DESCRIPTION OF THE VIP PROGRAM

FOR COMPUTING INCIDENT AND

ABSORBED ENERGY AT THE SURFACE

OF AN ORBITING SATELLITE

R. Goldstein
R. Nagel

March, 1969

Work Performed for

Goddard Space Flight Center

Greenbelt, Maryland

Contract No. NAS5-11702

Mathematical Applications Group, Inc.

180 South Broadway

White Plains, New York

TABLE OF CONTENTS

1. DESCRIPTION OF THE VIP PROGRAM

VIP (Vehicle Illumination Program) is a FORTRAN IV code for determining incident and absorbed radiant energy on the surface of orbiting space vehicles. Starting with a three-dimensional description of the spacecraft geometry, the code employs geometric ray tracing techniques for following rays of energy from their source of origin to a point of intersection on the spacecraft. External energy sources treated are direct emission from both the sun and an arbitrary planet and planetary reflection of sunlight. Incident and absorbed energy is computed for these sources as a function of location on the vehicle surface. In addition, the code is capable of reflecting rays from the vehicle surface and determining that component of the energy resulting from such reflections. Instantaneous results at specified locations in orbit as well as time integrated results over complete orbits may be obtained.

The program has been written to provide the user with a wide range of flexibility in the class of problems which can be solved. These capabilities are described in the remainder of this section. Section 2 discusses the basic concepts employed for three-dimensional geometry description. Section 3 provides a user's guide to the program, describing the details of preparing input, running a problem, and interpreting the output. This is followed, in Section 4, by a somewhat qualitative description of each subroutine in the code. The mathematical formulations of some of the more important routines are then given in Appendices 1-6.

1.1 Geometry Capabilities

The program employs a powerful technique for modeling three-dimensional geometries and inputting the model into a computer. This method, called the Combinatorial Geometry Technique, permits the accurate representation of space vehicles of essentially any degree of complexity. The details of the method are discussed in Section 3 but, in essence, it represents the vehicle and its components as an array of volumes (or regions) of arbitrary size, shape, and orientation. All results for energy deposition and absorption are computed and edited as function of region, so that a complete 'mapping' of energy vs. vehicle component is provided.

1.2 Surface Properties

The computation of energy absorption and reflection requires a description of the reflection properties of each region. A region may be designated as being either a specular reflector, a diffuse reflector, or transparent. In the former case, a 'mirror' reflection occurs at the surface with a single ray being emitted. A diffuse reflection results in a number of rays being emitted in predetermined directions. A transparent region acts like a vacuum, transmitting 100% of the energy. Region properties are entered in the form of one or more sets of reflection coefficients (fraction of energy reflected) for each region. Each set is composed of two coefficients with first applicable to solar wavelengths and second to planetary radiation. The source of the ray is used to select the proper region coefficient and this, in turn, determines both the absorbed surface energy and the reflected ray energy. Up to five sets of coefficients can be used and the code will provide separate energy vs. region results for each set. Thus, it is possible to obtain parametric studies of vehicle surface properties.

1.3 Energy Sources

A problem may be run to obtain results for various combinations of the following four sources of energy:

a) Solar radiation,

b) Planet radiation,

c) Planet reflection of sunlight,

d) Vehicle reflection of the above sources.

The output will provide separate results for each energy source. Solar radiation is treated as a set of parallel rays emitted from a plane having any desired orientation with respect to a fixed spacial coordinate system. The solar constant (energy/area-time) is supplied as input. Planet radiation consists of non-parallel rays originating on that part of the planet surface which is visible to the satellite. Since the radiating temperature and radius of the planet are input quantities, the code has the flexibility of handling any desired planet. Planet reflection results are obtained merely by inputting the reflection coefficient (albedo) of the planet surface for sunlight.

1.4 Allowed Orbits

One of the following orbit classifications may be selected for a problem:

a) Inertial (with no spin) - The vehicle maintains a fixed orientation with respect to the sun.

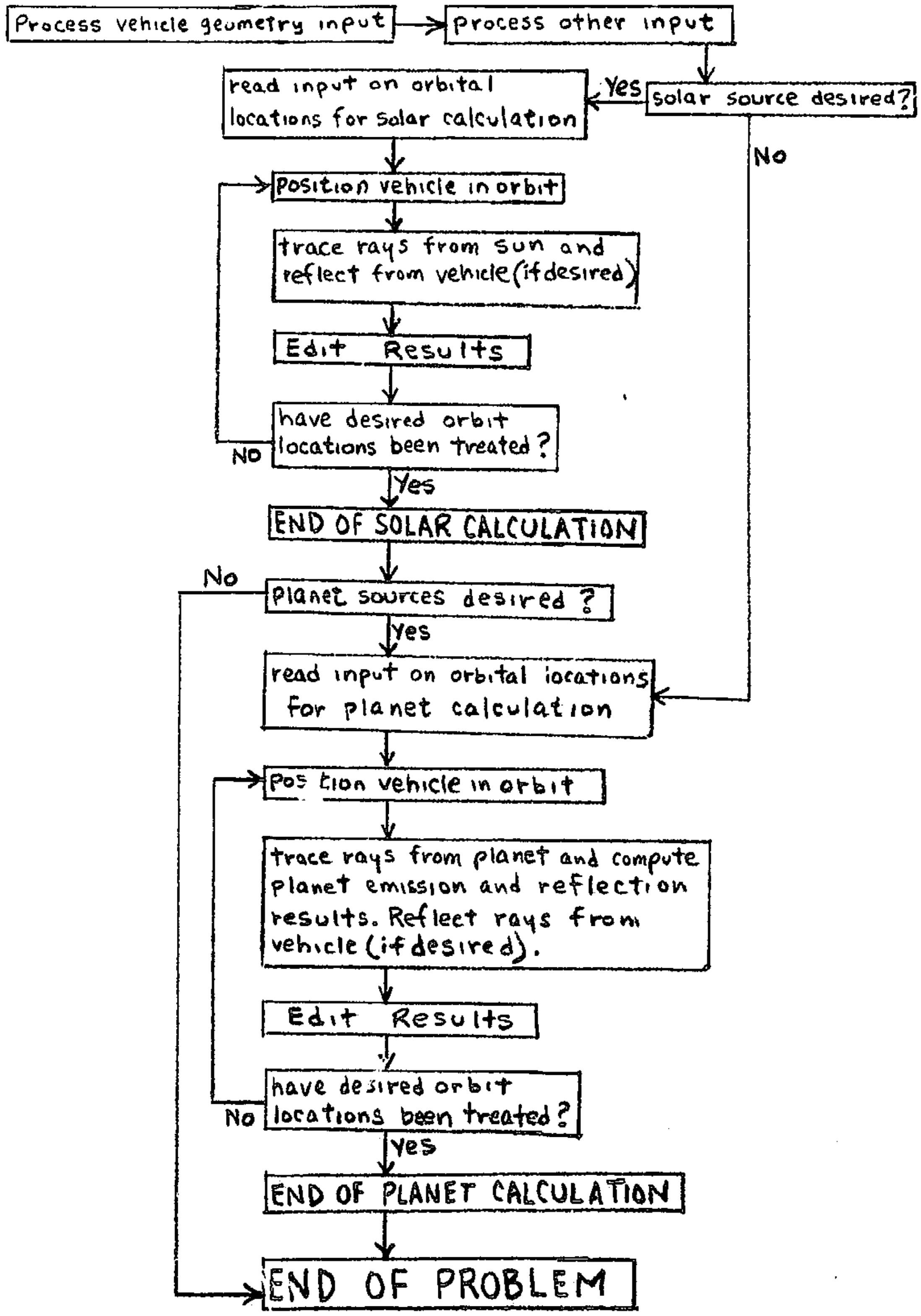
b) Spin Stabilized - Similar to the inertial orbit except that the vehicle is rotating about a fixed axis.

c) Gravity Gradient - The vehicle maintains a fixed orientation with respect to the planet.

Any of the above orbits is specified by its period, the vehicle altitudes at perigee and apogee, and the orientation of the orbital plane. The calculation is performed in stepwise fashion around the orbit. That is, the vehicle is initially positioned at a specified location and a set of rays are traced giving answers at that location. The vehicle is then moved through a given angle to its next position in orbit and another set of rays is traced. This procedure is repeated until the desired number of locations have been completed. The initial location, angular interval between locations, and number of locations are all input parameters.

1.5 Calculation Procedure

The first step in the calculation is to read and process input defining the vehicle geometry, region properties, orbit parameters, etc. In general, all input which is independent of the chosen energy sources is processed first. The actual computation is then split into two parts so that the solar and planetary sources are done completely independently, with the solar source done first. The following macroscopic flowchart indicates the major calculational steps performed by the program. It should be noted that for inertial or spin stabilized orbits, solar rays are traced at the initial location only and the results are applied to all locations. Correction factors are applied for those locations where the vehicle is shaded by the planet.


Process vehicle geometry input
process other input
solar source desired?
Yes
read input on orbital locations for solar calculation
No
Position vehicle in orbit
trace rays from sun and reflect from vehicle (if desired)
Edit Results
have desired orbit locations been treated?
NO
Yes
END OF SOLAR CALCULATION
No
Planet sources desired ?
Yes
read input on orbital locations for planet calculation
Position vehicle in orbit
trace rays from planet and compute planet emission and reflection results. Reflect rays from vehicle (if desired).
Edit Results
have desired orbit locations been treated?
No
Yes
END OF PLANET CALCULATION
END OF PROBLEM

VIP Flowchart

1.6 Edit Description

An edit of results is provided after each orbital location is completed. This permits the user to restart a problem in the middle with no loss of information. The edit provides the following categories of data for each energy source and for each region in the geometry:

a) incident energy/sec at the current location,

b) absorbed energy/sec at the current location,

c) time integrated incident energy from initial to current location,

d) time integrated absorbed energy from initial to current location.

This information is given in the following tabular form, where a, b, c, d refer to the above data categories.

SOLAR CALCULATION

	solar emission				vehicle reflection of sunlight*			
Region #	a	b	c	d	a	b	c	d

Tables repeated for each location.

PLANET CALCULATION

	planet emission				planet reflection				vehicle reflection of planet emitted and reflected rays*			
Region #	a	b	c	d	a	b	c	d	a	b	c	d

Tables repeated for each location.

* These tables are blank if vehicle reflection is not desired.

2. THE COMBINATORIAL GEOMETRY METHOD

Combinatorial Geometry is essentially a technique for representing, in a computer, a mathematical model of a three-dimensional geometric configuration. Once in the computer, the configuration can be analyzed in many different ways by ray tracing techniques. For example, quantities such as volumes, surface areas, object boundaries, line of sight distances, etc. are readily determined. Regardless of the application, however, the basic concepts employed are the same. A discussion of these concepts can logically be broken down into two topics. That is, geometry description and ray tracing, which are discussed separately below.

2.1 Description of the Geometry

In effect, the description of the geometry divides the problem space into unique volumes, called regions. Each region is, in turn, a combination of one or more standard geometric figures, called bodies. The allowed bodies (two examples of which are spheres and cylinders) and how they are described are discussed later. For the present, it is sufficient to say that a body is completely defined by its type, dimensions, and location in space relative to an X,Y,Z coordinate system.

Once all bodies have been described, each region may be defined by a simple equation which combines these bodies in the following way. Consider the equation

$$R = \pm B_1 \pm B_2 \pm B_3 \ldots \pm B_i$$

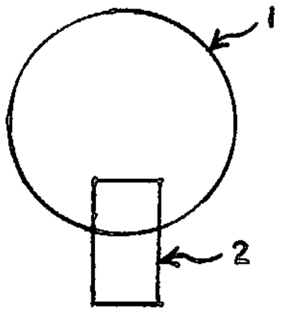
where R represents a particular region number and the B's represent body numbers. The + or - signs act as operators having the following significance. A (+ B) means that all points within region R lie inside of body B. A (- B) means that region R is wholly outside of body B. In addition to the + or -, there is a third allowed operator (written OR) which permits a region to be described in terms of subregions. An example might be

$$R = OR(+B_1)OR(B_2-B_3).$$

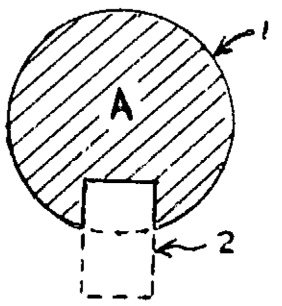
This defines region R as having two parts, one of which is wholly contained within body B_1. The second part is wholly within B_2 but completely outside of B_3.

There is no limit to the number of bodies which may appear in a region description equation as long as the following rule is obeyed. Every spatial point in the geometry must be located in one and only one region.

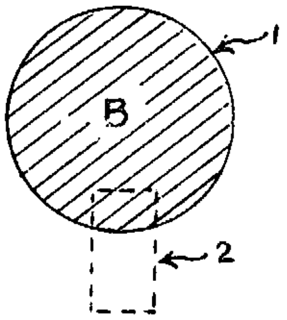
To illustrate the method, consider the following figures showing several different regions which can be constructed by combining a sphere and a cylinder.



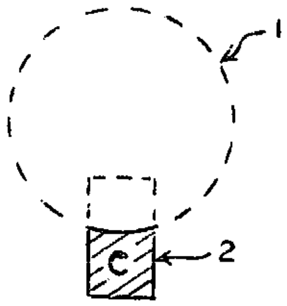
Assume that a sphere (body 1) and a cylinder (body 2) have been described such that the cylinder penetrates the sphere.



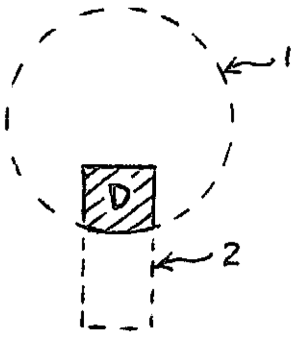
Region A (shaded) is defined as A=+1-2. This means that all points in region A lie inside the sphere and outside the cylinder.



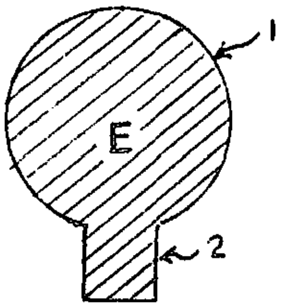
B=+1 Region B is the entire sphere. Notice that body 2 is not in the equation since the cylinder is not needed to describe the region.



C=+2-1 Region C is that part of the cylinder which lies outside of the sphere.

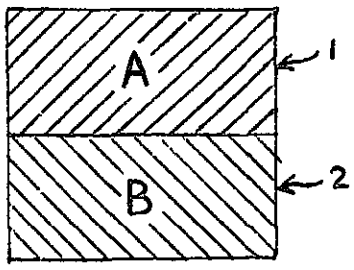


D=+1+2 This says that all points in region D are simultaneously within the sphere and within the cylinder. The only part of the geometry which obeys this description is the shaded area.



E=OR(+1)OR(+2) Part of E lies completely within the entire sphere and another part lies in the entire cylinder. Note that the use of the OR operators in the above equation does not require that the bodies intersect. The sphere and cylinder could have been widely separated giving a discontinuous region, which is perfectly legal.

There is an important rule of construction which is illustrated by the following example. Consider a cylinder (body 1) resting on top of another cylinder (body 2). The rule states that when describing a region, one must negate (- operator) any body that has a buttressing (coplanar) surface with the region. Thus, A=+1-2 and B=+2-1.



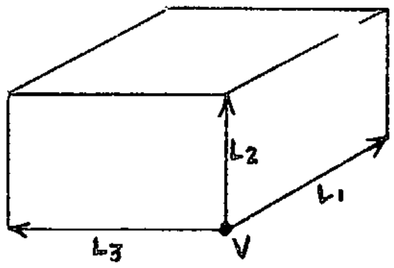
Although the above examples dealt only with spheres and cylinders, there are actually 9 body types available for modeling the geometry. The parameters needed to describe each body are discussed below. The three letter symbol in parentheses is called the 'body identifier' and, with the exception of the RPP is required input.

1. Rectangular Parallelepiped (RPP)

These bodies are used for gross subdivisions of the geometrv and must have bounding surfaces parallel to the coordinate axes. The input consists of the minimum and maximum X,Y,Z coordinates of the 6 planes which bound the body. The entire geometry must be enclosed in an RPP.

2. Box (BOX)

Specify the (X,Y,Z) coordinates of the vertex (V) at any one of the corners of the box. Also specify the (X,Y,Z) components of a set of three mutually perpendicular vectors (L_i) representing the height, width and length of the box. These vectors need not be parallel to the coordinate axes.

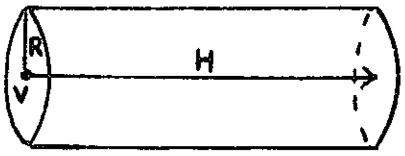


3. Sphere (SPH)

Specify the coordinates of the center of the sphere (V) and length of the radius R.

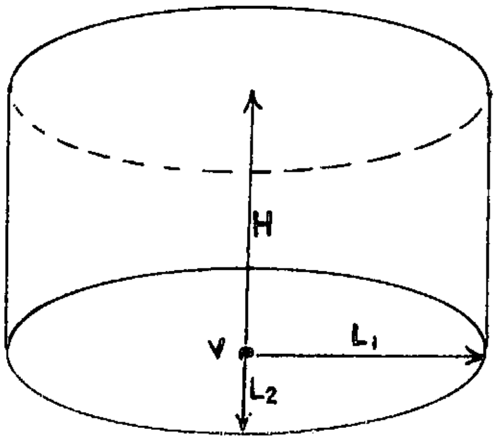
4. Right Circular Cylinder (RCC)

Specify the coordinates of the vertex (V) at the center of either base, a height vector H in terms of its X,Y,Z components, and a scalar R denoting the base radius. Note that V is the origin of the vector H so that the components of H may be either positive or negative depending on the cylinder's orientation.



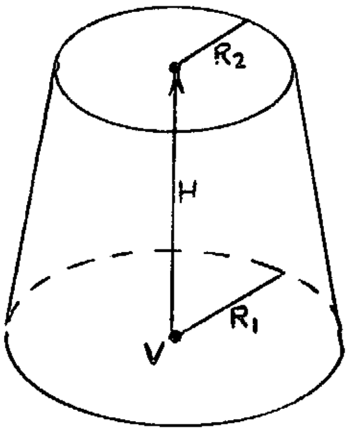
5. Right Elliptic Cylinder (REC)

Specify the coordinates of the center of the base ellipse, the components of the height vector H, and two vectors, L_1 and L_2, defining the semi-major and semi-minor axes of the base ellipse.



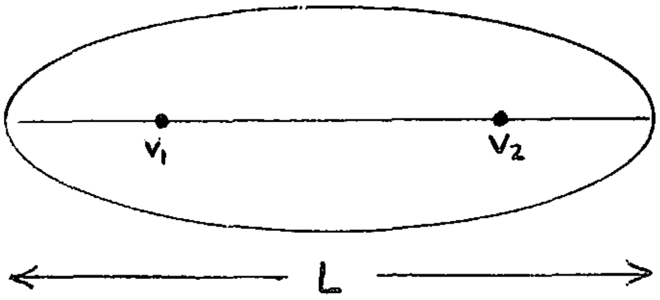
6. Truncated Right Angle Cone (TRC)

Specify the coordinates of the vertex V at the center of the lower base, the components of the height vector H, and two scalars, R_1 and R_2, giving the radii of the larger and smaller bases, respectively.



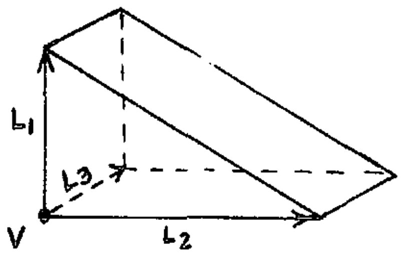
7. Ellipsoid (ELL)

Specify the coordinates of two vertices, denoting the locations of the foci of the ellipse and a scalar L giving the length of the major axis.



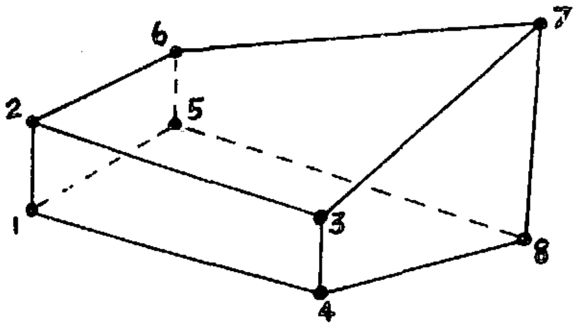
8. Right Angle Wedge (RAW)

Same input as for the BOX with the restriction that L_1 and L_2 describe the two legs of the right triangle of the wedge.



9. Arbitrary Polyhedron (ARB)

The ARB is bounded by any 6 planes so long as the figure is convex (the angle between any two sides is less than 180°). Assign an integer number N to each of the 8 vertices of the ARB. Give the coordinates of each vertex V_N. Then list the numbers of the 4 vertices at the corners of each side. The vertex numbers must be given either in clockwise or counterclockwise order.

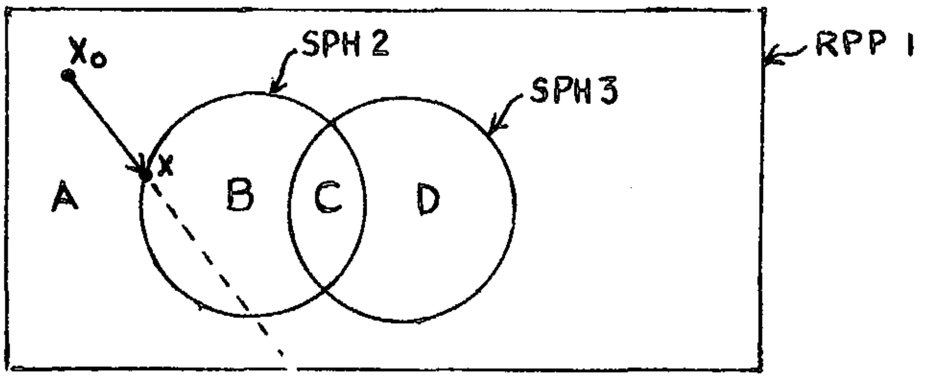


In this example, the integer descriptions of each side are 1234, 1265, 3784, 1485, 5678, 2376.

2.2 Ray Tracing

In effect, ray tracing is a means of following a straight line through the various regions comprising the geometry. In the VIP program the rays simulate non-penetrating electro-magnetic radiation (as opposed to x-rays, for example) so that each ray is terminated at the surface of a non-transparent region.

An explanation of the ray tracing technique is best accomplished by following the code through a simple example. Consider two intersecting spheres enclosed in an RPP. The bodies are numbered 1-3 and the regions are lettered A-D. (Letters are used for clarity only. Region are numbered in the code.)



The region descriptions for this geometry are:

A = +1-2-3

B = +2-3

C = +2+3

D = +3-2

Before proceeding with the discussion, let us list two tables of information which are constructed by the code during input processing. These are called 'entering' and 'leaving' tables which define the possible regions a ray might be in if it <u>enters</u> a given body and the regions a ray might be in if it <u>leaves</u> a given body. These tables are derived from the region description equations and would appear as follows for this problem.

Body	Entering	Leaving
1	A	escapes from geometry
2	B or C	A or D
3	C or D	A or B

For example, these tables tell the code that a ray which enters body 2 must be entering either region B or C. A ray leaving body 3 must be entering A or B.

, ssume now that the ray originates at point X_O in region A and that its direction of flight is known. The program must then determine the next region in the ray's path and the point of intersection (X) with that region.

The code first examines all the bodies in the region description of A for intersections with the ray. The program contains a separate routine for each body type for computing these intersections. These routines compute two parameters called RIN (the distance from X_O to the entering point on the body) and ROUT (from X_O to the leaving point on the body). If the ray passes through the body both numbers will be positive. If X_O is inside the body, RIN will be negative. If the ray misses the body entirely both RIN and ROUT are negative. After investigating bodies 1, 2 and 3 the code will have six numbers and will choose the smallest positive number to determine what has actually happened to the ray. From the picture of the geometry, it is obvious that RIN for body 2 must be chosen, so that the ray must be entering body 2.

The entering table for body 2 now tells the code that the ray could be entering region B or C. The validity of the region equations for B and C are then tested via a series of logical rules. Without going into detail on these rules it should be obvious that the equation for C cannot be satisfied. The +3 in this equation requires the ray to be in or on the surface of body 3 at point X. But, since ROUT for body 3 is negative, the ray must have missed the body entirely. Thus, it cannot be in region C. The B equation, however, is satisfied since the +2 is valid (the ray is known to have hit body 2) and the -3 is also valid (the ray cannot be in body 3). The ray is, thus, determined to be entering region B. The intersection point X is determined from the vector equation

$$\overline{X} = \overline{X}_O + (RIN_3)\overline{w}$$

where $\overline{w}$ represents the direction cosines of the ray. To transmit the ray through region B, X_O is set equal to X and B becomes the new region of origin. The above process is then repeated.

3. VIP USER'S GUIDE

This section is provided mainly as a guide to preparing input for a VIP calculation. It is assumed that the reader has read the remainder of the report and has become familiar with the meanings and purposes of the input variables.

3.1 Units

The units for most input quantities are arbitrary but care should be taken to use a consistent set of units throughout. Dimensions of length, for example, may be in centimeters, inches, feet, etc. Once chosen, however, the unit of length must remain constant. Thus, if vehicle dimensions are in inches, the planet radius and vehicle altitudes must also be in inches. Area units will then be in square inches, such as energy/in^2-sec. for the solar constant. Time and energy units are also arbitrary. The only required unit is that angles be given in degrees.

3.2 Card Input

Cards are described in the order in which they must appear in the input deck. A sample input form is included at the end of this section.

Card 1 - Title Card (Format 20A4)

This card may contain 80 columns of alphanumeric information. It serves as a title card and will appear as the first line of output.

Card 2 - Geometry Descriptors (Format 4I10)

The following four integer quantities are required

Column Input

10 NRPP The number of rectangular parallelepipeds in the geometry (enter a 1 in column 10)

11-20 NBODY The number of bodies in the geometry not including the RPP. The total number of bodies is, thus, NBODY + 1.

21-30 NRMAX The number of regions in the geometry.

40 ITBL Used to obtain a printout of the geometry data table. If ITBL=0, this table will be printed. If ITBL=1, the table will not be printed. The geometry input data are always printed but this table contains the data after processing. Unless one is familiar with the way this table is constructed, the printout probably won't be of much use.

NOTE: Before beginning to fill out geometry input the user should have at least a crude sketch of the satellite to refer to. An X,Y,Z coordinate system should be established with the origin at a convenient location near the geometric center of the vehicle. All body locations will be relative to this origin. Also keep in mind that if the vehicle is spinning, the Z axis will be assumed to be the axis of rotation.

Card 3 - RPP Input (Format 6E10.5)

This card gives the coordinates of the six bounding planes of the rectangular parallelepiped which encloses all other geometry. Each plane must be perpendicular to one of the coordinate axes and is, therefore, specified by a single coordinate. The order in which they are given are

$X_{minimum}$, $X_{maximum}$, Y_{min}, Y_{max}, Z_{min}, Z_{max}.

These boundaries should be very large compared to the dimensions of the satellite.

Card Set 4 - Body Input (Format 2X, A4, 4X, 6E10.5)

These cards describe the location and dimensions of each body in the geometry. A total of NBODY bodies must be described. The code will assign a number to each body which is determined by the order in which the bodies are described. The RPP will be body #1, the first body described here is body #2, the second is body #3, and so on. Each body description will require one or more cards depending on the body type. The format of each card is as follows.

Columns Input

1-2 The code ignores these columns but they may be used to number the bodies in order to help keep the deck in proper sequence.

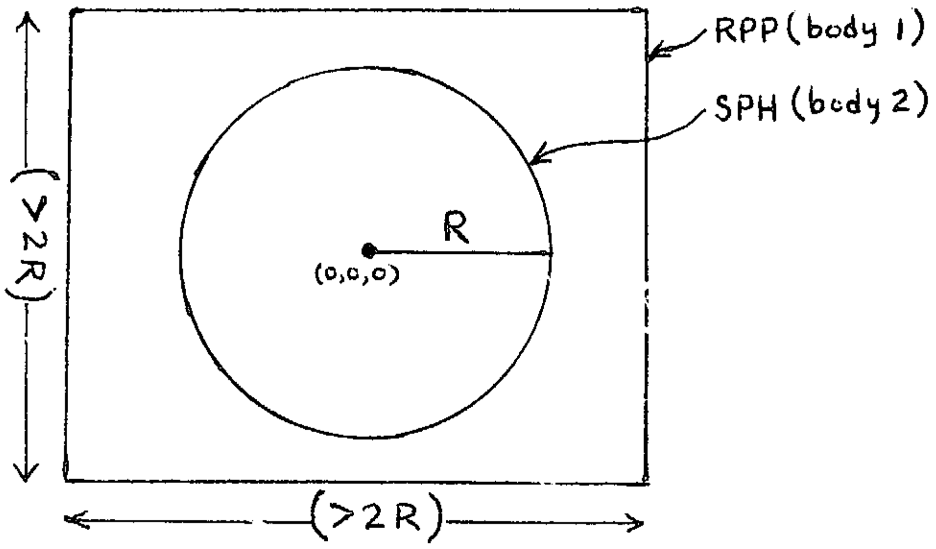
3-6 Column 3 is blank and is followed by a three letter identifier giving the body type (required input only on the first card used to describe a body).

Columns	Input
7-10	The code ignores these columns.
11-70	Divided into six fields of 10 columns each. Body data are entered here.

The following table gives the required input for each body type. Section 2.1 describes the meaning of each body variable.

		Card Columns						
body type	4-6	11-20	21-30	31-40	41-50	51-60	61-70	card #
box	BOX -	V_x L_{2x}	V_y L_{2y}	V_z L_{2z}	L_{1x} L_{3x}	L_{1y} L_{3y}	L_{1z} L_{3z}	1 of 2 2 of 2
sphere	SPH	V_x	V_y	V_z	R	-	-	1 of 1
circular cylinder	RCC -	V_x R	V_y -	V_z -	H_x -	H_y -	H_z -	1 of 2 2 of 2
elliptic cylinder	REC -	V_x R_{1x}	V_y R_{1y}	V_z R_{1z}	H_x R_{2x}	H_y R_{2y}	H_z R_{2z}	1 of 2 2 of 2
ellipse	ELL -	V_{1x} L	V_{1y} -	V_{1z} -	V_{2x} -	V_{2y} -	V_{2z} -	1 of 2 2 of 2
truncated cone	TRC -	V_x R_1	V_y R_2	V_z -	H_x -	H_y -	H_z -	1 of 2 2 of 2
wedge	RAW -	V_x L_{2x}	V_y L_{2y}	V_z L_{2z}	L_{1x} L_{3x}	L_{1y} L_{3y}	L_{1z} L_{3z}	1 of 2 2 of 2
arbitrary poly-hedron	ARB - - -	V_{1x} V_{3x} V_{5x} V_{7x}	V_{1y} V_{3y} V_{5y} V_{7y}	V_{1z} V_{3z} V_{5z} V_{7z}	V_{2x} V_{4x} V_{6x} V_{8x}	V_{2y} V_{4y} V_{6y} V_{8y}	V_{2z} V_{4z} V_{6z} V_{8z}	1 of 5 2 of 5 3 of 5 4 of 5
	enter the 4-digit face descriptions in columns 1-30 using Format 6(I4,1X)							5 of 5

It should be pointed out that, in addition to the RPP, one other body which is not part of satellite is required. That is a large sphere, centered at (0,0,0), of very large radius compared to the satellite dimensions. This sphere must, however, be enclosed in the RPP. It is a good idea to describe this body before any others, making it body #2. The geometry would then look like ...



The radius R can be determined only after all bodies describing the satellite have been defined. Find that point on the satellite which is furthest from origin and call its radial distance from the origin S. Then R>20S. The purpose of this large sphere will become clear later on.

Card Set 5 - Region Descriptions (Format I5, 1X, 9 (A2,I5))

Each region must be numbered and described by the combination of bodies which make up that region. A total of NRMAX regions must be described. The input format is indicated below.

Columns	Input
1-5	Region number (start with 1)
6	Blank
7-8	Insert the OR operator if needed. Otherwise leave blank.
9-13	Body number preceded by a + or - operator The + operator may be omitted since a blank is an inferred +.
14-69	Divided into eight fields of 7 columns, each being similar to columns 7-13. Thus, up to nine bodies can appear on one region description card.

Use as many of the above type as needed to describe a region but leave columns 1-6 blank on all continuation cards. Start each region on a new card.

The last card of Set 5 should contain a -1 in columns 4 and 5. This instructs the code that all regions have been described.

In the program, regions 1 and 2 have special meanings. Region 1 is the volume between the RPP and the large sphere (body 2). Its description is, thus, 1-2. Region 2 is the volume inside the sphere but outside the satellite bodies. Its description must start with a +2. All rays will originate in region 2 and be fired toward the satellite. If, however, the ray misses the satellite, it will pass into region 1. Any ray which enters region 1 is immediately terminated since it can no longer be of interest.

One might wonder, at this point, where the sun and planet come into the problem. The answer is, that as far as the geometry description is concerned, these two bodies do not exist. There is no requirement that regions 1 or 2 be large enough to include them.

Card 6 - Number of Reflection Coefficient Sets (Format I10)

Enter in column 10 the quantity NSET, which equals the number of reflection coefficient sets to be used in the problem. NSET can be any integer from 1 through 5.

Card Set 7 - Surface Proper+ies (Format 2I6, 10F6.4)

These cards give the reflection properties of each region. Use one card per region and a total of NRMAX cards. The input quantities are defined as follows.

Columns		Input
1-6	IR	Region number
11-12	IPROP	Region type
		For a transparent region enter -1
		For a diffuse region enter 0
		For a specular region enter 1
13-18	RC_1	Reflection coefficient for solar radiation.
19-24	RC_2	Reflection coefficient for planet radiation.

Columns	Input
25-72	Divided into eight fields of 6 columns each and used to enter additional reflection coefficient sets. For example, enter RC_1 for set #2 in columns 25-30 and RC_2 for set #2 in columns 31-36, etc. The last column number used should equal 12+2x6xNSET.

Card 8 - Imaginary Vehicle Sphere (Format E10.3)

Enter SRAD in columns 1-10.

After the vehicle geometry has been defined, determine the smallest sphere, centered at (0,0,0), which can enclose the entire vehicle. SRAD is the radius of this sphere. All rays will be fired at this sphere so the smaller it is, the more chance a ray will have of hitting the vehicle. On the other hand, if a vehicle region extends beyond the sphere it may not be hit at all, depending on the vehicle orientation. Note that this sphere is imaginary in the sense that it is not actually described as part of the Combinatorial Geometry input.

Card 9 - Orbit Parameters (Format I10, 4E10.3)

Enter the following quantities on this card.

Column		Input
1-10	IORB	Orbit Type. Enter 1 for inertial orbit, 2 for gravity gradient orbit, 3 for spin stabilized orbits.

Column		Input
11-20	RP	Altitude from the planet surface to the perigee point of the orbit.
21-30	RA	Altitude from the planet surface to the apogee point.
31-40	TAU	Orbit period.
41-50	PLRAD	Planet radius.

Card 10 - Solar Orientation (Format 3E10.3)

Determine the X,Y,Z components of a unit vector which points from the coordinate system origin to the sun. (i.e., enter the direction cosines of the vehicle - sun line)

Column		Input
1-10	WS_x	X component
11-20	WS_y	Y component
21-30	WS_z	Z component

Note that $WS_x^2+WS_y^2+WS_z^2 = 1.0$

Card 11 - Planet Orientation (Format 3E10.3)

The same as card 10 except that the unit vector points from the vehicle to the planet center.

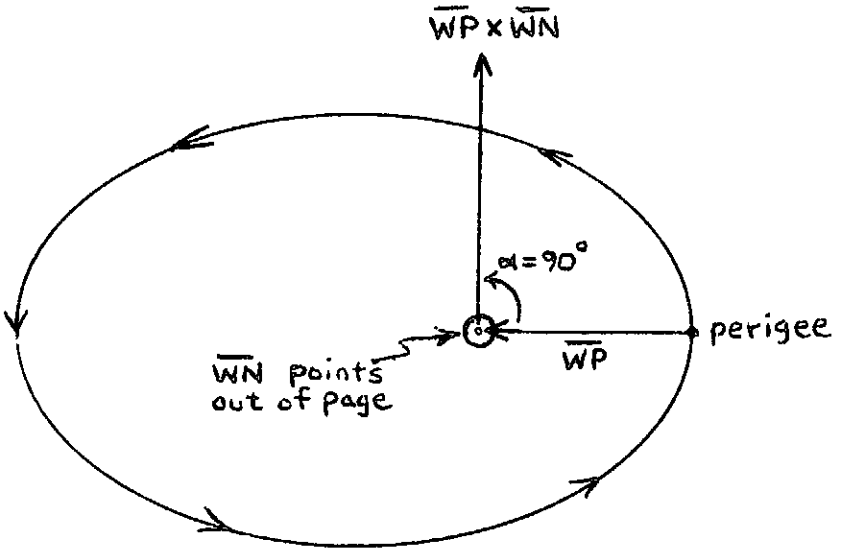
Column		Input
1-10	WP_x	X component
11-20	WP_y	Y component
21-30	WP_z	Z component

Card 12 - Orbit Orientation (Format 3E10.3)

Enter the three direction cosines of the line which is normal to the orbital plane.

Column		Input
1-10	WN_x	X component
11-20	WN_y	Y component
21-30	WN_z	Z component

The direction of $\overline{WN}$ should be such that the vector $\overline{WP} \times \overline{WN}$ points toward an orbital angle of 90 degrees, as shown below.


$\overline{WP} \times \overline{WN}$
$\alpha = 90°$
perigee
$\overline{WP}$
$\overline{WN}$ points out of page

Card 13 - Source Information (Format 2E10, 4E10.3)

Enter the following six quantities.

Column		Input
10	ISGO	Solar Option. Enter 1 if the solar source is desired and 0 if it is not.
20	IPGO	Planet Option. Enter 1 if planet source is desired and 0 if it is not.
21-30	SOLINT	Solar Constant (energy/area-time)
31-40	ALBEDO	Albedo of planet (fraction of sunlight reflected)
41-50	STEF	Stefan-Boltzmann Constant (energy/area-time-degree)
51-60	TEMP	Radiating temperature of planet (absolute)

If the planet option is not desired (IPGO=0), columns 31-60 may be left blank.

Card 14 - Debug Printout Options (Format 6I3)

Certain subroutines in the program contain provisions for obtaining intermediate printouts during the calculation. These were inserted to aid in program debugging and were left in should any unresolved bugs turn up. Normally, these printouts would not be desired and Card 14 should be included in the input deck but left blank. If they are desired, however, the input numbers go into an IDBUG array defined as follows.

Column		Input
1-3	IDBUG(1)	Any positive integer will cause subroutine UMBRA to print the minimum and maximum orbital angles where the vehicle is shaded by the sun.

Column   Input

4-6 IDBUG(2) Causes TIMER to print the range of angles covered by a location in orbit and the flight times from perigee to these angles. The value of IDBUG(2) is the number of locations where this is desired.

7-9 IDBUG(3) Causes EXPO to print certain variables. However, these require a detailed knowledge of the coding to interpret. EXPO is entered once per location and the printout will appear IDBUG(3) times.

10-12 IDBUG(4) Causes STRAK and SATREF to print information describing the progress of an individual solar ray. It is possible to follow a ray from its origin to its termination, including any vehicle reflections. The value of IDBUG(4) is the number of rays you wish to follow at each location. A knowledge of the coding would be required, however, to interpret the printout.

13-15 IDBUG(5) Causes WMAKER to print the direction cosines of the sun and planet at each orbital location. The printout will occur IDBUG(5) times.

16-18 IDBUG(6) Same definition as IDBUG(4) but follows rays from the planet.

Card 15 - Parameters for Solar Calculation (Format 4I10, 2E10.3)

These parameters describe how the solar calculation should be run. If ISGO=0, omit card 15.

Column	Input
1-10 NLOC	The number of orbital locations to be treated
11-20 NRS	Number of rays to be traced from the sun at each location (discussed in Section 3.4).
30 IVRGO	Vehicle reflection option. Enter 1 if rays are to be reflected from vehicle. Enter 0 if not.
31-40 NRDIFF	Number of reflected rays to be emitted per diffuse reflection. If IVRGO=0 or if there are no diffuse surfaces, enter any small positive integer (i.e., 1) but don't leave blank.
41-50 ALPH1	Initial orbital angle to be treated (need not be 0)
51-60 DALPH	Angular interval between locations (i.e., if ALPH1=20° and DALPH=10°, the second location in orbit will be at 30°).

Card 16 - Vehicle Spin Parameters for Solar Calculation
(Format I10, 2E10.3)

These parameters control vehicle rotation for spin-stabilized orbits. If ISGO=0, omit card 16.

Column	Input
1-10 NSPIN	Number of spin orientations (about the Z axis) to be treated at each orbital location. If IORB=1 or 2 (no spin), enter a 1.
11-20 THET1	Initial spin angle (measured from X axis). If IORB=1 or 2, enter 0.0.
21-30 DTHET	Angular interval between spin orientations. If IORB=1 or 2, enter 0.0.

Note that the value of NSPIN does not affect the number of rays to be fired at a given orbital location. At each location, NRS/NSPIN rays are fired a total of NSPIN times.

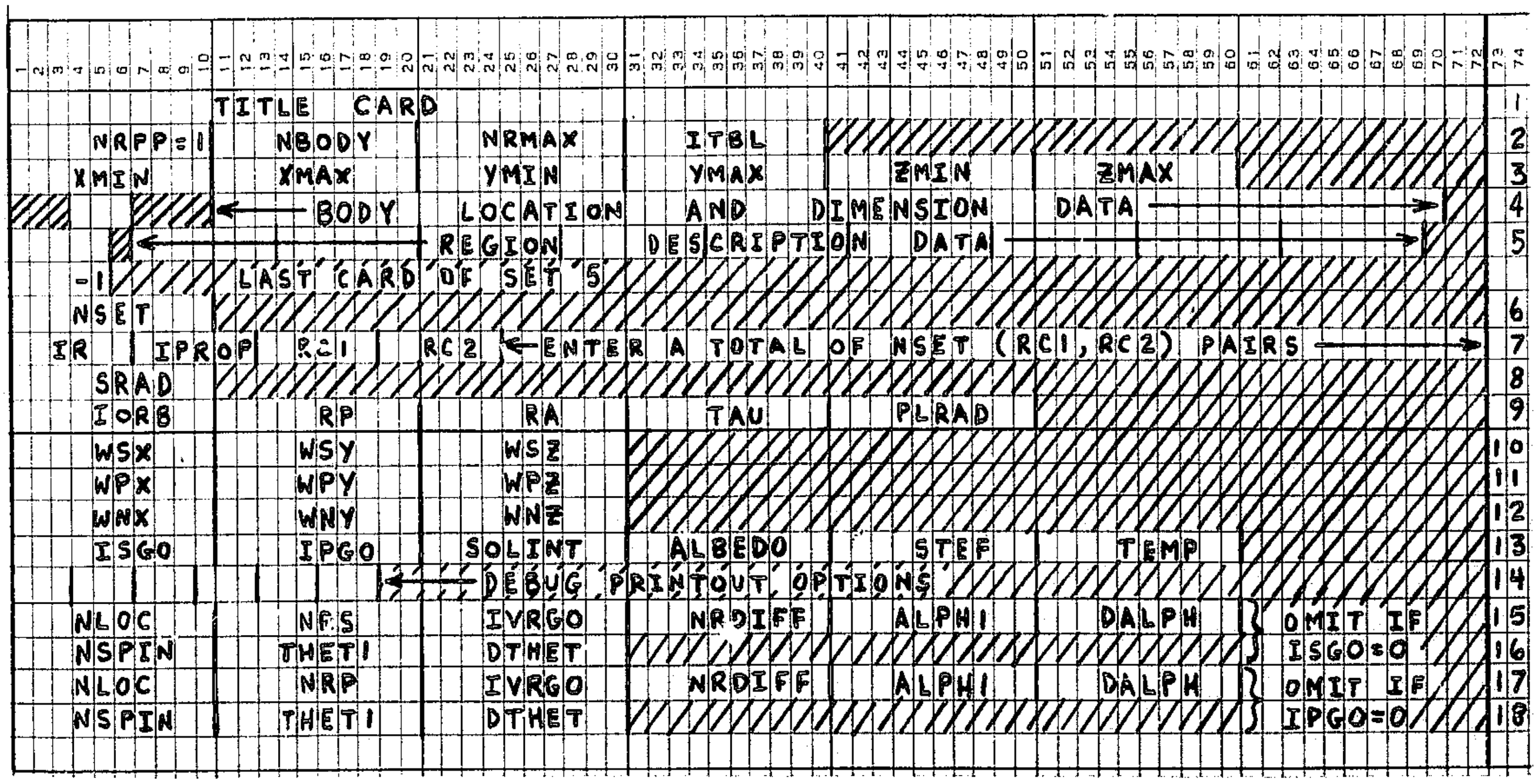
Card 17 - Parameters for Planet Calculation (Format 4I10, 2E10.3)

This card is analogous to Card 15 for solar rays. The six parameters may be entirely different for the planet calculation, however. Note that NRS is called NRP by the program at this point and equals the number of planet emitted rays per location. If IPGO=0, omit card 17.

Card 18 - Vehicle Spin Parameters for Planet Calculation (Format I10, 2E10.3)

Analogous to card 16 but different values may be used if IORB=3. If IPGO=0, omit card 18.

SAMPLE INPUT FORM



	TITLE CARD						1
NRPP=1	NBODY	NRMAX	ITBL				2
XMIN	XMAX	YMIN	YMAX	ZMIN	ZMAX		3
	← BODY	LOCATION	AND	DIMENSION	DATA →		4
	←	REGION	DESCRIPTION	DATA →			5
-1	LAST CARD	OF SET 5					
NSET							6
IR IPROP	RC1	RC2 ← ENTER	A TOTAL	OF NSET (RC1,	RC2) PAIRS →		7
SRAD							8
IORB	RP	RA	TAU	PLRAD			9
WSX	WSY	WSZ					10
WPX	WPY	WPZ					11
WNX	WNY	WNZ					12
ISGO	IPGO	SOLINT	ALBEDO	STEF	TEMP		13
	←	DEBUG PRINTOUT	OPTIONS				14
NLOC	NFS	IVRGO	NRDIFF	ALPH1	DALPH	} OMIT IF	15
NSPIN	THET1	DTHET				} ISGO=0	16
NLOC	NRP	IVRGO	NRDIFF	ALPH1	DALPH	} OMIT IF	17
NSPIN	THET1	DTHET				} IPGO=0	18

3.3 Error Messages

There are numerous places in the code where a printout will occur upon detection of an input error. The job will terminate but the source of the error will be indicated. There is one possible error which does not terminate a job. It occurs while tracing a ray in subroutine G1. In general, the cause of this error is that a ray gets 'lost' at the boundary between two regions. This is due to insufficient computing precision of the machine. If this occurs, some error printout will appear and the code will abandon this ray and go on to the next one. As long as this is a relatively rare event, the user need not be concerned with it. Experience has shown that it might be expected once every few thousand rays. If, however, it occurs much more often than this, the cause is probably an undetected input geometry error. The printout can usually provide clues to the exact source of the error but lacking a detailed knowledge of the coding, the best thing to do is to carefully go over the input. The error is likely either in a body dimension or a region description.

3.4 Selecting the Number of Rays

The accuracy of the results is determined by the number of rays fired (NRS from the sun, NRP from the planet). To estimate a reasonable value for NRS it is first necessary to select a desired average density of rays (rays/projected area). Since NRS rays will be fired at the imaginary vehicle sphere of radius SRAD, the density, D, will be $D = NRS/\pi(SRAD)^2$ or $NRS = \pi D(SRAD)^2$.

The value for D is a function of the surface area of the important vehicle regions. If, for example, a region of interest is only 1 in^2 in area, an average density of 1 ray/in^2 is inadequate, since there is a good chance that no rays will hit that region. In test problems involving the RAE satellite, however, a D of about 1 was adequate. The area of the solar paddles was about 100 in^2 so that, on average, 100 rays would strike a paddle that was fully exposed to the sun. The number of 'hits' would still be significant even for partial exposure. Since SRAD was 42 in. for that problem, the above equation (assuming D=1.0) would give NRS = 5500. The actual number used in the calculation was 6000.

Before leaving this discussion, there is one procedure that is wise to follow when running a problem. That is, to first run a short test problem to insure that the input represents the problem that is actually desired and that the computed results are at least reasonable. Even though the results are inaccurate, one can usually spot order of magnitude type errors. This trial run might use 5-10% of the number of rays selected for the final run.

3.5 Output Edit

The output format for the main results was described in Section 1.6. Additional output not discussed there, as well as all input quantities, are clearly labelled in the edit and require no explanation.

3.6 Estimated Running Times

CPU time for a given problem depends largely on three items.

a) Source options selected,

b) Number of rays to be fired,

c) Complexity of the geometry.

Since these depend on the particular problem, no general rule can be made for computing time.

However, one can get a fair idea from test problems run on Goddard's 360/91 for the RAE satellite geometry. In these calculations, solar rays were traced at the rate of about 4500 rays/minute. The same rate should apply to planet emitted rays. If vehicle reflection were included, computing speed would probably be cut roughly in half. Using these estimates and the above ray tracing rate it is possible to write a crude equation for the computing time, T.

$$T=(\mathrm{ISGO}+\mathrm{IVRGO})(\mathrm{NLOC})_S(\mathrm{NRS}/4500)+(\mathrm{IPGO}+\mathrm{IVRGO})(\mathrm{NLOC})_P(\mathrm{NRP}/4500).$$

where ISGO, IPGO, IVRGO are the source options desired (0 or 1) and NLOC is the number of locations where rays will actually be traced (i.e., for inertial orbits $(\mathrm{NLOC})_S=1$, regardless of the number of locations where answers are desired).

Estimates can also be made by scaling the running time of the trial problem, recommended earlier.

Remember, however, that a problem can be restarted in the middle so little is lost if it overruns the estimated time.

4. SUBROUTINE DESCRIPTIONS

This section provides brief descriptions of each subroutine in the code. The routines are discussed in the following order.

MAIN
GENI
UN3
BODY ROUTINES
RPP
ARBLAD
CROSS
DOT
DCOSP
XDIST
UNIT
S
RPP2
GJ
NORMAL

SHADOW
UMBRA
UMBRA 2
TIMER
EXPO
WCHEK
RANUM
CMAKER
WMAKER
SUN
STRAK
PLANET
PTRAK
SATREF

MAIN

Function:

Initialize certain variables and start the calculation.

Description:

After initializing variables the MAIN routine reads the problem title from cards, calls subroutine GENI to read and process geometry input and calls SHADOW to read the remaining input and begin the calculation.

GENI

Function:

Read, process, and store geometry input data.

Description:

GENI processes all input describing the geometry of the satellite vehicle. The routine also does some preliminary checking of the input for format, duplicate numbering, etc. The RPP input is read and processed first followed by the data for each body in the geometry. Region descriptions are then read in from which the entering and leaving tables described earlier are constructed. At the completion of GENI all geometry data are housed in the ASTER array and the code is ready to begin ray tracing.

UN3 (KK, LK1, LK2, LK3)

Function:

Unpacks the packed data stored in the ASTER array.

Call Arguments:

KK = location of a word in the ASTER array (input).

LK1, LK2, LK3 = the three unpacked components of word KK (output).

Description:

UN3 contains the following three entry point routines for unpacking the ASTER array.

A. UNNWW unpacks a word originally packed under a 5-bit - 13 bit - 13 bit format.

B. UNWNW unpacks information in a 13 bit - 5 bit - 13 bit word.

C. UNWWN unpacks information in a 13 bit - 13 bit - 5 bit word.

Note that the last three letters in the entry point name describe the packed format, where W refers to 'wide' (13 bits) and N to 'narrow' (5 bits).

BODY ROUTINES

Function:

Computes the distance fr[illegible] the origin of a ray to entering and leaving surfaces of a given body.

Call Arguments:

None

Description:

The code contains seven independent body routines, with each performing a similar calculation for a different body type. Given that a ray is initially at point $\overline{XB}$ and traveling in a direction $\overline{WB}$, each routine computes the distance RIN to the first (entering) intersection point with the body and the distance ROUT to the second (leaving) intersection point. The calculation for a given body may result in one of the following three possible combinations for RIN and ROUT.

1. RIN>0, ROUT>0 XB is outside the body and the ray intersects the body (2 intersections found)

2. RIN≤0, ROUT>0 XB is inside the body (1 intersection found)

3. RIN≤0, ROUT≤0 the ray misses the body completely

The following is a list of each routine and the body type it treats.

ELL - ellipsoid

RCC - right circular cylinder

REC - right elliptic cylinder

SPH - sphere

TRC - truncated cone

ARB - arbitrary polyhedron (also handles the box and wedge which were converted to polyhedrons during input processing)

RPP (NBO)

Function:

Performs the function of a body routine for a rectangular parallelepiped.

Call Arguments:

NBO = body number of the rectangular parallelepiped.

ARBLAD (X, NUMN, IWC)

Function:

Processes and stores arbitrary polyhedron (ARB) input and converts box (BOX) or wedge (RAW) input to ARB format.

Call Arguments:

X = body input numbers

NUMN = size of the X array

IWC = informs ARBLAD which body type is being processed
(1 if ARB, 2 if RAW, 3 if BOX).

Description:

The original implementation of Combinatorial Geometry treated the ARB, BOX, and RAW as separate and distinct body types with each having its own body (or ray tracing) routine. Subsequently, it was found that ray tracing for an ARB was faster than for the other two bodies. At the same time, however, body input for the BOX and RAW was often simpler to specify. Thus, it was decided to maintain separate input formats but, once read into the machine, input for these two bodies would be converted to ARB format. The BOX and RAW ray tracing routines were thereby eliminated from the code. ARBLAD performs this body conversion and also processes ARB input.

CROSS (ANSWER, FIRST, SECOND)

Function:

Computes the vector product of two vectors.

Call Arguments:

ANSWER = a vector giving the cross product

FIRST, SECOND = two input vectors.

Description:

ANSWER(1) = FIRST(2)*SECOND(3) - FIRST(3)*SECOND(2)

ANSWER(2) = FIRST(3)*SECOND(1) - FIRST(1)*SECOND(3)

ANSWER(3) = FIRST(1)*SECOND(2) - FIRST(2)*SECOND(1)

DOT (FIRST, SECOND)

Function:

Computes the scalar product of two vectors.

Call Arguments:

FIRST, SECOND = two input vectors.

Description:

$$\text{DOT} = \sum_{i=1}^{3} \text{FIRST}_i * \text{SECOND}_i$$

DCOSP (XA, XB, WA)

Function:

Computes the direction cosines of a line between two points.

Call Arguments:

WA = 3 direction cosines from point XA to point XB

XA, XB = subscripted arrays giving the (x,y,z) coordinates of XA and XB.

Description:

$WA_i = (XB_i - XA_i)/DIS$, where DIS is the distance between XA and XB.

XDIST (XA, XB)

Function:

Computes the distance between two points.

Call Arguments:

XA, XB = subscripted arrays giving the (x,y,z) coordinates of points XA and XB.

Description:

$$XDIST = \left[\sum_{i=1}^{3} (XA_i - XB_i)^2 \right]^{\frac{1}{2}}$$

UNIT (VECTOR)

Function:

Converts an arbitrary vector in space to a unit vector.

Call Arguments:

VECTOR = 3 components of any vector.

Description:

$VECTOR_i = VECTOR_i/VECT$, where

$VECT = (VECTOR \cdot VECTOR)^{\frac{1}{2}}$

S (I, N)

Function:

Retrieves the coordinate of any of the six sides of an RPP from the ASTER array.

Call Arguments:

I = number of the RPP

N = side (1 to 6) whose coordinate is desired.

Description:

The routine uses the input values I and N to compute the location (LL) of the desired coordinate in the ASTER array. Then S = ASTER(LL).

RPP2 (JSURF, XP, IRP)

Function:

Determines the number of the next region a ray will encounter when it leaves one RPP (rectangular parallelepiped) and enters another.

This routine was needed for other implementations of the Combinatorial Geometry technique, where more than one RPP could appear in the geometry. This version of the technique permits only one RPP to be specified so that subroutine RPP2 is never called. It has been left in the program, however, to facilitate possible future modifications.

G1 (S1, IRPRIM, XP)

Function:

Computes the next region in the path of a ray and the point of intersection with that region.

Call Arguments:

IRPRIM = next region which the ray will encounter after it leaves the region it is currently in.

XP = x,y,z coordinates of intersection point with IRPRIM.

S1 = distance between the ray's current location and the point XP (not used by the code).

Description:

The logic employed by G1 is described in the section that discusses the Combinatorial Geometry technique (section 2).

NORMAL (XI, WN)

Function:

Computes the components of the normal to the surface of a given body.

Call Arguments:

XI = point on the surface of a body at which the normal is desired.

XN = direction cosines of the outward directed normal at point XI.

Description:

Given a body number NBO, a surface number LSURF (both in COMMON), and a point on the surface XI, subroutine NORMAL computes the direction cosines of a vector normal to the required surface at that point. The routine is made up of independent sections each of which treats a different body type. It should be noted that the normals to the six sides of each arbitrary polyhedron (including the box and wedge) are precomputed and stored in ASTER during input processing and need only be retrieved by this routine. The normal to a sphere is merely the direction cosines of the line joining its center and the point XI. The remaining bodies employ straightforward mathematics for determining the orientation of a given surface with respect to a set of coordinate axes. Since this routine is only called when reflecting a ray off the surface of a body, it is important that the computed normal be directed outward from the surface. This is assured by setting the signs of WN such that the dot product WN·WB is negative, where WB is the direction of the incident ray.

SHADOW

Function:

Read and process non-geometry input and directs the overall flow of the calculation.

Description:

SHADOW is the input processor for all problem dependent input (i.e., that which is independent of the actual satellite geometry).

It reads, and in some cases checks for correctness, data on the reflection properties of region surfaces, input describing the orbit, source intensities, and the various calculation options available to the user. These options determine which sources are to be treated in the problem.

UMBRA

Function:

Computes the range of orbital angles in which the satellite is in the shadow of the planet.

Description:

This routine computes two angles, SHAD1 and SHAD2, which define the lower and upper limits of the planet's shadow. The angles are determined to the nearest degree and measured from the perigee line. UMBRA is the control routine for this calculation, while subroutine UMBRA2 performs the actual computations for a given angle.

UMBRA2 (ANG, ISEE)

Function:

Determines whether a given orbital angle is in the shadow of the planet.

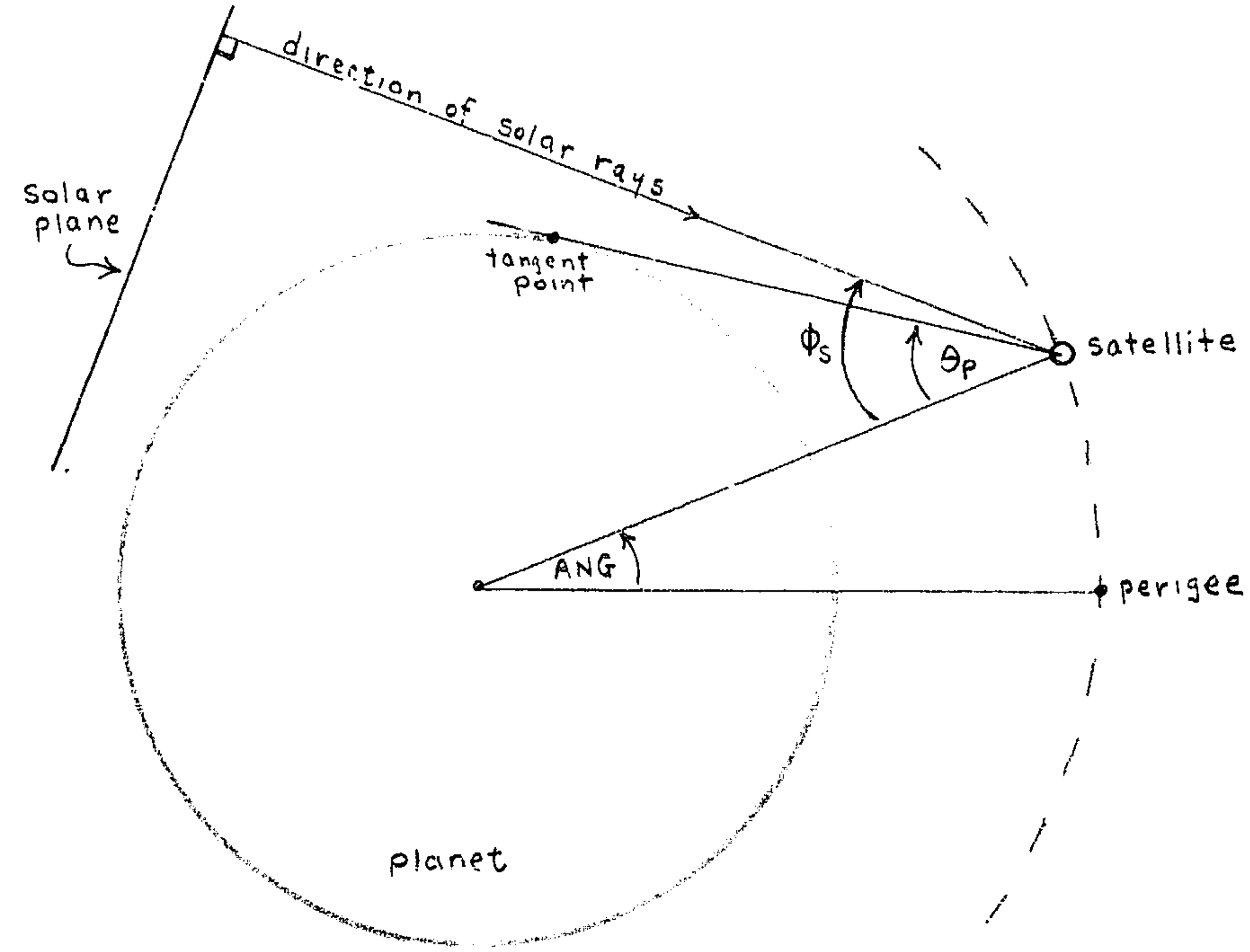
Call Arguments:

ANG = orbital angle in degrees (input).

ISEE = output flag indicating whether ANG is in shade.
If ISEE = 0, ANG is shaded. If ISEE = 1, ANG is exposed to sunlight.

Description:

UMBRA2 has the task of computing whether the given orbital angle (ANG) lies within the planet's shadow. This is done by computing the two angles θ_p and ϕ_s as defined by the following sketch.



The vehicle is shaded if $\theta_p \geq \phi_s$ and ISEE is set to 0. If $\theta_p < \phi_s$ the vehicle is in sunlight and ISEE = 1.

TIMER (ALPHA, DT)

Function:

Computes the vehicle flight time between two orbital angles.

Call Arguments:

ALPHA = orbital location of the vehicle measured in degrees from perigee.

DT = time spent at location ALPHA.

Description:

This routine computes the time spent in the angular interval between $(\alpha-\Delta\alpha/2)$ and $(\alpha+\Delta\alpha/2)$, where α = ALPHA and $\Delta\alpha$ is the angle between locations in orbit. The equations for computing DT are derived in Appendix 2.

EXPO (ALPHA, FEX)

Function:

Computes the fraction of time that the vehicle spends in sunlight at a given orbital location.

Call Arguments:

ALPHA = orbital location of the vehicle measured in degrees from perigee.

FEX = fraction of time exposed to sunlight.

Description:

This routine calculates the fraction of time exposed to sunlight by comparing the angular interval for a given location with the angles SHAD1 and SHAD2 (see Subroutine UMBRA). FEX is computed from the degree of overlap of these two intervals. It should be noted that EXPO is not an exact calculation since what is actually computed is the fraction of the angular interval in sunlight rather than the fraction of time. This is, however, an excellent approximation for relatively small angular intervals or low eccentricity orbits, since in either case the angular velocity is practically constant within the interval. In problems where a high degree of accuracy is desired, small angular intervals will be employed and any errors introduced by this approximation will be insignificant.

WCHEK (W, ICALL)

Function:

Checks and adjusts direction cosines.

Call Arguments:

W = any set of 3 direction cosines.

ICALL = a flag indicating which routine has called WCHEK.

Description:

This routine tests that no gross errors have been made in an input or computed direction cosine set. It first tests whether $\sum_{i=1}^{3} W_i^2 = 1.0 \pm 0.02$. If this test is passed the W values are assumed to be correctly given and the divergence from 1.0 (less than .02) is due to round off errors. W is then normalized so that the sum of the squares is precisely 1.0. If the above test fails, an error has occurred and the routine will print an error message containing W and ICALL and terminate the job with a STOP2.

RANUM (RN)

Function:

Generates random numbers from 0 to 1.

Call Arguments:

RN = random number.

Description:

This routine supplies random floating point numbers in the range 0.0 to 1.0. It is essentially the IBM random number generator library subroutine which makes use of the machine overflow characteristics.

CMAKER (W,C)

Function:

Computes certain constants used by other routines.

Call Arguments:

W = a set of 3 direction cosines supplied by the calling routine.

C = a set of 8 constants computed from W.

Description:

The constants generated by this routine are used by several other routines when a rotation of coordinate systems is required. The equations for these constants are derived in Appendix I for a generalized coordinate system rotation.

WMAKER (ALPHA, JCALL)

Function:

Computes the direction cosines of the vehicle-sun line or vehicle-planet line.

Call Arguments:

ALPHA = given orbital angle

JCALL = flag informing WMAKER as to which direction cosines are to be computed.

Description:

This routine computes the variables WSA and/or WPA at a given orbital angle, where

WSA = direction cosines from vehicle to solar plane,

WPA = direction cosines from vehicle to planet center.

The value of JCALL supplied by the calling routine determines both the variable to be computed and the orbit type as follows.

JCALL	Compute	Orbit	Calling Routine
1	WSA	gravity gradient	SUN
2	WSA,WPA	inertial	PLANET
3	WSA,WPA	gravity gradient	PLANET
4	WSA,WPA	spin stabilized	PLANET

The equations for WSA and WPA are derived in Appendix 3 .

SUN

Function:

Controls the calculation for solar emission.

Description:

Subroutine SUN controls the solar emission calculation and is entered just once during a problem. It also converts the ray tracing results to the proper form and prints the answers. The logic flow depends primarily on the orbit type (IORB) specified for the problem. Given that NRS rays are to be traced from the sun at each orbital location, the calculation proceeds as follows.

IORB	
1 (inertial)	Trace NRS rays at one location and apply the results to all locations.
2 (gravity gradient)	Trace NRS rays at each orbital location.
3 (spin stabilized)	Trace NRS rays at one location but divide them equally over NSPIN spin orientations. Apply results to all locations.

Note that for IORB = 1 or 3 rays are traced at only one location since the sun-vehicle orientation is constant for these orbits. The following is a simplified flowchart of SUN with the following variable definitions.

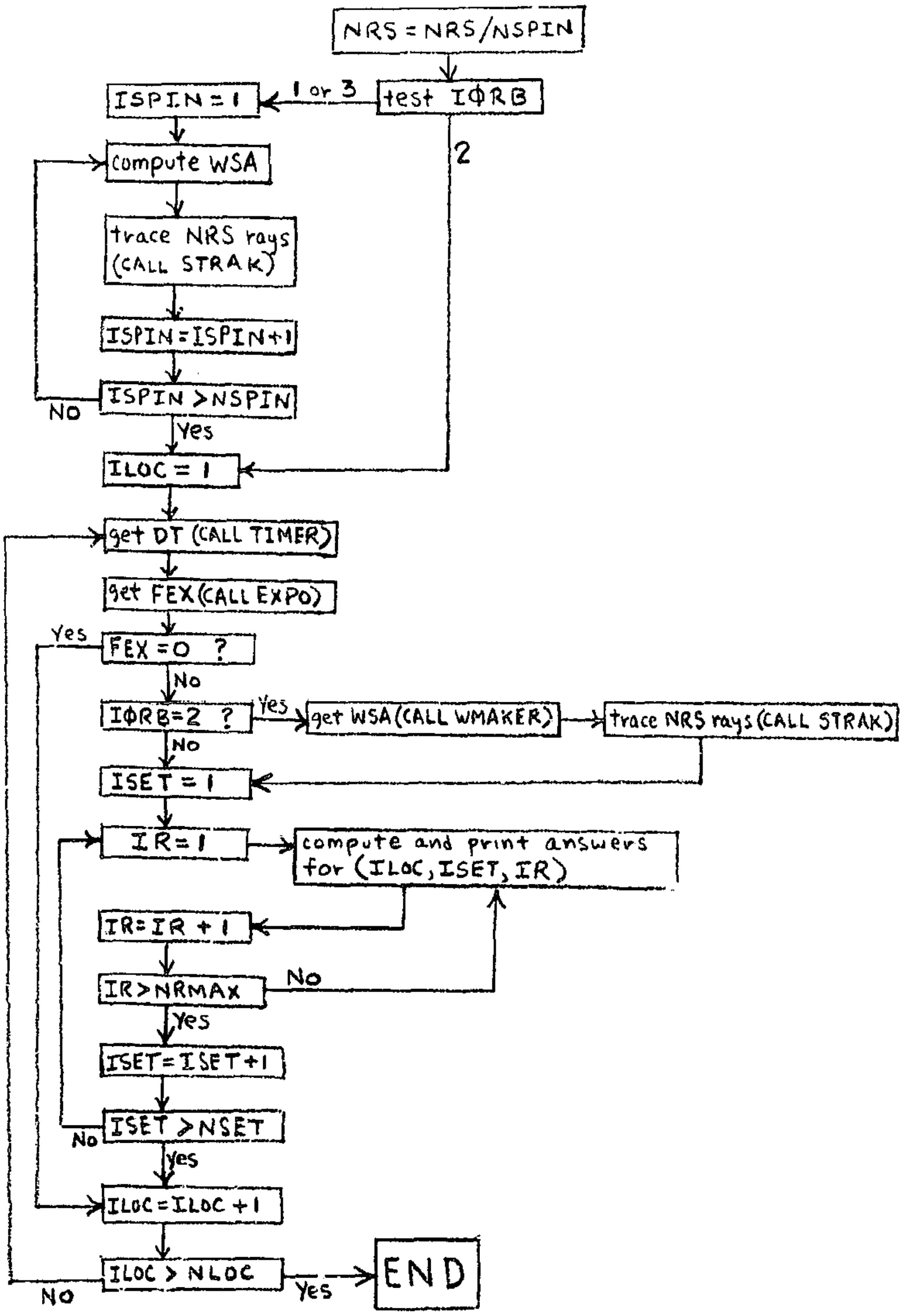
Variable	Definition	Maximum Value
ISPIN	spin orientation number	NSPIN*
ILOC	orbital location number	NLOC
ISET	reflection coefficient set number	NSET
IR	vehicle region number	NRMAX

*NSPIN = 1 for IORB = 1 or 2.

WSA = direction cosines from vehicle to sun.

DT = time spent by vehicle at location ILOC.

FEX = fraction of time exposed to sunlight at ILOC.



SUN Flowchart

STRAK (ISPIN)

Function:

Performs the ray tracing from the sun.

Call Arguments:

ISPIN = spin orientation number.

Description:

STRAK traces rays from the sun and computes the incident and absorbed energy contributions for each ray. The routine is called by subroutine SUN once for each location and a total of NRS rays are traced. The starting position of each ray is picked at random on the solar plane (a plane perpendicular to vehicle-sun line). The equations used for this purpose are derived in Appendix 4. The following flowchart illustrates the calculational procedure in STRAK, where the variable definitions are:

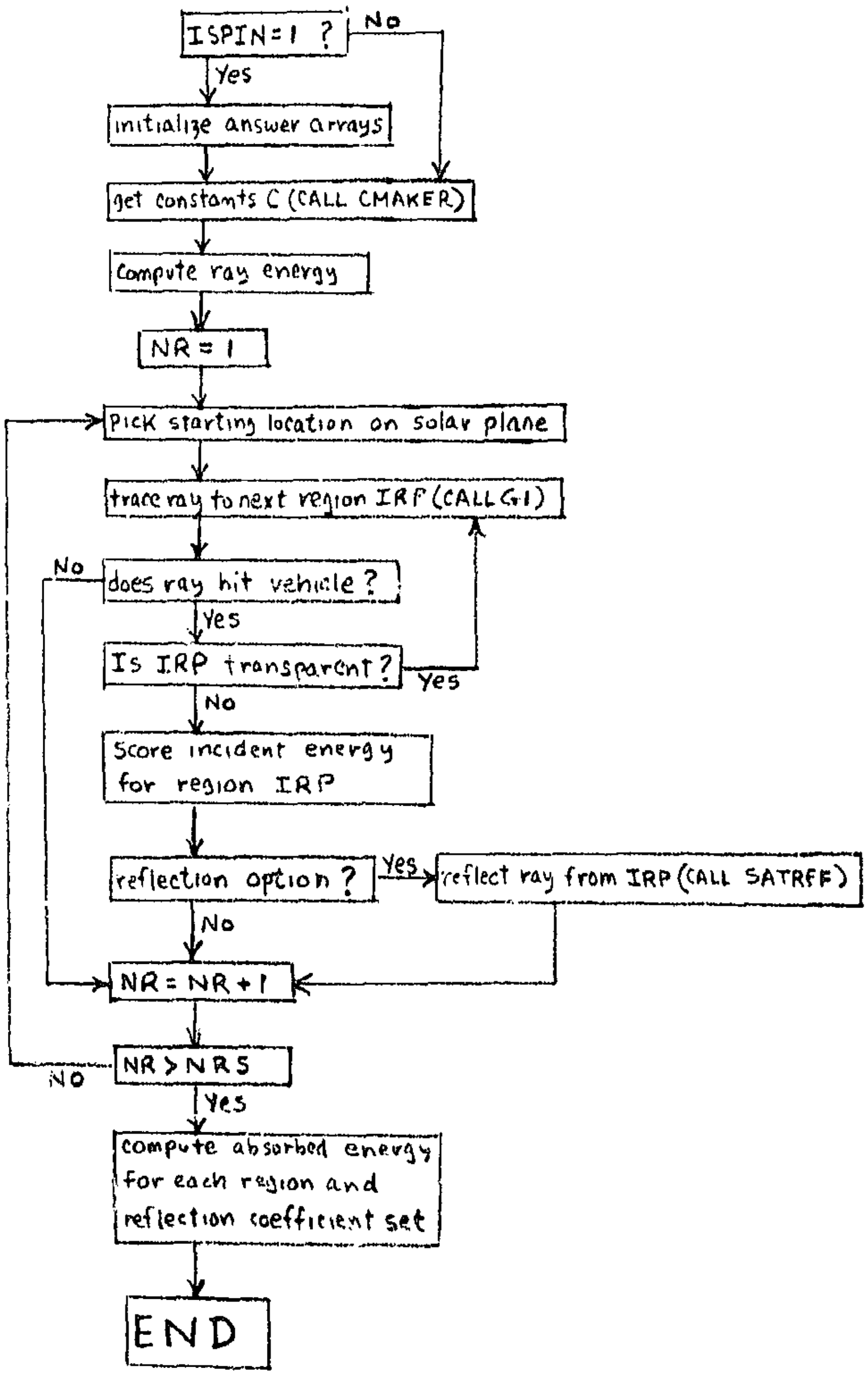
S_O = solar constant,

SRAD = radius of imaginary sphere around vehicle,

NSPIN = number of spin orientations to be treated,

NR = ray number,

IRP = next region in the ray's path.


ISPIN=1 ?
No
Yes
initialize answer arrays
get constants C (CALL CMAKER)
Compute ray energy
NR = 1
Pick starting location on solar plane
trace ray to next region IRP (CALL G1)
No
does ray hit vehicle ?
Yes
Is IRP transparent ?
Yes
No
Score incident energy for region IRP
reflection option ?
Yes
reflect ray from IRP (CALL SATREF)
No
NR = NR + 1
NR > NRS
NO
Yes
compute absorbed energy for each region and reflection coefficient set
END

STRAK Flowchart

PLANET

Function:

Controls the calculation for planet emission and reflection.

Description:

PLANET is the control and output routine for the planet emission and reflection calculation and is entered once during a problem. Its function is, therefore, analogous to that of subroutine SUN for solar emission and employs similar logic. The computation is again controlled by the orbit type (IORB) as follows.

IORB

1 or 2 Trace NRP rays at each location.

3 Trace NRP rays at each location but divide them equally among NSPIN spin orientations.

Note that rays are traced at every location even for IORB = 2 (gravity gradient) since planet reflection is location dependent for all orbit types.

PTRAK (ALPHA, ISPIN)

Function:

Performs the ray tracing from the planet.

Call Arguments:

ALPHA = orbital angle at current location.

ISPIN = spin orientation number.

Description:

PTRAK is called once by subroutine PLANET for each orbital location to perform the ray tracing from the planet. The overall logic is similar to that found in subroutine STRAK but the computations are somewhat more complex due mainly to planet emitted rays not having a constant direction. The equations used in this routine are derived in Appendix 5. The flowchart which follows illustrates the procedure used in STRAK where

NR = ray number,

NRP = total number of rays to be traced,

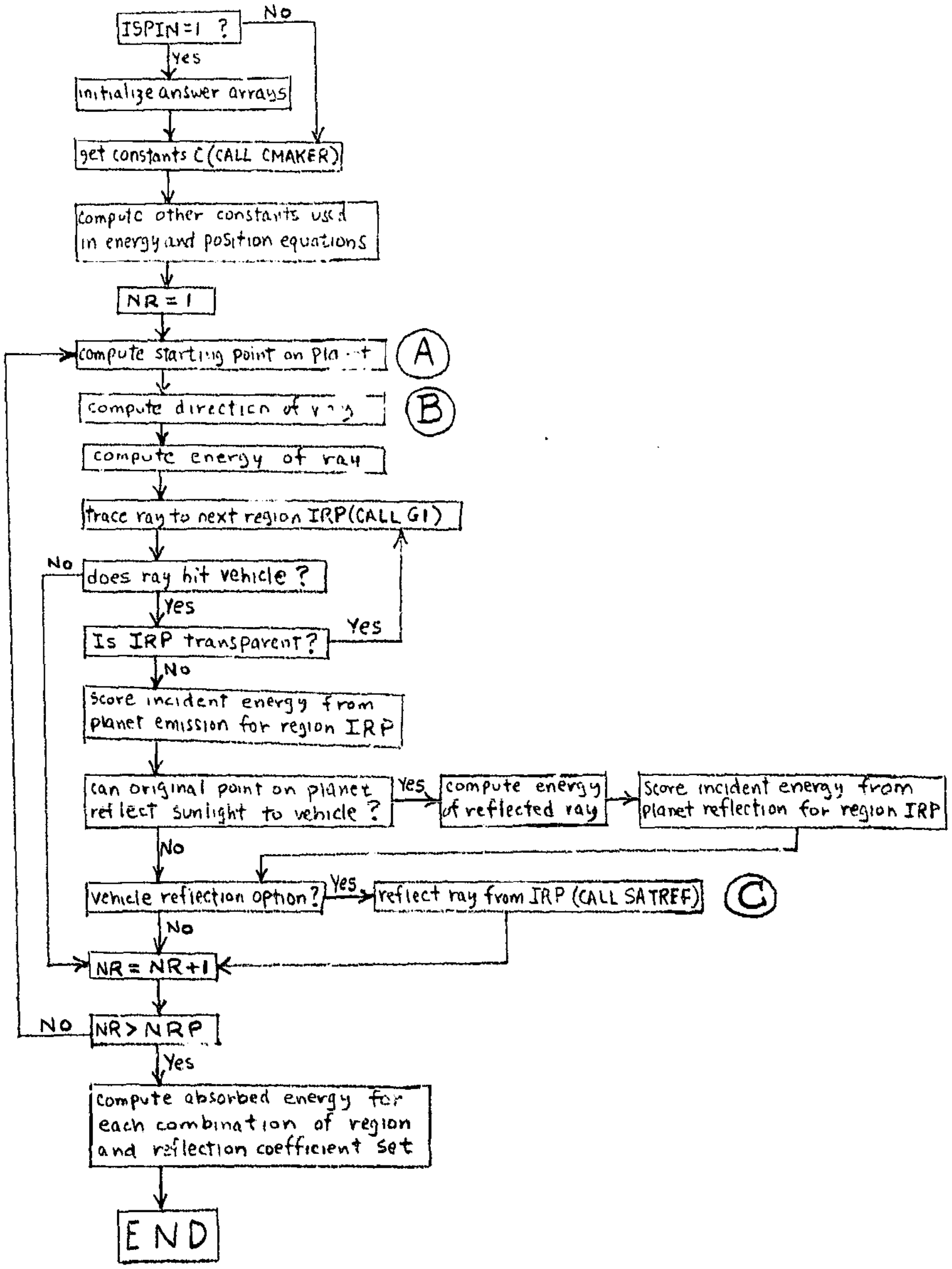
IRP = next region in the path of a ray.

Some of the boxes in the flowchart deserve some additional comments and these have been identified by letters.

Box A: The area of the planet surface visible to the vehicle is determined and the origin of a ray is picked at random within this area.

Box B: The direction of a ray is computed by picking a random point to shoot at within the imaginary vehicle sphere. That point and the origin point of the ray determine the ray's direction cosines.

Box C: Both the planet emission and planet reflection energies are supplied to SATREF. The ray to be reflected from the vehicle is then treated as a single ray but having the above two energy components.


ISPIN=1 ?
No
Yes
initialize answer arrays
get constants C (CALL CMAKER)
compute other constants used in energy and position equations
NR = 1
compute starting point on planet
A
compute direction of ray
B
compute energy of ray
trace ray to next region IRP (CALL G1)
No
does ray hit vehicle ?
Yes
Is IRP transparent ?
Yes
No
Score incident energy from planet emission for region IRP
can original point on planet reflect sunlight to vehicle ?
Yes
compute energy of reflected ray
Score incident energy from planet reflection for region IRP
No
Vehicle reflection option ?
Yes
reflect ray from IRP (CALL SATREF)
C
No
NR = NR+1
No
NR > NRP
Yes
Compute absorbed energy for each combination of region and reflection coefficient set
END

PTRAK Flowchart

SATREF (IRP, XP, E, EREF)

Function:

Performs ray tracing for vehicle reflected rays.

Call Arguments:

IRP = reflecting region number.

XP = reflection point on IRP.

E,EREF = energy components of the incoming ray. If called from STRAK, E = 0 and EREF = incident solar energy. If called from PTRAK, E = energy from direct planet emission and EREF = energy from planet reflection.

Description:

SATREF traces a primary ray through a series of vehicle reflections. It is called by either STRAK (solar rays) or PTRAK (planet rays) each time a ray is to be reflected. The routine is divided into two independent sections, the first of which treats specular reflections while the second handles diffuse reflection. The property of the primary reflecting surface (IRP) determines which section is entered.

Specular Reflection - The ray is reflected off the primary surface such that the angles of incidence and reflection are equal. Multiple reflections are permitted by continuing to track and reflect the ray until one of the following conditions occurs.

a) The ray 'escapes' from the vehicle.

b) The reflected ray strikes a diffusing surface.

c) The ray has undergone 5 specular reflections.

During the tracking, a list of successive regions hit by the ray is built up. After the ray is terminated, the routine steps through this list, computing incident, absorbed, and reflected energies for each region and reflection coefficient set.

Diffuse Reflection - If the primary reflector (IRP) is a diffusing surface, the routine traces NRDIFF rays from the reflection point, where NRDIFF is given as input. During input processing (subroutine SHADOW) a set of NRDIFF angular bins was established relative to the surface normal and one ray is emitted in each of these bins. In addition, the fraction of reflected energy as a function of bin angle was computed by SHADOW. The energy of the reflected ray is then simply the product of (incident energy) X (fraction emitted in bin) X (reflection coefficient). This energy is actually the sum of the contributions from E and EREF.

The mathematical formulation of subroutine SATREF is given in Appendix 6 . A flowchart is shown below with the following variable definitions.

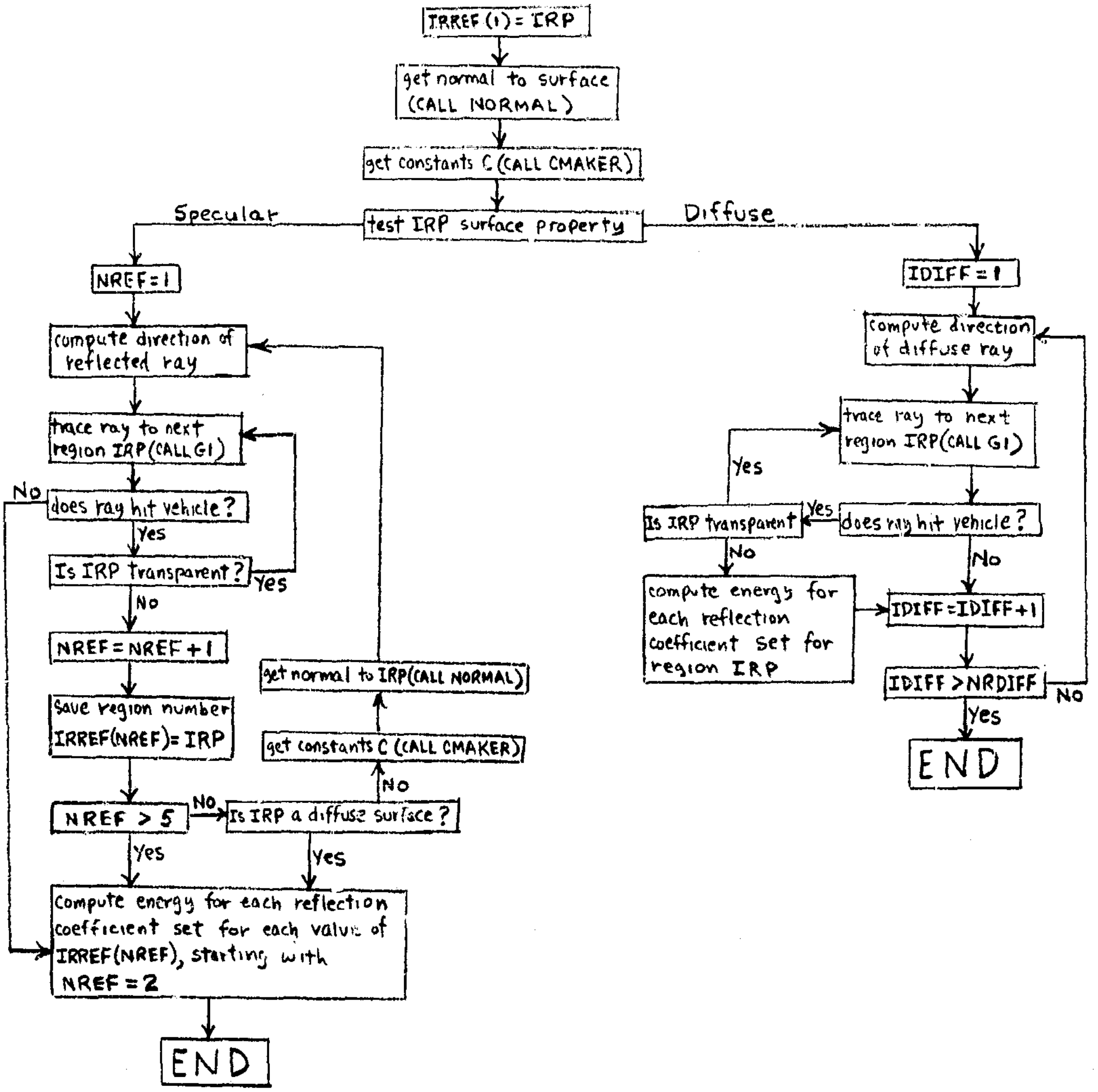
IRP = next region in ray's path (initial value is the primary reflecting region).

NREF = a counter on the number of specular reflections.

IRREF(NREF) = region hit by the ray at the NREFth reflection.

IDIFF = diffuse ray number.

NRDIFF = total diffuse rays to be emitted.

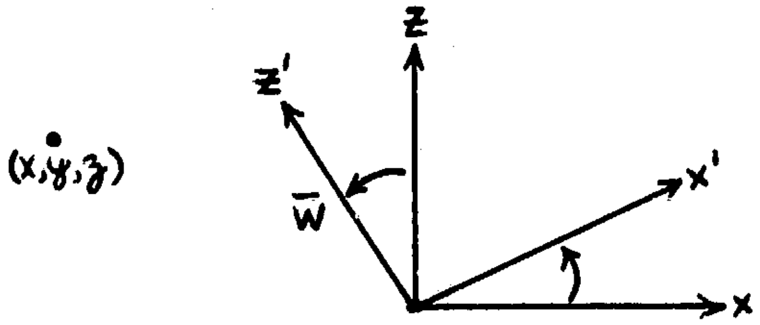

IRREF (1) = IRP
get normal to surface (CALL NORMAL)
get constants C (CALL CMAKER)
Specular
test IRP surface property
Diffuse
NREF=1
compute direction of reflected ray
trace ray to next region IRP(CALL G1)
No
does ray hit vehicle?
Yes
Is IRP transparent?
Yes
No
NREF=NREF+1
Save region number IRREF(NREF)=IRP
get normal to IRP(CALL NORMAL)
get constants C (CALL CMAKER)
No
NREF > 5
No
Is IRP a diffuse surface?
Yes
Yes
Compute energy for each reflection coefficient set for each value of IRREF(NREF), starting with NREF=2
END
IDIFF=1
compute direction of diffuse ray
trace ray to next region IRP(CALL G1)
Yes
Is IRP transparent
Yes
does ray hit vehicle?
No
No
compute energy for each reflection coefficient set for region IRP
IDIFF=IDIFF+1
IDIFF >NRDIFF
No
Yes
END

SATREF Flowchart

Appendix 1 - Rotation of Coordinates

The problem is to define a point in space (x,y,z) as a function of a given vector $\bar{W}$. The resulting equations for (x,y,z) are required by several routines in the program. For example, STRAK uses the results to pick a point on the solar firing plane relative to the direction cosines of the line joining the vehicle and sun.

The required equations can be derived by rotating the original X,Y,Z coordinate system into a primed system X',Y',Z' such that the Z' axis points along the vector $\bar{W}$. This is illustrated below in two dimentions.



The coordinates of the point (x,y,z) in the unprimed can be obtained from the following set of relationships for a generalized rotation of coordinates.

$$x = x'\cos\alpha_1+y'\cos\alpha_2+z'\cos\alpha_3$$

$$y = x'\cos\beta_1+y'\cos\beta_2+z'\cos\beta_3$$

$$z = x'\cos\gamma_1+y'\cos\gamma_2+z'\cos\gamma_3$$

where (x',y',z') are the coordinates of the point in the primed system and α β, γ are the angles between the primed and unprimed axes. These angles are defined as follows.

Unprimed Axes

	X	Y	Z
X'	α_1	β_1	γ_1
Y'	α_2	β_2	γ_2
Z'	α_3	β_3	γ_3

The problem, then, is to derive expressions for α, β, γ.

The primed axes, normalized to unit vectors, can be written as

$$X' = \frac{X - W_X Z'}{\sqrt{1 - W_X^2}} \qquad Y' = \frac{[Z'X]}{\sqrt{1 - W_X^2}} \qquad Z' = Z$$

where W_x is the X component of $\overline{W}$.

Since Z' has components W_x, W_y, W_z, the vector product $[Z'X]$ can be evaluated as

$$[Z'X] = \begin{vmatrix} i & j & k \\ W_X & W_Y & W_Z \\ 1 & 0 & 0 \end{vmatrix} = 0 + W_Z j - W_Y k$$

Now, letting $C_1 = \sqrt{1-W_x}^2$, the cosines of α, β, γ are given by the following expressions.

$$\cos\alpha_1 = x'\cdot x = \frac{x\cdot x - W_x z'\cdot x}{C_1} = \frac{1 - W_x W_x}{C_1}$$

$$\cos\beta_1 = x'\cdot y = \frac{x\cdot y - W_x z'\cdot y}{C_1} = -\frac{W_x W_y}{C_1}$$

$$\cos\gamma_1 = x'\cdot z = \frac{x\cdot z - W_x z'\cdot z}{C_1} = -\frac{W_x W_z}{C_1}$$

$$\cos\alpha_2 = y'\cdot x = \frac{W_z j\cdot x - W_y k\cdot x}{C_1} = 0$$

$$\cos\beta_2 = y'\cdot y = \frac{W_z j\cdot y - W_y k\cdot y}{C_1} = \frac{W_z}{C_1}$$

$$\cos\gamma_2 = y'\cdot z = \frac{W_z j\cdot z - W_y k\cdot z}{C_1} = -\frac{W_y}{C_1}$$

$$\cos\alpha_3 = z'\cdot x = W_x$$

$$\cos\beta_3 = z'\cdot y = W_y$$

$$\cos\gamma_3 = z'\cdot z = W_z$$

The original equations for (x,y,z) can now be written in terms of $\overline{W}$.

$$x = x'\sqrt{1-W_x^2} + z'W_x = x'C_1 + z'C_2$$

$$y = x'\left(-\frac{W_x W_y}{\sqrt{1-W_x^2}}\right) + y'\left(\frac{W_z}{\sqrt{1-W_x^2}}\right) + z'W_y = x'C_3 + y'C_4 + z'C_5$$

$$z = x'\left(\frac{-W_x W_z}{\sqrt{1-W_x^2}}\right) + y'\left(\frac{-W_y}{\sqrt{1-W_x^2}}\right) + z'W_z = x'C_6 + y'C_7 + z'C_8$$

The 8 constants C_i are computed by subroutine CMAKER for any given set W_x, W_y, W_z.

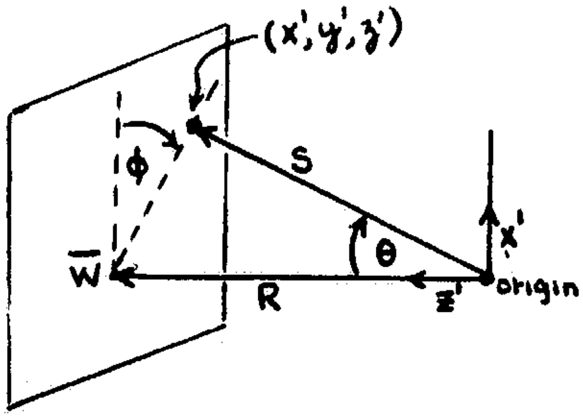
The values of x',y',z' will depend on the particular application but, in general, are given by the standard equations for a point in spherical coordinates.

$x' = S \sin\Theta \cos\phi$

$y' = S \sin\Theta \sin\phi$

$z' = R$

where the variables are defined by the following figure



Appendix 2 - Flight Time in Orbit

The problem is to compute the vehicle flight time between perigee and any orbital angle α . The variables used in the derivation are defined as follows.

t = time at angle α

t_o = time at α = o

P = period of orbit

e = eccentricity of orbit

L_p = altitude at perigee (measured from center of planet)

L_a = altitude at apogee (measured from center of planet)

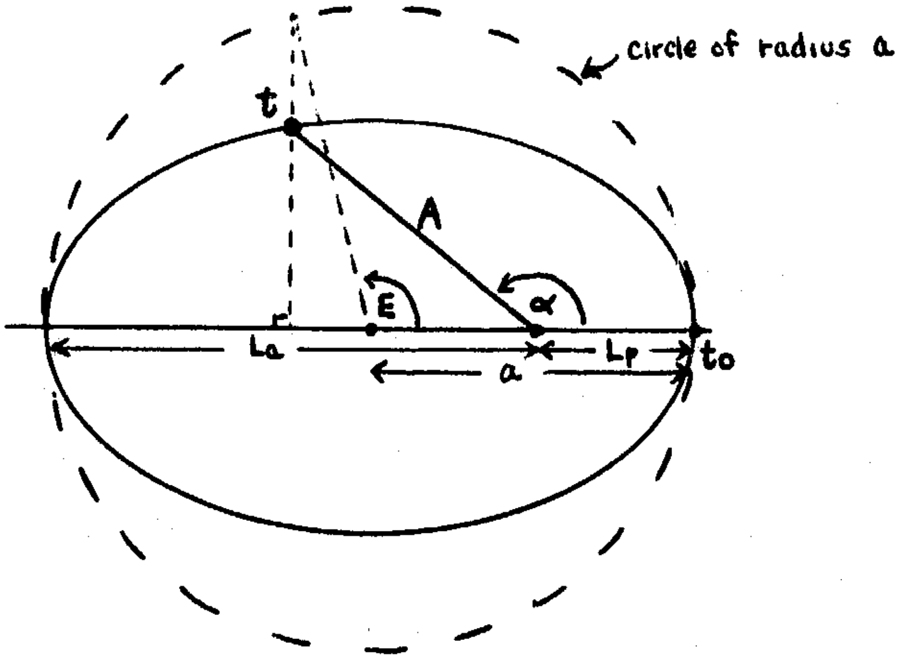
A = altitude at angle α (measured from center of planet)

a = semi-major axis = $(L_a+L_p)/2$

E = an angle referred to as the 'eccentric anomaly' and defined by the relationship

$$\cos E \equiv (a-A)/ea. \tag{1}$$

The following figure illustrates the geometry.



Using Kepler's equation for the orbital flight time, we have

$$(t-t_o) = \frac{P}{2\pi} (E-e \sin E). \qquad (2)$$

All that is required is to convert the above equation to a function of α .

We, first, define the eccentricity as

$e = (L_a-L_p)/(L_a+L_p)$.

Then, substituting for a and E into equation (1) we have

$$\cos E = (L_a+L_p-2A)/(L_a-L_p) \qquad (3)$$

But, from the polar equation of an ellipse

$$A = \frac{a(1+e^2)}{1-e\cos\alpha} = \frac{2L_aL_p}{(L_a+L_p)+(L_a-L_p)\cos\alpha} \ . \qquad (4)$$

Substituting for A in equation (3) gives

$$\cos E = \frac{1}{(L_a-L_p)} \cdot \left[(L_a+L_p) - \frac{4L_aL_p}{(L_a+L_p)+(L_a-L_p)\cos\alpha}\right]$$

Then, since

$E = \cos^{-1} (\cos E)$ and

$\sin E = (1-\cos^2 E)^{\frac{1}{2}}$,

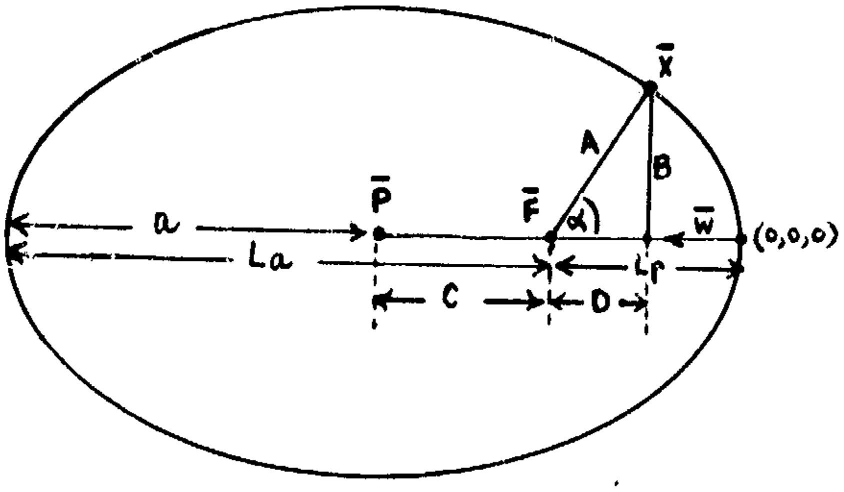
Kepler's equation can be put in terms of α and the three input constants P, L_a and L_p.

In subroutine TIMER, the quantity dt is computed, where dt is the flight time from α_1 to α_2. This is done simply by evaluating Kepler's equation at the two angles and taking the difference.

Appendix 3 - Deriving the Satellite-Sun-Planet Orientations for Different Orbits

A. Find the Direction Cosines of the Satellite-Planet Line for an Inertial Orbit

Consider the following figure, where the satellite is at point $\bar{x}$ with an orbital angle α .



The main problem is to derive the coordinates at $\bar{x}$ as a function of α , given

a) the direction cosines of the major axis = $\bar{w}$

b) altitude at perigee (from planet center) = L_p

c) altitude at apogee (from planet center) = L_a

d) direction cosines of normal to orbital plane = $\bar{n}$

The point $\bar{x}$ in vector notation is given by

$$\bar{x} = \bar{P} + \bar{\epsilon}B + (-\bar{w})\ (c+d), \tag{1}$$

where $\bar{\epsilon} = \bar{w} \times \bar{n}$.

From the above figure, it is obvious that

$$a = (L_a+L_p)/2$$

$$c = (L_a-L_p)/2$$

$$B = A \sin\alpha$$

$$D = A \cos\alpha$$

Also the point $\overline{P}$ at the center of the ellipse is equal to

$$\overline{w}a = \frac{\overline{W}}{2}(L_a+L_p).$$

Substituting for $\overline{P}$, C, B, D in equation (1) gives

$$\overline{X} = \overline{\epsilon}\ (A\ \sin\alpha)\ -\overline{w}\left[\left(\frac{La}{2} - \frac{Lp}{2} + A\cos\alpha\right) - \left(\frac{La}{2} + \frac{Lp}{2}\right)\right].$$

After combining terms

$$\overline{X} = \overline{\epsilon}\ (A\ \sin\alpha)\ +\overline{w}(L_p - A\ \cos\alpha). \qquad (2)$$

The definition of $\overline{\epsilon}$ gives

$$\overline{\epsilon} = \overline{w} \times \overline{n} = \begin{vmatrix} i & j & k \\ w_x & w_y & w_z \\ n_x & n_y & n_z \end{vmatrix} = (w_y n_z - w_z n_y)i + (w_z n_x - w_x n_z)j + (w_x n_y - w_y n_x)k$$

The components of equation (2) are now obtained by substituting the i, j, k components of $\overline{\epsilon}$. Thus,

$$x = (w_y n_z - w_z n_y)\ \ (A\ \sin\alpha) + w_x(L_p - A\ \cos\alpha)$$

$$y = (w_z n_x - w_x n_z)\ \ (A\ \sin\alpha) + w_y(L_p - A\ \cos\alpha)$$

$$z = (w_x n_y - w_y n_x)\ \ (A\ \sin\alpha) + w_z(L_p - A\ \cos\alpha). \qquad (3)$$

Having defined the point $\overline{X}$, it is now a simple matter to derive the direction cosines $\overline{WPA}$ of the line from point $\overline{X}$ to point $\overline{F}$. First, define the point $\overline{F}$ as

$$\overline{F} = \overline{X} + (\overline{WPA})A\ .$$

Then

$$\overline{WPA} = (\overline{F} - \overline{X})/A.$$

But $\overline{F}$ can also be written as

$$\overline{F} = L_p\overline{w}.$$

Thus,

$$\overline{WPA} = (L_p\overline{w}-\overline{x})/A.$$

Finally, substituting for $\overline{x}$ from equation (3) we get for the three components of $\overline{WPA}$

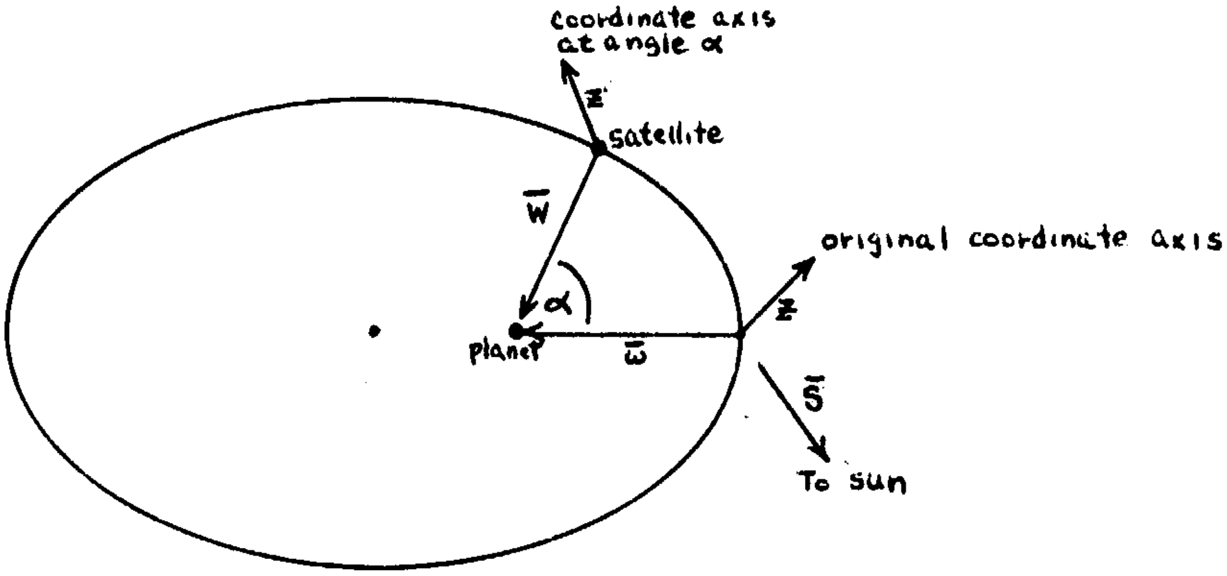
$$WPA_x = w_x \cos\alpha + (w_z n_y - w_y n_z) \sin\alpha$$

$$WPA_y = w_y \cos\alpha + (w_x n_z - w_z n_x) \sin\alpha$$

$$WPA_z = w_z \cos\alpha + (w_y n_x - w_x n_y) \sin\alpha \quad .$$

B. Find the Direction Cosines of the Satellite-Sun Line for a Gravity Gradient Orbit

Consider the following figure which defines the variables in the derivation.



For clarity, only the Z and Z' axes are shown in the above figure and the details of the derivation will be given for the Z components of the direction cosines.

The problem is to determine the direction cosines of the satellite-sun line at location α with respect to the Z' axis. Let this quantity be S_z'.

The following quantities are given:

a) direction cosines of the major axis = $\bar{w}$

b) direction cosines of sun at perigee (α =o) = $\bar{S}$

c) direction cosines of earth at α = $\bar{W}$

d) direction cosines of normal to orbital plane = $\bar{N}$

In general, the angle ϕ between two vectors $\overline{V}$ and $\overline{V}'$ is given by

$\cos \phi = V_x V_x' + V_y V_y' + V_z V_z'$ where

V_x = direction cosines of $\overline{V}$ with respect to the x axis, etc.

Then, the cosine of the angle between $\overline{S}$ and the Z' axis is

$$S_Z' = S_x Z_x' + S_y Z_y' + S_z Z_z' \quad \text{where} \qquad (1)$$

Z_x', Z_y', Z_z' = direction cosines of Z' with respect to the original x, y, z axes.

Since S_x, S_y, S_z are given, the problem is to find Z_x', Z_y', Z_z'. In order to solve for the three components of the Z' axis, the following three equations are required.

1. We first make use of the fact that for a gravity gradient orbit the relationship between $\overline{w}$ and Z is the same as between $\overline{W}$ and Z'. Thus, the cosine of the angle between $\overline{W}$ and Z' = w_z. Then

   $$w_z = W_x Z_x' + W_y Z_y' + W_z Z_z' \qquad (2)$$

2. Also note that the orientation of the normal ($\overline{N}$) is invariant. Thus, N_Z = cosine of the angle between $\overline{N}$ and Z'.

   $$N_Z = N_x Z_x' + N_y Z_y' + N_z Z_z' \qquad (3)$$

3. Finally, let M_Z be the cosine of the angle between $\overline{w}$ and Z'. Then

   $$M_Z = w_x Z_x' + w_y Z_y' + w_z Z_z' \qquad (4)$$

To find M_Z let

$\cos\beta$ = cosine of the angle between $\bar{W}$ and $Z' = w_Z$

$\cos\gamma$ = cosine of the angle between $\bar{N}$ and $Z' = N_Z$

$\cos\alpha$ = cosine of the angle between $\bar{w}$ and $\bar{W}$.

Using solid trigonometry we get

$$M_Z = \cos\alpha\ \cos\beta\ + \sin\alpha\ \sin\beta\ \cos\ (\phi+90)$$

where $\cos\phi = \cos\gamma\ /\sin\beta$ and $\cos\ (\phi+90) = -\sin\phi$.

Thus

$$\cos^2\phi = \frac{N_z^2}{1-\omega_z^2}$$

$$\sin\phi = (1-\cos^2\phi)^{1/2} = \left(\frac{1-\omega_z^2-N_z^2}{1-\omega_z^2}\right)^{1/2}$$

$$\sin\phi\sin\beta = (1-\omega_z^2-N_z^2).$$

Substitution in the above equation for M_Z gives

$$M_Z = w_z\ \cos\alpha\ - \sin\alpha\ (1-w_z^2-N_z^2)$$

Solving equations (2), (3), and (4) simultaneously gives the following set of equations for Z'_x, Z'_y, Z'_z.

$$Z'_x = \frac{A_1M_Z+B_1N_Z+C_1w_Z}{A_1w_x+B_1N_x+C_1W_x} \qquad (5)$$

$$Z'_y = \frac{A_2M_z+B_2N_z+C_2w_z}{A_2w_y+B_2N_y+C_2W_y} \qquad (6)$$

$$Z'_z = \frac{A_3M_z+B_3N_z+C_3w_z}{A_3w_z+B_3N_z+C_3W_z} \qquad (7)$$

The desired quantity S'_z is obtained by substitution into equation (1). The constants in these equations are defined as follows:

$$A_1 = (W_zN_y-W_yN_z)$$
$$B_1 = (W_yw_z-W_zw_y)$$
$$C_1 = (w_yN_z-w_zN_y)$$
$$A_2 = (W_xN_z-W_zN_x)$$
$$B_2 = (W_zw_x-W_xw_z)$$
$$C_2 = (w_zN_x-w_xN_z)$$
$$A_3 = (W_xN_y-W_yN_x)$$
$$B_3 = (W_yw_x-W_xw_y)$$
$$C_3 = (w_yN_x-w_xN_y)$$

The x, y components of S' are obtained from

$$S'_x = S_xX'_x+S_yX'_y+S_zX'_z$$

$$S'_y = S_xY'_x+S_yY'_y+S_zY'_z.$$

The equations analogous to (5), (6), (7) for $\overline{X}'$ and $\overline{Y}'$ are obtained from the same sort of analysis as above for $\overline{Z}'$. The constants A, B, C are identical and the general form of the equations are

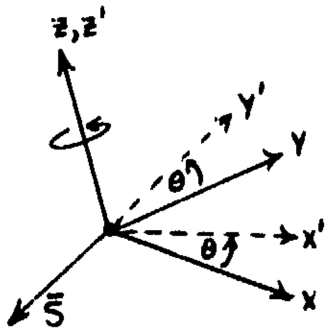
$$X_i' = \frac{A_i M_x + B_i N_x + C_i w_x}{A_i w_i + B_i N_i + C_i W_i}$$

$$Y_i' = \frac{A_i M_y + B_i N_y + C_i w_y}{A_i w_i + B_i N_i + C_i W_i}$$

where the subscripts i = 1, 2, 3 refer to the x, y, z components, respectively.

C. Find Direction Cosines of Satellite-Sun Line for a Spin Stabilized Orbit

A spin stabilized orbit is merely an inertial orbit with the satellite rotating about the Z axis. Thus, after rotating through an angle θ , the original x, y, z coordinate axes will have rotated into a new set of coordinates x', y', z'. This is shown below.



Given that the original direction cosines of the sun (at $\theta=0$) are S_x, S_y, S_z (the unit vector $\bar{S}$), the problem is to find the direction cosines with respect to x', y', z'. Let these be called S_x', S_y', S_z' and let

X_x' = X component of the X' axis

X_y' = Y component of the X' axis

X_z' = Z component of the X' axis

with similar definitions for the Y' and Z' axes.

Then

S_x' = cosine of angle between $\bar{S}$ and X' axis = $S_xX_x'+S_yX_y'+S_zX_z'$

S_y' = cosine of angle between $\bar{S}$ and Y' axis = $S_xY_x'+S_yY_y'+S_zY_z'$

S_z' = cosine of angle between $\bar{S}$ and Z' axis = $S_xZ_x'+S_yZ_y'+S_zZ_z'$

From the above figure it is obvious that

$$X'_x = \cos\theta \qquad Y'_x = -\sin\theta \qquad Z'_x = 0$$

$$X'_y = \sin\theta \qquad Y'_y = \cos\theta \qquad Z'_y = 0$$

$$X'_z = 0 \qquad Y'_z = 0 \qquad Z'_x = 1$$

Thus

$$S'_x = S_x \cos\theta + S_y \sin\theta$$

$$S'_y = -S_x \sin\theta + S_y \cos\theta$$

$$S'_z = S_z$$

D. Find Direction Cosines of Satellite-Planet Line for Spin Stabilized Orbit

Let the direction cosines of the planet at a rotation angle θ be WPA'_x, WPA'_y, WPA'_z. The same analysis as given in part C results in the following set of equations.

$$WPA'_x = WPA_x \cos\theta + WPA_y \sin\theta$$

$$WPA'_y = -WPA_x \sin\theta + WPA_y \cos\theta$$

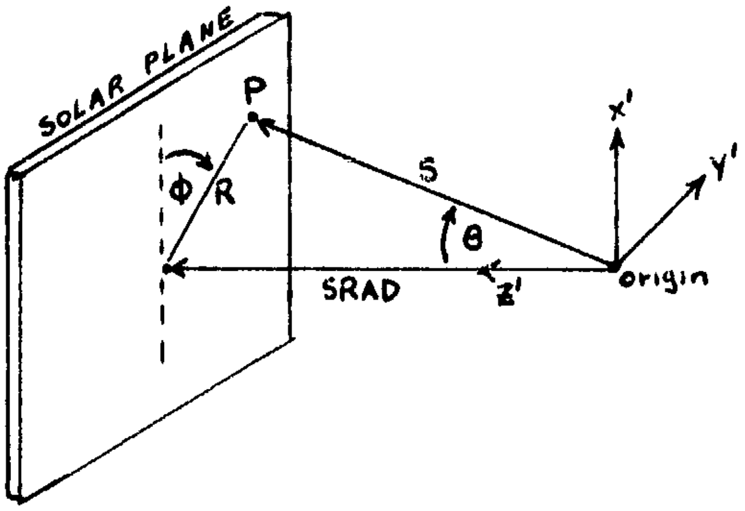
$$WPA'_z = WPA_z$$

where $WPA_{x,y,z}$ are the direction cosines of the planet at $\theta = 0$. These are derived in part A as a function of orbital angle α .

Appendix 4 - Originating a Ray From the Sun

A. Picking a Point of Origin

The Solar source is considered to be a set of plane parallel rays originating from a plane which is perpendicular to the satellite-sun line. Let the direction cosines of this line ($\overline{WSA}$) be WSA_x, WSA_y, WSA_z. The solar plane will be established at a distance SRAD from the origin of coordinates, where SRAD is the radius of the imaginary sphere enclosing the satellite. We first rotate the (X,Y,Z) coordinate system into a new system (X', Y', Z') such that the Z' axis points along the vector $\overline{WSA}$. This is shown in the following figure.



The problem is to find the coordinates of the point P relative to the original (X,Y,Z) system. To do this we apply the equations for a coordinate rotation derived in Appendix 1. Thus,

$$X = X'C_1 + Z'C_2$$
$$Y = X'C_3 + Y'C_4 + Z'C_5$$
$$Z = X'C_6 + Y'C_7 + Z'C_8$$

Where the C's are functions of WPA_x, WPA_Y, WPA_z and are defined in Appendix 1. The parameters X', Y', Z' are defined as follows

$$X' = S \sin\theta \cos\phi$$
$$Y' = S \sin\theta \sin\phi$$
$$Z' = SRAD$$

We now pick a random point per unit area on the solar plane as follows.

1. Pick a value of R between O and SRAD
   $R = \sqrt{\xi_1}\ SRAD$, where ξ_1 is a random number between 0 and 1.
   But, from the above figure $R = S \sin\theta$. Thus,
   $(S \sin\theta) = \sqrt{\xi_1}\ SRAD$
2. Pick an azimuthal angle ϕ at random
   $\phi = 2\pi\xi_2$, where ξ_2 is a second random number.

Thus

$$X' = (\sqrt{\xi_1}\ SRAD) \cos(2\pi\xi_2)$$
$$Y' = (\sqrt{\xi_1}\ SRAD) \sin(2\pi\xi_2)$$
$$Z' = SRAD \ .$$

Substitution of these value into the equations for X,Y,Z then gives the required coordinates of the point P.

B. Finding the Direction of the Ray

The ray is simply fired back toward the satellite with direction cosines $-WSA_x$, $-WSA_y$, $-WSA_z$.

C. Computing the Energy of the Ray

The energy of each ray is given by

$$E = \frac{S_o \pi \, SRAD^2}{N}$$

S_o = Solar constant (energy/area)

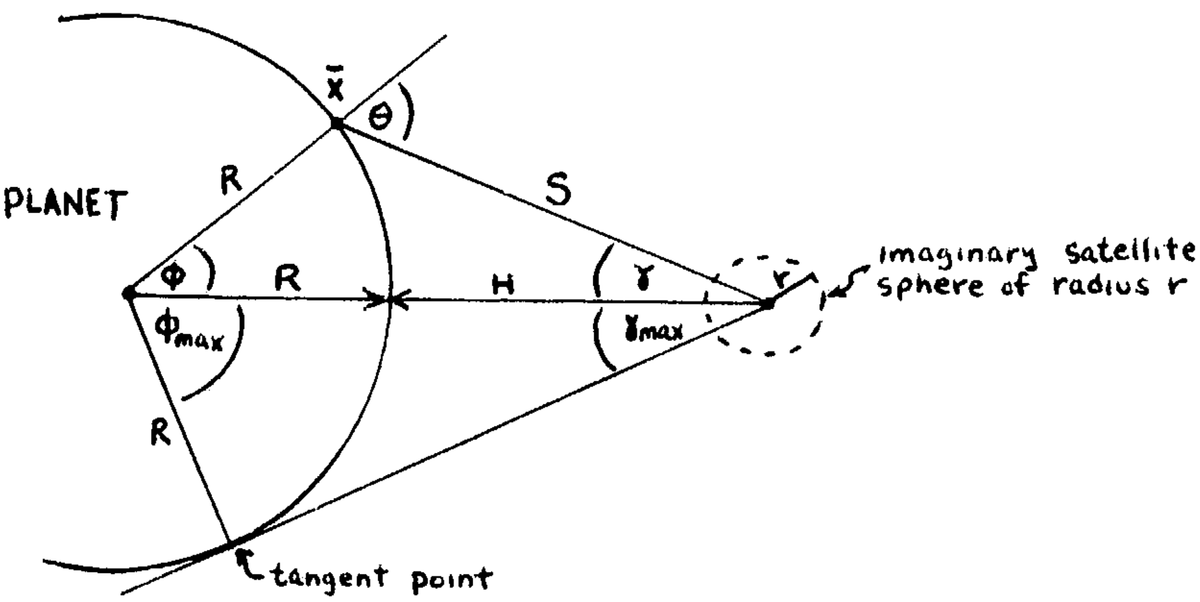
N = number of rays to be fired from the sun.

The numerator, of course, is the total energy fired from the solar plane within the imaginary satellite sphere. Dividing by N merely divides this energy equally among all rays.

Appendix 5 - Originating a Ray From the Planet

A. Picking the Point of Origin and Direction for a Ray

The geometry of the problem is shown in the following figure.



$\bar{X}$ = point to be picked on the planet

H = altitude above planet surface at orbit angle α

R = radius of planet

S = distance from satellite to $\bar{X}$

The point $\bar{X}$ will be computed such that the angle $\gamma \leq \gamma_{max}$.

The altitude H can be defined immediately from equation (4) of Appendix 2. Thus,

$$H = \frac{2 L_a L_p}{(L_a + L_p) + (L_a - L_p)\cos\alpha} - R \qquad (1)$$

L_a, L_p = altitudes (from the planet center) at apogee and perigee, respectively.

The value of γ_{max} is also easily obtained, since

$$\cos \Phi_{max} = R/(R+H) \qquad (2)$$

and

$$\cos \gamma_{max} = \cos(90-\phi_{max}) = \sin \phi_{max} = (1-\cos^2 \phi_{max})^{1/2}$$

Thus

$$\cos \gamma_{max} = (H^2 + 2RH)^{1/2}/(R+H) \qquad (3)$$

We next compute θ from the law of sines.

$$R/\sin \gamma = (R+H)/\sin \theta$$

$$\cos\theta = (1-\sin^2\theta)^{1/2} = \left(\frac{R^2 - (R+H)^2 \sin^2 \gamma}{R^2}\right)^{1/2}$$

$$\cos\theta = \frac{1}{R}\left[R^2 - (R+H)^2(1-\cos^2\gamma)\right]^{1/2} \qquad (4)$$

We are now in a position to define the initial conditions for a ray. First rotate the coordinate system into a primed system such that the Z' axis points toward the planet center. Then, using the results of Appendix 1, the point $\overline{X}$ on the planet is defined as

$$\begin{aligned} X &= X'C_1 + Z'C_2 \\ Y &= X'C_3 + Y'C_4 + Z'C_5 \qquad (5) \\ Z &= X'C_6 + Y'C_7 + Z'C_8 \end{aligned}$$

where the C's are functions of the direction cosines of the satellite-planet line. The X', Y', Z' values are defined as

$$\begin{aligned} X' &= S \sin \gamma \cos \beta \\ Y' &= S \sin \gamma \sin \beta \\ Z' &= S \cos \gamma \end{aligned}$$

where β is the azimuthal angle around the Z' axis.

It is now necessary to determine S, γ and β.

The angle β is chosen at random such that

$$\beta = 2\pi \xi$$

where ξ is a random number from 0 to 1.

Using the law of cosines

$$S^2 = R^2 + (R+H)^2 - 2R(R+H)\cos\phi \, .$$

Rearranging terms gives

$$S = \left[H^2 + 2R(R+H)(1-\cos\phi)\right]^{1/2} . \qquad (6)$$

From the law of sines

$$\sin\gamma = \frac{R}{S}\sin\phi \qquad (7)$$

Thus, once ϕ is determined, S can be computed from equation (6) and γ is obtained from equation (7).

The angle ϕ is chosen such that $\cos\phi$ is random between $\cos\phi_{\max}$ (equation (2)) and 1. Thus,

$$\cos\phi = \cos\phi_{max} + \xi\,(1 - \cos\phi_{max})$$

where ξ is a new random number.

The point on the planet (X,Y,Z) can then be computed and the direction cosines along the line S are given by

WS_1 = X/S

WS_2 = Y/S

WS_3 = Z/S

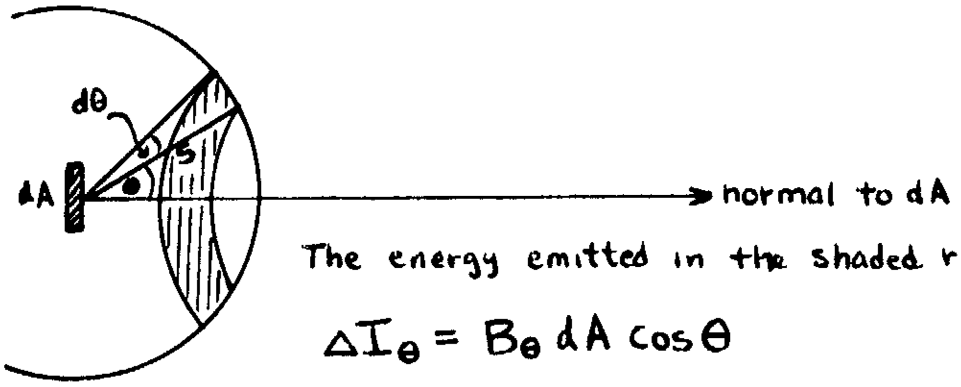
In theory, the ray could now be fired from (X,Y,Z) in the direction $-\overline{WS}$. There are two things wrong with this, however. Due to the finite precision capabilities of the computer, it is unwise to fire a ray at a relatively small object (of radius r) from a very large distance (S). The origin of the ray must, therefore, be moved closer to the satellite. To accomplish this, a disc is established perpendicular to S, a distance r from the satellite center. Secondly, the ray should not be fired from the center of this disc since it would always be aimed at the center of the satellite. Instead, the starting position of the ray on the disc should be picked randomly with area. This is done by rotating the coordinate system into a double primed system with Z" axis directed along S. A random starting point on the disc is then picked in the new system as follows.

$$X'' = r\sqrt{\xi_1}\ \cos(2\pi\xi_2)$$
$$Y'' = r\sqrt{\xi_1}\ \sin(2\pi\xi_2)$$
$$Z'' = r$$

The starting point in the unprimed system (X_o, Y_o, Z_o) is then computed from equations similar to equation (5) (i.e., $X_o = X''C_1 + Z''C_2$, etc.) The C's are a new set of constants computed from $\overline{WS}$.

B. Computing the Energy of Ray Emitted by the Planet

Referring to the figure below, we first compute the energy emitted by an element of area on the planet dA at an angle θ in the interval $d\theta$.



The energy emitted in the shaded ring is

$$\Delta I_\theta = B_\theta \, dA \cos\theta$$

ΔI_θ = energy/solid angle in θ direction = $dE_\theta / d\Omega$

B_θ = energy/area in θ direction.

From Lambert's Law

$$\Delta I_\theta = \Delta I_n \cos\theta \quad \text{where}$$

ΔI_n = energy/solid angle in normal direction $(\theta = 0^\circ)$.

Thus,

$$B_\theta = \Delta I_n / dA = \text{constant}.$$

The energy is therefore

$$dE_\theta = \Delta I_\theta \, d\Omega = B_\theta \, dA \cos\theta \, d\Omega = \left(\frac{\Delta I_n}{dA}\right) dA \cos\theta \, d\Omega .$$

But the solid angle subtended by the shaded ring is

$$d\Omega = 2\pi \sin\theta \, d\theta$$

Thus,

$$dE_\theta = \left(\frac{\Delta I_n}{dA}\right) dA \; 2\pi \cos\theta \sin\theta \, d\theta \quad .$$

The total energy, E_{TOT} is then

$$E_{TOT} = \int_0^{E_{TOT}} dE_\theta = 2\pi \left(\frac{\Delta I_n}{dA}\right) dA \int_0^{\pi/2} \cos\theta \sin\theta \, d\theta = 2\pi \left(\frac{\Delta I_n}{dA}\right) dA \cdot \frac{1}{2}$$

$$E_{TOT} = \pi \Delta I_n \; .$$

Therefore, ΔI_n is equal to E_{TOT}/π .

Lambert's Law now becomes

$$\Delta I_\theta = \frac{E_{TOT}}{\pi} \cos\theta$$

or

$$\Delta E_\theta = \frac{E_{TOT}}{\pi} \cos\theta \, d\Omega$$

, where $d\Omega$ is now the solid angle subtended by the imaginary satellite sphere of radius r. Since this is equal to $\pi r^2/S^2$, the energy incident on the satellite sphere is

$$\Delta E_\theta = E_{TOT} \cos\theta \left(\frac{r}{S}\right)^2 .$$

Now let $E_{TOT} = QA$ where

Q = energy radiated/surface area = σT^4

A = area of planet surface visible to the satellite

σ = Stefan - Boltzmann constant

The total energy emitted at the satellite is then

$$\boxed{\Delta E_\theta = (\sigma T^4) A \cos\theta \left(r/S\right)^2} . \qquad (8)$$

The energy of each ray is merely ΔE_θ divided by the number of rays to be fired from the planet. The value of cos θ is computed from equation (4) derived earlier, so that only A is still to be computed. Referring to the figure at the beginning of this appendix, we see that the visible area can be written in terms of R and ϕ. Thus,

$$dA = 2\pi R^2 \sin\phi \, d\phi$$

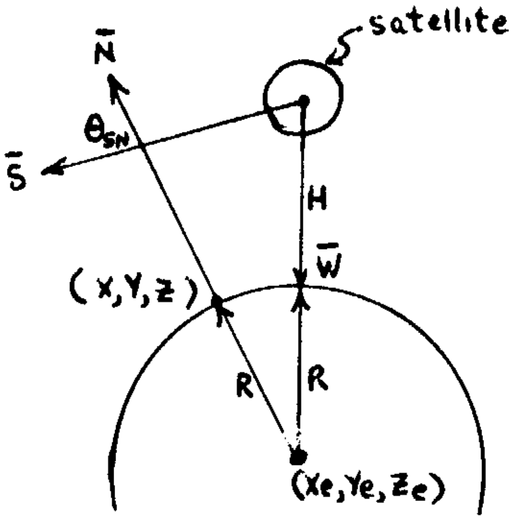
$$A = \int_{\phi_{max}}^{0} dA = 2\pi R^2 \cos\phi \Big]_{\phi_{max}}^{0} = 2\pi R^2 (1 - \cos\phi_{max}) .$$

But from equation (2), cos ϕ_{max} = R/(R+H) so that

$$A = 2\pi R^2 H / (R+H) .$$

C. Determine if a Point on the Planet Can Reflect Sunlight

Once a ray has been fired from the planet, it is necessary to determine if the point where the ray originated from on the planet can reflect sunlight to the satellite. The variables for this calculation are defined by the following figure.



$\bar{N}$=normal to planet at x,y,z

$\bar{S}$=direction of satellite-sun line

$\bar{W}$=direction of satellite-planet line

$\bar{N},\bar{S},\bar{W}$ are unit vectors whose components are direction cosines

The angle θ_{SN} between $\bar{N}$ and $\bar{S}$ is given by

$$\cos\theta_{SN} = N_x S_x + N_y S_y + N_z S_z \quad \text{where}$$

$$N_x = \frac{X-X_e}{R} \qquad N_y = \frac{Y-Y_e}{R} \qquad N_z = \frac{Z-Z_e}{R}$$

and

$$X_e = (R+H)W_x \qquad Y_e = (R+H)W_y \qquad Z_e = (R+H)W_z \text{ .}$$

Now, by inspection one can see that a point on the planet can only reflect sunlight to the satellite if $\cos\theta_{SN} > 0$.
This test is made in the code and if $\cos\theta_{SN}$ is positive, the energy of the reflected ray is computed.

D. Computing the Energy of a Planet-Reflected Ray

The energy of planet reflected rays is analogous to equation (8) for planet radiation energy except for the following changes.

(1) A new source term is needed. This is equal to $S_o \alpha_p$ where S_o is the solar constant and α_p is the reflection coefficient (albedo) of the planet.

(2) A $\cos\theta_{SN}$ factor must be inserted to account for the projected visible area on the planet surface.

Thus, the reflected energy E_{ref} is

$$E_{ref} = \frac{1}{N} \cdot (S_o \alpha_p) A \cos\theta \cos\theta_{SN} \left(r/S \right)^2$$

Where N is the total number of rays emitted from the planet and A is the total area of the planet visible to the satellite.

We can show that A_{TOT}/N_{TOT} is the correct factor to use, even for planet reflection, in the following way. The correct factor should actually be

$$\frac{\text{reflecting area}}{\text{number of reflected rays}} = \frac{A_{Ref}}{N_{Ref}} .$$

But $A_{ref} = A_{TOT} \left(\frac{N_{ref}}{N_{tot}} \right).$

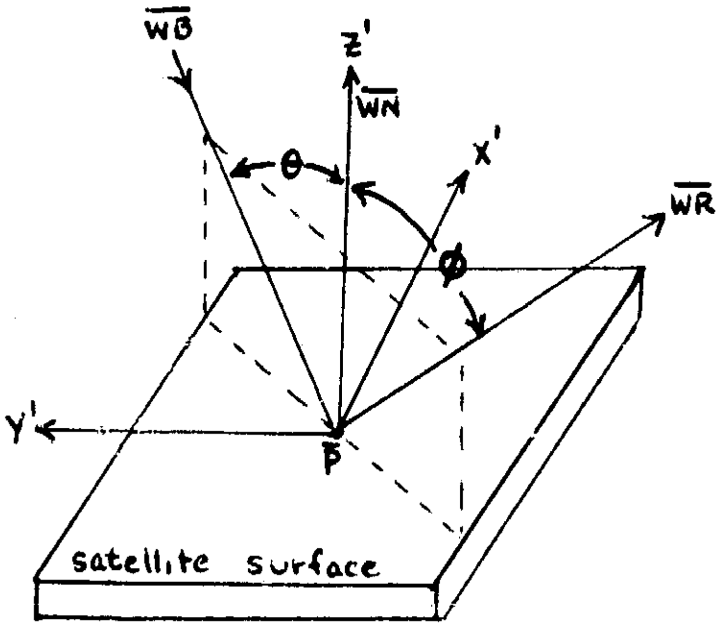
Thus,

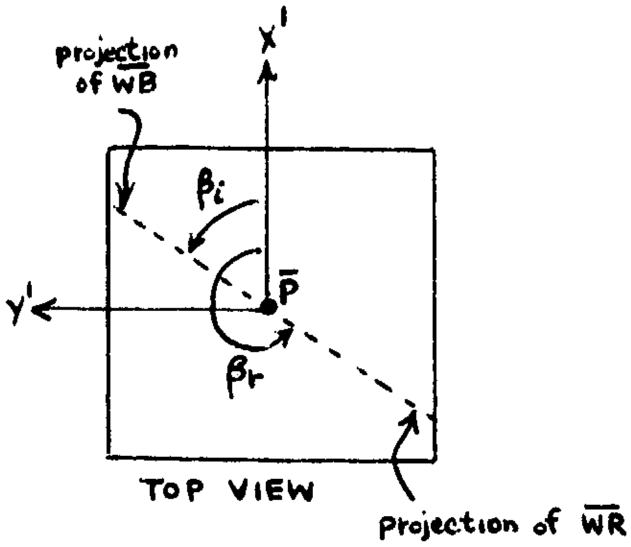
$$\frac{A_{ref}}{N_{ref}} = A_{TOT} \left(\frac{N_{ref}}{N_{TOT}} \right) \left(\frac{1}{N_{ref}} \right) = \frac{A_{TOT}}{N_{TOT}} .$$

Appendix 6 - Energy and Direction of Reflected Rays From the Satellite

A. Specular Reflection

The problem is to derive expressions for the direction cosines of a ray reflected from the surface of the satellite. The geometry is defined by the following figures.





$\overline{WB}$ = direction of incident ray

$\overline{WR}$ = direction of reflected ray

$\overline{P}$ = reflection point on satellite surface

$\overline{WN}$ = direction of normal to surface at point $\overline{P}$

θ = angle of incidence

Φ = angle of reflection

β_i = angle between the X' axis and the projection of the incident ray on the surface

β_r = angle between the X' axis and the projection of the reflected ray on the surface

The coordinate axes have been rotated into a primed system (x',y',z') such that the z' axis points along the normal $\overline{WN}$. In addition, the origin of coordinates has been translated to the reflection point $\overline{P}$. The equations of rotation, derived in Appendix 1, are then

$$(X-P_x) = X'C_1+Z'C_2$$

$$(Y-P_y) = X'C_3+Y'C_4+Z'C_5$$

$$(Z-P_z) = X'C_6+Y'C_7+Z'C_8 \qquad (1)$$

where (X,Y,Z,) are the coordinates of any point on the reflected ray. The c's, also defined in Appendix 1, are functions of the direction cosines of the normal (WN_x, WN_y, WN_z). If (X,Y,Z) is assumed to be a unit distance from $\overline{P}$, the equations for (X',Y',Z') become

$$X' = \sin\phi \cos\beta r$$

$$Y' = \sin\phi \sin\beta r$$

$$Z' = \cos\phi \qquad (2)$$

Thus, if X',Y',Z' are known, they can be substituted into equation set (1) to compute ($X-P_x$), etc. But, since (X,Y,Z) is a unit distance from $\overline{P}$, the left sides of equations (1) are merely the direction cosines of the reflected ray. Thus,

$WR_x = (X-P_x) \quad WR_y = (Y-P_y) \quad WR_z = (Z-P_z)$.

The problem then is to derive expressions for ϕ and βr, subject to the constraints that the angle of incidence equals the angle of reflection and that $\overline{WB}$, $\overline{WN}$, and $\overline{WR}$ all lie in a plane.

To compute ϕ, we first define the cosine of the angle between $\overline{WB}$ and $\overline{WN}$. Since the incident ray points away from the normal, this angle is actually $(180-\theta)$. Thus

$$\cos(180-\theta) = -\cos\theta = \overline{WB}\cdot\overline{WN}$$

and, since $\phi = \theta$,

$$\cos\phi = -\overline{WB}\cdot\overline{WN} = -(WB_xWN_x+WB_yWN_y+WB_zWN_z)$$

$$\sin\phi = (1-\cos^2\phi)^{\frac{1}{2}} .$$

To define β_r, we establish a point on the <u>incident</u> ray a unit distance from $\overline{P}$. Then a set of equations for the incident ray analogous to (1) and (2) can be established, where (X,Y,Z) lie on the incident ray and θ and β_i replace ϕ and β_r. Using the first equation of set (1)

$$(X-P_x) = -WB_x = C_1(\sin\theta\ \cos\beta_i) + C_2\cos\theta$$

But, since $\overline{WB}$, $\overline{WN}$, $\overline{WR}$ lie in the same plane, $\beta_r = 180+\beta_i$. Thus, $\cos\beta_i = -\cos\beta_r$. Substituting for $\cos\beta_i$ in the above equation, setting $\theta = \phi$ and solving for $\cos\beta_r$ gives

$$\cos\beta_r = \frac{WB_x + C_2\cos\phi}{C_1\sin\phi} .$$

Using the second equation of set (1) and applying a similar analysis gives

$$\sin\beta_r = \frac{WB_y - C_3\sin\phi\cos\beta_r + C_5\cos\phi}{C_4\sin\phi} .$$

Thus, having solved for cos ϕ , sin ϕ , cos β_r , sin β_r , the direction cosines of the reflected ray can be computed from equation set (1).

The energy of the reflected ray is simply the product of the incident energy and the reflection coefficient of the surface.

B. Diffuse Reflection

The direction, $\overline{WR}$, of a diffuse ray also involves defining the angles ϕ and β_r . In this case, however, β_r is chosen at random. Thus, $\beta_r = 2\pi\xi$, where ξ is a random number between 0 and 1. The reflection angle ϕ is chosen from a predetermined set of angular 'bins'. Once ϕ and β_r are defined, the components of $\overline{WR}$ are computed in the same manner as for specular reflection.

In the program, ϕ is computed in the following way. A set of angular bins of equal width are established, with the number of bins equal to the number of rays to be emitted per diffuse reflection (an input quantity). These bins extend from 0 to 90° with respect to the surface normal. One ray is emitted in each bin with the angle ϕ within the bin being chosen at random. Thus, if n rays are to be emitted, the width of each bin is $\pi/2n$. The angle ϕ_i for the ith ray is then

$$\phi_i = \frac{\pi}{2n}(i-1) + \frac{\pi}{2n}\xi$$

where ξ is a random number.

Although the number of rays are uniformly distributed with angle, the energy of each ray is computed so as to give the correct energy distribution. The energy equation is derived as follows. Let dE_ϕ be the energy reflected at an angle ϕ . Then

$$dE_\phi = \left(\frac{\text{energy}}{\text{solid angle}}\right)\left(\text{solid angle}\right) = \Delta I_\phi \, d\Omega .$$

Using Lambert's Law this becomes

$dE_\phi = (\Delta I_n \cos\phi)d\Omega$ where ΔI_n is the energy per unit solid angle along the normal.

In Appendix 5, Part B, it was shown that $\Delta I_n = E_{TOT}/\pi$ where E_{TOT} is the total reflected energy.

Since $d\Omega = 2\pi \sin\phi \; d\phi$,

$$dE_\phi = \left(\frac{E_{TOT}}{\pi}\cos\phi\right)\left(2\pi\sin\phi\, d\phi\right).$$

Therefore, the energy reflected in the angular bin ϕ_1 to ϕ_2 is

$$\int_{\phi_1}^{\phi_2} dE_\phi = 2E_{TOT}\int_{\phi_1}^{\phi_2}\cos\phi\sin\phi\, d\phi = E_{TOT}\left[\sin^2\phi_2 - \sin^2\phi_1\right].$$

This equation is used to determine the energy of each diffuse ray, where E_{TOT} = incident energy x surface reflection coefficient. To save needless computing, the size of the angular bins and the $\Delta\sin^2\phi$ factor for each bin are calculated in subroutine SHADOW and stored for later use by subroutine SATREF.