INTERACTIVE COMPUTATION OF RADIATION VIEW FACTORS*

A.F. EMERY
H.R. MORTAZAVI
C.J. KIPPENHAN

UNIVERSITY OF WASHINGTON
SEATTLE, WASHINGTON

ABSTRACT

The development of a pair of computer programs to calculate the radiation exchange view factors is described. The surface generation program is based upon current graphics capabilites and includes special provisions which are unique to the radiation problem. The calculational program uses a combination of contour and double area integration to permit consideration of radiation with obstructing surfaces. Examples of the surface generation and the calculation are given.

INTRODUCTION

The calculation of the radiation exchange between two surfaces by the usual engineering method

$$\frac{\sigma(T_1^4 - T_2^4)}{\frac{1-\varepsilon_1}{\varepsilon_1 A_1} + \frac{1}{A_1 F_{12}} + \frac{1-\varepsilon_2}{\varepsilon_2 A_2}} \qquad (1)$$

first requires the evaluation of the view factor, F_{12}, which represents the fraction of energy leaving one diffuse surface which is intercepted by the second surface. F_{12} is defined by[1]

$$A_1 F_{12} = \iint_{A_1 A_2} \frac{\cos\theta_1 \cos\theta_2}{\pi r_{12}^2} \, dA_1 dA_2 \qquad (2)$$

*This work was supported by NASA grant NAG-1-41.

Although the equation is simple in appearance, its evaluation is fraught with difficulties since it requires a full and precise description of both surfaces. In addition, if the view from surface 1 to surface 2 is obscured by an interposed surface or object, some provisions must be made to account for that porion of the view which is occluded. It is important to recognize that the obstructed view is not a constant, but varies, depending upon the position of the elemental area dA_1 .

The calculation of the view factor is thus really two problems in one:

1. To devise an efficient way of describing the surfaces and their spatial relationship to each other.

2. To calculate the resulting view factors.

This paper describes the development of a pair of computer programs to do this. The calculational program was developed first and used with hand input in a batch mode. It later became apparent that efficient use of the program required some form of automatic surface generation and the appropriate subroutines were added. It also became painfully obvious that the average user made so many errors (both in key punching and in defining the surfaces) that an interactive program, with graphic capability, was necessary.

SURFACE GENERATION

The first impression was that a standard CAD/CAM program could be easily modified to provide the needed interactive capability. However, some of the unique requirements of the view factor calculation program led us to conclude that a special program would be more appropriate. This program was based upon the following assumptions:

1. Surfaces would be only 3 or 4 sided with straight sides
2. To simplify the obstruction calculations, all surfaces would be planar
3. A surface would radiate from one side only. Thus every plate, no matter how thin, would require 2 surfaces.
4. The direction of the surface normal must be uniquely and simply defined.
5. Surfaces could not penetrate one another.

Some of these requirements were needed to calculate the view factors. Others were imposed to simplify the surface generation. Several additional requirements were found to be useful, although not absolutely necessary, namely:

1. All surfaces are initially generated in the x,y plane, facing upwards. Three dimensional surfaces (cylinders etc) would be oriented along the z axis
2. Only a global coordinate system would be used. Individual surfaces would not carry along an embedded coordinate system.
3. Surface corner nodes would always be numbered in a counter clockwise direction to specify the surface normal.
4. Curved surfaces would be represented by a combination of triangles and quadrilaterals, the size of each adjusted by the user to give the best representation.

The different capabilities of the generation program are illustrated in figure 1. Because radiation is a surface phenomena, the surface orientation (i.e., the direction of the surface normal) must be uniquely defined. In testing the program, particularly with inexperienced users, we have found that it must not only be interactive but must permit the user to modify any command at any time and must have a very complete graphics capability. The following sections describe the different features of the program.

A. Surface Types

Three different surface types are used: minor, major and groups. A major surface is one that is generated by a single command. Typical major surfaces are listed in table 1. A major surface can be manipulated as a single entity. A group is a collection of major surfaces and is also a single entity. When the major surface is generated, it may be composed of many minor or sub-surfaces. Figure 2 illustrates the development of a group and shows the minor surfaces. Minor surfaces cannot be treated as independent entities since this would lead to possible erratic and unacceptable distortions of the major surfaces.

TABLE 1
Typical Major Surfaces

Two Dimensional
1. Triangle
2. Rectangle
3. Quadrilateral
4. Circle (annulus, sector)
5. Ellipse (annulus, sector, orientation)

Three Dimensional (inside or outside radiation, top and/or bottom)
1. Box
2. Cylinder (right, slant, annular, sector)
3. Cone (right, slant, frustrum, annular, sector)
4. Sphere (sector, cutting planes)

B. Command Data Base

Because even experienced users tend to make many mistakes or desire to make a substantial number of modifications, it is important that all commands be stored for future use and modification. Commands are either informative or functional. All functional commands are stored in a data base and the user can:

1. List any or all of the commands
2. Insert new commands between existing commands.
3. Delete commands
4. Temporarily suspend commands
5. Repeat the commands
6. Interrupt the repeat
7. Call for information or graphical display at any time during the repetition of the commands.

Since surface movements are non-commutative and because the current structure cannot be generated by only a portion of the commands, any changes in the data base must be accomplished by restarting the command sequence (i.e. repeating) and modifying during the subsequent generation process.

C. Surface Manipulations

The user must be able to manipulate each entity (major or group) by such movements as:

1. Translation
2. Rotation
3. Replication (i.e, duplicating an original set of surfaces and manipulating the new set)
4. Scaling in x,y or z coordinates

In addition, all surfaces (including minor surfaces) must be subject to:

1. Deleting (temporary or permanent)
2. Adding
3. Restoring (if previously deleted)
4. Joining to another surface to form a new entity
5. Separating from another
6. Reversing the normal direction
7. Combining with others to form a single radiating surface but still capable of independent manipulation.

D. Graphical Display

The key to effective surface generation is a high speed graphical display. Our experience has been that the typical user will call for a display after every one or two functional commands. Furthermore, the user will generally ask for more than one view. Thus the graphics must be fast and versatile. The minimum number of graphical views and commands appears to be:

1. Orthographic views (top,bottom,left,right,front,back) singly or in groups
2. Perspective or isometric views
3. Hidden line removal from any or all views
4. Variable field of view
5. Rotation and translation of the object.
6. Deletion of selected surfaces from the view
7. Numbering of selected surfaces or corner nodes
8. Display of surface normals
9. Views as seen from one surface to another

Although the isometric view gives the best overall picture, the orthographic views prove to be the most useful when precise movements are necessary. Because obstructions so completely change the radiation exchange, it is critical that the user be able to look from any one surface to any other to check for obstructions. This must be done with hidden line removal to yield the maximum information.

Figure 3 illustrates the incorrect movement of panels on a cylinder as shown by an isometric view and an orthographic view--the latter being necessary for the correct placement.

Although a wire frame display is useful, the usual structure has so many surfaces that a hidden line removal is necessary. Unfortunately hidden views are very time consuming and consequently the program incorporates a somewhat inexact, but very fast, routine. Part of its speed is based upon the use of planar, straight sided surfaces with counter clockwise numbering. Any other surface requires some type of raster scan hidden surface algorithm with an unacceptable increase in computing time[2]. It is interesting to note that a fast graphics display may not give the desired information. We have found that if the lines are drawn slowly enough so that the viewer can get a feeling for the order of generation, it is easier to visualize the structure. On the other hand, if the display is nearly instantaneous, many viewers cannot recognize the structure. This is particularly true if an old command data base is being reused. Finally the surface normals must be displayed since a surface has only one active side. Most user errors appear to be related to an incorrect orientation of a surface after a series of manipulations have been made. Figure 4 illustrates a set of double surfaces to represent panels and the erroneous orientation of the replicated set is clearly seen.

E. Surface Refinement

After the basic structure has been constructed, three additional functions are needed:

1. Refinement by subdividing surfaces
2. Assuring planarity of surfaces
3. Condensing nodes

Refinement is simply the subdivision of surfaces into triangular sub-surfaces. Since triangles are always plane, planarity is assured by dividing any non-planar quadrilateral into 2 triangles. Obviously, refinement also guarantees planarity. Finally, upon exit any two nodes which are within a prescribed distance ε are condensed into one node; all inactive surfaces and unused nodes are deleted. The data file is created for use by the view factor calculating progam and for subsequent plotting. The command data base is closed for future use.

CALCULATING THE VIEW FACTORS

Because analytical expressions for F_{ij} exist only for simple surfaces [3], the computation of the view factors for complex structures is practical only by:

1. Double area integration
2. Contour integration
3. Nusselt projection
4. Monte Carlo

The Monte Carlo method[4] tends to be too expensive for most surfaces even if one of the adaptive techniques [5] is used. Nusselt projection (i.e., the use of the unit sphere) generally produces a curved projected surface and consequently requires a complex integration to find the projected area. In addition, obstructions will distort the contour of the viewed surface, rendering the calculation of the projected edges very difficult. In this case, one simply projects the position of an elemental area, not the edges of the contour. Since the projection must be done from every point on the viewing surface, it essentially reduces to the double area integration method.

A. Double Area Integration

The double area integration is performed in two steps (figure 5).

1. Mapping the surface onto a unit square by

$$x' = \sum N_i (\xi,\eta)\, \hat{x}_i \tag{3}$$

The functions N_i are the usual finite element isoparametric functions[6].

2. Evaluating the integral

$$A_i F_{ij} = \sum_{k=1}^{n_i} \sum_{\ell=1}^{n_j} h_k^i\, h_\ell^j\, I(P_k,P_\ell)\, J^i(P_k)\, J^j(P_\ell) \tag{4}$$

where $P_k\ (\xi,\eta)$ are the Gaussian points in the unit square, $J_i(P_k)$ are the Jacobian of transformation evaluated at these points, h_k are the Gaussian weights and the superscripts i and j refer to the surfaces and

$$I(P_k,P_\ell) = \frac{\cos\Theta_k \cos\Theta_\ell}{\pi r_{ij}^2 (k,\ell)} \tag{5}$$

B. Contour Integration

The double area method is the best of the three, but it is not sufficiently accurate when portions of the surfaces are close to each other since the denominator of equation 1 becomes very small and the integral nearly singular. For these reasons we adopted the contour integration method as described by Mitalas and Stephenson [7]. In this method, the double area integration can be replaced by

$$A_i F_{ij} = \frac{1}{2\pi} \oint_{A_i} \oint_{A_j} \ell n\, r_{ij}\, (dx_i\, dx_j + dy_i\, dy_j + dz_i\, dz_j) \tag{6}$$

If the edges of all surfaces are straight lines and divided into short segments, equation 6 can be expressed as

$$A_i F_{ij} = \frac{1}{2\pi} \sum_k \sum_\ell D(k,\ell)\ (L_\alpha \ell n L_\alpha \cos\alpha + L_\beta \ell n L_\beta \cos\beta + P\gamma - L)dt \tag{7}$$

where the various terms are defined on figure 6 and

$$D(k,\ell) = \ell_s \ell_t + m_s m_t + n_s n_t \quad (\ell\ m\ n = \text{direction cosines})$$

This formulation is not sensitive to the separation distance of the edges since the term $L_\alpha \ell n L_\alpha$ is not singular as the distance r goes to zero. We note that planar or non-planar surfaces can be treated equally well.

C. Comparison

In general, when numerically implemented, either Contour integration or Double Area integration will produce acceptable results. However, when two surfaces have a common edge (the adjoint problem) the Double Area integration method may perform very poorly[8, 9]. Table 2 lists comparable values obtained by the two methods for the surface shown on figure 7. Because the greatest portion of the radiation occurs in the corner, where the surfaces are the most proximate, the Double Area method tends to be inaccurate. By contrast the Contour integration method is very accurate, even with very few edge elements. When the surfaces are separated slightly, there is some improvement in the Double Area results, but not sufficient to justify its choice over the contour method.

TABLE 2
Percentage Error in the Numerical Calculation of the View Factor between Two Surfaces of Equal Breadth (L=H)
(see figure 7)

	Contour Integration		Double Area Integral			
	Infinite Strip	Finite Area	Infinite Strip		Finite Area	
d/L	S=0.0	S=0.0	S=0.0	S=0.1 L	S=0.0	S=0.1 L
1.0	0.2%	3.7	20.7	21.5	59.1	57.6
0.5	0.	0.8	12.9	11.2	32.0	24.1
0.3		0.4	9.1	6.6	21.8	12.1
0.2		0.1	5.7	2.8	13.2	4.9
0.1		0.	2.9	0.7	6.7	1.2
0.05			1.5	0.2	3.4	0.3

d = dx=dy=dz for the Double Area Integration
edge segment length for the Contour Integration

S = Separation distance along the x coordinate

We note that the use of the Contour integral with as few as 5 segments on an edge gives acceptable results, while the use of the Double Integral with a corresponding spacing is quite unacceptable.

D. Program Structure

The basic program structure is illustrated in Table 3.

TABLE 3
PROGRAM FLOW

1. Geometrical Input (definition of surfaces)
2. Unobstructed View Factor Computations using contour integration
3. Obstructed View Factor Computations
   1. determining if two surfaces, i and j, can see each other
   2. determining if a 3rd surface is interposed between the pair
   3. subdividing surfaces i and j into elemental areas
   4. constructing a ray between points on surfaces i and j and determining if it is intercepted
   5. computing F_{ij} using the double area integral

Each of the three major functions of the program will be described in detail in the following sections.

D1. Input

Data input is from the output file of the Surface Generation program.

D2. Unobstructed View Factor Calculations

The values of F_{ij} are computed directly by using the Contour Integration equation. Calculational time is reduced by the use of the reciprocity relationship, at the cost of the loss of some additional information.

D3. Obstructed View Factor Calculations

(A) Elimination of non-obstructing Surfaces

The calculation of F_{ij} when one or more surfaces are interposed between surfaces i and j is the most difficult and time consuming part of the calculation. In order to accelerate this calculation, every possible effort must be made to eliminate all non-obstructing surfaces from consideration. Consider the situation schematically shown on figure 8. The best calculational procedure found to date is:

1. Using the original x,y,z coordinate system, define the smallest rectangular parallelepiped (the view prism) which contains surfaces i and j (indicated by the dashed lines). Eliminate all surfaces which do not penetrate this view prism. This process is usually referred to as "clipping". Check to see if any surface completely fills the view prism, since if the view is totally obscured no further computations are needed.
2. Define a new coordinate system, $\bar{x},\bar{y},\bar{z}$, with the $\bar{z}$ axis directed along the line connecting the centroids of surfaces i and j and eliminate all possible surfaces by another clipping pass. Again test for total blockage.
3. Define a third coordinate system, $\bar{\bar{x}},\bar{\bar{y}}$ by rotating the $\bar{x},\bar{y}$ coordinate system until the rectangular area which encloses both surfaces i and j is a minimum (figure 8b) and perform another clipping test and test for total blockage.

It cannot be emphasized too strongly that acceptance or rejection of a possible obstructing surface can only be done by the simple clipping operation associated with view prisms which are constructed from rectangular parallelepipeds. Any other procedure to test for penetration of the view prism calls for geometric calculations which are unacceptably expensive. Since the transformations involved in steps 2 and 3 require matrix operations on all of the corner coordinates of all candidate obstructions, it is imperative that each successive step eliminate as many obstructing surfaces as possible. We have considered using the graphic capability of the generating program to detect the obstructing surfaces, but it has proven to be too time consuming and inefficient.

Once the final set of possible obstructing surfaces has been determined, F_{ij} can be calculated in two ways. Consider the view of surface j as seen from a segment on the contour of surface i as shown in figure 9. The obstructing surfaces obscure part of surface j, leading to the formation of the two visible sub-areas indicated by the dotted line contours. We note that: a) the determination of the unobstructed part of the surface is the classical hidden surface problem of graphic display for which, currently, there are no efficient methods; b) the number of non-contiguous visible sub-areas and the number of straight line segments encompassing these sub-areas may vary considerably as seen from different points of the contour of surface i. For this reason, Contour Integration is not an acceptable method; hence, we must utilize Double Area Integration.

(B) Ray Interception

We express F_{ij} as

$$A_i F_{ij} = \sum_{k=1}^{n_i} \sum_{\ell=1}^{n_j} f_{k\ell}\, I(k,\ell)\, dA_k\, dA_\ell \tag{8}$$

where $f_{k\ell}$= 1 if the ray between dA_k and dA_ℓ is unobstructed

= 0 if obstructed.

Since $f_{k\ell}$ is a discontinous function of position, the higher accuracy of Gaussian quadrature is not always realized; and since Gaussian quadrature requires substantially more numerical operations, it has proven best to use Newton-Cotes integration.

Once each surface has been divided into elemental areas, the centroids of these areas determined, and the rays between each of the points on surface i to each of the points on surface j defined, the calculation of $f_{k\ell}$ proceeds as follows (figure 10):

1. Determine if the angle between the outward normals to surfaces i and j and the ray $P_{k\ell}$ is greater than 90 degrees. If so, set $f_{k\ell}$=0 since the surfaces cannot see each other. Note that this must be done for every ray, since the inability of one point on surface i to see any given point on surface j does not ensure that other parts of the two surfaces cannot see each other.
2. Determine the intersection of the ray $P_{k\ell}$ with each of the possible obstructing surfaces by examining each in turn. This is done by:

    1. Finding the intersection of the ray $P_{k\ell}$ with the infinite surface which contains the surface. Because the intersection requires an iteration for arbitrary surface, but not for planes, only plane surfaces are permitted. The determination of the intersection is best effected by pre-calculating the equation of the plane of each surface and the transformation R

$$\begin{Bmatrix} x' \\ y' \\ z' \end{Bmatrix} = [R] \begin{Bmatrix} x \\ y \\ z \end{Bmatrix} \tag{9}$$

    which produces the coordinate system x',y',z' for which x' and y' are in the plane of the obstructing surface and z' normal to it.

2. Determining if the point of intersection is within the surface or not. If it is, $f_{k\ell}=0$; if not, $f_{k\ell}=1$. Determining whether the intersection point is within the surface by mapping the surface to a square and checking the values of ξ and η is not efficient because the non-linear mapping requires an iterative solution for ξ and η . The most efficient way appears to be by computing the angle between the line drawn from the corner node to the intersection point and between the corner node and the next corner node. For convex surfaces, if this angle is negative for any corner, the point is not within the surface and the ray is not blocked.

The most efficient calculation is one in which the obstruction is detected early, thus eliminating the consideration of the remaining surfaces. This is only possible if the candidate obstructing surfaces can be sorted in the order of their included angle as seen from the elemental area, dA_k. Unfortunately, this determination of the included angle is equivalent to finding the view factor from the elemental area dA_k to that portion of the obstructing surface which is within the view prism between points dA_k and dA_ℓ . Thus far we have not found a sorting method which gives a net reduction in computational times. Our experience is that maintaining a fixed order of checking the rays for interception in a single view prism is an effective method. (It should be noted that this entire problem can be compared to the usual graphical display problem for perspective views of hidden surfaces, but since in essence the viewing point must range over the entire surface i, calculationally it is the square of this classical problem.)

That the Double Area Integration is inaccurate for surfaces with a common edge has already been discussed; yet only the Double Area Integration is useful for the obstructed view calculations. Therefore it is important to avoid this situation by appropriately defining the surfaces.

EXAMPLES

We present two examples of the use of the program. Figure 11 represents an enclosure with a dividing panel, with the surfaces as numbered. The dividing partition must be expressed as two surfaces, infinitesimally separated, not one. Because surface 1 has an obstructed view of surface 9 with which it has a common edge, it must be represented as two surfaces, 1A and 1B. In this way, surface 1A which adjoins surface 9 has an unobstructed view and is treated by Contour Integration. Surface 1B, which has an obstructed view, is not adjoining and can be successfully treated by Double Area Integration. Figure 12 represents an enclosed cylinder which obstructs the view of the other surfaces. Table 4 lists typical execution times. The obstruction calculations were carried out using the three different elimination methods and different numbers of rays per surface and the accelerating effect of the clipping routines is clearly shown. Because of the simple geometries and the orientation of the surfaces along coordinate axes, the second clipping was not effective since the slight reduction in computational time due to the elimination of a few additional surfaces was less than the time necessary to

perform the coordinate transformations. When the x,y axes are rotated by about 22 degrees, the first clipping effectiveness is reduced by about 30% and the value of the second clipping is more pronounced. For general problems, all clipping procedures must be used.

TABLE 4
Calculation Times
for the Example Problems depicted in Figures 11 and 12

	Adjoint Surfaces figure 10 (12 surfaces)		Enclosed Cylinder figure 11 (48 surfaces)		
Number of rays per surface	9	16	9	16	16 (rotated coordinates)
Clipping routine	Calculational Times in Central Processor Seconds on CDC Cyber 175/750				
none	5.6	14.8	99.0	274.1	274.1
1st pass	2.9	6.3	35.7	82.8	110.8
2nd pass	3.0	6.3	38.9	82.5	82.5
2nd pass(opt=2)					76.3

CONCLUSIONS

Our experience in using the programs has emphasized that radiation view factors of complex structures can only be accomplished if a fast, interactive program with good graphics capability is used. Although it would be more satisfying if curved surfaces could be treated, high program efficiency requires that they be modelled by an assemblage of flat triangles or quadrilaterals.

The view-factor calculational algorithm described has proven to be very effective for surfaces which have an obstructed view of each other. Under some conditions, statistical methods may prove to be more efficient, but for general configurations the combination of the Contour Integration and Double Area Integration methods, in concert with ray intersection calculations, has proven to be about the fastest method currently available. Further increases in computing efficiencies are possible if hardware perspective view, hidden surface devices are used, but such devices are not currently available for digital computers used in thermal analyses. From another point of view, the use of the rays may be regarded as a highly adaptive form of the Monte Carlo method which bears the same relationship to the usual method that the quasi-deterministic Exodus [10] method bears to the usual Monte Carlo method for solving diffusion problems.

NOMENCLATURE

A_i	Area of surface i
dA_i	Elemental area of surface
F_{ij}	Viewfactor from surface i to surface j
h_k	Gaussian weight for point k in surface i
$J(P_k)$	Jacobian of transformation at point k
n_j	Number of elemental areas in surface j
N	Isoparametric shape functions
P_k	Gaussian point k on surface i
r_{ij}	Distance from surface i to surface j
R	Rotation matrix
x,y,z	Global coordinate system
$\bar{x},\bar{y},\bar{z}$	Rotated coordinate system
θ	Angle between r_{ij} and a surface normal
ξ,η	Surface coordinates of the unit square.

REFERENCES

1. Sparrow, E.M. and Cess, R.D., Radiation Heat Transfer, Brooks/Cole Publishing Co., Belmont, Calif, 1966
2. Sutherland, I.E., Sproull, R.F. and Schumacker, R.A., 'A Characterization of Ten Hidden-Surface Algorithms' ONR Contract Report, Contract N00014-72-C-0346
3. Hamilton, D.C. and Morgan, W.R., 'Radiant Interchange Configuration Factors', NACA Technical Note 2836 (1952)
4. Siegel, R. and Howell, J.R., Thermal Radiation Heat Transfer, McGraw-Hill Book Company, New York, 1972
5. Hitzl, D.L. and Maltz, F.H.,'Adaptive Estimation Procedures for Multi-Parameter Monte Carlo Computations', Journal of Computational Physics, vol 21, 1980, pp 218-241
6. Zienkiewicz, O.C., The Finite Element Method in Engineering Science, McGraw-Hill Book Co., New York, 1971
7. Mitalas, G.P. and Stephenson, D.G.,'Fortran IV Programs to Calculate Radiant Energy Interchange Factors', National Research Council of Canada, Div. Bldg. Res., DBR Computer Program 25, CP25 Ottawa, 1966
8. Look, DC. and Love, T.J., Numerical Quadrature and Radiative Heat-Transfer Computations, AIAA, vol8, pp819-820, 1970
9. Rasmussen, M.L. and Jischke, M.C., Radiant Heat Transfer at the Vertex of Adjoint Plates, AIAA, vol 8, pp 1360-1361, 1970
10. Emery, A.F. and Carson, W.W. 'A Modification of the Monte Carlo Method--the Exodus Method' ASME J. of Heat Transfer, August, 1968, pp. 328-333

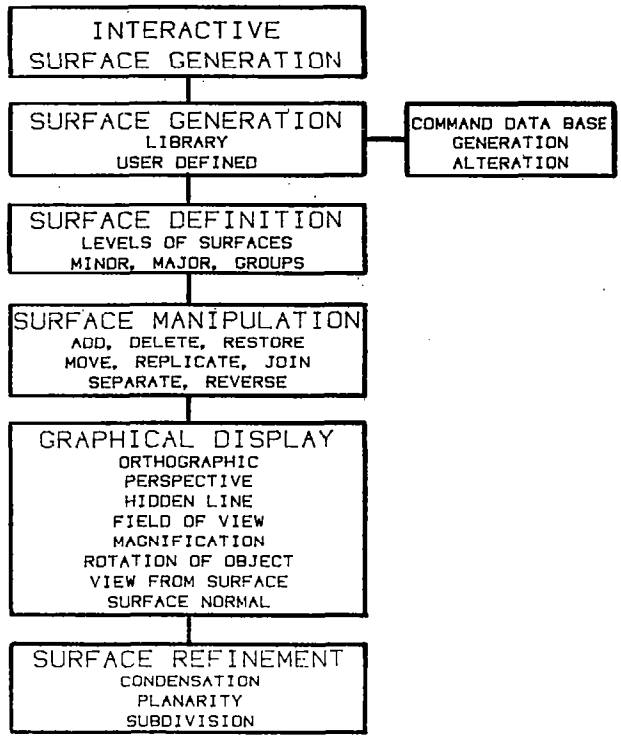


Figure 1.- Surface generation-program capabilities.

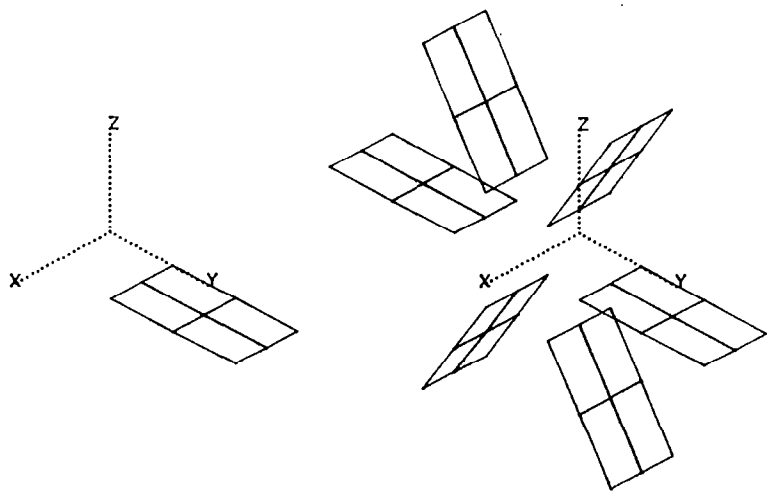


Figure 2.- Development of a group of surfaces.

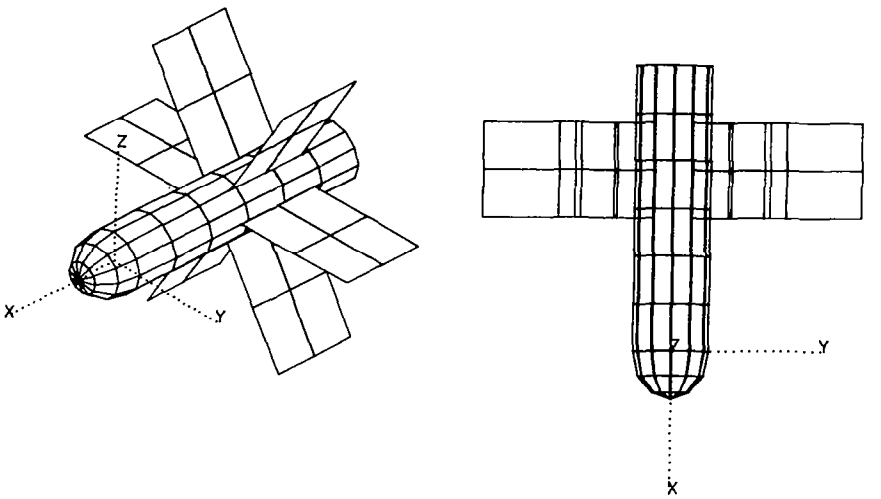


Figure 3.- Incorrect placement of panels, isometric and orthographic views.

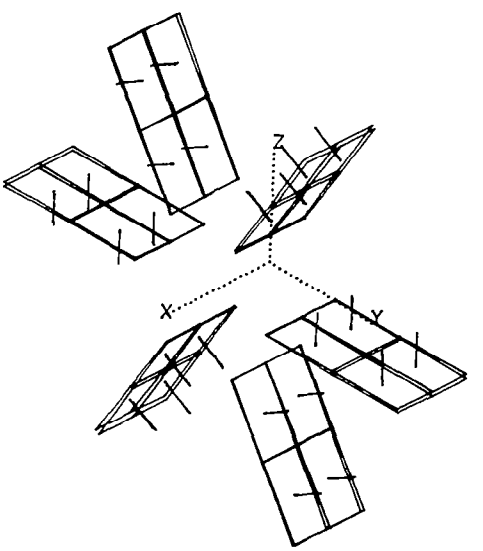


Figure 4.- Display of surface normals showing incorrect orientation.

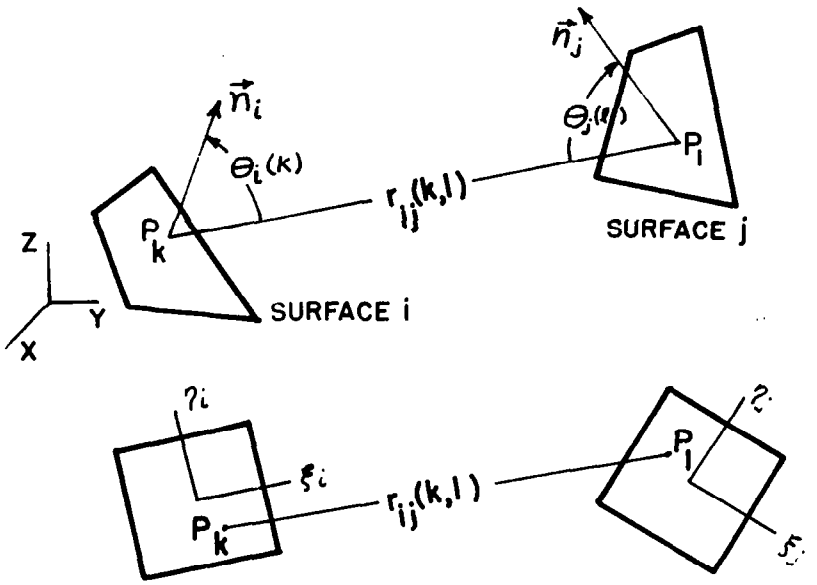


Figure 5.- Double area integration (showing mapping onto a unit square).

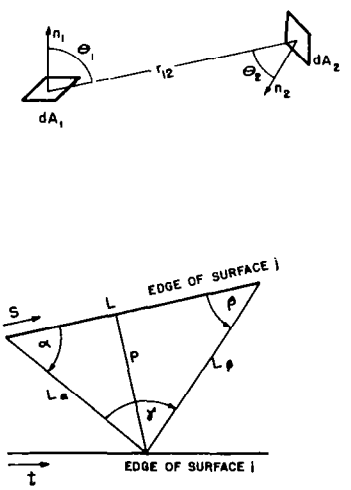


Figure 6.- Definition of angles and lengths for contour integration.

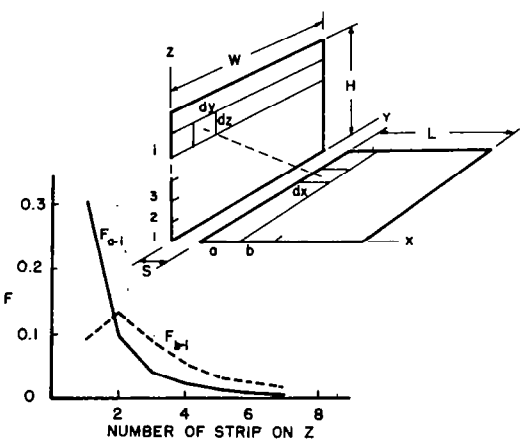
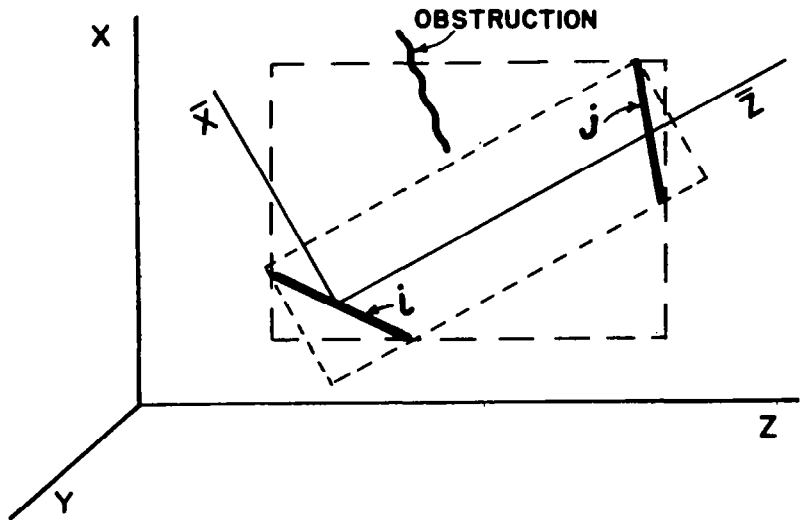
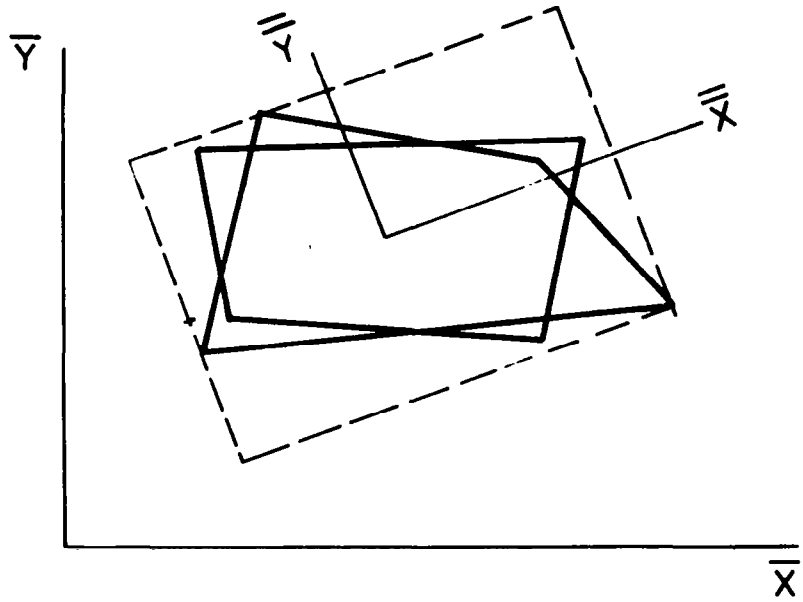


Figure 7.- View factors between surfaces at right angles to each other and separated by the distance S.



(a) Projected view of a view prism between surfaces i and j.



(b) Projected end view of the prism showing the rotation to achieve minimum cross section.

Figure 8.- Views of prisms between surfaces i and j.

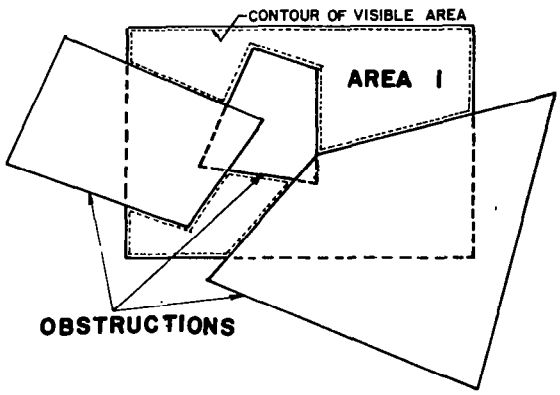


Figure 9.- View of surface i with interposed obstructions showing the multiple segmented contours.

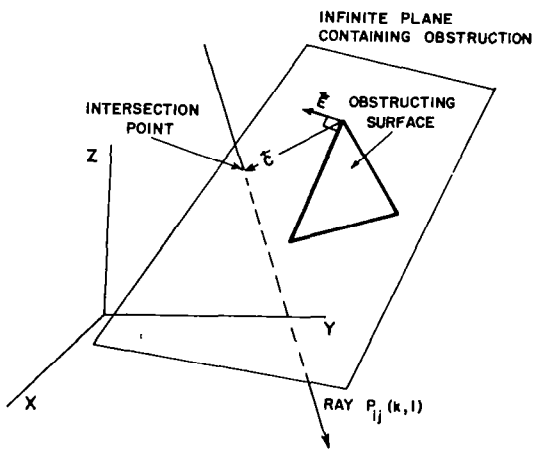


Figure 10.- Geometry for determining if the ray P(k,l) intercepts an obstructing surface.

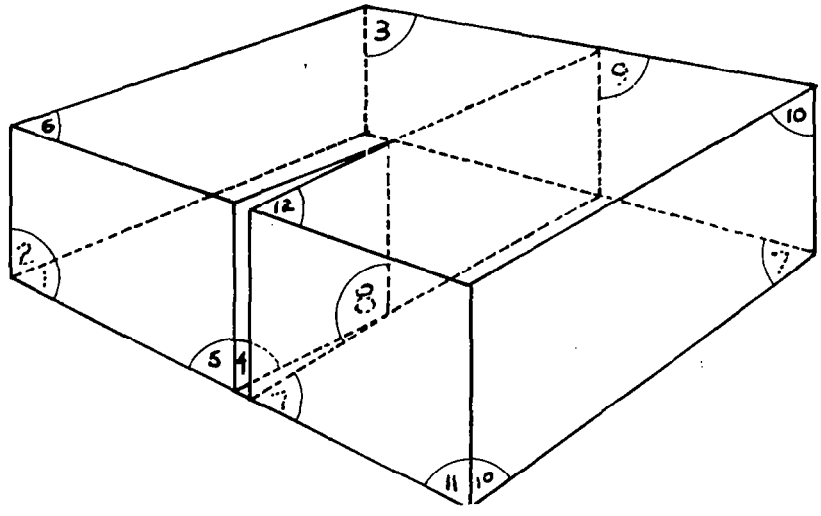


Figure 11.- The adjoint problem in which two contiguous surfaces have an obstructed view of each other.

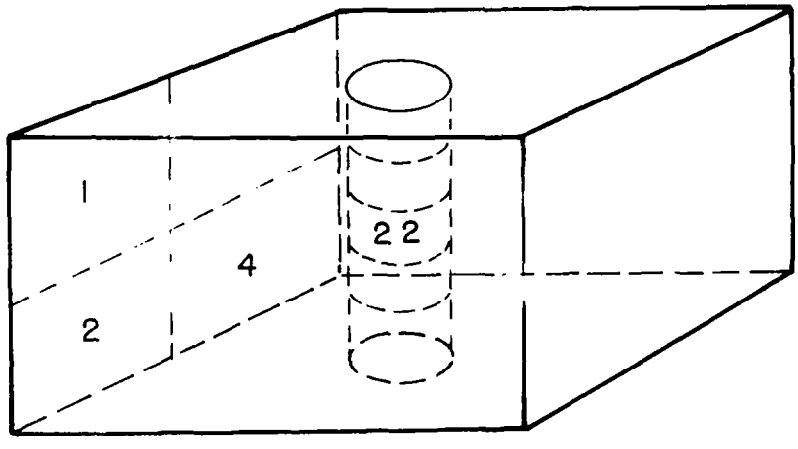


Figure 12.- The enclosed cylinder.