i) + raz + 90.
  end do

end if

do i = 1,5
  if(azangle(i).lt.-180.) then
    azangle(i) = 360.+azangle(i)
  end if
```

```
      if(azangle(i).gt.180.) then
        azangle(i) = azangle(i)-360.
      end if
    end do

    k = 0
    do i = 1,5
      do j = 1,5
        k = k+1
       call sky(azangle(i),altangle(j),sunaz,sunalt,diffuse,
@              total,beta,omega,phase,skylum(k))
      end do
    end do

    lumwin = 0.
    do i = 1,25
       lumwin = lumwin + skylum(i)
    end do

    lumwin = lumwin/25. * t0                              ! [lm]

    return
    end


C *** SKY calculates the luminance of the patch of sky seen by the window.
C    parameters :
C    diffuse  = Diffuse horizontal soalr radiation (W/(m*m))
C    total    = Total hoprizontal solar radiation (W/)m*m))
C    paz      = Azimuth angle of point in sky
C    palt     = Altitude angle of point in sky
C    sunaz    = Solar azimuth angle
C    sunalt   = Solar altitude angle
C    beta     = Climatic region (1=rural,2=urban,3=industrial)
C    omega    = Water vapor content of atmosphere in cm of water
C    cr       = Cloud ratio (diff. horiz. radiation/total hor. rad.)
C    phase    = Clear sky fraction
```

```
C    jul      = Julian date

     Subroutine sky(paz,palt,sunaz,sunalt,diffuse,total,beta,omega,
     @                phase,skylum)

     real lpclear,lpover
     integer beta

     pi = 4.*atan(1.)

C *** Phasing technique according to Gary Gillette (NBS) to find
C    intermediate state between clear and overcast sky conditions
     cr = diffuse/total

     phase = (1.+cos(cr*pi))/2.

     skylum = (lpclear(paz,palt,sunaz,sunalt,beta,omega)*phase
     @         +lpover(diffuse,palt)*(1.-phase))              ! [cd/(m*m)]

     return
     end


C *** LPOVER calculates of the overcast sky luminance distribution
C    with an assumed diffuse horizontal luminous efficacy of 115 lm/W

     Function lpover(diffuse,palt)

     real lpover

     pi = 4.*atan(1.)

     lpover = 3./7./pi*115.*diffuse*(1.+2.*sind(palt))       ! [cd/(m*m)]

     return
     end
```

```
C *** LPCLEAR calculates of the clear sky luminance at a specific point
C    in the sky according to KITTLER.
C    Parameters :
C    paz        = Azimuth angle of sky element
C    palt       = Angle between sky element and horizontal
C    sunalt     = Solar altitude angle
C    sunaz      = Solar azimuth angle
C    lzclear    = Zenith clear sky luminance
C    lpclear    = Clear sky luminance at specified point
C    coef(3,3,6) = Array containing coefficients for Kittler zenith
C                  clear sky luminance formula
C    psi        = great circle angle between specified point and sun
C    beta       = climactic region (1=rural,2=urban,3=industrial)
C    omega      = water vapor content of atmosphere in cm of water
C    thetaz     = solar zenith angle

     Function lpclear(paz,palt,sunaz,sunalt,beta,omega)

     real coef(3,3,6),lzclear,lpclear
     integer beta

     pi = 4.*atan(1.)

     data (coef(1,1,i),i=1,6) /936.104,909.278,872.698,854.731,846.000,
    @                          840.993/
     data (coef(1,2,i),i=1,6) /2.015,1.792,1.433,1.239,1.146,1.103/
     data (coef(1,3,i),i=1,6) /-0.0002,-0.0001,4*0./
     data (coef(2,1,i),i=1,6) /1032.875,1020.647,1007.411,1001.928,
    @                          999.066,998.493/
     data (coef(2,2,i),i=1,6) /2.418,2.215,1.894,1.718,1.631,1.575/
     data (coef(2,3,i),i=1,6) /2*-0.0002,4*-0.0001/
     data (coef(3,1,i),i=1,6) /1139.868,1149.857,1169.805,1183.266,
    @                          1189.851,1196.765/
     data (coef(3,2,i),i=1,6) /3.049,2.831,2.506,2.319,2.220,2.141/
     data (coef(3,3,i),i=1,6) /2*-0.0003,4*-0.0002/

     if (beta.eq.1) then
```

```
      i = 1

      else if (beta.eq.2) then
        i = 2

      else if (beta.eq.3) then
        i = 3

      else
        write(*,*) ' *** Error : Input for climate has to be'
        write(*,*) ' 1 (rural), 2 (urban) or 3 (industrial) !'
        stop

    end if

    if (omega.eq.0.5) then
      j = 1

      else if (omega.eq.1.0) then
      j = 2

      else if (omega.eq.2.0) then
      j = 3

      else if (omega.eq.3.0) then
      j = 4

      else if (omega.eq.4.0) then
      j = 5

      else if (omega.eq.5.0) then
      j = 6

      else
      write(*,*) ' *** Error : The water vapor content input (omega)'
       write(*,*) '            has to be 0.5, 1.0, 2.0, 3.0, 4.0 or'
       write(*,*) '            5.0 !'
       stop
```

```
      end if

      a = coef(i,1,j)
      b = coef(i,2,j)
      c = coef(i,3,j)

      lzclear = a+b*sunalt*sunalt+c*sunalt*sunalt*sunalt    ! [cd/(m*m)]

C *** solar zenith angle
      thetaz = 90.-sunalt

C *** great circle angle between sky element and sun (scattering angle),
      psi = cosd(palt)*sind(thetaz)*cosd(abs(paz-sunaz))
     @    +sind(palt)*cosd(thetaz)
      psi = acos(psi)

C *** luminance at point p
      lpclear = lzclear*(1.-exp(-.32/sind(palt)))*(.91+10.
     @         *exp(-3.*psi)+.45*cos(psi)*cos(psi))
     @         /(.274*(.91+10.*exp(-3.*(pi/2.-sunalt*pi/180.))
     @         +.45*sind(sunalt)*sind(sunalt)))

      return
      end


C *** TRANSMIT calculates of the angular dependent transmittance
C     of a vertical window for the direct beam component.
C     Nomenclature:
C     surfaz = Azimuth angle of considered room surface
C     sunalt = Solar altitude angle
C     sunaz  = Solar azimuth angle
C     raz    = building azimuth angle
C     inc    = incidence angle
C     t0     = Visible transmittance value (from manufacturer)
```

```
      Function transmit(raz,sn,sunalt,sunaz,t0)

      real inc
      integer sn

C *** Determine azimuth angle of room surface
      if (sn.eq.1) then
        surfaz = raz

      else if (sn.eq.2) then
        surfaz = raz-90.

      else if(sn.eq.3) then
        surfaz = raz-180.

      else if (sn.eq.4) then
        surfaz = raz+90.

      end if

      if (surfaz.lt.(-180.)) then
        surfaz = 360.+surfaz
      end if

      if (surfaz.gt.180) then
        surfaz = surfaz-360.
      end if

C *** Calculate incidence angle (angle between sun and normal of
C     window plane)
     inc = acosd(sind(90.-sunalt)*cosd(sunaz-surfaz))

C *** Angular dependent transmittance
      transmit = 1.018 * t0 * cosd(inc) * (1.+sind(inc)**3)

      return
      end
```

```
C *** BEAM calculates the direct beam illuminance using an atmospheric
C     extinction coefficient of 0.21

      Function beam(sunalt,jul)

      real jul

      beam = 127500.*(1.+0.033*cosd(360.*j/365.))
     @       *exp(-0.21/sind(sunalt))

      return
      end
```

BIBLIOGRAPHY

Beckman, W.A., "Solution of Heat Transfer Problems on a Digital Computer," *Solar Energy*, Pergamon Press, vol. 13, pp. 293-300, 1971.

Dogniaux, R., "Représentation Analytiques des Composantes du Rayonnement Lumineux Solaire," *Publication Série A*, Institut Royal Météorologique de Belgique, No. 83, 1974.

Dongarra, J.J., Moler, C.B., Bunch, J.R., and Stewart, G.W., *LINPACK*, Philadelphia, Siam, 1979.

Duffie, J.A., and Beckman, W.A., *Solar Engineering of Thermal Processes*, New York, Wiley-Interscience, 1980.

Gillette, G., "A Daylighting Model for Building Energy Simulation," *NBS Building Science Series 152*, 1983.

Hamilton, D.C., and Morgan, W.R., "Radiant-Interchange Configuration Factors," *Technical Note 2836*, National Advisory Committee for Aeronautics, 1984.

Karayel, M., Navvab, M., Ne'eman, E., and Selkowitz, S., "Zenith Luminance and Sky Luminance Distribution for Daylighting Calculations," University of California - Berkeley, Lawrence Berkeley Laboratory, LBL-15622, 1983.

Kaufman, J.E., ed., *IES Lighting Handbook*, Illuminating Engineering Society of North America, Baltimore, Waverly Press, 1981.

Klein, S., Class Handouts from "Heat Transfer", Mechanical Engineering 364, University of Wisconsin - Madison.

Littlefair, P.J., "The Luminous Efficacy of Daylight: A Review," *Lighting Research and Technology*, vol. 17, No. 4, 1985.

Narasimhan, V., Saxena, B.K., and Maitreya, V.K., "The Internal Reflected Component of Daylight," *Indian Journal of Pure Applied Physics*, vol. 6, February 1968.

Pierpoint, W., and Hopkins, J., "The Derivation of a New Area-Source Equation," *Journal of IES*, April 1984.

Solar Energy Laboratory, "A Transient Simulation Program," University of Wisconsin - Madison, 1983.