GEORGIA INSTITUTE OF TECHNOLOGY — OFFICE OF CONTRACT ADMINISTRATION

CORRECTED

PROJECT ADMINISTRATION DATA SHEET

[x] ORIGINAL
[ ] REVISION NO. ____

Project No. A-2986 — DATE: 7/23/81

Project Director: H. L. Bassett — ~~School~~/Lab RAIL/MAD

Sponsor: Rockwell International Corp., Columbus, OH

Type Agreement: Agreement for Services V161-SA-113203

Award Period: From 6/1/81 To ~~8/31/81~~ 9/30/81 ~~3/31/82~~ 4/30/82 (Performance) ______ (Reports)

Sponsor Amount: $10,000 — Contracted through:

Cost Sharing: none — GTRI/~~GIT~~

Title: Development of Tank RCS Model

ADMINISTRATIVE DATA — OCA CONTACT William F. Brown x4820

1) Sponsor Technical Contact: Mr. R. C. Perry, Manager; Systems Engineering; Missile System Division; P. O. Box 1259; Columbus, OH 43216

2) Sponsor Admin./Contractual Contact: Mr. A. George Miller, Senior Buyer, Rockwell International Corp., Missile Systems Division; 4300 E. 5th Ave.; P. O. Box 1259; Columbus, OH 43216 (614) 239-2440

Reports: See Deliverable Schedule — Security Classification: none

Defense Priority Rating: none

RESTRICTIONS

See Attached ______ Supplemental Information Sheet for Additional Requirements.

Travel: Foreign travel must have prior approval - Contact OCA in each case. Domestic travel requires sponsor approval where total will exceed greater of $500 or 125% of approved proposal budget category.

Equipment: Title vests with none proposed

COMMENTS:

AUG 1981

COPIES TO:

Administrative Coordinator
Research Property Management
Accounting Office
Procurement/EES Supply Services
Research Security Services
~~Reports Coordinator (OCA)~~
~~Legal Services (OCA)~~
Library, Technical Reports
EES Research Public Relations
Project File (OCA)
Other:

GEORGIA INSTITUTE OF TECHNOLOGY   OFFICE OF CONTRACT ADMINISTRATION

SPONSORED PROJECT TERMINATION/CLOSEOUT SHEET

Date April 10, 1984

Project No. A-2986   XXXXXX Lab RAIL-MSD

Includes Subproject No.(s)

Project Director(s) H.L. Bassett   GTRI / XXXX

Sponsor Rockwell International Corporation, Columbus, Ohio

Title "Development of Tank RCS Model"

Effective Completion Date: 4/30/82 (Performance) 4/30/82 (Reports)

Grant/Contract Closeout Actions Remaining:

- [x] None
- [ ] Final Invoice or Final Fiscal Report
- [ ] Closing Documents
- [ ] Final Report of Inventions
- [ ] Govt. Property Inventory & Related Certificate
- [ ] Classified Material Certificate
- [ ] Other

Continues Project No.   Continued by Project No.

COPIES TO:

Project Director
Research Administrative Network
Research Property Management
Accounting
Procurement/EES Supply Services
Research Security Services
Reports Coordinator (OCA)
Legal Services

Library
GTRI
Research Communications (2)
Project File
Other

Form OCA 60:1028

COMPLEX SCATTERING RADAR CROSS SECTION MODEL

by

D. A. Newton

Phase One Final Technical Report
GIT/EES Project A-2986

Prepared under

Rockwell International
Purchase Order V161-SA-113203

GEORGIA INSTITUTE OF TECHNOLOGY
Engineering Experiment Station
Atlanta, Georgia 30332

September 1981

TABLE OF CONTENTS

LIST OF FIGURES

COMPLEX SCATTERING RADAR CROSS SECTION MODEL

A mathematical model has been developed at Georgia Tech for estimating the radar cross section (RCS) of complex targets. The model was originally developed to predict the RCS of submarine masts and periscopes [1,2], and later expanded to include larger targets such as patrol boats, amphibious assault landing craft, ships, and tanks [3,4,5]. Under Purchase Order V161-SA-113203 from Rockwell International, these RCS modeling techniques have now been applied to the Soviet T-72 Main Battle Tank.

1. MODEL METHODOLOGY

The target is represented as a collection of many scatterers of simple geometric shapes. The major types of scatterers handled by the model at this time are triangular flat plates, spheres, truncated cone frusta, and dihedral and trihedral corners. The computer model calculates the RCS of the individual scatterers on the target by physical optics methods and forms the phasor sum to arrive at the total cross section. This RCS value is the effective cross section since the model takes into account the multipath effects. RCS calculations based on the use of physical optics theory to derive a closed form analytic solution for the cross section of the simple geometric shapes are preferable to "brute force" numerical integration techniques when considering the enormous amount of computer time to implement the latter.

1.1 SCATTERER SHAPES

The geometric shapes chosen to be scatterer types are general enough to be implemented in a variety of modeling tasks including ships, landing craft and planes, as well as the Soviet T-72 Main Battle Tank modeled for Rockwell International. Flat plates, the most common and versatile of the scatterer types representing the T-72, are used to model flat areas of the tank, such as the walls, and many rows of small flat plates are utilized in areas with some curvature, such as the turret. The second most prevalent geometric shape used on the T-72 is a cylinder. The gun barrel is composed of a collection of cylinders with different radii. Also the modeling scheme includes dihedrals and trihedrals which are used when two or three flat plates meet at near right angles, creating effective corner reflectors.

The triangle is chosen as the geometric shape for flat plate scatterers for several important reasons. For one, three noncolinear points define a plane. A polygon of more than three points is not used because of the difficulty encountered when locating more than three coplanar points. Thus, it is much easier with triangles to ensure that scatterers lie in the plane of the two-dimensional drawings used to construct the model. Another reason for choosing triangles is that any contoured surface can be approximated with triangles. The tank turret is a prime example of the use of triangles to approximate smooth contours. Triangular flat plates are defined by the Cartesian coordinates of the vertices as shown in Figure 1. These plates are assumed to be one-sided so that a counterclockwise numbering scheme of the vertices defines the direction of the outward pointing normal $\hat{n}$ as shown. This one-sidedness is useful in shadowing algorithms to determine whether the plate is facing towards or away from the radar.

The truncated cone frustum is another very general shape including both the cylinder and the cone as special cases. It is defined by the coordinates of the two ends of its central axis and by the radii of the two ends (see Figure 2). Note that when the two radii are equal, the frustum reduces to a cylinder, and when one of them is zero, it becomes a cone. All frusta are assumed open ended, so only the side of the frustum is modeled, not the ends or the interior. If the ends of a particular frustum are of importance, they can be covered by one or more flat plate scatterers.

1.2 TOTAL TARGET RCS

For each scatterer on the target, an effective RCS value is calculated by taking into account the multipath effects from the earth's surface. Multipath is the interference between the radar waves that travel directly from the radar to the scatterer and those that bounce off the earth's surface before and/or after reaching the scatterer. The model utilizes inputs from the user to calculate the attenuation of the signal that bounces off the earth's surface.

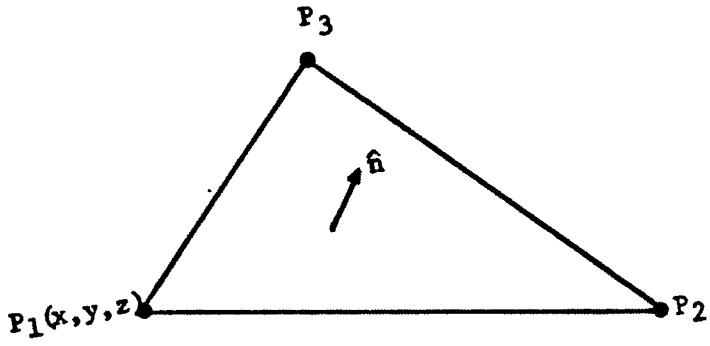


Figure 1. Triangular flat plate.

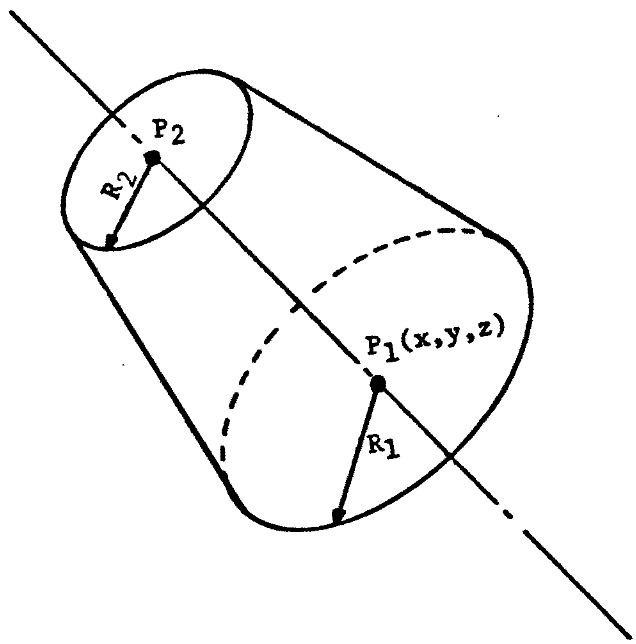


Figure 2. Truncated cone frustum.

The computer model has several options for adding the returns from the individual scatterers to form an RCS for the entire tank. One of the methods is known as a coherent summation in which the relative phase angle of each scatterer is included in the summation. This can cause effects such as constructive and destructive interference between the scatterers. The phase of the return signal from each scatterer can also be calculated in one of two ways, depending upon whether the target is in the near or far field. The far field is where the incident waves are approximately planar as opposed to spherical. The usual criterion for determining whether a target is in the far field is:

$$R > 2d^2/\lambda$$

where R is the range from the radar to the target, d is the maximum dimension of the target element, and λ is the radar wavelength. Whether the target is in the far field or the near field, the length of the path from the radar to each particular scatterer is correctly calculated to determine that scatterer's phase angle. Note that the model always assumes a plane wave over the dimensions of an individual scatterer. It is only in looking at the target as a whole and in the relative phases between the scatterers that the near/far effects are considered.

Another method for finding the total RCS of a target is known as incoherent summation. Here, the phase angles are not considered so only the magnitudes are added. An incoherent summation would be deemed more useful when the main question is detectability, since in detection analysis, the target will usually be quite some distance from the radar and the radar return will typically be integrated over some number of radar pulses. At these distances, the target subtends a very small solid angle with respect to the radar so a plane wave front of constant phase is incident over the entire target, and pulse-to-pulse phase effects are averaged out over the integration time of the radar signal processor. The major differences between coherent and incoherent predictions are in the depth and frequency of the nulls and the heights of the peaks, due to phase effects.

For much closer ranges, however, a coherent summation is used, since a near field prediction is totally dependent on the phase of each scatterer's return. Calculations on a pulse-by-pulse basis, such as those required for an impact point prediction for a seeker, also depend strongly on phase effects. For this case, a coherent summation is used.

Before a scatterer's RCS is added to the total sum, it is multiplied by a weighting factor known in the program as WT. This "weighting factor" can be any real number between 0 and 1, where a WT = 1 represents a perfectly reflecting metallic surface. These WTs are used quite extensively to simulate nonmetallic scatterers such as dielectrics (fiberglass, for example) or radar absorbent material (RAM).

1.3 MODEL LIMITATIONS

One feature not considered by the model is the effect of cavities, or re-entrant features, such as the driver's hatch. The theoretical problems associated with developing an accurate analytical description of the return from arbitrary re-entrant structures are practically insurmountable at the present time. Another scatterer type not considered in the model is the general doubly curved surface (hyperboloid, ellipsoid, etc.). The only doubly curved surface available is the spheroid, which is used mainly to represent antenna radomes on ships and other naval craft. The major feature with a doubly curved shape on the T-72 is the turret, but the radii of curvature of the curved sections of the turret are generally so large that, to a good approximation, the turret is relatively flat over small regions. Thus, the turret is most easily modeled as a collection of triangular facets (flat plates), and there is no need to consider general doubly-curved surfaces as a unique scatterer type.

2. T-72 TANK RCS MODEL

Any results derived from the analysis of a model are no better than the model itself. Consequently, a crucial consideration in evaluating the results of this project is the fitness of the model. However, before one can examine the question of the fitness of a model, one must establish a measure of fitness; that is, one must examine the function or purpose of a model and determine how well the model satisfies this function or purpose.

The model is first and foremost a radar model. The fitness of the model is a function only of how well it simulates what a radar would "see" when illuminating a real tank. The modeling effort is not intended to be a scaled down replica of the actual target, correct in every detail. Rather it is an attempt to simulate the target as it appears to a radar at millimeter wavelengths. Most details are significant to the radar at millimeter wavelengths, therefore the model includes many small structures that would be excluded if the model were to be constructed solely for use at longer wavelengths. Yet, there are details that are significant to the human eye which are insignificant to the radar; for example, the support brackets used to carry the snorkel on the left side of the turret. These brackets are partially obscured by the snorkel and may not be constructed of metal. Items like these are not good reflectors and are usually excluded from the model whether it is created for microwave or millimeter wavelengths. Nonetheless, details which are significant to the radar are present and are modeled to simulate as nearly as is feasible what the radar would actually "see" when illuminating the actual tank.

The above introductory remarks aside, the following discussion examines how the individual components of a typical tank model are determined, how the model is structured in the computer, how the model is handled and manipulated by the user, and how various features of the tank are modeled. Before surveying the model as a whole, one must examine each of the individual components.

2.1 SCATTERER COORDINATE DETERMINATION

First, the question of how these components are initially created must be addressed. The creation of a component is actually the determination of the weight factor, identification code, and coordinates of a particular element of the model and the entry of this information into a data file. Since weight factors (to be discussed later) are determined by straightforward calculation, and identification codes (also to be discussed later) by some reasonable, although often arbitrary, naming scheme, the discussion to follow will focus on the determination of component coordinates.

Three methods are available for use in the development of the tank model for determining component coordinates and entering these coordinates into a data file. The Strip Mode Digitization (SMD) method is the most efficient way to enter coordinates. In this digitization process, one first places a photograph or drawing of some portion of the tank on a digitizer tablet and enters three reference points of known coordinates. After entering the reference points, one enters unknown points from the drawing or photograph, and the computer then determines the coordinates of these points and enters them in a data file.

The SMD method can be illustrated by considering the turret of the T-72. Figure 3 shows the turret at an 90° elevation angle, hence the observer is looking straight down on the turret. The cross-sectional lines which are drawn for the front left quarter of the turret represent cuts through the tank in given z-planes. In general, if the contour lines of the turret, in the x-y plane, are defined by N rows of M points per row, one would have to digitize M points per row and enter Z-coordinates of each rib cross section. The digitization algorithm then fills in each of the M-1 strips from the base of the turret to the top of the turret with triangles.

The second method of entering coordinates is the manual calculation of these coordinates from the tank drawings. The coordinates are then entered into a data file by hand. This method, although very simple and straightforward, is quite tedious and time consuming. The digitization process, as a whole, is much faster than the manual one, yet for some of the tank armament, it is more efficient to enter the points by hand because of the complexity of the armament. An example of this is the laser rangefinder.

A third method for entering coordinates is automatic digitization (AD). This method was developed to speed the manual coordinate determination and hand entry process. The difference in the AD process and the SMD process is that this method utilizes detailed drawings of two perspectives of the same portion of the tank simutaneously [6]. The available drawings of the T-72 were not adequate to use this process. This method, although not as efficient or accurate as the SMD method, could be used to determine the coordinates more precisely and probably quicker than the hand entry process for the equipment on the turret.

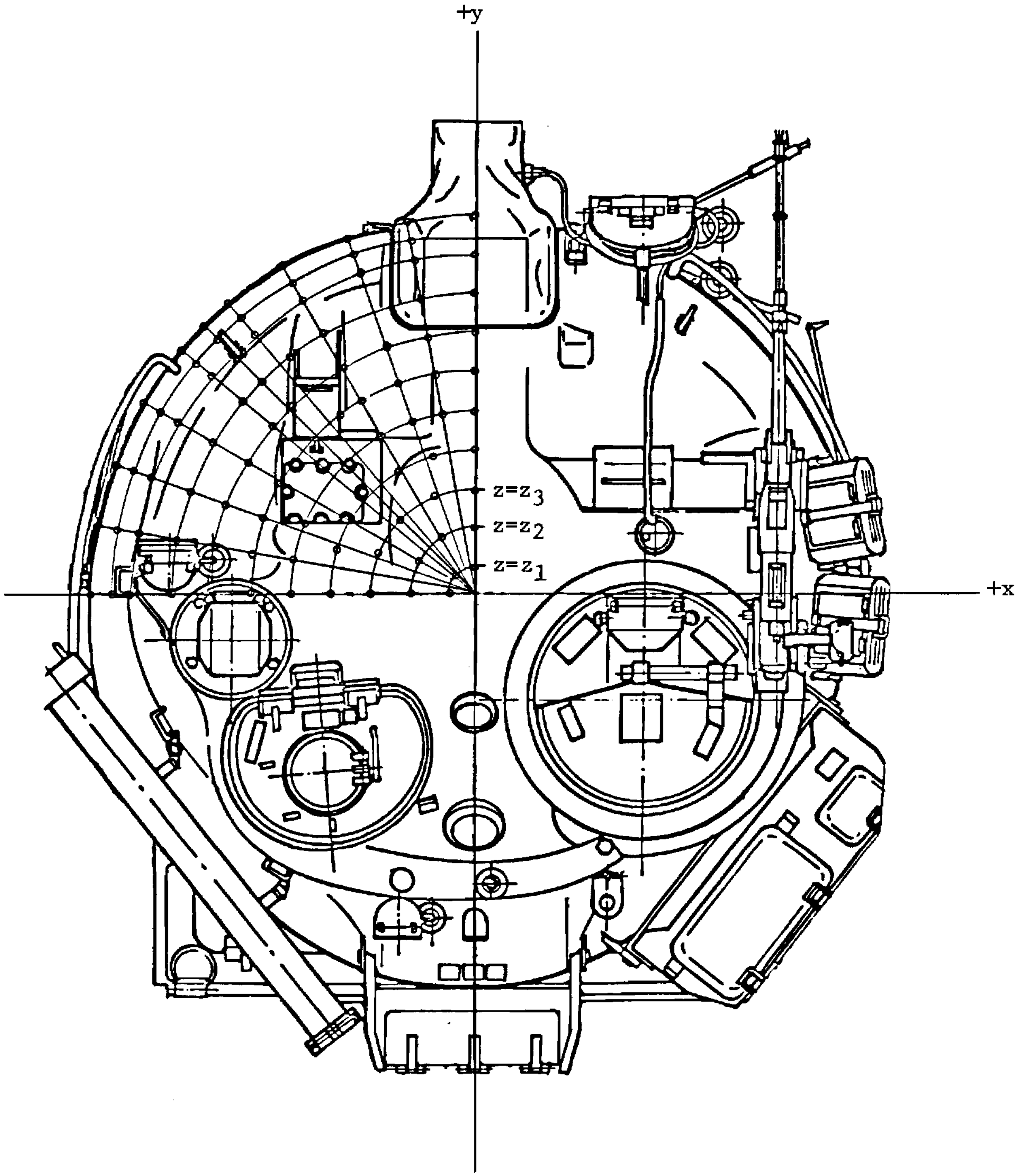


Figure 3. Blueprint of the T-72 turret used in the strip mode digitization method. Each cross-sectional line, as illustrated in the front left quarter, represents a cut through the turret in a given z - plane.

2.2 SCATTERER IDENTIFICATION CODES

A model can represent a large data file, requiring many kilobytes of computer storage. Because of volume, the data are stored as a binary rather than ASCII file to reduce memory requirements. Furthermore, it is a random access file, which minimizes the time required to access any given record within the file. Each record within the data file represents a single component in the tank model, and among other information, each record contains a unique record number and unique identification code. Whereas record numbers represent merely a sequential ordering of the components, without regard for function or physical location within the model, the identification code is much more informative from a model standpoint. For example, one record may have the identification code TURRET.HATCH.GUNNER.00040. This record is the fortieth triangular flat plate which, together with the other plates identified as TURRET.HATCH.GUNNER.*, composes the gunner's hatch located on the turret. Another example of the identification code is TURRET.BARREL.000001. This component is the first cylinder attached to the turret and, as the label suggests, is a portion of the gun barrel. The entire gun barrel is constructed with cylinders which are identified by the label TURRET.BARREL.*. These examples indicate the ease with which the records pertaining to any given portion of the model can be accessed in the data file.

2.3 SCATTERER WEIGHTING FACTORS

In addition to a unique number and identification code, each record also contains a weight factor (reflection coefficient) which is proportional to the attenuation of the electric field reflected by the component when illuminated by a radar. Such a weight factor can be used to take into account the presence of RAM on a particular model component, as well as the frequency-dependent attenuation of waves illuminating non-metallic surfaces.

2.4 DATA FILE MANIPULATION

Because of the size and structure of a typical tank data file, an efficient means is needed to access and manipulate records within the file. This requirement is satisfied by a computer program called EDITR. Among other capabilities this program permits the user to specify (by identification code or record number) and display a single record or a range of records. Thus, one might display the 114th and the 115th records in the file, or, by specifying TURRET.HATCH.GUNNER.*, for example, one can display the entire gunner hatch on the turret. Clearly, any portion of the model can be easily accessed by using the identification code scheme in conjunction with the wild card character *.

In addition to the two access modes described above, EDITR can delete a range of records or transfer a range of records from one tank file into another. Thus, one can easily investigate the effect on RCS of removing or adding equipment on the turret, such as tool boxes or supply drums, for example.

In addition to adding and deleting records from a data file, one can also use EDITR to modify existing records within that file. The user can modify the coordinates, the identification code, and the weight factor for any range of records, that range again being specified either by record numbers or by identification codes. The user also can shift a range of components by given distances in the x, y, and z directions. Using this capability, one can easily investigate the effects of moving a part of the tank, for example, a tool box, to another storage location. The EDITR also provides the capability of reflecting or rotating a specified range of records. In reflecting a range of records, EDITR creates the reflection of a given range of components about a plane specified by the user. The reflection capability of EDITR is illustrated in Figure 4. Obviously, such a tool can greatly reduce the amount of work in creating a model where symmetry exists, as in the case of the track and wheels, since one can create the left side of the tank and then obtain the right side by reflecting the left portion about the y plane.

The rotate option, illustrated in Figure 5, is another powerful tool for manipulating a model. To use the option, one merely specifies a line and an angle of rotation about the line. Thus, for example, to rotate a segment of the turret 30° off center, one simply specifies two points on the line determined by the intersection of the base of the tank and the turret, and the angle 30°. EDITR deletes the existing portion of the turret which was specified, and replaces it with the corresponding rotated segment.

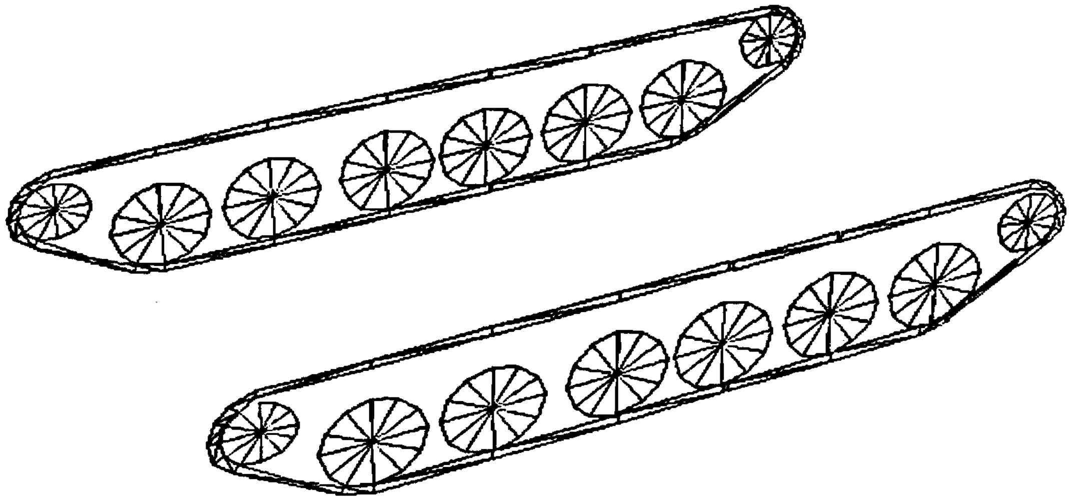


Figure 4. Computer-generated drawing illustrating the REFLECT capability of EDITR. Shown in the drawing are the left tracks and wheels and the right track and wheels are constructed by reflecting the left portion about the x-plane.

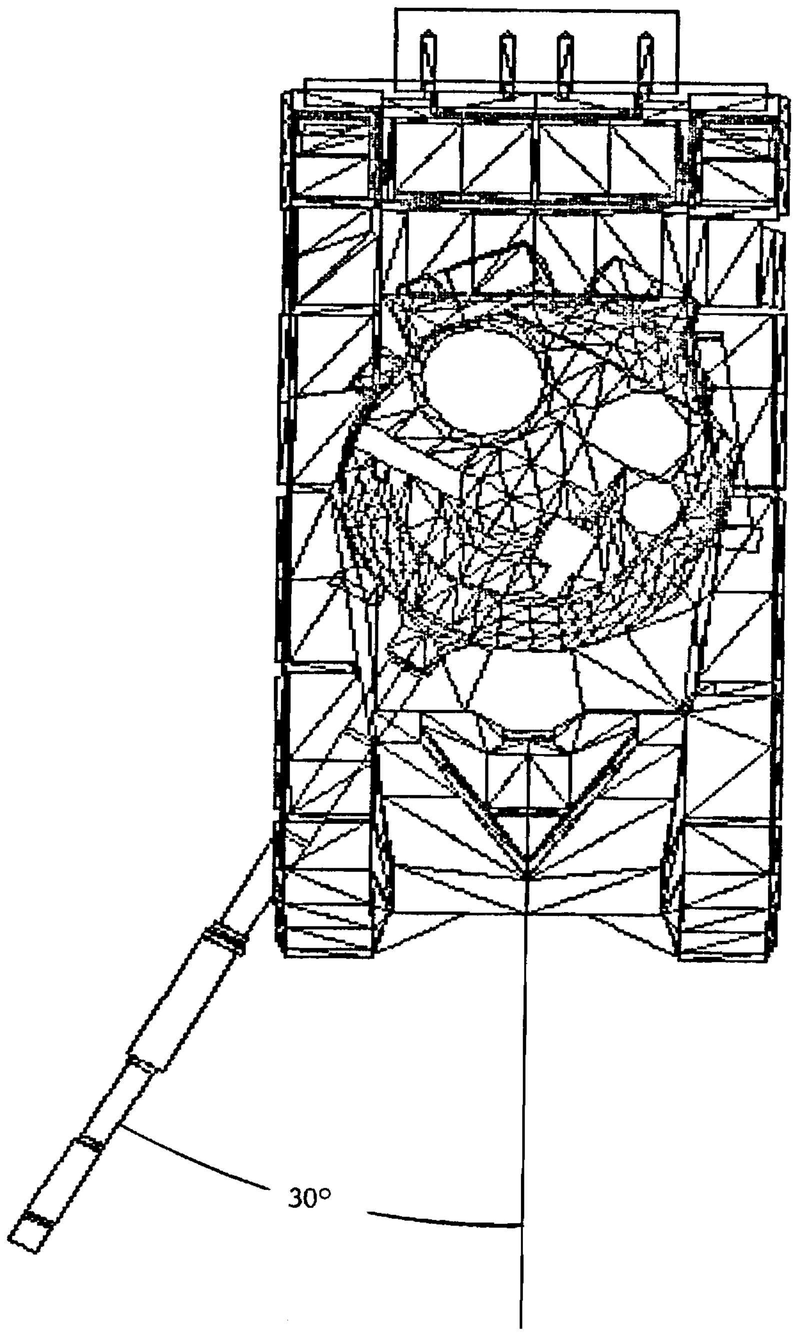


Figure 5. Computer-generated drawing illustrating the ROTATE capability of EDITR. Shown in the drawing is the entire turret of the T-72 rotated 30° from the center line of the tank.

The above discussion of the EDITR, while not exhaustive, is sufficient to indicate the great power of this program for manipulating models. Although a data file may be quite large, manipulating such a file is quite easy. With a basic knowledge of the identification code scheme used for the tank and a familiarity with the capabilities of EDITR, one can examine with a few command lines the RCS effects on a given tank of a great number of variations and changes: rotating the turret and moving the snorkel from its storage position on the side of the turret into its upright functional position on the top of the turret, or replacing the existing machine gun with a gun of different caliber. The list is endless.

Figures 6 through 8 illustrate the radar models of the Soviet T-72 Main Battle Tank which are developed using the techniques described above. Obviously, the value of any study of tank RCS performed on the computer is directly proportional to the radar accuracy of the model. That is, the model must approximate as nearly as possible the actual tank from a radar point of view. Features which seem important to the human eye may or may not be important from the standpoint of the radar. One should keep this distinction firmly in mind, for the model is first and foremost a radar model. The radar model of the tank is modeled specifically for millimeter wavelengths; therefore most small details which are normally omitted at longer wavelengths, are included in this model. Some features which might seem important to a casual observer (for example, the support brackets on the left rear side of the turret) are judged to be relatively insignifcant to the eye of the radar and are not included in the model. The features which are excluded normally are excluded because they are poor reflectors, not simply because of size. In each case, the question of whether to include a particular feature or not is answered by consulting the radar rather than the eye.

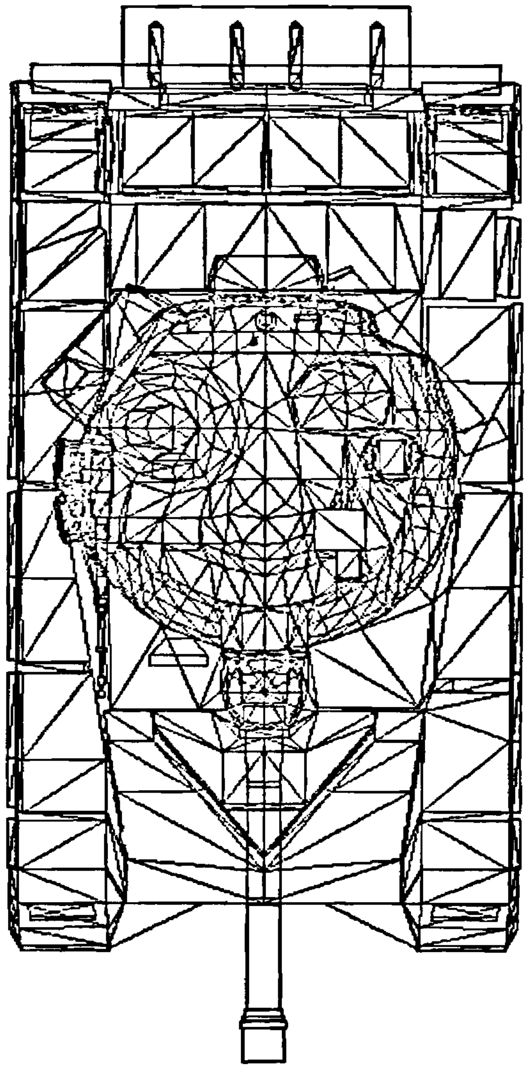


Figure 6. Computer model of the Soviet T-72 tank at an elevation angle of 90°.

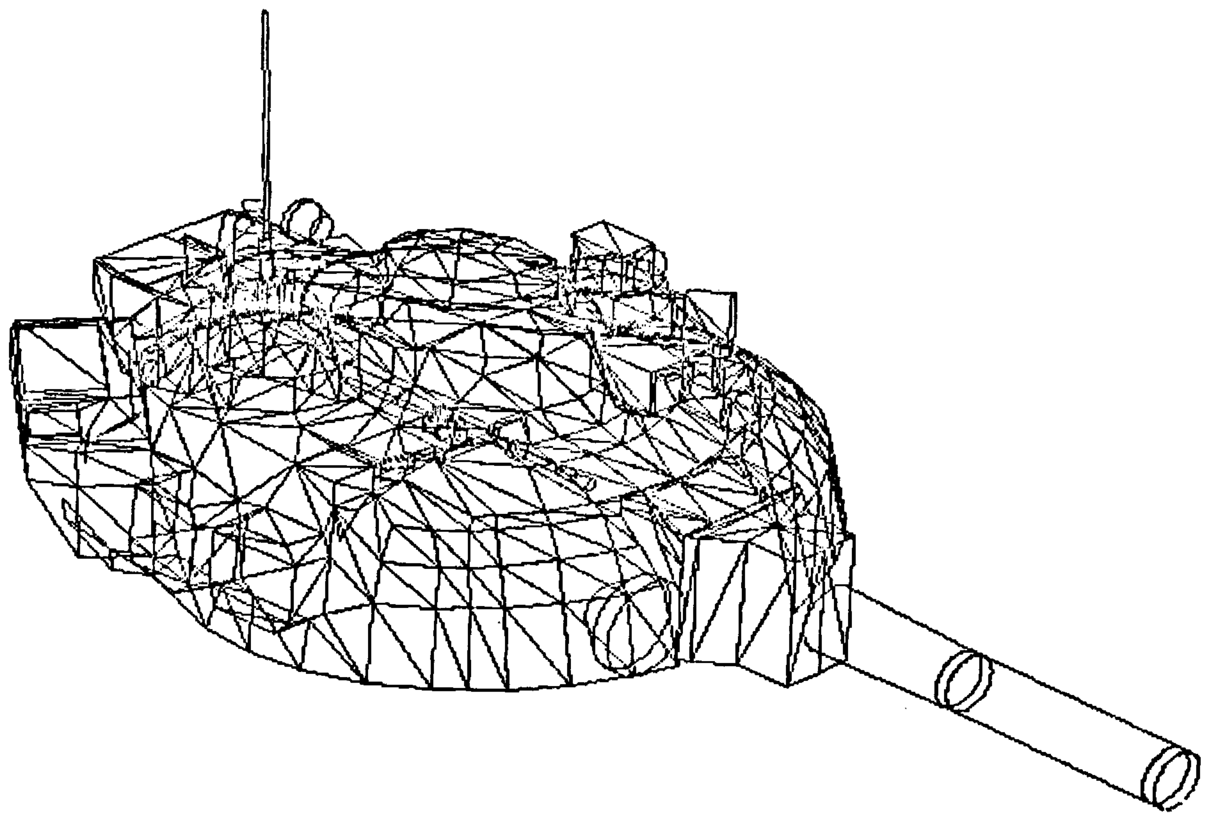


Figure 7. Computer model of the T-72 turret at an elevation angle of 30° and an azimuth angle of 45°. (Note: 0° and 90° azimuth angles represent the center front and the right side of the tank respectively).

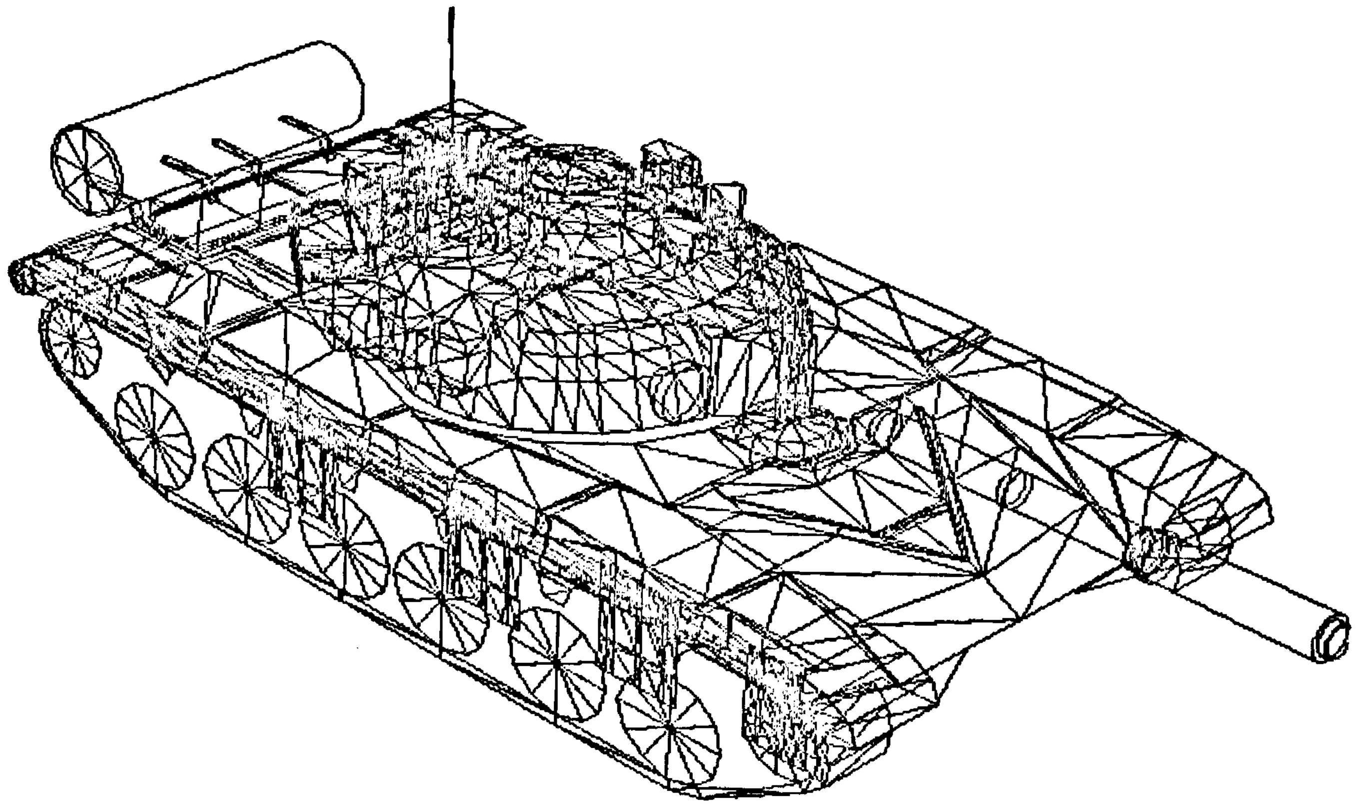


Figure 8. Computer model of the T-72 at an elevation angle of 30° and an azimuth angle of 45°.

REFERENCES

1. M.T. Tuley, M.M. Horst, and K.B. Langseth, "Radar Modeling Studies (U)," Task 7 Final Report on APL Subcontract 600403, Georgia Institute of Technology, Engineering Experiment Station, July 1978, SECRET, XGDS.

2. M.M. Horst, D.A. Newton, and R.B. Rakes, "RCS Modeling of Submarine Masts (U)," Task 2 Final Technical Report on Contract N00039-79-C-0467, Georgia Institute of Technology, Engineering Experiment Station, June 1981, SECRET.

3. M.T. Tuley, W.M. O'Dowd, and F. B. Dyer, "Radar Cross Section Reduction of Ships (U)," Phase One Final Technical Report on Contract N00039-79-C-0676, Georgia Institute of Technology, Engineering Experiment Station, April 1975, SECRET, XGDS.

4. M.M. Horst, D.A. Newton, and E.E. Martin, "JEFF AALC RSC Measurements and Modeling (U)," Final Report on Contract N00612-79-D-8004 HR-18, Georgia Institute of Technology, Engineering Experiment Station, January 1981, SECRET NOFORN.

5. M.M. Horst, et al., "Ship Radar Cross Section Preditions (U)," Final Report on Contract N00167-80-C-0048, Georgia Institute of Technology, Engineering Experiment Station, March 1981, SECRET.

6. J. E. Sutherland, "Three Dimensional Data Input by Tablet," Proc. IEEE, Vol. 62, No. 4, April 1974.

Phase II
PROGRESS REVIEW

Rockwell International
Purchase Order V161-SA-113203

17 September 1981

GEORGIA INSTITUTE OF TECHNOLOGY
Engineering Experiment Station
Atlanta, Georgia 30332

GIT/EES Project A-2986
Deliverable Number 2

A-2986
PROGRESS REVIEW
17 September 1981

A review of Phase II efforts under Rockwell Purchase Order V161-SA-113203 was held at Georgia Tech on 17 September 1981. Those present were Carl Bates and Bob Bensinger from Rockwell International, and Margaret Horst, Bruce Rakes, and Debbie Newton from Georgia Tech. The overall effort of the project was discussed by phase, with emphasis on Phase II efforts.

Phase I

A preliminary copy of the Phase I Final Report, " A Complex Scatterer RCS Model" was given to Carl Bates. Included in the report are line drawings of the T-72 computer model from different perspectives. Color reproductions of these line drawings as well as color pictures of the T-72 computer model using a light shadowing algorithm were also given to Carl Bates.

A demonstration of the completed computer model of the T-72 tank was presented to Carl Bates and Bob Bensinger using the VAX 11/780 computer and the Chromatics, an eight-color high resolution graphics device. Some of the capabilities of other Georgia Tech software were illustrated also, including the program EDITR, which accesses, edits, manipulates, and displays records within the model data file. A video presentation, which included the T-72 as well as other models developed by Georgia Tech, was given by Bruce Rakes. The possibility of Bruce presenting this short video tape of Georgia Tech's modeling capabilities to others at Rockwell was also discussed.

Phase II

A flow chart of the computer simulation to be developed under Phase II is presented on the following page. The Rockwell Driver was not discussed in detail. The remaining routines in the flow chart were discussed in detail and are to be supplied to Rockwell by Georgia Tech.

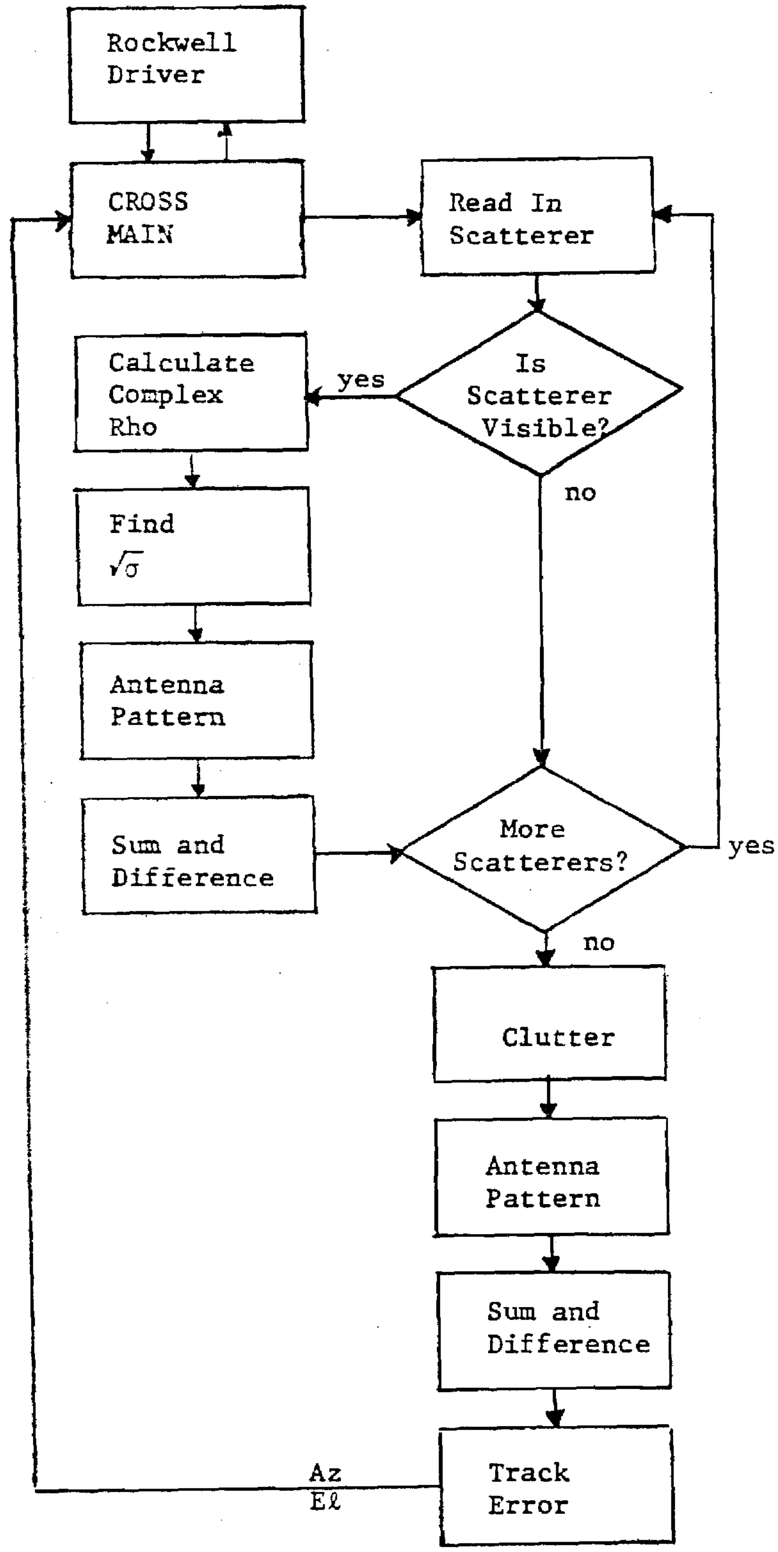


Figure 1. Flow chart of RCS and Track Error routines to be developed under Phase II.

CROSS MAIN is the main program which calls the remaining portion of the Radar Cross Section (RCS) and Track Error program. Basically, CROSS MAIN receives the inputs from the Rockwell driver, initializes and converts the inputs to their correct units, and calls the Read in Scatterer procedure.

This procedure reads in the record of each individual scatterer of the target model, including its record number, its scatterer type (either a flat plate or a cylinder), its identification code, its weighting factor, and either three end points, if the scatterer type is a flat plate, or two end points and two radii if the scatterer type is a cylinder. The routine then determines if the scatterer is visible to the radar. If the scatterer cannot be seen by the radar (e.g., a flat plate facing away from the radar), its contribution to target RCS is zero, and the routine proceeds to the next scatterer in the file.

The next step in calculating the RCS of a scatterer that is visible to the radar is determination of the complex reflection coefficient of the surface, which is needed for the multipath calculations. The reflection coefficient, ρ, can be either calculated using the Calculate Complex Rho routine, or can be an input, in which case this routine will be ignored. In both cases a value for ρ will be needed in the input file. If a calculated value for ρ is desired, then the value for ρ in the input file is a flag for the program to calculate a complex ρ for each scatterer. If the value for ρ in the input file is the actual value, then the value will be used as a flag to skip the Calculate Complex ρ procedure and call the Find $\sqrt{\sigma}$ routine directly.

$\sqrt{\sigma}$, the quantity calculated in the routine named Find $\sqrt{\sigma}$, is the complex scattering length. The quantity desired from the model is Radar Cross Section, σ, which has units of area, and is a real number associated with received power in the radar equation. Since the return comprises several contributions, the complex voltages, $\sqrt{\sigma}$, should be summed before the amplitude is found and squared. The scattering length is determined from:

$$\sqrt{\sigma} = G_f + 2\rho\ G_d + \rho^2 G_I \tag{1}$$

ρ = complex voltage reflection coefficient

G_f = component of return due to the target, the direct-direct path, as if the ocean were not present (the free space contribution)

G_I = component of return due to the image, the indirect-indirect path. This term must be multiplied by the square of the voltage reflection coefficient of the ocean surface, because the path involves two bounces from the surface.

G_d = component of return due to the diplane term, the indirect-direct path or the direct-indirect path. This term must be multiplied by the voltage coefficient because of the single bounce and doubled because of the two reciprocal propagation paths.

After a value for the scattering length is found for a scatterer, the program calls the Antenna Pattern routine. This routine uses a standard antenna pattern and multiplies the antenna gain in the appropriate direction by $\sqrt{\sigma}$. This value is then passed to the Sum and Difference routine to determine the sum and difference patterns, which are needed for the Track Error calculation.

The next step of the program decides if there are more scatterers left in the model which have not been read into the program. If so, the program calls the Read in Scatterer routine and each appropriate subsequent routine through the Sum and Difference routine. When all the scatterers in the model have been stepped through, the clutter routine is then called.

The clutter routine is a standard algorithm which calculates the clutter from a flat homogeneous earth and then calculates the antenna gain for the clutter contribution. The Sum and Difference patterns are updated with clutter information, and the Track Error is calculated in both the azimuth and elevation directions in the Track Error routine. The track errors in elevation and azimuth are proportional to the difference signal divided by the sum signal. These values are then passed to CROSS MAIN and are returned to the Rockwell Driver.

The inputs from the Rockwell Driver to the CROSS MAIN program were also discussed in detail. The following are the radar parameters to be supplied by Rockwell: frequency (GHz), pulse width (μs), and azimuth and elevation beamwidth (degrees). Clutter per unit area (dBsm/m^2) and the complex reflection coefficient, ρ, will also be inputs. The position of the target is specified as an input. The x, y, and z coordinates of the tank, from the radar, are specified in feet, along with the distance, Δx, to the center of the tank (feet).

The orientation of the model can be specified by the yaw, pitch, and roll angles (degrees) which are input by the Rockwell Driver. Yaw, pitch, and roll are right handed rotations about the z, x and y axes, respectively. The fixed body rotations are as follows: a positive yaw rotates the tank to the left, a positive pitch rotates the tank upwards, and a positive roll rotates the tank to the right.

The T-72 tank model can probably be used to simulate the newer T-80 tank; however, there is a possibility that the T-80 tank is equipped with a hydraulic system which elevates the tank some unknown distance. Because of the limited availability of information on the T-80, it is not known when the tank is raised (while moving or stationary) or the elevation to which it is raised. Therefore, to accommodate for the unknown, the height of the target can be manipulated by changing the orientation of the tank in the z-position. Hence, if more information becomes available in this area, the model can accommodate either a T-72 or T-80 model.

Phase III

The last major topic of discussion for the Review meeting involved suggestions and recommendations for further work. Several tasks were identified as immediately necessary or desirable as part of Phase III:

(1) Develop software for the coordinate transformations necessary to make the GIT RCS model and Track Error routines compatible with the Rockwell missile simulation; i.e., write the software interface between the Rockwell Driver and CROSS MAIN.

(2) Deliver to Rockwell International a digital magnetic tape containing:

a) Source code for GIT RCS/Track Error model and software interface to Rockwell Driver.

b) Source code for FLECS preprocessor

c) Data file for the tank model developed under Phase I.

d) Data file for one simple test target.

e) Source code for a simplified translator program for target data files.

(3) Deliver User Documentation for the GIT RCS/Track Error model, including documentation on model methodology.

(4) Provide support to Rockwell International in Columbus, Ohio, to aid in implementing the software delivered in (2) above, to verify that the software performs accurately on their PDP-11 computer, and to train Rockwell personnel in exercising the software.

(5) As part of the validation effort to ensure that the software is properly installed on Rockwell's PDP-11, prepare a set of benchmark runs at Georgia Tech for comparison with results obtained at Rockwell. The benchmark data set shall include, at a minimum, 360° azimuthal polar plots of tank RCS and track error angle at 1° azimuthal increments for elevations of 15° and 25°. The main beam of the antenna shall be directed toward the defined origin for the tank data file coordinates.

(6) Analyze the effects of the "poor man's" hidden surface removal algorithm delivered with the GIT RCS model, as compared to a more sophisticated hidden surface removal algorithm, by computing RCS and track error using full hidden surface removal for a selected subset of conditions covered by the benchmark data described in (5) above. Compare the results with the benchmark data and make recommendations in a short memorandum report.

(7) Provide a table of suggested values for σ^{o} (clutter cross section per unit area), σ_s (rms surface roughness), and ρ (the complex reflection coefficient), for different types of terrain of interest (e.g., open field, forest, urban area, snow), in a brief memorandum report explaining the suggested choices.

Other tasks identified as desirable outgrowths of current and proposed efforts, but probably scheduled for farther downstream, include:

(8) Explore means of identifying missile impact points on the tank, possibly through identification of the scatterer in the tank model whose midpoint is closest to the impact point. Assist Rockwell International in the preparation of visual aids (photographs, slides, vu-graphs) depicting results of selected simulations, utilizing GIT computer graphics software and output devices.

(9) Implement a diffuse scattering term in the RCS model to allow for surface roughness in target elements.

(10) Validate the RCS model by comparing predicted RCS with actual measured data. This could be accomplished in one of three ways:

a) Georgia Tech could be tasked to develop a computer model for a specific target for which high quality, well-calibrated measured RCS data at frequencies of interest already exist in the literature (open or classified).

b) Georgia Tech or some other agency could be tasked to record calibrated RCS data at the appropriate radar frequencies on the T-72, for geometries similar to those encountered in the simulation.

c) A vehicle other than the T-72 could be selected; Georgia Tech tasked to model it; and Georgia Tech or some other agency tasked to measure its RCS.

Option (a) would probably have the lowest cost and, not surprisingly, the greatest uncertainty in the results. It is very difficult to conduct a model validation using data not gathered specifically for that purpose and possibly not documented adequately for that purpose. Option (b) is reasonable, subject to the availability of a T-72 tank. Georgia Tech possesses the appropriate instrumentation radars and expertise to carry out such a measurement program, or to act in an advisory capacity to others. If an appropriate T-72 is not available, Option (c) is a possibility. The vehicle choice would depend on availability as well as interest in the vehicle; if possible, selection of a vehicle for which some measured RCS data exist in the literature would provide an independent check on measurement accuracy as well as model validity.

SOFTWARE USER DOCUMENTATION
PROJECT NO. A–2986

RADAR GLINT MODEL

By

R. B. Rakes

Prepared for

ROCKWELL INTERNATIONAL
MISSILE SYSTEMS DIVISION
COLUMBUS, OHIO 43216

APRIL 1982

GEORGIA INSTITUTE OF TECHNOLOGY

A Unit of the University System of Georgia
Engineering Experiment Station
Atlanta, Georgia 30332

Software User Documentation

Georgia Tech/EES Project A-2986

RADAR GLINT MODEL

By

R. B. Bakes

Prepared for
Rockwell International
Missile Systems Division
Columbus, Ohio 43216

under Agreement V161-SA-113203

Prepared by

GEORGIA INSTITUTE OF TECHNOLOGY
Engineering Experiment Station
Atlanta, Georgia 303332

April 1982

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SECTION 1. INTRODUCTION

This user's manual for the Georgia Tech Radar Glint computer model is intended as an aid for the programmer in the implementation of these routines into an existing missile seeker model. It is also meant to be used as a reference by the user of the missile model to aid in constructing the data files used by the Radar Glint routines.

The Radar Glint model enhances a missile model by simulating a monopulse tracking radar employed on the missile. The primary outputs of the model are the azimuth (yaw) and elevation (pitch) track errors which are used as inputs to the guidance portion of the missile simulation. One of the main advantages of this program is that the target of the missile is modeled as an extended target, i.e., as a large collection of individual scatterers, for the calculation of the radar cross section (RCS) as opposed to a single point target with a constant cross section. This allows for the effects of glint to be modeled more accurately. Another advantage of this computer model is that the effects of multipath and clutter from the ground plane are also included. As an added bonus, the program returns the location of the missile's impact point on the target (assuming it didn't miss!).

The implementation of these routines is relatively straightforward. Due to the modular nature of the Radar Glint model, the missile program needs to be modified at only two points with a minimum number of variables being passed. For ease of readability, these routines were written in FLECS (an extended version of Fortran) and developed on a Vax 11/780 computer. They are also available for a SYSTEMS 32/7780 computer. Detailed instructions for the implementation of the Radar Glint Model are found in Section 2.

Two data files are used for input to the Radar Glint model. The first of these is a geometry file that describes the spacial orientation of the target in a fixed reference frame. This data file also contains the terrain parameters for multipath and clutter. The other data file is the target model itself. It is a direct access, record oriented file that contains the three dimensional coordinate information to describe each of the scatterers that make up the target. The format of the target data file as well as the geometry file are described thoroughly in Section 3. As an additional programmer's aid, a listing of the entire program as it presently exists on the VAX 11/780 is included in Appendix A.

The Radar Glint model was developed from two earlier Georgia Tech computer programs: the Multipath/Clutter routines for the TAC ZINGER program[1] and the Radar Cross Section (RCS) model known as CROSS.[2] The Multipath/Clutter model produces track errors for a monopulse radar, but is restricted to a single point target. The CROSS model was originally designed to calculate the RCS of ships, but has since been adapted to find the near field RCS of many other three dimensional targets including land as well as sea targets. Thus, a very powerful tool has been developed by combining these two programs into the present Radar Glint algorithm.

SECTION 2. IMPLEMENTATION OF PROGRAM MODULES

The Radar Glint model is designed as a set of subprograms to be inserted into a missile seeker program. The purpose of this section is to describe the procedures involved in implementing these routines.

The set of subroutines that make up the Radar Glint model is written in a structured superset of Fortran 77 known as FLECS (Fortran Language Extended Control Structures). Before these routines can be compiled by the Fortran compiler, they must be run through the FLECS preprocessor to translate the FLECS sources to standard Fortran. The FLECS preprocessor is available for the VAX 11/780 and the PDP 11 family as well as many other computers. Since FLECS is completely compatible with standard Fortran, there should be no problem in interfacing the Radar Glint model with a missile model written in standard Fortran.

To simplify the implementation of this model, only two subroutines are to be called directly by the main missile program with a minimum number of arguments being passed. The I/O used by the Radar Glint routines such as the handling of the target and geometry data files (see Section 3) is taken care of entirely by the subprograms and should not interfere at all with the operation of the main routine. Thus, the only modification to the main missile model should be the placement of the two subroutine calls. To aid in this placement, Figure 1 illustrates a simplified flow chart showing the primary Radar Glint modules and how they would interface with a typical missile model. The modules surrounded by the dashed lines represent the Radar Glint subroutine calls.

The first of the two subroutine calls to be placed in the missile program is a call to the subroutine MULTIN (for MULTIpath INitialization). This subroutine call should be inserted in the initialization portion of the missile program and should only be called once since it serves to initialize the Radar Glint model. The MULTIN routine opens three data files used internally by the Radar Glint model. The programmer must insure that the unit numbers used for these three data files are unique, i.e., they must not be used anywhere else in the missile program. The unit numbers currently used in the Radar Glint routines are 2, 3, and 5, which refer to the target data file,

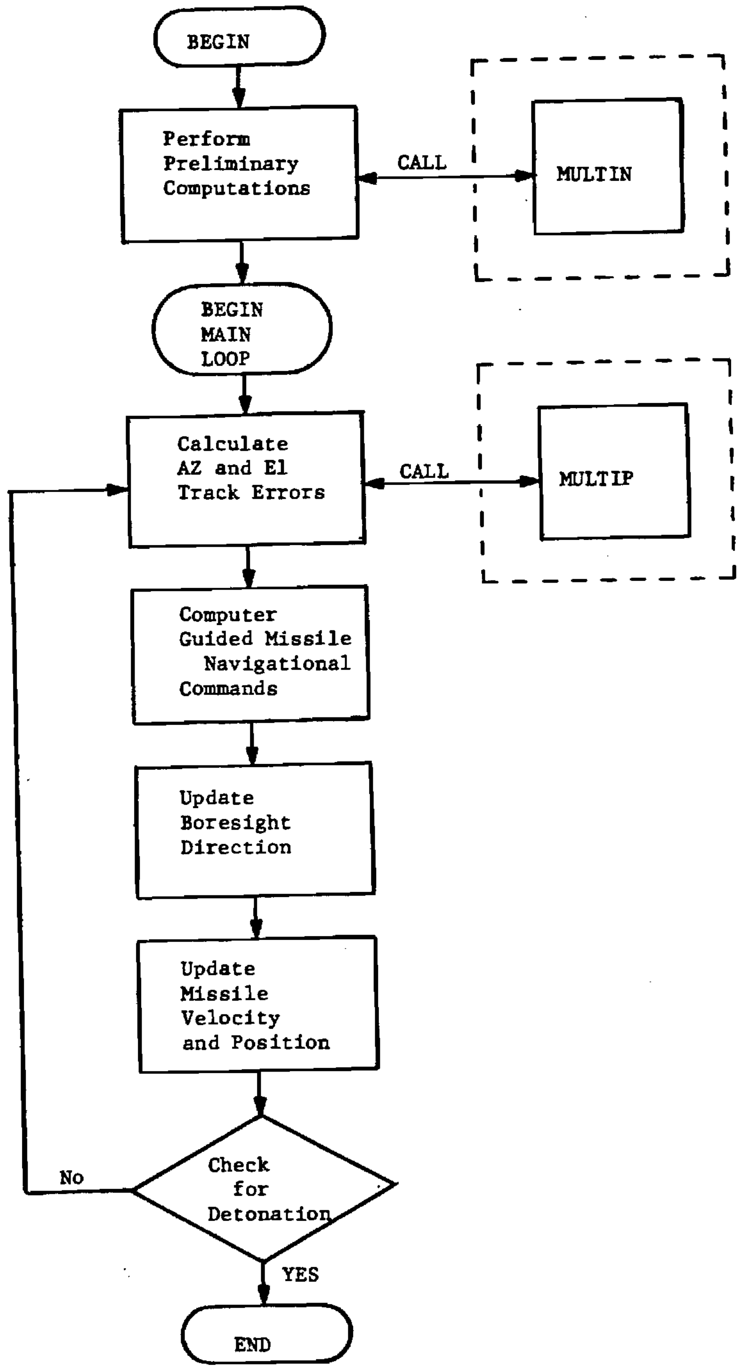


Figure 1. Simplified flow chart of a typical Missile model illustrating the placement of calls to the Track Error/RCS subroutines.

the geometry file, and the scratchpad data file, respectively. For details on the formats of these data files, see Section 3.

The subroutine MULTIN passes only three real arguments and will appear as shown below:

```
CALL MULTIN (XTARG, YTARG, ZTARG)
```

The arguments XTARG, YTARG, and ZTARG are the X, Y, and Z coordinates in feet of the target's position in a fixed earth reference frame. Since the target actually covers an extended volume, this XYZ location is the position of the origin of the target's internal coordinate system. This information is returned from MULTIN where it is read from the geometry data file and then passed to the main missile program. These coordinates will then remain unchanged throughout the remainder of the simulation since the target is assumed to be stationary. It is especially important to remember that the arguments for MULTIN must be variables and NOT constants since the data are to be returned from MULTIN to the missile program and not the other way around.

Another purpose for the MULTIN subroutine is to set the radar parameters for the missile seeker system. The parameters used by the Radar Glint program are the frequency, the pulse width, and the antenna beamwidths. These parameters are set by inserting several assignment statements at the beginning of the MULTIN subroutine. The variable FREQ must be set to the frequency in Gigahertz. The variable PW is the pulse width in nanoseconds. The variables BWT1 and BWT2 are the transmitted 3 dB elevation and azimuth beamwidths, respectively. The variables BWR1 and BWR2 are the received elevation and azimuth beamwidths, respectively.

The other subroutine to be called by the missile model is MULTIP (for MULTIPath, track error, and RCS). Since its purpose is to return the azimuth and elevation track errors which need to be supplied to the missile guidance algorithm, it should be placed inside the main loop of the program where it can be called once for every update of the missile's position and velocity vectors as well as the radar's boresight direction. The call to MULTIP has 10 real arguments and will appear as follows:

```
CALL MULTIP (XS,YS,ZS,XT,YT,ZT,AZS,ELS,AZERR,ELERR)
```

The first eight of the arguments are the input variables to MULTIP. The variables XS, YS, and ZS are the X, Y, and Z coordinates in feet of the seeker position in the fixed earth frame. XT, YT, and ZT are the coordinates of the target's origin, also in feet. The variables AZS and ELS are the azimuthal (yaw) and elevation (pitch) directions of the missile's radar boresight. They are both measured in radians. The range of values for the azimuth is from 0 to 2π, measured clockwise from the Y-axis in the fixed earth frame. The positive sense for the elevation is above the horizontal plane.

The two output variables are AZERR and ELERR which are the azimuthal and elevation track errors. These two values indicate the direction off boresight at which the phase center of the target appears. Thus, these are the values that need to be input to the missile guidance algorithm so that the missile may make the proper adjustments in tracking the target. Both of the track error angles are measured in radians, with respect to the boresight. The positive sense of the azimuthal error is to the right of the boresight, and for the elevation error, the positive sense is up.

SECTION 3. INPUT DATA FILE FORMATS

3.1 INTRODUCTION

The Georgia Tech Radar Glint program utilizes three data files in its operation. Two of these are input data files for describing the orientation and position of the target as well as its shape. The third data file is an internal unformatted file used solely as 'scratch pad' memory by the program due to memory limitations of most minicomputers. The primary purpose of this section is to describe the format of the two input files to aid the user in both the maintenance and understanding of existing data files as well as the creation of new data files for the running of different scenarios.

3.2 GEOMETRY DATA FILE

The geometry data file is a relatively simple data file that describes the orientation and position of the missile's target. It is an ASCII data file that can be easily updated or created by most text editors. The geometry file is opened, read, and closed by the module MULTIN which uses device number 2 as its input channel. This number could be changed, of course, if it conflicts with any files opened by the main missile program.

The format of the geometry data file is as follows:

```
Line 1 :  ROUGH,DEC,SIGODB
     2 :  XTARG,YTARG,ZTARG
     3 :  IORDER
     4 :  ALPHA,BETA,GAMMA
```

Since this data file is read in using a free field format, no special columnar formatting is needed. Commas, spaces, and/or carriage returns may be used as variable delimiters.

The variables on line 1 of the geometry data file refer to the electrical characteristics of the ground plane. The Radar Glint model assumes a flat, homogeneous terrain upon which the target sits. Multipath reflection of the radar's signal as well as background clutter can occur from the ground plane, hence the necessary parameters to describe these phenomena are found in line

1. The real variable ROUGH is the root-mean-square surface roughness in feet of the terrain. DEC is a complex variable that contains the complex dielectric constant of the terrain. The last variable, SIGODB, is a clutter parameter for the terrain. This real variable represents the average cross section per unit area of the clutter (σ^0), measured in decibels (dB). Some typical values for the complex dielectric constant for various terrain types are listed in Table 1. Figures 2 and 3 are supplied to give some indication of possible values for σ^0. These graphs present some land clutter data taken by Georgia Tech in 1975.[3]

Line 2 of the geometry data file contains the X, Y, and Z coordinates of the target's position as expressed by the variables XTARG, YTARG, and ZTARG. The Radar Glint model assumes a fixed earth, right handed Cartesian coordinate system in which the Z axis points towards the zenith and Z = 0 is on the earth's surface. The "target's position" is defined as being the origin of its own internal coordinate system, i.e., the point on the target from which all the internal target's coordinates are measured as described by the target data file (see Section 3.3).

The third line in this input data file is the integer array IORDER which is dimensioned to three. This array describes the order of rotations which may be performed on the target to obtain the desired orientation. By default, the target is oriented in the directions shown in Figure 4. That is, the front of the target points in the Y direction and the right side of the target points in the X direction. The vertical direction of the target is parallel to the Z axis. Successive yaws, pitches, and rolls may be performed to reorient the target into a new position. The definition of yaw, pitch, and roll in the context of the target's orientation are defined in Figure 4. Yaw may be thought of as rotation about the Z axis, pitch as a rotation about the X axis, and roll as a rotation about the Y axis. The positive sense for all three rotations is counterclockwise.

The purpose of the IORDER array is to define the order of these rotations since doing a yaw before a pitch has a different outcome than performing a pitch before a yaw. The rotations of yaw, pitch, and roll are designated by the integers 1, 2, and 3, respectively. The first angle to be rotated is IORDER(1), the second is IORDER(2), and the third is IORDER(3). An example of IORDER is 2,1,3 which means do pitch first, yaw next, and roll last. Another

TABLE 1

DIELECTRIC CONSTANTS OF TERRAINS
FOR COMPUTATION OF REFLECTION COEFFICIENTS

Terrain Type		ε_1	ε_2
1. Bare Soil	dry wet	2.44 20.0	.00267 2.4
2. Grass	dry wet	2.0 20.0	0 0
3. Sand	dry wet	2.55 20.0	.016 .26

where ε_r is the complex dielectric constant

and $\varepsilon_r = \varepsilon_1 - j\,\varepsilon_2$

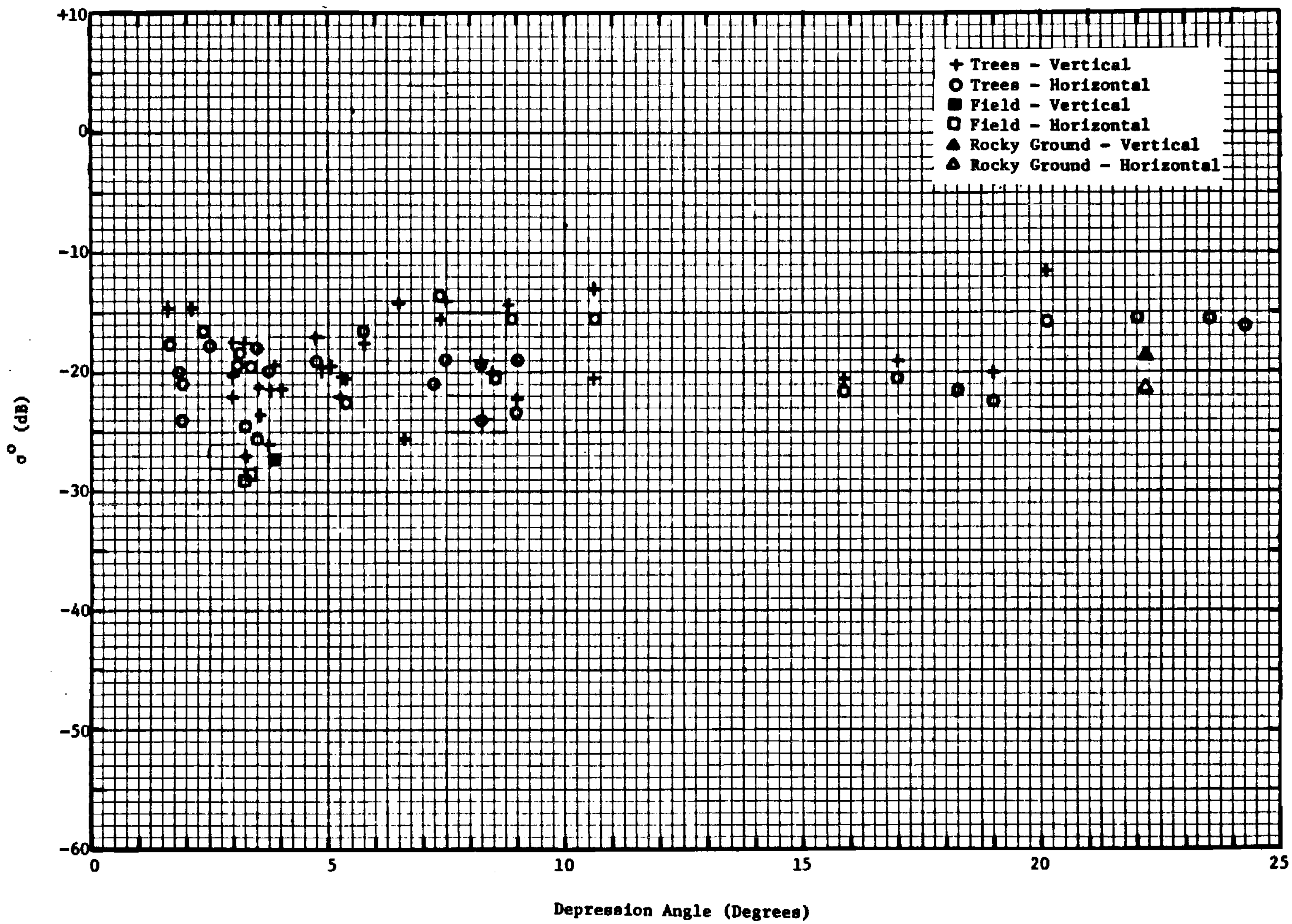


Figure 2. Comparison of the average backscatter per unit area for vertical and horizontal polarizations; 35 GHz.

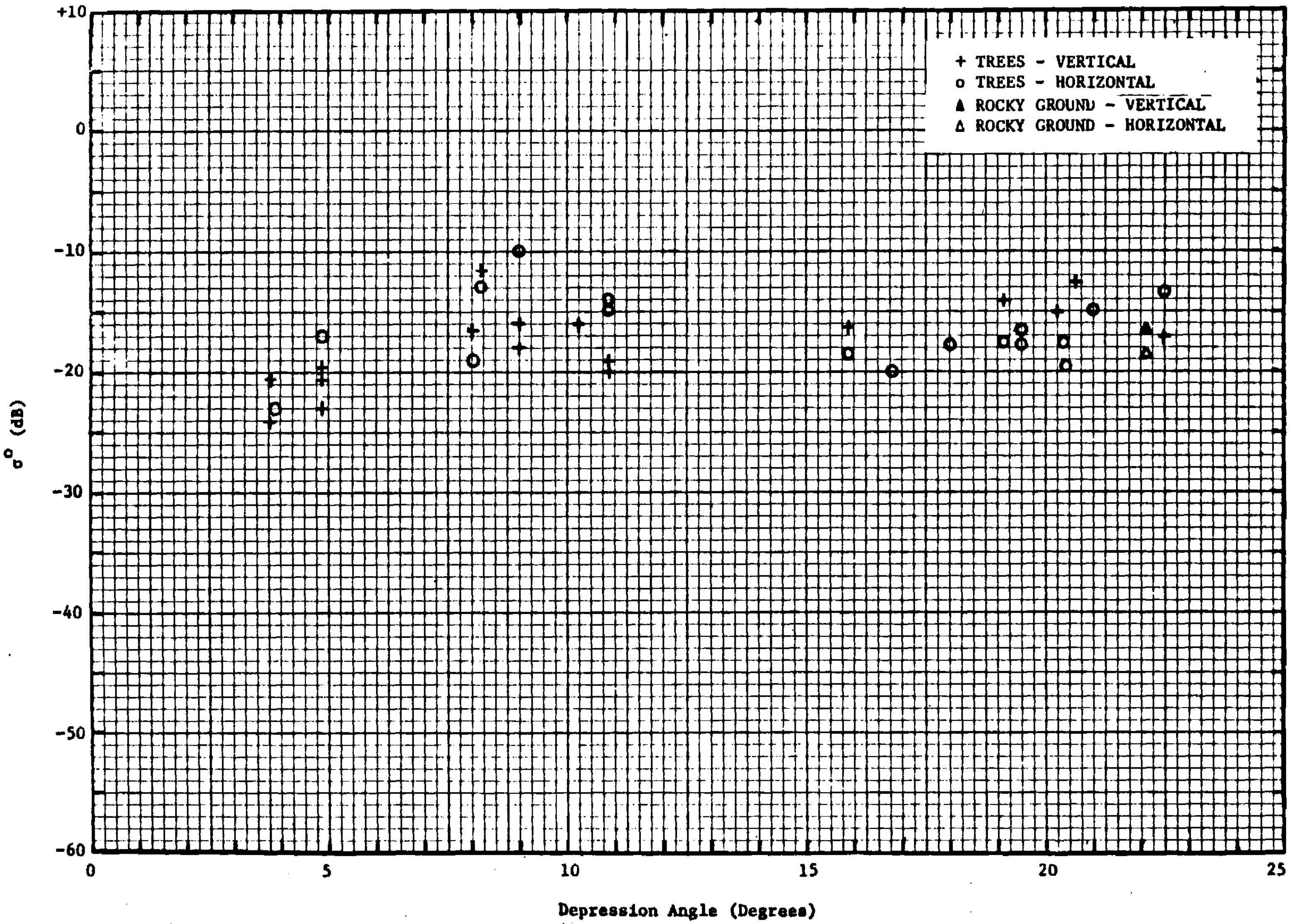


Figure 3. Comparison of the average backscatter per unit area for vertical and horizontal polarizations; 95 GHz.

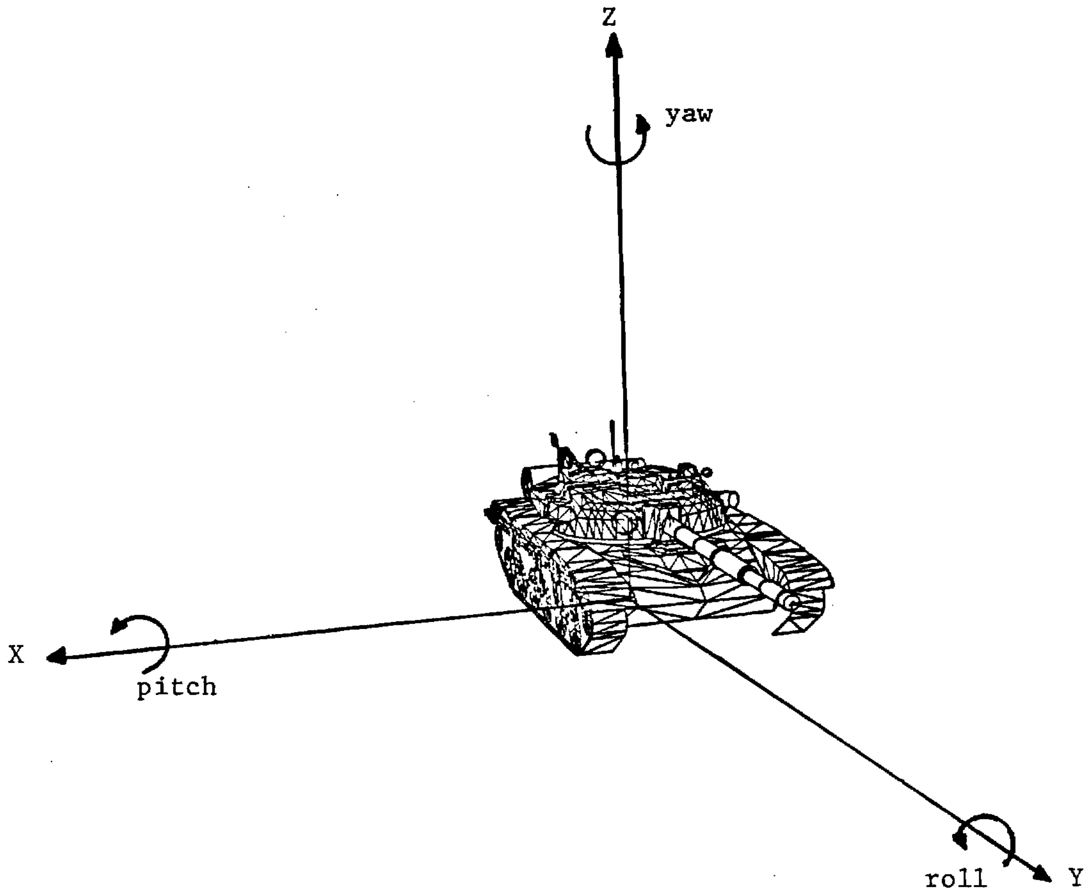


Figure 4. Target coordinate axes with rotation angles.

example would be 3,1,2 which means do roll first, then yaw, and finally pitch. The integers 1,2, and 3 must each be used once and only once in the array IORDER. Thus IORDER = 2,2,1 is obviously unacceptable.

The last line of the geometry input file contains the rotation angles themselves for the amount of yaw, pitch, and roll to be performed. Here, ALPHA is the amount of the first rotation specified by IORDER, measured in degrees. BETA corresponds to the second rotation, and GAMMA is the third rotation. Thus if IORDER = 3,2,1, then ALPHA, BETA, and GAMMA would correspond to the amounts of roll, pitch, and yaw, respectively. Remember that a positive value for any of these angles means a counterclockwise rotation.

3.2 TARGET DATA FILE

The target data file describes the missile's target by representing it as a discrete set of radar scattering elements. A typical target may contain as many as several thousand of these scattering elements which will henceforth be referred to simply as scatterers. The Radar Glint model presently handles five types of scatterers. They are the triangular flat plate, the truncated cone frustum, a dihedral corner, a trihedral corner, and an ellipsoid.

The triangular flat plate is the most prevalent scattering type in a target model. Because almost any surface can be defined as a mesh of triangular facets. In the Radar Glint model, the triangular flat plate is described by the X, Y, and Z coordinates of the three vertices. Since these flat plates are considered to be single sided, the direction of the outward pointing normal to the surface is defined by a counterclockwise ordering (right-handed) of the vertices as illustrated in Figure 5. The right circular truncated cone frustum scattering type covers many singly curved surfaces. It is defined by the X, Y, and Z coordinates of the centers of the two end faces as shown in Figure 6. The radii of both the faces are also necessary to completely define the cone frustum. Note that the first end point is always the one with the larger radius. Two special cases of this scattering type are the cylinder (both end radii are equal) and the cone (the second end radius equals zero).

The other two scattering types are the multifaceted dihedral and trihedral which are illustrated in Figure 7. The purpose for defining these

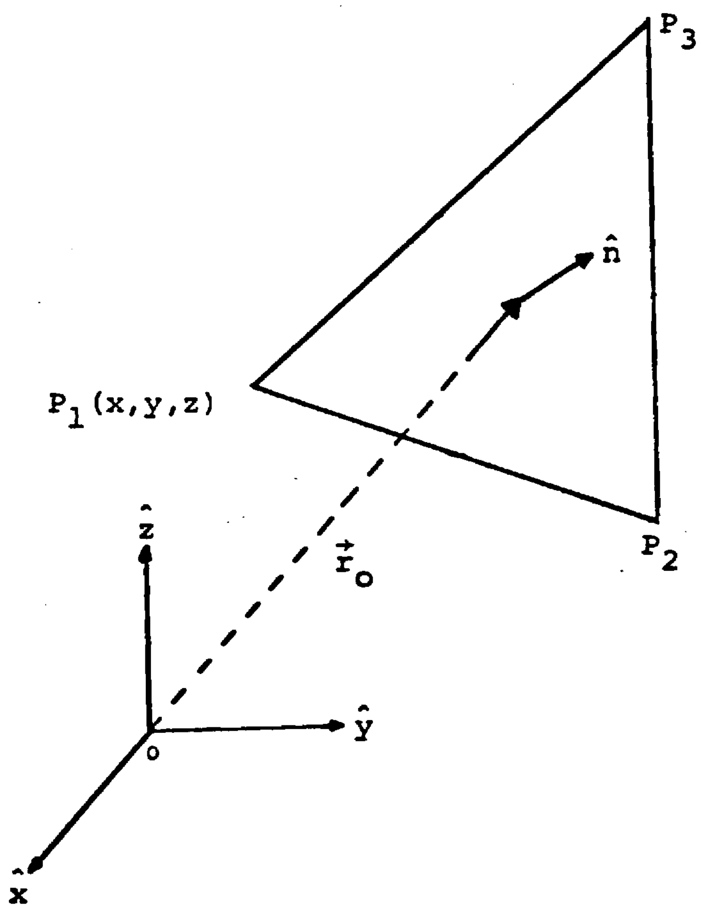


Figure 5. Triangular flat plate geometry.

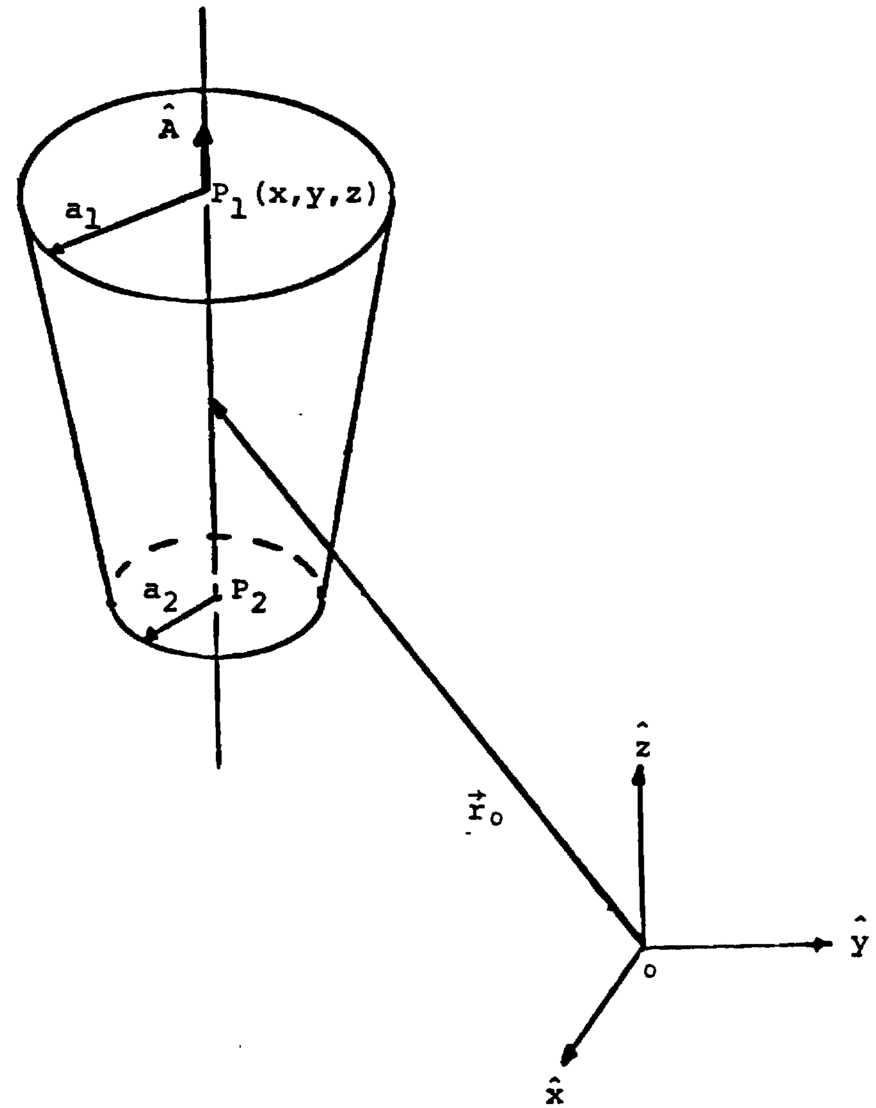
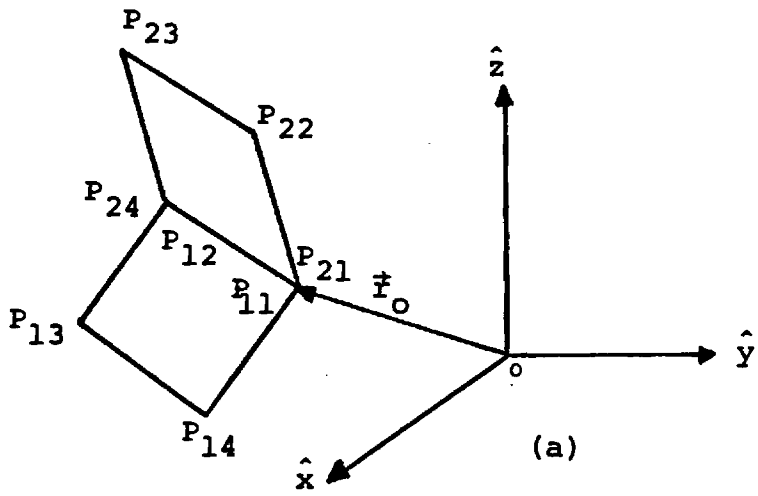
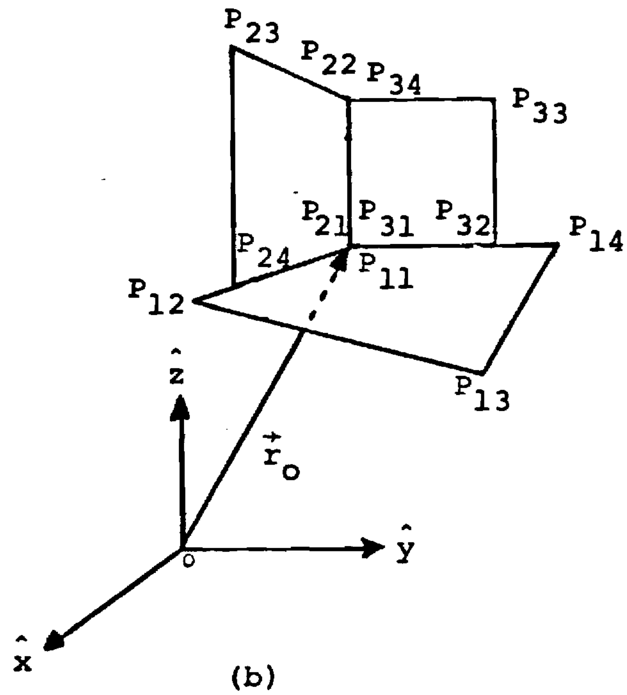


Figure 6. Truncated cone frustum geometry.



(a)



(b)

Figure 7. (a) Dihedral and (b) trihedral geometry.

scatterers as a separate type is to take into account the multiple bounce effects that enhance the RCS wherever right-angled corners occur. These corners could not be described by individual triangles since the Track Error RCS model has no provision for multiple scatterer interactions. The trihedral is made up of three quadrilaterals whose points are ordered counterclockwise to orient the outward pointing normals. The central vertex of the trihedral is defined as point 1 for each of the three quadrilateral faces. The dihedral is defined similarly, except that there are only two quadrilaterals. In this case, point 1 for each face is defined to be the leftmost common vertex.

Due to the immense size of the target models, a special editor has been written to create or modify a target data file. This editor is known as SAMURAI, for Simulation And Modeling Used in Radar Analytical Investigations. It was designed at Georgia Tech and was originally intended for the building of ship data files, but has since been used for other targets such as airplanes and tanks as well. The SAMURAI editor is a very powerful editor in that it allows a user to both edit and/or graphically display the contents of the data base for the three dimensional model. The editor's command structure is line oriented with a complete HELP facility. All commands may be entered directly from the keyboard or from a separate command file. Commands include translation, reflection, rotation, scaling, deletion, and merging as well as many graphical display commands. Along with the geometric information, the editor tags each record with a descriptive identifier which may be referenced by many of the commands to allow operation on a single element or a selected subset of elements.

SAMURAI supports a number of graphic output devices as well as a variety of display formats. Displays may be simple line drawings with no hidden line removal, line drawings with "poor man's" hidden surface removal, or a shaded image with complete hidden surface removal. Shaded drawings may also be used to display radar information by allowing the intensity of color of each displayed scatterer to represent the relative radar return due to that scatterer. SAMURAI currently supports four color raster displays, two incremental plotters and a dot matrix printer/plotter; the addition of more devices is a relatively simple task. Figure 8 illustrates a typical target data file created using SAMURAI and drawn by SAMURAI.

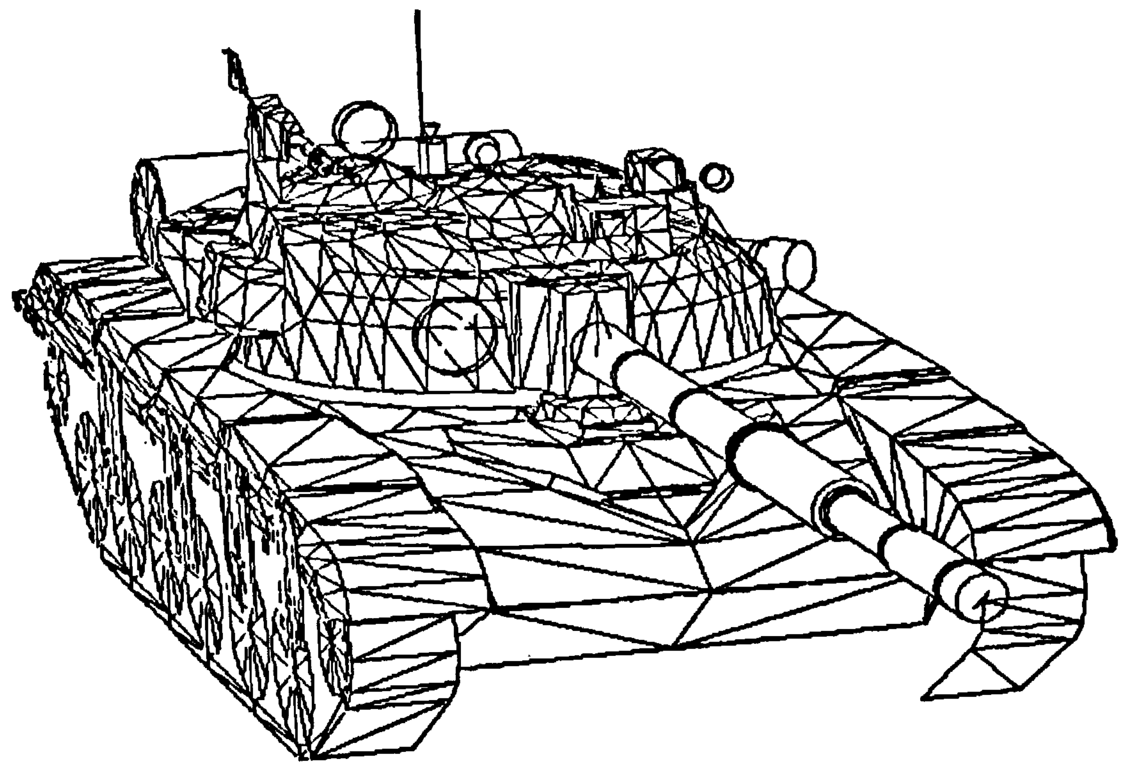


Figure 8. Typical target drawn by SAMURAI.

The format of the target data files created by SAMURAI and used by the Radar Glint program is a direct access disk file with fixed length records (128 bytes per record). The first 66 records of each file form a header block and are reserved for information which identifies the target being modeled and the date and time of creation for the file. The remaining records contain the information about the individual scatterers and are arranged as a doubly linked list.

Of the 66 records at the beginning of the file, only the first five are currently used. The rest are reserved for future enhancements. The first record contains the name of the target, while the second record has the name of the original author of the data file. The next three records are used to record the date and time of creation of the file. The most important record in the header block is record 66. The first two bytes of this record contain the pointer to the first scatterer in the data file. This pointer is stored in an INTEGER*2 format and has an offset of 66. That is, if this pointer contained the value 1, it would actually reference the 67th record in the data file. The programmer would have to add the offset of 66 to the pointer value to actually reference the specified data record. The other important information stored in the 66th record is the total number of scatterers contained in this target data file. This value is also stored as an INTEGER*2 value and is found in bytes 9 and 10 of the 66th record.

The rest of the file is made up of the data records that describe the scatterers which form the target model. In general, one record is used to describe a single scatterer. In some special cases, an additional record is needed to supply more coordinate information for a complicated scatterer (dihedral or trihedral). The standard format for a data record is as follows:

```
+------+------------+------+------+------+------+-------+----+-------+
! WHO  !  IDSTRING  ! TIME ! TYPE ! FPTR ! BPTR ! SPECL ! WT ! COORD !
+------+------------+------+------+------+------+-------+----+-------+
0      4            64     68     70     72     74      76   80      127
```

FIELD DESCRIPTIONS

WHO 4 bytes, the first three bytes are the initials of the last person to edit this particular record.

IDSTRING 60 bytes which are used to store an identification code unique to the particular scatterer. This code may be used in a large number of selective search operations. Each ID consists of a number of text fields separated by periods. A possible ID could be:

TURRET.GUN.BARREL.000010

This would identify this scatterer as a component of the main gun which is part of the entire turret structure.

TIME 4 bytes which contain the date and time of the last update made to this scatterer. The time is given as the number of minutes since 1-JAN-80 00:00.

TYPE 2 bytes which contain an integer that identifies the type of scatterer. The current types are:

1 - A truncated cone frustum.
2 - A triangular flat plate.
3 - An ellipsoid.
4 - A trihedral.
5 - A dihedral.

FPTR 2 bytes which contain a pointer to the next scatterer in the file. A value of 0 indicates that this is the last scatterer in the file. All pointers have the intrinsic offset of 66 as described above.

BPTR 2 bytes which contain a pointer to the previous scatterer in the file. A value of 0 for this pointer indicates that this is the first scatterer in the file.

SPECL 2 bytes which contain a pointer to an auxiliary record which is used to hold additional coordinate data for trihedral and dihedral reflector types. A value of 0 indicates that there are no other records associated with this scatterer.

WT 4 bytes which contain a real floating point value used as a weighting factor for this particular scatterer. This weighting factor is usually the reflection coefficient of the material that this particular scatterer is made of. It can be used to simulate dielectrics or Radar Absorbent Material (RAM). For metal scatterers, the value is usually 1.

COORD 48 bytes which contain 12 real values treated as an array of coordinates which define the actual scatterer. If there is an auxiliary record specified, it is treated as an extension of this field. These values are best thought of as an array dimensioned as COORD (3, 4, 3). The first index selects the X, Y, or Z value of the vertex which is specified by the second index. The third index identifies which plate of a trihedral or dihedral is to be used. For frusta, the first two XYZ points are the centers of the faces of the frustum, and the next two reals are their respective radii.

SECTION 4. REFERENCES

1. S. P. Zehner and M. T. Tuley, Eds. "Development and Validation of Multipath and Clutter Models for TAC ZINGER In Low Altitude Scenarios," Final Report on Contract No. F49620-78-C-0121, Georgia Institute of Technology, Engineering Experiment Station, March 1978, UNCLASSIFIED.

2. M. M. Horst et al., "Ship RCS Predictions (U)," Final Report on Contract N00167-82-M-0737, Engineering Experiment Station, Georgia Institute of Technology, January 1982, SECRET.

3. N. C. Currie et al., "Radar Land Clutter Measurements at Frequencies of 9.5, 16, 35, and 95 GHz," Technical Report No. 3 on Contract DAAA 25-73-C-0256, Engineering Experiment Station, Georgia Institute of Technology, 2 April 1975, UNCLASSIFIED.

APPENDIX

RADAR GLINT MODEL

PROGRAM LISTING

```
---------------------------------------

         C     TRERR3.FLX     -    TRACK ERROR CALCULATIONS.
         C
         C      MODIFIED BY R. B. RAKES ON 2/23/82

         C
    1           PROGRAM DRIVER

         C      This stand-alone program is a front end driver for the GA. TECH.
         C      Track Error/RCS model.  It is used only as a debugging tool and
         C      will not be included in the implementation of these routines into
         C      an existing missile seeker program.
         C

    2           PARAMETER  PI = 3.141593

    3           COMMON /RCS/ TOTAL_RCS  !  Used only in this driver & MULTIP


    4           OPEN ( UNIT = 0, NAME = 'OUT$FILE', TYPE = 'NEW' )

    5           CALL MULTIN (XT,YT,ZT) ! Initialize multipath and clutter.

    6           XTARG = XT
    7           YTARG = YT
    8           ZTARG = ZT

    9           TYPE *,'Slant range (ft), elevation (deg) : '
   10           ACCEPT *,RANGE,ELDEG
   11           TYPE *,'Azimuth : Inital, final, & increment (deg) = '
   12           ACCEPT *,AZI,AZF,AZINC

   13           WRITE ( 0, * ) AZI,AZF,AZINC

   14           EL = ELDEG * PI / 180.
   15           AZDEG = AZI

   16           REPEAT UNTIL (AZDEG .GT. AZF)
   18           .  AZ = AZDEG * PI / 180.
   19           .  AZ_BORE =  AZ - PI
   20           .  UNTIL (AZ_BORE .GE. 0.) AZ_BORE = AZ_BORE + 2.*PI
   22           .  EL_BORE = -EL
   24           .  XS = RANGE*COS(EL)*SIN(AZ)
   25           .  YS = RANGE*COS(EL)*COS(AZ)
   26           .  ZS = RANGE*SIN(EL)
                .
   27           .  XT = XTARG
   28           .  YT = YTARG
   29           .  ZT = ZTARG
                .
   30           .  CALL MULTIP(XS,YS,ZS,XT,YT,ZT,AZ_BORE,EL_BORE,
              1.  AZER,ELER)  !COMPUTE MUTIPATH,CLUTTER,TRACK ERROR.
                .
   31           .  AZERDEG = AZER * 180./PI
```

```
32       .  ELERDEG = ELER * 180./PI
         .
33       .  WRITE ( 0, * ) AZERDEG,ELERDEG,TOTAL_RCS
         .
34       .  AZDEG = AZDEG + AZINC
         ...FIN
35       END

     (FLECS 77  VERSION 22.38)

MODULE CONTAINS  NO MINOR ERRORS
MODULE CONTAINS  NO MAJOR ERRORS
```

```
-----------------------------------------

1        SUBROUTINE MULTIN(XT,YT,ZT)
      C
      C#ABSTRACT INITIALIZE MULTIPATH AND CLUTTER CONSTANTS
      C
      C#PURPOSE  INITIALIZE FREQUENCY, BEAMWIDTH AND PULSE  WIDTH DEPENDING ON
      C          MISSILE MODEL USED. COMPUTE GROUND CORRELATION CONSTANTS.
      C
      C REFERENCES 'DEVELOPMENT AND VALIDATION OF MULTIPATH AND CLUTTER MODELS
      C            FOR TAC ZINGER IN LOW ALTITUDE SCENARIOS', CONTRACT NO
      C            F49620-78-C-0121, HQ USAF/SAGF, WASHINGTON, D.C. 20330.
      C            MARCH 1979

      C     INPUT VARIABLES :

      C                        none

      C     OUTPUT VARIABLES :

      C        XT, YT, ZT    -      The x, y, and z coordinates of the target'
      C                             origin expressed in a fixed earth, right-
      C                             handed Cartesian coordinate system.  These
      C                             coordinates are read in from the geometry
      C                             data file and are in units of feet.

      C  NOTE:    All variables dimensioned to three are assumed to be 3
      C           dimensional vectors.  If a vector's variable name includes
      C           the letters 'HAT', then it is a unit vector.  If the vector
      C           ends with the letter 'P', then it is measured in the target's
      C           (primed) coordinate system.

      C*COMMONS
2        COMMON/COUNTERS/ COUNTER
3        COMMON/FACET/ OLDNFACET
4        COMMON/CTARA/ ROUGH
5        COMMON/MULCLT/DEC,ZERO,DC,NUMBER_OF_SCATTERERS
        *,PI,CO,CO1,C3,C31,XK,BWT1,BWT2,ZHATP
        *,BWR1,BWR2,P1,P2,IFCL,RSIGC,HEIGHT_INCREMENT
        *,WL,FREQ,R,HAV,BWAZ,FLENGTH,EYE,ZHAT,VERTICAL,Q,QI
        *,SIGO
6        INTEGER*2 POINTER,TRIPOINTER,ITYPE,IFIRST,IFIRSTTEMP,NSCATTEMP
7        BYTE BUFF(256),POINTERTEMP(2),TRIPTEMP(2)
8        INTEGER PRESENT_RECORD,COUNTER

9        COMMON /BUFF/  IFIRST

10       EQUIVALENCE (BUFF(69),ITYPE),(BUFF(71),POINTER),
        &(BUFF(75),TRIPOINTER),(BUFF(77),WT),(BUFF(81),VF(1,1,1)),
        &(BUFF(1),IFIRSTTEMP),(BUFF(9),NSCATTEMP)
11       EQUIVALENCE (POINTERTEMP(1),POINTER),(TRIPTEMP(1),TRIPOINTER)

12       LOGICAL VERTICAL
13       DIMENSION ZHAT(3),IORDER(3),Q(3,3),QI(3,3),S(3,3),SI(3,3)
14       REAL MATA(3,3),MATB(3,3),MATC(3,3),MATAI(3,3),ZHATP(3),
```

```
                &MATBI(3,3),MATCI(3,3),VF(3,4,3)

 15             COMPLEX DEC,ZERO,EYE
        C*

 16             DATA ZHAT/0.,0.,1./,VERTICAL/.FALSE./,EYE/(0.,1.)/


        C        Initialize matrices to identity matrices.
 17             DATA MATA/1.,3*0.,1.,3*0.,1./
 18             DATA MATAI/1.,3*0.,1.,3*0.,1./
 19             DATA MATB/1.,3*0.,1.,3*0.,1./
 20             DATA MATBI/1.,3*0.,1.,3*0.,1./
 21             DATA MATC/1.,3*0.,1.,3*0.,1./
 22             DATA MATCI/1.,3*0.,1.,3*0.,1./

        C
        C
 23             ZERO=CMPLX(0.,0.)
 24             PI = 3.141593
 25             OLDNFACET = 100


 26             OPEN-TARGET-FILE
        C        OPEN(UNIT=3,NAME=.'TARG$FILE'   ,TYPE='OLD')
        C        OPEN(UNIT=3,FILE=TARGFILE,STATUS='OLD',BLOCKED=.TRUE.) !SEL OPEN

 28             OPEN(UNIT=2,NAME='GEOM$FILE',TYPE='OLD')
 29             OPEN(UNIT=5,NAME='RTS.BIN',FORM='UNFORMATTED') ! Scratchpad data file.
 30             READ-IN-GEOMETRY-DATA

 32             FLENGTH = 500./.3048       ! Length of a ground facet (or increment
        C                                !i.e., 10 facets for an orignal range of 5 km
 33             HEIGHT_INCREMENT = 12.       ! 12. ft. intervals in ht. for rho cal

        C  INITIAL VALUE FOR CLUTTER CROSS SECTION FOR SIGC
 34             RSIGC=.0000001

        C
        C  INITIALIZE FREQUENCY, BEAMWIDTH, AND PULSEWIDTH
        C
        C  DEFAULT
 35             FREQ=95.0                          ! GHz
 36             BWR1=1.0         ! Receiver beamwidth - Elevation (deg)
 37             BWT1=1.0         ! Transmitter  "          "        "
 38             BWR2=1.0         ! Receiver     "       Azimuth     "
 39             BWT2=1.0         ! Transmitter  "          "        "
 40             PW=600.          ! Pulsewidth ( nanoseconds )

        C
 41             WL=.984252/FREQ             !  Wavelength in feet.
 42             XK=2.*PI/WL                 !  Wave number ( rad/ft )
 43             CO=SQRT(2.)*XK
 44             CO1=4.*SQRT(PI)

 45             SET-UP-ROTATION-MATRIX-FOR-YAW-PITCH-AND-ROLL
```

```
     C  BEAMWIDTH IN ALPHA(1)  (DEGREES TO RADIANS)
47         BWR1=BWR1*PI/180.
48         BWR1=.7992*BWR1
49         BWT1=BWT1*PI/180.
50         BWT1=.7992*BWT1
     C
     C  BEAMWIDTH IN BETA(2) (DEGREES TO RADIANS)
51         BWR2=BWR2*PI/180.
52         BWR2=.7992*BWR2
53         BWT2=BWT2*PI/180.
54         BWAZ=BWT2        ! This unmodified az beamwidth is used with clutter
55         BWT2=.7992*BWT2
56         SBW1=BWR1*.01
57         SBW2=BWR2*.01
58         CALL PATT(D1,D2,SS,SBW1,SBW2,BWR1,BWR2)
59         P1=SBW1*SS/D1
60         P2=SBW2*SS/D2
     C

     C PW IS PULSEWIDTH IN NANOSECONDS
61         C3=PW*.5   !  C3 IS C*TAU/2
62         C31=C3/SQRT(2 )
     C
     C  GROUND CORRELATION
63         DC=30./.3048        ! in feet.
     C
64         RETURN

     ---------------------------------------

65         TO OPEN-TARGET-FILE
66           .  OPEN(UNIT=3,NAME='TARG$FILE',TYPE='OLD',ACCESS='DIRECT',
            &.  FORM='UNFORMATTED',RECORDSIZE=32,RECORDTYPE='FIXED',
            &.  ASSOCIATEVARIABLE=ICRUD)
67           .  READ(3'66) (BUFF(I),I=1,128)
68           .  IFIRST=IFIRSTTEMP
69           .  NUMBER_OF_SCATTERERS=NSCATTEMP
70           .  POINTER=IFIRST
             .
     C       .  The following section of commented code may be used if the compute
     C       .  program is being run on has enough memory (or virtual memory) to
     C       .  read the entire target data file into main memory.
             .
     C       .   DO(II=1,NUMBER_OF_SCATTERERS)
     C       .   IRECORD=POINTER+66
     C       .   READ(3'IRECORD) (IBUFF(I,II),I=1,128)
     C       .   DO(I=1,2)
     C       .   POINTERTEMP(I)=IBUFF(70+I,II)
     C       .   TRIPTEMP(I)=IBUFF(74+I,II)
     C       .   FIN
     C       .   IF(TRIPOINTER.NE.0)
     C       .   IREC=TRIPOINTER+66         ! Read in remaining dihedral or trihedral
     C       .   READ(3'IREC) (IBUFF(I,II),I=129,256)        ! coordinates.
```

```
C0223        C    .   FIN
00224        C    .   FIN
00225        C    .   CLOSE (UNIT=3)
00226             .
00227             ...FIN

         ----------------------------------------

00228   71        TO READ-IN-GEOMETRY-DATA
00229   73        .  READ(2,*) ROUGH,DEC,SIGODB   ! RMS surface roughness (ft)
00230   74        .  READ(2,*) XT,YT,ZT   ! Initial target coordinates (ft.)
00231   75        .  READ(2,*) IORDER
00232   76        .  READ(2,*) ALPHA,BETA,GAMMA         !  Yaw,pitch, and roll ang
00233   77        .  CLOSE (UNIT=2)
00234   78        .  IF (ROUGH.LE.0.) ROUGH=.0001       ! To prevent division by 0
00235   79        .  SIGO = 10.**(SIGODB/10.)   ! 'Un-dB' sigma O (clutter)
00236             .
00237   80        .  ALPHA = ALPHA*PI/180.       ! \
00238   81        .  BETA  = BETA*PI/180.        !  >  Convert degrees to radians.
00239   82        .  GAMMA = GAMMA*PI/180.       ! /
00240             ...FIN

         ----------------------------------------

00241   83        TO SET-UP-ROTATION-MATRIX-FOR-YAW-PITCH-AND-ROLL
00242   85        .  CALL YPR(ALPHA,MATA,MATAI,IORDER(1))
00243   86        .  CALL YPR(BETA ,MATB,MATBI,IORDER(2))
00244   87        .  CALL YPR(GAMMA,MATC,MATCI,IORDER(3))
00245             .
00246   88        .  CALL MATMU(MATB,MATA,S,3,3,3 )    !  Matrix B times matrix A = matr
00247   89        .  CALL MATMU(MATAI,MATBI,SI,3,3,3)
00248   90        .  CALL MATMU(MATC,S,Q,3,3,3) ! Q ROTATES A VECTOR TO TARGET'S FRAME.
00249   91        .  CALL MATMU(SI,MATCI,QI,3,3,3) ! QI IS INVERSE (TRANSPOSE) OF Q.
00250             .
00251   92        .  CALL MATMU(Q,ZHAT,ZHATP,3,3,1)
00252             ...FIN
00253   93
00254   94        END

         ----------------------------------------

                  PROCEDURE CROSS-REFERENCE TABLE

               00200 OPEN-TARGET-FILE
                     00137

               00228 READ-IN-GEOMETRY-DATA
                     00143

               00241 SET-UP-ROTATION-MATRIX-FOR-YAW-PITCH-AND-ROLL
                     00169

         (FLECS 77  VERSION 22.38)

     MODULE CONTAINS  NO MINOR ERRORS
     MODULE CONTAINS  NO MAJOR ERRORS
```

```
 ------------------------------
 1          SUBROUTINE MULTIP(XS,YS,ZS,XT,YT,ZT
           1,AZS,ELS,AZERR,ELERR)
     C
     C
     C REFERENCES 'DEVELOPMENT AND VALIDATION OF MULTIPATH AND CLUTTER MODELS
     C              FOR TAC ZINGER IN LOW ALTITUDE SCENARIOS', CONTRACT NO
     C              F49620-78-C-0121, HQ USAF/SAGF, WASHINGTON, D.C. 20330.
     C              MARCH 1979

     C      INPUT VARIABLES :

     C        . XS, YS, ZS      -  The x, y, and z coordinates of the seeker in
     C                             a right-handed, fixed earth Cartesian referen
     C                             frame.  The z-direction points towards the
     C                             zenith (up).  Zero z is always on the ground.
     C                             (feet)

     C          XT, YT, ZT      -  The x, y, and z coordinates of the target's
     C                             intrinsic origin specified in the same frame
     C                             as XS, YS, and ZS.   (feet)

     C          AZS, ELS        -  The azimuth and elevation directions of the
     C                             boresight in radians.  AZS is measured
     C                             clockwise from the Y-axis and ELS is the angl
     C                             of elevation above the horizontal, also in
     C                             radians.

     C  OUTPUT VARIABLES :

     C          AZERR, ELERR    -  The azimuth and elevation track errors
     C                             measured with respect to the boresight.
     C                             AZERR is considered positive to the right,
     C                             and ELERR is positive up.  Both are in radian
     C
     C   NOTE :  An exp(jwt) phase convention is assumed throughout this
     C           model except in the CROSS portions.  Thus, the need for
     C           the complex conjugating of the output of CROSS (or RCS).
     C

 2          PARAMETER TWOPI = 6.28318406


            INCLUDE TRERR3.FLX (COMMONS)
     &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
     C*COMMONS
 3          COMMON/COUNTERS/ COUNTER
 4          COMMON/FACET/ OLDNFACET
 5          COMMON/CTARA/ ROUGH
 6          COMMON/MULCLT/DEC,ZERO,DC,NUMBER_OF_SCATTERERS
           $,PI,CO,CO1,C3,C31,XK,BWT1,BWT2,ZHATP
           $,BWR1,BWR2,P1,P2,IFCL,RSIGC,HEIGHT_INCREMENT
           $,WL,FREQ,R,HAV,BWAZ,FLENGTH,EYE,ZHAT,VERTICAL,Q,QI
           $,SIGO
```

```
7         INTEGER*2 POINTER,TRIPOINTER,ITYPE,IFIRST,IFIRSTTEMP,NSCATTEMP
8         BYTE BUFF(256),POINTERTEMP(2),TRIPTEMP(2)
9         INTEGER PRESENT_RECORD,COUNTER

10        COMMON /BUFF/  IFIRST

11        EQUIVALENCE (BUFF(69),ITYPE),(BUFF(71),POINTER),
         &(BUFF(75),TRIPOINTER),(BUFF(77),WT),(BUFF(81),VF(1,1,1)),
         &(BUFF(1),IFIRSTTEMP),(BUFF(9),NSCATTEMP)
12        EQUIVALENCE (POINTERTEMP(1),POINTER),(TRIPTEMP(1),TRIPOINTER)

13        LOGICAL VERTICAL
14        DIMENSION ZHAT(3),IORDER(3),Q(3,3),QI(3,3),S(3,3),SI(3,3)
15        REAL MATA(3,3),MATB(3,3),MATC(3,3),MATAI(3,3),ZHATP(3),
         &MATBI(3,3),MATCI(3,3),VF(3,4,3)

16        COMPLEX DEC,ZERO,EYE
     &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

17        COMMON /PHASE/  RD,DO   !   Path length difference for phase.

18        COMMON /RCS/ TOTAL_RCS ! Used only in the front end driver.

19        COMPLEX RHO(10,10),RHOP,FAC2,FAC3,ZTRI,ZTRI2,FE,FA
20        COMPLEX FAC4C,FAC5,DENOM,NUMEL,NUMAZ,PHASE
21        COMPLEX RTS,ROOT_SIGMA(66),RCS_SUM
22        REAL IHAT(3),I2HAT(3),IHATP(3),I2HATP(3),SHATP(3),I3HAT(3),
         &RADAR_POSITION(3),THETA1(10,10),RHOD(10,10),I3HATP(3)
     C
23        LOGICAL SPECULAR

24        DIMENSION COORDS(36),TARGET_POSITION(3),ANP(3)
25        DIMENSION RO(3),ROP(3),PT1(3),PT2(3),PT3(3)
26        DIMENSION AN(3)
27        INTEGER OLDNFACET

28        EQUIVALENCE (COORDS(1),PT1(1)),(COORDS(4),PT2(1)),
         &                           (COORDS(7),PT3(1)),
         &                 (COORDS(7),RADIUS1),(COORDS(8),RADIUS2)
29        EQUIVALENCE (COORDS(1),VF(1,1,1))

30        DATA COUNTER /1/

     C
31        COMPLEX RICE,TEMP
32        CHARACTER*8 TARGFILE

33        NFACET = 0             ! # of ground facets (increments).
34        OWL = 0.     !  Distance of target origin ABOVE water line (ft.)
35        RCS_SUM = (0.,0.)

36        RADAR_POSITION(1) = XS
37        RADAR_POSITION(2) = YS
38        RADAR_POSITION(3) = ZS

39        TARGET_POSITION(1) = XT
```

```
40       TARGET_POSITION(2) = YT
41       TARGET_POSITION(3) = ZT

42       DO(I=1,3) AN(I)=RADAR_POSITION(I) - TARGET_POSITION(I)
44       CALL MATMU(Q,AN,ANP,3,3,1) ! Radar antenna's position in target fr

46       COMPUTE-HEIGHTS-AND-RANGES
48       FIND-CLUTTER-RCS

50       NFACET = G/FLENGTH  ! NOTE: Facet underneath radar is ignored.
51       IF(NFACET.GT.10) NFACET = 10
52       NFACET = 0    !  To simulate free space!!!
53       IF( NFACET .NE. OLDNFACET .AND. NFACET .GE. 1)
54       .  PERFORM-MULTIPATH-CALCULATIONS-FOR-COMPLEX-RHO
         ...FIN
56       OLDNFACET=NFACET
59       CALCULATE-TRACK-ERROR
61       INCREMENT-COUNTER

63       RETURN


    ----------------------------------------

64       TO FIND-CLUTTER-RCS
         .
     C   .        * * * * * * CLUTTER SECTION  * * * * * * * * * * * * * *
65       .  SIGOB=0.
66       .  IFCL=1
     C   .
     C   .
67       .  THETA=ATAN(POSS/G)
68       .  SIGOB=SIGO*R*BWAZ              ! OK.
69       .  WHEN(SIGOB.GE.1.E-7)
70       .  .  EPSC=THETA+ELS
71       .  .  AZCLUT=0.  ! Clutter in same azimuthal direction as boresight
72       .  .  RSIGC=SQRT(SIGOB*C31)
         .  ...FIN
73       .  ELSE
74       .  .  IFCL=0
75       .  .  RSIGC=.0000001
         .  ...FIN
76       .
     C   .       * * * * * * * END OF CLUTTER SECTION * * * * * * * * * * * *
         .
         ...FIN
77

    ----------------------------------------

78       TO PERFORM-MULTIPATH-CALCULATIONS-FOR-COMPLEX-RHO
     C   .        * * * * * * MULTIPATH SECTION  * * * * * * * * * * * * * *
         .
79       .  DO(IHT=1,10)
80       .  .  DO(K=1,NFACET)
```

```
 81        . . . .  RHOD(K,IHT)=0.
 82        . . . .  RHO(K,IHT)=ZERO
      C    . . . .
 83        . . . .  H1=POSS                ! Height of radar above ground plane.
 84        . . . .  H2=IHT*HEIGHT_INCREMENT
 85        . . . .  G1=G-K*FLENGTH-FLENGTH/2.  !Ground distance from radar to facet cen
 86        . . . .  G2=G-G1
 87        . . . .  IF(ELS+ATAN(H1/G1).GT.20.*BWT1) GOTO 997
      C    . . . .
 88        . . . .  R1=SQRT(G1*G1+H1*H1)
 89        . . . .  R2=SQRT(G2*G2+H2*H2)
 90        . . . .  DELO=R1+R2-R  !  Path length difference.
 91        . . . .  IF(DELO.GE.C3)GOTO 997
      C    . . . .
 92        . . . .  BETAM=1.
 93        . . . .  SH=ROUGH                ! RMS surface roughness (ft.)
 94        . . . .  BETAO=2.*SH/DC
 95        . . . .  THETA1(K,IHT)= ATAN( H1 / G1 )    ! Depression angle from rad. to f
 96        . . . .  THETA2 = ATAN(H2/G2)              !    "        "     "   TARGET  "
 97        . . . .  PSI1=THETA1(K,IHT)
 98        . . . .  IF(PSI1.LE.0.) GO TO 997
 99        . . . .  BETA = (THETA2-THETA1(K,IHT))/2.
100        . . . .  ABETA = ABS(BETA)
101        . . . .  H1H2M=H1+H2
102        . . . .  RL=G*SQRT(1.+XK*DELO/PI)/(1.+(H1H2M)**2/(WL*G)) !1st Fresnel zone
103        . . . .  RL2=RL/2.
104        . . . .  GA=G1-RL2
105        . . . .  GB=G1+RL2
      C    . . . .   BETA_TEST = WL*SQRT(1.+2.*DELO/WL)/(4.*PSI1*R1)
106        . . . .  G2SPEC = G*H2/H1H2M
107        . . . .  GTEST1 = (K-1)*FLENGTH
108        . . . .  GTEST2 = K*FLENGTH
109        . . . .  WHEN(G2SPEC.GT.GTEST1.AND.G2SPEC.LE.GTEST2) SPECULAR = .TRUE.
111        . . . .  ELSE SPECULAR = .FALSE.
113        . . . .  RHOS2=0.
115        . . . .  RHOS1=0.
116        . . . .  SINP=SIN(PSI1)
117        . . . .  SINP2=SIN(THETA2)
118        . . . .  COSP=COS(PSI1)
119        . . . .  FAC1=CO*SH                ! SQRT(2) * XK * SIGMA H
120        . . . .  FAC4=FAC1*SINP
121        . . . .  IF(FAC4.LT.7.) RHOS1=EXP2(-1.*FAC4*FAC4)
122        . . . .  FAC4=FAC1*SINP2
123        . . . .  IF(FAC4.LT.7.) RHOS2=EXP2(-1.*FAC4*FAC4)
124        . . . .  FAC2=CSQRT(DEC-COSP*COSP)
125        . . . .  WHEN(VERTICAL) FAC3=DEC*SINP
127        . . . .  ELSE FAC3=SINP
129        . . . .  FE=FAC3-FAC2
131        . . . .  FA=FAC3+FAC2
132        . . . .  RHOP=FE/FA                ! Fresnel reflection coefficient.
           . . . .
133        . . . .  RHOD(K,IHT)=0.
           . . . .
134        . . . .  NEVER
      C    . . . . .      NOTE: Assume specular reflection only for tank target.
      C    . . . . .                           For the time being that is.
```

```
                  .  .  .  .
                  .  .  .  .
135               .  .  .  .  .  RHOS1S=0.
136               .  .  .  .  .  RHOS2S=0.
137               .  .  .  .  .  IF(RHOS1.GT.1.0E-10) RHOS1S=RHOS1*RHOS1
138               .  .  .  .  .  RHOS2S = RHOS2*RHOS2
139               .  .  .  .  .  FD2=SQRT((1.-RHOS1S)*(1.-RHOS2S))
140               .  .  .  .  .  B_OVERO=BETA/BETAO
141               .  .  .  .  .  RHOD2=R*FD2*(THETA1(K,IHT)+THETA2)*FLENGTH*EXP(-B_OVERO*B_OVERO)*
                &.  .  .  .  .  RHOP*CONJG(RHOP)/(R1*R2*CO1*BETAO)
142               .  .  .  .  .  IF(RHOD2.GT.1.) RHOD2=1.
143               .  .  .  .  .  RHOD(K,IHT)=SQRT(RHOD2)  !  Diffuse reflection coefficient.
                  .  .  .  .  ...FIN
144               .  .  .  .  IF(.NOT. SPECULAR) GOTO 997
145 994           .  .  .  .  CONTINUE
146               .  .  .  .  RHO(K,IHT)=RHOP*RHOS1        ! Fresnel times Rayleigh roughness.
147 997           .  .  .  .  CONTINUE
    C             .  .  .        * * * * * * END OF MULTIPATH SECTION * * * * * * * * * * *
                  .  .  .
                  .  .  ...FIN
148               .  ...FIN
149               ...FIN

            --------------------------------------

150               TO CALCULATE-TRACK-ERROR
    C             .   MONOPULSE
152               .  NUMEL = ZERO           !  Numerator for elevation diff/sum ratio.
153               .  NUMAZ = ZERO           !     "        "   azimuth     "      "
154               .  DENOM = ZERO
                  .
    C             .   READ (3,*) NUMBER_OF_SCATTERERS
155               .  REWIND 5
156               .  POINTER=IFIRST
157               .  DO (J=1,NUMBER_OF_SCATTERERS)
158               .  .  DO (I=1,66) ROOT_SIGMA(I) = ZERO
160               .  .  READ-IN-SCATTERER-COORDINATES
163               .  .  CONVERT-SCATTERER-MIDPOINT-TO-FIXED-EARTH-FRAME
165               .  .  COMPUTE-HEIGHTS-AND-RANGES
167               .  .  FIND-AZ-AND-EL-OFF-BORESIGHT
169               .  .  UNLESS(RO(3) .LT. 0.)
    C             .  .  .                              Check for scatterer under water.
                  .  .  .
170               .  .  .  CALL PATT(RD1,RD2,RS,TEI,TAI,BWR1,BWR2)     ! Receiving pattern.
171               .  .  .  CALL PATT(DUME,DUME,TS1,TEI,TAI,BWT1,BWT2) ! Transmission patte
                  .  .  .
172               .  .  .  WHEN(COUNTER .EQ. 1)
173               .  .  .  .  DO(I=1,3) IHAT(I)=RO(I)-RADAR_POSITION(I)
175               .  .  .  .  CALL NORMLZ(IHAT,IHAT)
177               .  .  .  .  CALL MATMU(Q,IHAT,IHATP,3,3,1) ! Incident unit vector in target fr
178               .  .  .  .  DO(I=1,3) SHATP(I)=-IHATP(I)       ! Scattered unit vector.
180               .  .  .  .
181               .  .  .  .  CALL RCS(ITYPE,VF,ROP,ZHATP,ANP,IHATP,SHATP,XK,
                &.  .  .  .  VERTICAL,RTS)                  ! Freespace term.
                  .  .  .  .
182               .  .  .  .  ROOT_SIGMA(1)=CONJG(RTS)*CEXP(-EYE*XK*2.*RD)*WT
```

```
                . . .   ...FIN
183             . . --  ELSE READ (5) ROOT_SIGMA
185             . . .   RCS_SUM = RCS_SUM + ROOT_SIGMA(1)
                . . .
     C          . . .     PHASE_ANGLE = ATAN2(AIMAG(ROOT_SIGMA(1)),
     C          . . .   &REAL(ROOT_SIGMA(1)))*180./PI
     C          . . .    XMAG = CABS(ROOT_SIGMA(1)) * .3048
187             . . .   TEMP=ROOT_SIGMA(1)*TS1
188             . . .   NUMEL=RD1*TEMP + NUMEL
189             . . .   NUMAZ=RD2*TEMP + NUMAZ
190             . . .   DENOM=RS*TEMP  + DENOM
                . . .
191             . . .   UNLESS (NFACET .EQ. 0)
192             . . . .   DO ( K = 1,NFACET ) ADD-IN-MULTIPATH-CONTRIBUTIONS
195             . . .   ...FIN
197             . . .
                . . ...FIN
198             . ...FIN
199             . IF (COUNTER .EQ. 1) WRITE (5) ROOT_SIGMA
201             . UNLESS(IFCL.EQ.0)
     C          . .     CHECKED FOR NO CLUTTER
202             . . ZTRI=RICE(ZERO,RSIGC)
203             . . ECLUT=-EPSC
204             . . ACLUT=AZCLUT
205             . . CALL PATT(D1CP,D2CP,SCP,ECLUT,ACLUT,BWR1,BWR2)
206             . . DENOM=DENOM+ZTRI*SCP
207             . . NUMEL=NUMEL+ZTRI*D1CP
208             . . NUMAZ=NUMAZ+ZTRI*D2CP
                . ...FIN
209             .
210             . ELERR = P1*REAL(NUMEL/DENOM)
211             . AZERR = P2*REAL(NUMAZ/DENOM)
                .
212             . RCS_SUM = RCS_SUM * .3048  !  Convert Feet to Meters.
                .
213             . WHEN ( RCS_SUM .EQ. (0.,0.) ) TOTAL_RCS = -500.
215             . ELSE    TOTAL_RCS = 20.*ALOG10(CABS(RCS_SUM))
217             ...FIN
218

        ----------------------------------------

219             TO READ-IN-SCATTERER-COORDINATES
                .
220             . READ (3'POINTER+66) (BUFF(I),I=1,128)
221             . DO(I=1,2)
222             . . POINTERTEMP(I) = BUFF(70+I)
223             . . TRIPTEMP(I) = BUFF(74+I)
                . ...FIN
224             . IF (TRIPOINTER .NE. 0)
226             . . IREC = TRIPOINTER + 66
227             . . READ (3'IREC) (BUFF(I),I=129,256)
                . ...FIN
228             .
     C          .  DO(I=1,256) BUFF(I)=IBUFF(I,J)
                .
```

```
        ...FIN

        ------------------------------------

        TO CONVERT-SCATTERER-MIDPOINT-TO-FIXED-EARTH-FRAME
        .  SELECT(ITYPE)
        .  .  (1)
        .  .  .  PT1(3)=PT1(3)+OWL       ! Add distance from origin to waterline
        .  .  .  PT2(3)=PT2(3)+OWL       ! Z-coordinates. (Redefine origin to waterl
        .  .  .  DO(I=1,3) ROP(I)=(PT1(I)+PT2(I))/2.   ! Frustum midpoint.
        .  .  ...FIN
        .  .  (2)
        .  .  .  PT1(3)=PT1(3)+OWL               ! Triangular flat plate.
        .  .  .  PT2(3)=PT2(3)+OWL
        .  .  .  PT3(3)=PT3(3)+OWL
        .  .  .  DO(I=1,3) ROP(I)=(PT1(I)+PT2(I)+PT3(I))/3.
        .  .  ...FIN
        .  .  (4)
        .  .  .  DO(KK=1,3)
        .  .  .  .  DO(JJ=1,4) VF(3,JJ,KK)=VF(3,JJ,KK)+OWL       ! Trihedral.
        .  .  .  ...FIN
        .  .  .  DO(I=1,3) ROP(I)=VF(I,1,1)
        .  .  ...FIN
        .  .  (5)
        .  .  .  DO(KK=1,2)
        .  .  .  .  DO(JJ=1,4) VF(3,JJ,KK)=VF(3,JJ,KK)+OWL       ! Dihedral.
        .  .  .  ...FIN
        .  .  .  DO(I=1,3) ROP(I)=VF(I,1,1)
        .  .  ...FIN
        .  ...FIN
        .  CALL MATMU(GI,ROP,RO,3,3,1) ! ROTATE ROP TO EARTH'S FRAME.
        .  DO(I=1,3) RO(I)=RO(I)+TARGET_POSITION(I)
        .  XT=RO(1)
        .  YT=RO(2)
        .  ZT=RO(3)
        ...FIN
      C

        ------------------------------------

        TO COMPUTE-HEIGHTS-AND-RANGES
        .  POSS=ZS
        .  POST=ZT
        .
        .  G=SQRT((XT-XS)**2+(YT-YS)**2)    !  GROUND RANGE
        .  R=SQRT(G*G+(POST-POSS)**2) !DIRECT RANGE FROM ANTENNA TO TARGET.
        ...FIN

        ------------------------------------

        TO FIND-AZ-AND-EL-OFF-BORESIGHT
        .  ELT=ATAN((RO(3)-ZS)/G)
        .  AZT=ATAN3((RO(1)-XS),(RO(2)-YS))
        .  TEI=ELT-ELS   !  Target elevation (pitch), wrt boresight(UP? +)
        .  TAI=AZT-AZS   !  Target azimuth (yaw) , wrt boresight (clockwise +
        .  UNTIL (TAI .GT. -PI) TAI = TAI + TWOPI
```

```
         .  UNTIL (TAI .LE. PI )  TAI = TAI - TWOPI
         ...FIN

      ------------------------------------

         TO ADD-IN-MULTIPATH-CONTRIBUTIONS
         .  DETERMINE-NEAREST-HEIGHT-INCREMENT
         .  TEI=(-THETA1(K,IHT)-ELS) ! OK if ELS & TEI are defined positive
         .  TAN=TAI
         .  UNLESS(RHO(K,IHT).EQ.ZERO.AND.RHOD(K,IHT).LE.0.)
         .  .  ZTRI=RHO(K,IHT)
         .  .  IF(RHOD(K,IHT).GT.0.) ZTRI=RICE(RHO(K,IHT),RHOD(K,IHT))
         .  .  CALL PATT(RD1N,RD2N,RSN,TEI,TAN,BWR1,BWR2)
         .  .  CALL PATT(DUME,DUME,TSN,TEI,TAN,BWT1,BWT2)
         .  .
         .  .  INDX = 1 + K
         .  .  IF(COUNTER .EQ. 1)
         .  .  .  II=K
         .  .  .  FIND-PATH-LENGTHS
         .  .  .  I2HAT(1)=COS(THETA2)*SIN(AZT) !  Assuming AZT is measured
         .  .  .  I2HAT(2)=COS(THETA2)*COS(AZT)  !  clockwise from the Y-axis.
         .  .  .  I2HAT(3)= SIN(THETA2)
         .  .  .  CALL MATMU(Q,I2HAT,I2HATP,3,3,1) !Indirect incident vector i
         .  .  .
         .  .  .  DO(I=1,3) SHATP(I)=-IHATP(I)
         .  .  .  CALL RCS(ITYPE,VF,ROP,ZHATP,ANP,I2HATP,SHATP,XK,
        &.  .  .  VERTICAL,RTS)       !  Diplane term.
         .  .  .
         .  .  .  DELTA=R1+R2-DO
         .  .  .  PHASE=CEXP(-EYE*XK*(RD+DELTA))
         .  .  .  ROOT_SIGMA(INDX)=CONJG(RTS)*PHASE*WT
         .  .  ...FIN
         .  .  RCS_SUM = RCS_SUM + ROOT_SIGMA(INDX) * ZTRI*2.
         .  .
         .  .  TEMP=TSN*ROOT_SIGMA(INDX)*ZTRI
         .  .  NUMEL=RD1*TEMP + NUMEL              !  Cross terms.
         .  .  NUMAZ=RD2*TEMP + NUMAZ
         .  .  DENOM=RS*TEMP  + DENOM
         .  .
         .  .  TEMP=TS1*ROOT_SIGMA(INDX)*ZTRI
         .  .  NUMEL=RD1N*TEMP + NUMEL              !  More cross terms.
         .  .  NUMAZ=RD2N*TEMP + NUMAZ
         .  .  DENOM=RSN*TEMP  + DENOM
         .  .
         .  .  DO(L=K,NFACET)
         .  .  .  ZTRI2=RHO(L,IHT)
         .  .  .  IF(RHOD(L,IHT).GT.0.) ZTRI2=RICE(RHO(L,IHT),RHOD(L,IHT))
         .  .  .
         .  .  .  EVALUATE-RTS-INDEX
         .  .  .  IF(COUNTER .EQ. 1)
         .  .  .  .  II=L
         .  .  .  .  FIND-PATH-LENGTHS
         .  .  .  .
         .  .  .  .  I3HAT(1)=COS(THETA2)*SIN(AZT)
         .  .  .  .  I3HAT(2)=COS(THETA2)*COS(AZT)
         .  .  .  .  I3HAT(3)= SIN(THETA2)
```

```
337     . . . . .  CALL MATMU(Q,I3HAT,I3HATP,3,3,1)
338     . . . . .  DO(I=1,3) SHATP(I)=-I3HATP(I)
340     . . . . .
341     . . . . .  CALL RCS(ITYPE,VF,ROP,ZHATP,ANP,I2HATP,SHATP,XK,
       &. . . . .  VERTICAL,RTS)  ! Indirect-Indirect term.
        . . . . .
342     . . . . .  DELTA2=R1+R2-D0
343     . . . . .  PHASE=CEXP(-EYE*XK*(DELTA+DELTA2))
344     . . . . .  ROOT_SIGMA(INDX)=CONJG(RTS)*PHASE*WT
        . . . . ...FIN
345     . . . .
346     . . . .  RCS_SUM = RCS_SUM + ROOT_SIGMA(INDX) * ZTRI * ZTRI2
347     . . . .  TEI=(-THETA1(L,IHT)-ELS)
348     . . . .  CALL PATT(RD1I,RD2I,RSI,TEI,TAN,BWR1,BWR2)
349     . . . .  CALL PATT(DUME,DUME,TSI,TEI,TAN,BWT1,BWT2)
        . . . .
350     . . . .  TEMP=TSN*ROOT_SIGMA(INDX)*ZTRI*ZTRI2
351     . . . .  NUMEL=RD1I*TEMP + NUMEL
352     . . . .  NUMAZ=RD2I*TEMP + NUMAZ
353     . . . .  DENOM=RSI*TEMP  + DENOM
        . . . .
354     . . . .  UNLESS(K.EQ.L)
355     . . . . .  TEMP=TSI*ROOT_SIGMA(INDX)*ZTRI*ZTRI2
356     . . . . .  NUMEL=RD1N*TEMP + NUMEL
357     . . . . .  NUMAZ=RD2N*TEMP + NUMAZ
358     . . . . .  DENOM=RSN*TEMP  + DENOM
        . . . . ...FIN
359     . . . ...FIN
360     . . ...FIN
361     . ...FIN

        -----------------------------------------

362     TO DETERMINE-NEAREST-HEIGHT-INCREMENT
364     .  NHT=RO(3)/HEIGHT_INCREMENT
        .  CONDITIONAL
365     . .  (NHT.GE.10) IHT=10
367     . .  (NHT.LE.1)        IHT= 1
369     . .  (OTHERWISE)
370     . . .  WHEN(RO(3).GT.(NHT+.5)*HEIGHT_INCREMENT) IHT=NHT+1
372     . . .  ELSE IHT=NHT
374     . . ...FIN
        . ...FIN
375     ...FIN

        -----------------------------------------

376     TO FIND-PATH-LENGTHS
378     .  G2=FLENGTH*II-FLENGTH/2.
379     .  G1=G-G2
380     .  THETA2=ATAN(RO(3)/G2)
381     .  R1=SQRT(ZS*ZS+G1*G1)
382     .  R2=SQRT(ZT*ZT+G2*G2)
        ...FIN

        -----------------------------------------
```

```
00696   383        TO INCREMENT-COUNTER
00697   385        .  COUNTER = COUNTER + 1 ! RCS calculated only once per 10 track loop
00698   386        .  IF ( COUNTER .GT. NFACET ) COUNTER = 1
00699   387        .  COUNTER = 1        !  Always calculate RCS.
00700              ...FIN

           ----------------------------------------

00701   388        TO EVALUATE-RTS-INDEX    INDX=(K-1)*10+L-(K-1)*K/2+11
00702   391        END

           ----------------------------------------

                   PROCEDURE CROSS-REFERENCE TABLE

              00597 ADD-IN-MULTIPATH-CONTRIBUTIONS
                    00508

              00464 CALCULATE-TRACK-ERROR
                    00352

              00582 COMPUTE-HEIGHTS-AND-RANGES
                    00477      00342

              00549 CONVERT-SCATTERER-MIDPOINT-TO-FIXED-EARTH-FRAME
                    00476

              00678 DETERMINE-NEAREST-HEIGHT-INCREMENT
                    00598

              00701 EVALUATE-RTS-INDEX
                    00640

              00589 FIND-AZ-AND-EL-OFF-BORESIGHT
                    00478

              00358 FIND-CLUTTER-RCS
                    00343

              00689 FIND-PATH-LENGTHS
                    00643      00610

              00696 INCREMENT-COUNTER
                    00353

              00381 PERFORM-MULTIPATH-CALCULATIONS-FOR-COMPLEX-RHO
                    00349

              00534 READ-IN-SCATTERER-COORDINATES
                    00475

       (FLECS 77  VERSION 22.38)

   MODULE CONTAINS  NO MINOR ERRORS
   MODULE CONTAINS  NO MAJOR ERRORS
```

```
----------------------------------------

 1        SUBROUTINE PATT(D1,D2,S,T1,T2,B1,B2)
    C
    C#PURPOSE
    C       COMPUTES SUM AND 2 DIFFERENCE PATTERNS
    C      FOR 4 SYMMETRICALLY PLACED FEEDS
    C
    C#PARAMETER DESCRIPTOR
    COUT D1        - DIFFERENCE IN THE 1 DIRECTION
    COUT D2       - DIFFERENCE IN THE 2 DIRECTION
    COUT S        - SUM PATTERN
    CIN  T1       - POSITION IN THE 1 DIRECTION
    CIN  T2        - POSITION IN THE 2 DIRECTION
    CIN  B1       -BEAMWIDTH IN THE 1 DIRECTION
    CIN B2       BEAMWIDTH IN THE 2 DIRECTION
    C
 2        A=.3
 3        TQ1=B1*A
 4        TQ2=B2*A
 5        TP1=T1+TQ1
 6        TM1=T1-TQ1
 7        TP2=T2+TQ2
 8        TM2=T2-TQ2
 9        P1=FBEAM(TP1,TM2,B1,B2)
10        P2=FBEAM(TM1,TM2,B1,B2)
11        P3=FBEAM(TP1,TP2,B1,B2)
12        P4=FBEAM(TM1,TP2,B1,B2)
13        S=P1+P2+P3+P4
14        D2=P1+P2-P3-P4
15        D1=P2+P4-P1-P3
16        RETURN
17        END

    (FLECS 77  VERSION 22.38)

MODULE CONTAINS  NO MINOR ERRORS
MODULE CONTAINS  NO MAJOR ERRORS
```

```
--------------------------------------

  1          FUNCTION ATAN3(YPARM, XPARM)
           C
           C#ABSTRACT ARC TANGENT FUNCTION FOR ANGLE BETWEEN 0 AND 2 PI
           C
           C
  2          PIX2=8.*ATAN(1.)
           C
  3          X=XPARM
  4          Y=YPARM
           C
           C
           C  COMPUTE TANGENT
  5          ATAN3=ATAN2(Y, X)
  6          IF(ATAN3.LT.0.)ATAN3=PIX2+ATAN3
  7          RETURN
  8          END

           (FLECS 77  VERSION 22.38)

      MODULE CONTAINS  NO MINOR ERRORS
      MODULE CONTAINS  NO MAJOR ERRORS
```

```
----------------------------------------

 1          FUNCTION EXP2(X)
   C
   C#ABSTRACT CHECK LIMITS ON EXPONENT OVERFLOW AND UNDERFLOW
   C
   C
 2          WHEN(X.LT.-88.028 ) EXP2=0.
 4          ELSE EXP2 = EXP(X)
 6 C
 7          RETURN
 8          END

   (FLECS 77  VERSION 22.38)

MODULE CONTAINS  NO MINOR ERRORS
MODULE CONTAINS  NO MAJOR ERRORS
```

```
---------------------------------------

1         COMPLEX FUNCTION RICE(A,B)
     C    SUBPROGRAMS CALLED
     C         UNIRAN - UNIFORM RANDOM NUMBER GENERATOR
2         COMPLEX A
3         UR=UNIRAN(0.)*6.2831853
4         XR=B*SQRT(-ALOG(UNIRAN(0.)))
5         RICE=CMPLX(REAL(A)+XR*COS(UR),AIMAG(A)+XR*SIN(UR))
6         RETURN
7         END

        (FLECS 77  VERSION 22.38)

   MODULE CONTAINS  NO MINOR ERRORS
   MODULE CONTAINS  NO MAJOR ERRORS
```

```
      FUNCTION UNIRAN(X)  !  SIMULATE A RANDOM # GENERATOR.
      UNIRAN = .5
      RETURN
      END
```

(FLECS 77 VERSION 22.38)

MODULE CONTAINS NO MINOR ERRORS
MODULE CONTAINS NO MAJOR ERRORS

```
--------------------------------------

 1        FUNCTION ENVEL(A,B1,B2)
     C
     C#PURPOSE
     C    COMPUTES THE FIELD PATTERN EFFECT FOR USE IN CLUTTER CALCULATION
     C          INPUTS REQUIRED AND ITS PRINCIPAL OUTPUTS; ALSO INDICATE
     C          ANY RESTRICTIONS OR OPTIONS BUILT INTO THE PROGRAM
     C
     C     INSTALLED 17 JUNE BY S.P.STUK
 2        ENVEL=1.
 3        RETURN
 4        END

     (FLECS 77  VERSION 22.38)

MODULE CONTAINS  NO MINOR ERRORS
MODULE CONTAINS  NO MAJOR ERRORS
```

```
 1        FUNCTION FUNCA(RM,DEL)
 2        FUNCA = 0.
          CONDITIONAL
 3        .  (RM .EQ. 0.) RETURN
 5        .  (RM .LT. 0.)
 6        .  .  RM = -RM
 7        .  .  SIGN = 1.
          .  ...FIN
 8        .  (RM .GT. 0.) SIGN = 1.
          ...FIN
10        MI=INT(RM)
12        A=RM-MI
          CONDITIONAL
13        .  (A .GE. (1-DEL)) FUNCA=SIGN*(2*(MI+1.)*DEL+A-1.)
15        .  (A .GE. DEL) FUNCA=SIGN*(DEL*(2*MI+1.))
17        .  (OTHERWISE)  FUNCA=SIGN*(2*MI*DEL+A)
          ...FIN
19        RETURN
21        END

     (FLECS 77  VERSION 22.38)

  MODULE CONTAINS  NO MINOR ERRORS
  MODULE CONTAINS  NO MAJOR ERRORS
```

```
------------------------------------

      FUNCTION FBEAM(A1,A2,B1,B2)
      D=.3
      CD1=3/(2.+D)
      U=2.7831*SQRT((A1/B1)**2+(A2/B2)**2)
      U2=U*U
      CD2=2*(1-D)/U2
      SU=SIN(U)/U
      FBEAM=CD1*(D*SU+CD2*(SU-COS(U)))
      RETURN
      END

        (FLECS 77  VERSION 22.38)

 MODULE CONTAINS  NO MINOR ERRORS
 MODULE CONTAINS  NO MAJOR ERRORS
```

```
------------------------------------

      C       RCS.FLX      10/7/81            R. B. Rakes



      C  This version of CROSS is used as a subroutine for TRERR.FLX.
      C  Includes trihedrals and dihedrals also.
      C  A new scatterer - partial frustum - is included.


 1          SUBROUTINE RCS (ITYPE,VFIN,RO,ZHAT,AN,IHAT,SHAT,XK,VERTICAL,RTS)

 2          DIMENSION COORDS(36),P1(3),P2(3),P3(3),VFIN(36)
 3          DIMENSION VF(3,4,3),TEMP(3),ZHAT(3),AN(3)
 4          DIMENSION EHAT(3),DHAT(3),RO(3),ROIMAGE(3)
 5          DIMENSION PP1(3),PP2(3),PP3(3)
 6          REAL DO
 7          INTEGER*2 ITYPE

 8          REAL IHAT(3),SHAT(3),NHAT(3),B1(3),B2(3)
 9          COMPLEX EYE,IK,RTS

 10         LOGICAL CANTSEE,VERTICAL

      C*COMMON
 11         COMMON/EXTRA/TWOPI,ROOTPI,EYE,IK,PI,PIO4
 12         COMMON/PHASE/RD,DO
      C*
 13         COMMON/DATUM/CNVT,FTM,NTM,PIO2 !Deg to Rad.,Feet to Meters,NMI to

 14         EQUIVALENCE (COORDS(1),P1(1)),(COORDS(4),P2(1)),(COORDS(7),P3(1)),
           &                   (COORDS(7),RADIUS1),(COORDS(8),RADIUS2)

 15         EQUIVALENCE (COORDS(1),VF(1,1,1))



 16         DO(I=1,36) COORDS(I)=VFIN(I) ! Since you cannot equiv a subr. para
 18
 19         DO=VMAG(AN)       !Distance between radar and ship origin.  Note th
      C!                      !AN is the vector between the present ship origin
      C!                      !and the radar antenna.
 20         DO(I=1,3) DHAT(I) = -AN(I)/DO  ! Unit vector pointing from radar t
 22         IK=EYE*XK
 24         RTS=(0.,0.)
 25         CANTSEE=.FALSE.
 26         HT=RO(3)

      C       POOR MAN'S HIDDEN SURFACE ALGORITHM

 27         IF(ITYPE.EQ.2)
 28         .  DO(I=1,3)
 29         .  .  B1(I)=P2(I)-P1(I)
 30         .  .  B2(I)=P3(I)-P1(I)
```

```
             .  ...FIN
31           .  CALL VCROSS(B1,B2,NHAT)
33           .  IF(VDOT(NHAT,IHAT).GT.O) CANTSEE=.TRUE.
             ...FIN
34
35           UNLESS(CANTSEE)
36           .  FIND-PATH-LENGTH-DIFFERENCES
38           .  CALCULATE-RTS
             ...FIN
40
42           RETURN


    ---------------------------------------


43           TO FIND-PATH-LENGTH-DIFFERENCES
44           .  DO(I=1,2) ROIMAGE(I)=RO(I)
46           .  ROIMAGE(3) = -RO(3)
48           .  DO(I=1,3) TEMP(I)=RO(I)-AN(I)
50           .  RS=VMAG(TEMP)  !  Direct distance from antenna to center of scatte
52           .  DO(I=1,3) TEMP(I)=ROIMAGE(I)-AN(I)
54           .  RSI=VMAG(TEMP) ! Indirect distance from ant. to center of scattere
56           .  DELTA = RS-RSI ! Difference between direct & indirect path lengths
57           .  WHEN(DO.LT. .01)
58           .  .  RD=RS-DO       !Difference between direct path and distance between
     C!                      !radar and origin of ship.
             .  ...FIN
59           .  ELSE
60           .  .  RD=VDOT(RO,DHAT)    ! Plane wave term.
61           .  .  CALL VCROSS(RO,DHAT,TEMP)    !  Find near field correction.
62           .  .  COST=VDOT(IHAT,DHAT)
63           .  .  IF (COST .LT. -1. ) COST=-1.
64           .  .  IF (COST .GT.  1. ) COST= 1.
65           .  .  RD=RD+VMAG(TEMP)*SQRT((1.-COST)/(1+COST)) ! Tangent half-angle for
             .  ...FIN
66           .  RDD=2.*RD       !direct-direct
68           .  RII=RDD+2.*DELTA       !indirect-indirect
69           .  RDI=RDD+DELTA      !direct-indirect=indirect-direct
             ...FIN

    ---------------------------------------

70           TO CALCULATE-RTS
             .  SELECT(ITYPE)
72           .  .  (1)
73           .  .  .  CALL FRUST(P1,P2,RADIUS1,RADIUS2,RO,XK,IHAT,SHAT,RTS)
             .  .  ...FIN
74           .  .  (2)
75           .  .  .  RE-ORIGIN-TRIANGLES-COORDINATES-TO-CENTROID
77           .  .  .  CALL TRIPLATE(PP1,PP2,PP3,RO,XK,IHAT,SHAT,RTS)
             .  .  ...FIN
78           .  .  (4)
79           .  .  .  FIND-EHAT
81           .  .  .  CALL CORNER(VF,XK,IHAT,SHAT,EHAT,RO,VERTICAL,RTS)
             .  .  ...FIN
82           .  .  (5)
```

```
00105    83          .  .  .  FIND-EHAT
00106    85          .  .  .  CALL DIPLANE(VF,XK,IHAT,SHAT,EHAT,RO,VERTICAL,RTS)
00107                .  .  ...FIN
00108    86          .  .  (OTHERWISE) RTS=(0.,0.)
00109                .  ...FIN
00110    88          ...FIN

         ----------------------------------------

00111    89          TO RE-ORIGIN-TRIANGLES-COORDINATES-TO-CENTROID
00112    91          .  DO(I=1,3)
00113    92          .  .  PP1(I) = P1(I) - RO(I)
00114    93          .  .  PP2(I) = P2(I) - RO(I)
00115    94          .  .  PP3(I) = P3(I) - RO(I)
00116                .  ...FIN
00117    95          ...FIN

         ----------------------------------------

00118    96          TO FIND-EHAT
00119                .
00120          C   Incident electric field unit vector.
00121                .
00122    98          .  WHEN(VERTICAL)
00123    99          .  .  CALL VCROSS(IHAT,ZHAT,TEMP)
00124   100          .  .  CALL VCROSS(TEMP,IHAT,EHAT)
00125                .  ...FIN
00126   101          .  ELSE CALL VCROSS(IHAT,ZHAT,EHAT)
00127   103          .  CALL NORMLZ(EHAT,EHAT)
00128                ...FIN
00129   105          END

         ----------------------------------------

                     PROCEDURE CROSS-REFERENCE TABLE

              00091 CALCULATE-RTS
                    00062

              00118 FIND-EHAT
                    00105       00101

              00067 FIND-PATH-LENGTH-DIFFERENCES
                    00061

              00111 RE-ORIGIN-TRIANGLES-COORDINATES-TO-CENTROID
                    00097

         (FLECS 77  VERSION 22.38)

    MODULE CONTAINS  NO MINOR ERRORS
    MODULE CONTAINS  NO MAJOR ERRORS
```

```
        ------------------------------------------
        C      YPR.FR              R. B. Rakes      4/15/80
        C      This subroutine is used to select the order of yaw,pitch,and roll.
        C      All angles are measured counter-clockwise or "right-handed".  MATI
        C      is the inverse matrix of MAT.


   1           SUBROUTINE YPR(ANG,MAT,MATI,IORDER)
   2           REAL MAT(3,3),MATI(3,3)

   3           COSA=COS(ANG)
   4           SINA=SIN(ANG)
               SELECT(IORDER)
   5           .  (1) FIND-YAW
   8           .  (2) FIND-PITCH
  12           .  (3) FIND-ROLL
  16           ...FIN
  17           RETURN
        ------------------------------------------

  19           TO FIND-YAW
  20           .  MAT(1,1)=COSA
  21           .  MAT(2,2)=COSA
  22           .  MAT(1,2)=SINA
  23           .  MAT(2,1)=-SINA
  24           .  MATI(1,1)=COSA
  25           .  MATI(2,2)=COSA
  26           .  MATI(1,2)=-SINA
  27           .  MATI(2,1)=SINA
               ...FIN
        ------------------------------------------

  28           TO FIND-PITCH
  30           .  MAT(2,2)=COSA
  31           .  MAT(3,3)=COSA
  32           .  MAT(2,3)=SINA
  33           .  MAT(3,2)=-SINA
  34           .  MATI(2,2)=COSA
  35           .  MATI(3,3)=COSA
  36           .  MATI(2,3)=-SINA
  37           .  MATI(3,2)=SINA
               ...FIN
        ------------------------------------------

  38           TO FIND-ROLL
  40           .  MAT(1,1)=COSA
  41           .  MAT(3,3)=COSA
  42           .  MAT(1,3)=-SINA
  43           .  MAT(3,1)=SINA
  44           .  MATI(1,1)=COSA
  45           .  MATI(3,3)=COSA
```

```
00045     46        .  MATI(3,1)=SINA
00046     47        .  MATI(1,3)=-SINA
00047               ...FIN
00048     48        END

            ---------------------------------------

                    PROCEDURE CROSS-REFERENCE TABLE

                 00028 FIND-PITCH
                       00014

                 00038 FIND-ROLL
                       00015

                 00018 FIND-YAW
                       00013

            (FLECS 77  VERSION 22.38)

       MODULE CONTAINS  NO MINOR ERRORS
       MODULE CONTAINS  NO MAJOR ERRORS
```

```
      ----------------------------------------

        C      TRIPLATE2.FR       R.B. Rakes       6/5/80

        C      This subroutine calculates the complex square root of the RCS for
        C      a triangular flat plate scatterer.  The vertices are specified by
        C      the vectors P1,P2,P2 such that the normal vector is defined right
        C      handedly.  IHAT is the incident vector.
        C      NOTE: All vectors denoted by HAT are unit vectors.

        C        Modified for TRERR.FL on 10/7/81 by RBR.

   1          SUBROUTINE TRIPLATE(P1,P2,P3,RO,K,IHAT,SHAT,RTS)
   2          DIMENSION P1(3),P2(3),P3(3),TEMP(3),A(3),B(3),C(3),
             1CMINUS(3),THAT(3),TAU(3),RO(3),RA(3),RB(3),RC(3)
   3          REAL NHAT(3),IHAT(3),SHAT(3),IMS(3),K,KT

   4          COMPLEX RTS,EYE,IK,TERM,TERMTEMP
        C     INCLUDE RCS.FLX (COMMON)
   5          COMMON /EXTRA/ TWOPI,ROOTPI,EYE,IK,PI,PIO4

   6          TOL=.00001
   7          DO-PRELIMINARY-CALCULATIONS

   9          DO(I=1,3) IMS(I)=IHAT(I)-SHAT(I) ! Find incident minus scatter vec
  11
  12          FIND-VECTOR-RELATIONS
  14          CALCULATE-COMPLEX-TERM
  16          IF(CABS(TERM).GT.TERMO) TERM= CMPLX(TERMO,0.)
  17 100      CONTINUE

  18          RTS=-EYE*K*TERM/ROOTPI

  19          RETURN

      ----------------------------------------

  20          TO DO-PRELIMINARY-CALCULATIONS
  21          .  DO(I=1,3)
  22          .  .  A(I)=P2(I)-P1(I)
  23          .  .  B(I)=P3(I)-P2(I)
  24          .  .  C(I)=P1(I)-P3(I)
  25          .  .  RA(I)=(P1(I)+P2(I))/2.
  26          .  .  RB(I)=(P2(I)+P3(I))/2.
  27          .  .  RC(I)=(P3(I)+P1(I))/2.
              .  ...FIN
  28          .  DO(I=1,3) CMINUS(I)=-C(I)
  31          .  CALL VCROSS(A,CMINUS,TEMP)
  33          .  AREA2=VMAG(TEMP)
  34          .  IF(AREA2.LE.0.)
  35          .  .  RTS=(0.,0.)
  36          .  .  RETURN
              .  ...FIN
  37          .  TERMO=AREA2/2.
  39          .  DO(I=1,3) NHAT(I)=TEMP(I)/AREA2
```

```
00099     41        ...FIN

               -------------------------------------

00100     42        TO FIND-VECTOR-RELATIONS
00101     44        .  CALL VCROSS(NHAT, IMS, TEMP)
00102     45        .  CALL VCROSS(TEMP, NHAT, TAU)
00103     46        .  T=VMAG(TAU)
00104     47        .  IF(T.LT.TOL)
00105     48        .  .  TERM=CMPLX(TERMO, 0.)
00106               .  .
00107               .  .
00108     49        .  .  GOTO 100
00109               .  ...FIN
00110     50        .  KT=K*T
00111     52        .  DO(I=1,3) THAT(I)=TAU(I)/T
00112     54        ...FIN
00113     55

               -------------------------------------

00114     56        TO CALCULATE-COMPLEX-TERM
00115     57        .  CALL VCROSS(NHAT, THAT, TEMP)
00116     58        .  CALL SIDE(A, RA, IMS, K, TEMP, TERM)
00117     59        .  CALL SIDE(B, RB, IMS, K, TEMP, TERMTEMP)
00118     60        .  TERM=TERM+TERMTEMP
00119     61        .  CALL SIDE(C, RC, IMS, K, TEMP, TERMTEMP)
00120     62        .  TERM=TERM+TERMTEMP
00121     63        .  TERM=-TERM/(EYE*KT) ! Minus sign added 3/11/82, RBR
00122     64        .  POLTERM=VDOT(NHAT, IMS)/2.       ! Polarization term.
00123     65        .  IF(POLTERM.GE.0.) TERM=0.       ! Poor man's hidden surface check.
00124     66        .  TERM=TERM*POLTERM
00125               ...FIN
00126     67        END

               -------------------------------------

                    PROCEDURE CROSS-REFERENCE TABLE

                 00114 CALCULATE-COMPLEX-TERM
                       00074

                 00081 DO-PRELIMINARY-CALCULATIONS
                       00069

                 00100 FIND-VECTOR-RELATIONS
                       00073

               (FLECS 77  VERSION 22.38)

          MODULE CONTAINS  NO MINOR ERRORS
          MODULE CONTAINS  NO MAJOR ERRORS
```

```
---------------------------------------

  1         SUBROUTINE SIDE(X,RX,IMS,K,TEMP,TERM)

  2         DIMENSION X(3),RX(3),IMS(3),TEMP(3)
  3         REAL K,IMS
  4         COMPLEX TERM

  5         ARG=K*VDOT(X,IMS)/2.
  6         WHEN(ARG.EQ.0.) TERM=1.
  8         ELSE TERM=SIN(ARG)/ARG
 10         TERM=VDOT(TEMP,X)*TERM
 12         ARG=K*VDOT(RX,IMS)
 13         TERM=TERM*CMPLX(COS(ARG),SIN(ARG))
 14         RETURN
 15         END

       (FLECS 77  VERSION 22.38)

   MODULE CONTAINS  NO MINOR ERRORS
   MODULE CONTAINS  NO MAJOR ERRORS
```

```
-----------------------------------------

      C     FRUST.FR      7/1/80           R.B. Rakes

      C     This subroutine calculates the complex square root of the radar
      C     cross section for a frustum.  All variables are in MKS units.
      C     Inputs are as follows:

      C     P1,P2      : XYZ coordinates of the large and small ends of the a
      C              of the frustum respectively.
      C     A1,A2    : radii of the large and small ends respectively.
      C     RO       : XYZ position of the center of the frustum.
      C     K       : wave number (2*pi/lamda).
      C     IHAT      :  incident vector (unit magnitude).
      C      SHAT                :  scattered vector.

      C     Modified for TRERR.FL on 10/7/81 by RBR.


  1         SUBROUTINE FRUST(P1,P2,A1,A2,RO,K,IHAT,SHAT,RTS)

  2         DIMENSION P1(3),P2(3),AHAT(3),AXIS(3),TEMP(3),RO(3),THAT(3)
  3         REAL K,KA,KQ,L,IHAT(3),SHAT(3),IMS(3),NHAT(3)
  4         COMPLEX RTS,EYE,IK,TERM
      C     INCLUDE CROSS5.FLX (COMMON)
  5         COMMON /EXTRA/ TWOPI,ROOTPI,EYE,IK,PI,PIO4

  6         DO-PRELIMINARY-CALCULATIONS

      C      Find incident vector minus scatter vector (i - s).
  8         DO(I=1,3) IMS(I) = IHAT(I) - SHAT(I)
 10         P=-VDOT(IMS,AHAT)        !projection of i - s on frustum axis.
 12         CALL VCROSS(AHAT,IMS,TEMP)
 13         T=VMAG(TEMP)                !projection of i - s onto plane perpendicu
      C!                             to frustum axis.
 14         WHEN (T .LT. .0001) TERM=(0.,0.)
 16         ELSE CALCULATE-COMPLEX-TERM
 19
 21         RTS=-EYE*S*SQRT(2.*KA)*TERM

 22         RETURN


-----------------------------------------

 23         TO DO-PRELIMINARY-CALCULATIONS
 24         .  DO(I=1,3) AXIS(I)=P1(I)-P2(I)
 26         .  L=VMAG(AXIS)              ! length of frustum axis.
 28         .  DO(I=1,3) AHAT(I)=AXIS(I)/L
 30         .  AP=A1-A2               !difference between large and small radii.
 32         .  AMEAN=(A1+A2)/2.
 33         .  KA=K*AMEAN
 34         .  TANT=AP/L              !tangent of half-angle tau.
 35         .  S=SQRT(AP*AP+L*L)        !slant length.
 36         .  SINT=AP/S
```

```
00191     37        .  COST=L/S
00192               ...FIN

                ---------------------------------------

00193     38        TO CALCULATE-COMPLEX-TERM
00194     40        .  G=P+T*TANT
00195     41        .  COEF=COST/T
00196     42        .  DO(I=1,3) NHAT(I)=-COEF*(G*AHAT(I)+IMS(I))
00197     44        .  KG=K*G
00198     46        .  ARG=KG*L/2.
00199     47        .  WHEN(ARG.EQ.0.) TERM=1.
00200     49        .  ELSE TERM=SIN(ARG)/ARG
00201     51        .  UNLESS(TANT.EQ.0.)
00202     53        .  .  TOKGA=TANT/(KG*AMEAN)
00203     54        .  .  TERM=TERM*CMPLX(1.,-TOKGA)+CMPLX(0.,COS(ARG)*TOKGA)
00204               .  ...FIN
00205     55        .  ARG=PI04-KA*T
00206     57        .  TERM=CMPLX(COS(ARG),SIN(ARG))*TERM
00207     58        .  TERM=TERM*VDOT(NHAT,IMS)/2.        !Polarization term.
00208     59        .  TERM=-TERM/SQRT(T) ! Minus sign added 3/11/82; RBR
00209               ...FIN
00210     60
00211     61        END

                ---------------------------------------

                    PROCEDURE CROSS-REFERENCE TABLE

                00193 CALCULATE-COMPLEX-TERM
                      00175

                00181 DO-PRELIMINARY-CALCULATIONS
                      00166

              (FLECS 77  VERSION 22.38)

        MODULE CONTAINS  NO MINOR ERRORS
        MODULE CONTAINS  NO MAJOR ERRORS
```

```
               --------------------------------------
      C   CORNER.FLX        E.F. Knott & R.B. Rakes      11/12/80

      C   ORCHESTRATES THE VARIOUS SUBPROGRAMS USED IN COMPUTING THE RADAR
      C   ECHO FROM AN ARBITRARY TRIHEDRAL CORNER REFLECTOR.  THE COORDI-
      C   ATES OF THE VERTICES OF THE THREE FACES COMPRISING THE REFLECTOR
      C   MUST BE READ FROM THE FILE IFACE(2).
      C   This version of corner also includes multipath effects from the
      C   sea surface and shadowing from the backs of plates.
      C
      C      Modified for TRERR.FL on 10/7/81 by RBR.


  1         SUBROUTINE CORNER(VG,AK,INK,S,EHAT,ROP,VERTICAL,RTS)

  2         INTEGER ONE
  3         REAL INK(3),N(3,3),IS(3),I2HAT(3),IHAT(3),IMS(3)
  4         COMPLEX SS,SUM,RTS,EYE,IK
  5         LOGICAL VERTICAL,SHADOW,YES

  6         DIMENSION S(3),A(3),B(3),EHAT(3),HHAT(3),U(3,3),V(3,3)
  7         DIMENSION IFACE(5),LIST(6),HHS(3),ROP(3)
  8         DIMENSION VA(2,20),VB(2,20),VC(2,20),VF(3,4,3),VD(3,20),VE(3,20)
  9         DIMENSION ERHAT(3),VG(3,4,3)

      C     INCLUDE CROSS (COMMON)
 10         COMMON /EXTRA/ TWOPI,ROOTPI,EYE,IK,PI,PI04


      C   VA AND VB CONTAIN THE LOCAL X,Y COORDINATES OF THE VERTICES OF A
      C   PAIR OF PLANE POLYGONS WHOSE COMMON AREA IS TO BE FOUND.  THESE
      C   CAN BE PROJECTIONS OF OTHER POINTS ONTO THE PLANE OF A PARTICULAR
      C   TRIHEDRAL FACE. THE ARRAY VC CONTAINS THE 2D COORDINATES OF THE
      C   COMMON AREA.  VF CONTAINS THE X,Y,Z COORDINATES OF THE FOUR VER-
      C   TICES IF THE THREE FACES AND IS READ FROM THE DATA FILE, ALONG
      C   WITH THE COORDINATES OF THE FACE NORMALS N AND UNIT VECTORS U,V.
      C   LINEAR DIMENSIONS ARE ASSUMED GIVEN IN FEET.

      C  Translate points such that the main vertex is the new origin.
 11         DO(K=1,3)
 12         .  DO(J=1,4)
 13         .  .  DO(I=1,3) VF(I,J,K)=VG(I,J,K)-VG(I,1,1)
 15         .  ...FIN
 16         ...FIN
 17
 18         CALCULATE-UNIT-NORMALS

 20         CALL VCROSS(INK,EHAT,HHAT)
 21         WHEN(VERTICAL) CALL VCROSS(S,HHAT,ERHAT)
 23         ELSE
 24         .  DO(I=1,3) ERHAT(I)=EHAT(I)
 26         ...FIN
 27
```

```
    C   INK AND S ARE THE DIRECTIONS OF INCIDENCE AND SCATTERING, RESPEC-
    C   TIVELY.  EHAT AND HHAT ARE THE UNIT VECTORS ALONG THE INCIDENT ELEC-
    C   TRIC AND MAGNETIC FIELDS RESPECTIVELY.  ERHAT IS ALONG THE RECEIVED
    C   ELECTRIC FIELD. II,JJ,KK ARE FACE ID NUMBERS WHICH WILL BE PERMUTED
    C   TO GIVE 6 POSSIBLE COMBINATIONS OF INTERACTIONS.

28        SS=CMPLX(0.,0.)
29        ONE=-1
30        II=1
31        JJ=3
32        KK=2

    C   KLUX IS JUST A DUMMY INDEX AND DOES NOTHING ELSE.

33        DO (KLUX =1,6)
34        .  ONE=-ONE
35        .  KU=JJ
36        .  WHEN(ONE.LE.0)
37        .  .  JJ=II
38        .  .  II=KU
          .  ...FIN
39        .  ELSE
40        .  .  JJ=KK
41        .  .  KK=KU
          .  ...FIN
42        .
    C   THERE ARE ONLY 3 SINGLE-BOUNCE CONTRIBUTIONS, SO WE SKIP THE CALL
    C   TO SCAT HALF THE TIME.
          .
43        .  IF(VDOT(INK,N(1,II)).GE.0) GOTO 300
44        .  IF (ONE.GT.0)
45        .  .  NA=4
46        .  .  NC=4
47        .  .  NE=4
48        .  .  J2=II
49        .  .  DO (J=1,4)
50        .  .  .  DO (I=1,3) VD(I,J)=VF(I,J,II)
52        .  .  ...FIN
53        .  .  IF(VDOT(INK,N(1,JJ)).GT.0.)
55        .  .  .  I2=JJ
56        .  .  .  CALCULATE-UNSHADOWED-REGION
          .  .  ...FIN
58        .  .  IF(VDOT(INK,N(1,KK)).GT.0.)
61        .  .  .  I2=KK
62        .  .  .  NE=NC
63        .  .  .  NC=4
64        .  .  .  CALCULATE-UNSHADOWED-REGION
          .  .  ...FIN
66        .  .
    C  VD is now what is left of the original plate after shadowing (if any)
          .  .
68        .  .  CALL SCAT(INK,S,N(1,II),AK,VD,NC,SUM)
69        .  .  G=TRIPLE(ERHAT,HHAT,N(1,II))
70        .  .  SS=SS+G*SUM
          .  ...FIN
71        .
```

```
72         .  SHADOW = .FALSE.
           .
     C   WE HAVE TO BE LOOKING AT THE FRONT SIDE OF FACE II IN ORDER TO IN-
     C   CLUDE ITS RCS CONTRIBUTION.  THE UNIT VECTOR IS POINTS ALONG THE
     C   DIRECTION OF A RAY REFLECTED FROM FACE II; WHEN WE LOOK ALONG THIS
     C   DIRECTION, WE HAVE TO SEE THE FRONT SIDE OF FACE JJ IN ORDER TO
     C   TALLY A DOUBLE-BOUNCE CONRIBUTION.
           .
73         .  CALL IMAGE(INK,N(1,II),IS)
74         .  Q=VDOT(IS,N(1,JJ))
75         .  IF (Q.GE.0.0) GO TO 300
           .
     C   WE ALSO HAVE TO FIND THE IMAGE OF THE MAGNETIC POLARIZATION.
           .
76         .  CALL IMAGE(HHAT,N(1,II),HHS)
           .
     C   NOW FIND THE PROJECTION OF FACE II ONTO THE PLANE OF FACE JJ.
           .
77         .  NC=4
78         .  NE=4
79         .  J2=JJ
80         .  IF(ONE.LT.0)
81         .  .  DO(J=1,4)
82         .  .  .  DO(I=1,3) VD(I,J)=VF(I,J,II)
84         .  .  ...FIN
85         .  ...FIN
86         .  DO(J=1,4)
88         .  .  DO(I=1,3) VE(I,J)=VF(I,J,J2)
90         .  ...FIN
91         .  PROJECT-FACE
           .
     C   DEPENDING ON WHICH PARTICULAR FACE COMBINATION WE'RE WORKING WITH,
     C   THE SENSE OF ONE OR THE OTHER POLYGON HAS TO BE REVERSED.
           .
94         .  WHEN(ONE.LE.0)
95         .  .  DO (I=1,2)
96         .  .  .  Q=VA(I,2)
97         .  .  .  VA(I,2)=VA(I,4)
98         .  .  .  VA(I,4)=Q
           .  .  ...FIN
99         .  ...FIN
100        .  ELSE
101        .  .  DO (I=1,2)
102        .  .  .  Q=VB(I,2)
103        .  .  .  VB(I,2)=VB(I,4)
104        .  .  .  VB(I,4)=Q
           .  .  ...FIN
105        .  ...FIN
106        .
     C   NOW FIND THE COMMON AREA AND CONVERT THE 2D COODINATES BACK TO 3D.
           .
107        .  NA=4
108        .  FIND-COMMON-REGION
           .
     C   GET THE SCATTERING FROM THE RESULTING POLYGON AND TALLY ALL THE
     C   POLARIZATION COMBINATIONS FOR THE DOUBLE-BOUNCE RETURNS.
```

```
                     .
110                  .  CALL SCAT(IS,S,N(1,JJ),AK,VD,NC,SUM)
111                  .  SUM=-ONE*SUM
112                  .  Q=TRIPLE(ERHAT,HHS,N(1,JJ))
113                  .  SS=SS+Q*SUM
                     .
        C   GET THE IMAGE DIRECTION OF THE REFLECTED DIRECTION OFF THE SECOND
        C   FACE, BUT BE SURE WE'RE LOOKING AT THE FRONT SIDE OF FACE KK WHEN
        C   SEEN ALONG THIS DIRECTION.
                     .
114                  .  CALL IMAGE(IS,N(1,JJ),IS)
115                  .  Q=VDOT(IS,N(1,KK))
116                  .  IF (Q.GE.0.0) GO TO 300
117                  .  CALL IMAGE(HHS,N(1,JJ),HHS)
                     .
        C   FIND THE PROJECTIONS AND CONVERT TO 2D
                     .
118                  .  NA=NC
119                  .  J2=KK
120                  .  DO(J=1,4)
121                  .  .  DO(I=1,3) VE(I,J)=VF(I,J,J2)
123                  .  ...FIN
124                  .  PROJECT-FACE
                     .
        C   IT TURNS OUT WE HAVE TO REVERSE THE SENSE OF VA FOR ALL 6 COMBINA-
        C   TIONS INVOLVING A TRIPLE BOUNCE, BUT VB FOR ONLY 3.
                     .
127 200              .  LL=NC/2+1
129                  .  DO (J=2,LL)
130                  .  .  L=NC-J+2
131                  .  .  DO (I=1,2)
132                  .  .  .  Q=VA(I,L)
133                  .  .  .  VA(I,L)=VA(I,J)
134                  .  .  .  VA(I,J)=Q
                     .  .  ...FIN
135                  .  ...FIN
136                  .  IF (ONE.GT.0.)
138                  .  .  DO (I=1,2)
139                  .  .  .  Q=VB(I,2)
140                  .  .  .  VB(I,2)=VB(I,4)
141                  .  .  .  VB(I,4)=Q
                     .  .  ...FIN
142                  .  ...FIN
143                  .
        C   GET THE DOUBLY ILLUMINATED PATCH ON FACE KK AND CONVERT BACK TO 3D.
                     .
144                  .  FIND-COMMON-REGION
                     .
        C   GET THE TRIPLE-BOUNCE SCATTERING.
                     .
146                  .  CALL SCAT(IS,S,N(1,KK),AK,VD,NC,SUM)
147                  .  SUM=-ONE*SUM
148                  .  Q=TRIPLE(ERHAT,HHS,N(1,KK))
149                  .  SS=SS+Q*SUM
150 300              .  CONTINUE
                     ...FIN
```

```
151
152        RTS = SS

153        RETURN

           ----------------------------------------

154        TO CALCULATE-UNIT-NORMALS
155        .  DO(J=1,3)
156        .  .  DO(I=1,3)
157        .  .  .  A(I)=VF(I,2,J)-VF(I,1,J)
158        .  .  .  B(I)=VF(I,4,J)-VF(I,1,J)
           .  .  ...FIN
159        .  .  CALL VCROSS(A,B,N(1,J))
161        .  .  CALL NORMLZ(N(1,J),N(1,J))
162        .  .  CALL NORMLZ(A,U(1,J))       ! U and V are orthogonal unit vectors
163        .  .  CALL VCROSS(N(1,J),U(1,J),V(1,J)) ! in the plane of the face.
           .  ...FIN
164        ...FIN

           ----------------------------------------

165        TO CALCULATE-UNSHADOWED-REGION
167        .  DO(I=1,3) IS(I)=INK(I)
169        .  DO(J=1,NE)
171        .  .  DO(I=1,3) VE(I,J)=VD(I,J)
173        .  ...FIN
174        .  DO(J=1,4)
176        .  .  DO(I=1,3) VD(I,J)=VF(I,J,I2)
178        .  ...FIN
179        .  PROJECT-FACE
182        .  DO(I=1,2)
183        .  .  TEMP=VA(I,2)      ! Change numbering order sense of polygon A
184        .  .  VA(I,2)=VA(I,4) ! to counterclockwise to remain consistent
185        .  .  VA(I,4)=TEMP      ! with polygon B.
           .  ...FIN
186        .  SHADOW=.TRUE.
188        .  FIND-COMMON-REGION
190        .  IF(YES) GOTO 300      ! Region B is completely shadowed by A.
           ...FIN

           ----------------------------------------

191        TO PROJECT-FACE
           .
      C   Now find the projection of face I2 onto the plane of face J2.
           .
193        .  DO(J=1,NC)
194        .  .  CALL PROJECT(VD(1,J),IS,N(1,J2),VD(1,J))
195        .  .  VA(1,J)=VDOT(VD(1,J),U(1,J2))
196        .  .  VA(2,J)=VDOT(VD(1,J),V(1,J2))
197        .  .  NA=NC
           .  ...FIN
198        .  DO(J=1,NE)
200        .  .  VB(1,J)=VDOT(VE(1,J),U(1,J2))
```

```
00480   201        .  .  VB(2,J)=VDOT(VE(1,J),V(1,J2))
00481   202        .  .  NB=NE
00482              .  ...FIN
00483   203        ...FIN

        ----------------------------------------

00484   204        TO FIND-COMMON-REGION
00485   206        .  CALL PATCHES(NA,4,NC,VA,VB,VC,SHADOW,YES)
00486   207        .  DO(J=1,NC)
00487   208        .  .  DO(I=1,3) VD(I,J)=VC(1,J)*U(I,J2)+VC(2,J)*V(I,J2)
00488   210        .  ...FIN
00489   211        ...FIN
00490   212        END

        ----------------------------------------

                   PROCEDURE CROSS-REFERENCE TABLE

              00438 CALCULATE-UNIT-NORMALS
                    00256

              00450 CALCULATE-UNSHADOWED-REGION
                    00311      00305

              00484 FIND-COMMON-REGION
                    00465      00423      00373

              00468 PROJECT-FACE
                    00458      00399      00350

          (FLECS 77  VERSION 22.38)

      MODULE CONTAINS  NO MINOR ERRORS
      MODULE CONTAINS  NO MAJOR ERRORS
```

```
-----------------------------------------

      C   DIPLANE.FLX        E.F. Knott & R.B. Rakes       11/12/80

      C   ORCHESTRATES THE VARIOUS SUBPROGRAMS USED IN COMPUTING THE RADAR
      C   ECHO FROM AN ARBITRARY DIHEDRAL CORNER REFLECTOR.  THE COORDI-
      C   ATES OF THE VERTICES OF THE THREE FACES COMPRISING THE REFLECTOR
      C   MUST BE READ FROM THE FILE IFACE(2). THIS ROUTINE IS A MUCH SIMPLER
      C   VERSION OF CORNER.FLX WHICH CALCULATES TRIHEDRAL RETURNS.



 1          SUBROUTINE DIPLANE(VG,AK,INK,S,EHAT,RO,VERTICAL,RTS)

 2          INTEGER ONE
 3          REAL INK(3),N(3,3),IS(3),IMS(3)
 4          COMPLEX SS,SUM,RTS,EYE,IK
 5          LOGICAL VERTICAL,SHADOW,YES
 6          DIMENSION S(3),A(3),B(3),EHAT(3),HHAT(3),U(3,2),V(3,2),D(3)
 7          DIMENSION IFACE(5),LIST(6),HHS(3),RO(3)
 8          DIMENSION VA(2,20),VB(2,20),VC(2,20),VF(3,4,3),VD(3,20)
 9          DIMENSION ERHAT(3),VG(3,4,3)

      C     INCLUDE CROSS (COMMON)
10          COMMON /EXTRA/ TWOPI,ROOTPI,EYE,IK,PI,PI04

      C   VA AND VB CONTAIN THE LOCAL X,Y COORDINATES OF THE VERTICES OF A
      C   PAIR OF PLANE POLYGONS WHOSE COMMON AREA IS TO BE FOUND.  THESE
      C   CAN BE PROJECTIONS OF OTHER POINTS ONTO THE PLANE OF A PARTICULAR
      C   TRIHEDRAL FACE. THE ARRAY VC CONTAINS THE 2D COORDINATES OF THE
      C   COMMON AREA.  VF CONTAINS THE X,Y,Z COORDINATES OF THE FOUR VER-
      C   TICES IF THE THREE FACES AND IS READ FROM THE DATA FILE, ALONG
      C   WITH THE COORDINATES OF THE FACE NORMALS N AND UNIT VECTORS U,V.
      C   LINEAR DIMENSIONS ARE ASSUMED GIVEN IN FEET.

      C   Translate points such that the main vertex is now the origin.
11          DO(K=1,2)
12          .  DO(J=1,4)
13          .  .  DO(I=1,3) VF(I,J,K)=VG(I,J,K)-VG(I,1,1)
15          .  ...FIN
16          ...FIN
17
18          CALCULATE-UNIT-NORMALS

20          CALL VCROSS(INK,EHAT,HHAT)
21          WHEN(VERTICAL) CALL VCROSS(S,HHAT,ERHAT)
23          ELSE
24          .  DO(I=1,3) ERHAT(I)=EHAT(I)
26          ...FIN
27

      C   INK AND S ARE THE DIRECTIONS OF INCIDENCE AND SCATTERING, RESPEC-
      C   TIVELY.  EHAT AND HHAT ARE THE UNIT VECTORS ALONG THE INCIDENT ELEC-
      C   TRIC AND MAGNETIC FIELDS RESPECTIVELY.  ERHAT IS ALONG THE RECEIVED
      C   ELECTRIC FIELD.
```

```
 28          SS=CMPLX(0.,0.)
 29          ONE=-1
 30          JJ=2

       C   THERE ARE ONLY 2 SINGLE-BOUNCE CONTRIBUTIONS.

 31          DO(II=1,2)
 32          .  ONE=-ONE
 33          .  G=VDOT(INK,N(1,II))
 34          .  IF(G.GE.0.0) GOTO 300
 35          .  G=VDOT(INK,N(1,JJ))
 36          .  WHEN(G.GT.0.0) CALCULATE-UNSHADOWED-REGION
 39          .  ELSE
 41          .  .  NC=4
 42          .  .  DO (J=1,4)
 43          .  .  .  DO (I=1,3) VD(I,J)=VF(I,J,II)
 45          .  .  ...FIN
 46          .  ...FIN
 47          .  CALL SCAT(INK,S,N(1,II),AK,VD,NC,SUM)
 49          .  G=TRIPLE(ERHAT,HHAT,N(1,II))
 50          .  SS=SS+G*SUM
             .
       C   WE HAVE TO BE LOOKING AT THE FRONT SIDE OF FACE II IN ORDER TO IN-
       C   CLUDE ITS RCS CONTRIBUTION.  THE UNIT VECTOR IS POINTS ALONG THE
       C   DIRECTION OF A RAY REFLECTED FROM FACE II; WHEN WE LOOK ALONG THIS
       C   DIRECTION, WE HAVE TO SEE THE FRONT SIDE OF FACE JJ IN ORDER TO
       C   TALLY A DOUBLE-BOUNCE CONRIBUTION.
             .
 51          .  CALL IMAGE(INK,N(1,II),IS)
 52          .  G=VDOT(IS,N(1,JJ))
 53          .  IF (G.GE.0.0) GO TO 300
             .
       C   WE ALSO HAVE TO FIND THE IMAGE OF THE MAGNETIC POLARIZATION.
             .
 54          .  CALL IMAGE(HHAT,N(1,II),HHS)
             .
 55          .  PROJECT-FACE
             .
       C   DEPENDING ON WHICH PARTICULAR FACE COMBINATION WE'RE WORKING WITH,
       C   THE SENSE OF ONE OR THE OTHER POLYGON HAS TO BE REVERSED.
             .
 57          .  WHEN(ONE.LE.0)
 58          .  .  DO (I=1,2)
 59          .  .  .  G=VA(I,2)
 60          .  .  .  VA(I,2)=VA(I,4)
 61          .  .  .  VA(I,4)=G
             .  .  ...FIN
 62          .  ...FIN
 63          .  ELSE
 64          .  .  DO (I=1,2)
 65          .  .  .  G=VB(I,2)
 66          .  .  .  VB(I,2)=VB(I,4)
 67          .  .  .  VB(I,4)=G
             .  .  ...FIN
 68          .  ...FIN
```

```
 69            .
       C   NOW FIND THE COMMON AREA AND CONVERT THE 2D COODINATES BACK TO 3D.
               .
 70            .   SHADOW=.FALSE.
 71            .   FIND-COMMON-REGION
               .
       C   GET THE SCATTERING FROM THE RESULTING POLYGON AND TALLY ALL THE
       C   POLARIZATION COMBINATIONS FOR THE DOUBLE-BOUNCE RETURNS.
               .
 73            .   CALL SCAT(IS,S,N(1,JJ),AK,VD,NC,SUM)
 74            .   SUM=-ONE*SUM
 75            .   Q=TRIPLE(ERHAT,HHS,N(1,JJ))
 76            .   SS=SS+Q*SUM
 77 300        .   CONTINUE
 78            .   JJ=1
               ...FIN
 79
 80            RTS = EYE

 81            RETURN


       --------------------------------------

 82            TO CALCULATE-UNIT-NORMALS
 83            .   DO(J=1,2)
 84            .   .   DO(I=1,3)
 85            .   .   .   A(I)=VF(I,2,J)-VF(I,1,J)
 86            .   .   .   B(I)=VF(I,4,J)-VF(I,1,J)
               .   .   ...FIN
 87            .   .   CALL VCROSS(A,B,N(1,J))
 89            .   .   CALL NORMLZ(N(1,J),N(1,J))
 90            .   .   CALL NORMLZ(A,U(1,J))          ! U and V are orthogonal unit vectors
 91            .   .   CALL VCROSS(N(1,J),U(1,J),V(1,J)) ! in the plane of the face.
               .   ...FIN
 92            ...FIN

       --------------------------------------

 93            TO CALCULATE-UNSHADOWED-REGION
 95            .   SWAP-II-AND-JJ
 97            .   DO(I=1,3) IS(I)=INK(I)
 99            .   PROJECT-FACE
102            .   DO(I=1,2)
103            .   .   TEMP=VA(I,2)       ! Change numbering order sense of polygon A
104            .   .   VA(I,2)=VA(I,4) ! to counterclockwise to remain consistent
105            .   .   VA(I,4)=TEMP       ! with polygon B.
               .   ...FIN
106            .   SHADOW=.TRUE.
108            .   FIND-COMMON-REGION
110            .   SWAP-II-AND-JJ
112            .   IF(YES) GOTO 300       ! Region B is completely shadowed by A.
               ...FIN

       --------------------------------------
```

```
00647   113        TO SWAP-II-AND-JJ
00648   115        .  ITEMP=II
00649   116        .  II=JJ
00650   117        .  JJ=ITEMP
00651              ...FIN

          ----------------------------------------

00652   118        TO PROJECT-FACE
00653              .
00654          C   NOW FIND THE PROJECTION OF FACE II ONTO THE PLANE OF FACE JJ.
00655              .
00656   120        .  DO (J=1,4)
00657   121        .  .  CALL PROJECT(VF(1,J,II),IS,N(1,JJ),VD(1,J))
00658   122        .  .  VA(1,J)=VDOT(VD(1,J),U(1,JJ))
00659   123        .  .  VA(2,J)=VDOT(VD(1,J),V(1,JJ))
00660   124        .  .  VB(1,J)=VDOT(VF(1,J,JJ),U(1,JJ))
00661   125        .  .  VB(2,J)=VDOT(VF(1,J,JJ),V(1,JJ))
00662              .  ...FIN
00663   126        ...FIN

          ----------------------------------------

00664   127        TO FIND-COMMON-REGION
00665   129        .  CALL PATCHES(4,4,NC,VA,VB,VC,SHADOW,YES)
00666   130        .  DO (J=1,NC)
00667   131        .  .  DO (I=1,3)  VD(I,J)=VC(1,J)*U(I,JJ)+VC(2,J)*V(I,JJ)
00668   133        .  ...FIN
00669   134        ...FIN
00670   135        END

          ----------------------------------------

                   PROCEDURE CROSS-REFERENCE TABLE

                00621 CALCULATE-UNIT-NORMALS
                      00531

                00633 CALCULATE-UNSHADOWED-REGION
                      00556

                00664 FIND-COMMON-REGION
                      00643      00604

                00652 PROJECT-FACE
                      00636      00581

                00647 SWAP-II-AND-JJ
                      00644      00634

          (FLECS 77  VERSION 22.38)

      MODULE CONTAINS  NO MINOR ERRORS
      MODULE CONTAINS  NO MAJOR ERRORS
```

```
      C     SCAT.FLX              11/12/80

      C     SUBROUTINE RETURNS THE COMPLEX SCATTERING FROM AN ARBITRARY
      C     PLANE POLYGON.  IT NEEDS THE DIRECTIONS OF INCIDENCE AND SCAT-
      C     TERING (INC AND S), THE UNIT SURFACE NORMAL N, THE WAVENUMBER
      C     AK (EXPRESSED IN RADIANS PER FOOT), THE 3D COORDINATES OF THE
      C     POLYGON VERTICES VD, AND THE NUMBER OF VERTICES NC.  THE COM-
      C     PLEX SCATTERING AMPLITUDE IS RETURNED IN SUM.

  1          SUBROUTINE SCAT(INK,S,N,AK,VD,NC,SUM)
  2          COMPLEX SUM
  3          REAL N(3),INK(3)
  4          DIMENSION S(3),VD(3,20),P(3),W(3),AM(3),RM(3)
  5          ROOTPI=1.772454
  6          PI=3.141593
  7          SUM=CMPLX(0.0,0.0)
  8          IF ( NC .LE. 2 ) RETURN
  9          DEL=0.0001
 10          MC=NC+1
 11          DO (I=1,3) VD(I,MC)=VD(I,1)
 13          CALL MINUS(S,INK,W)
 15          CALL VCROSS(N,W,P)
 16          Q=VMAG(P)
 17          IF (Q.GT.DEL)
 18          .  T=TRIPLE(P,N,W)
 19          .  DO (J=1,NC)
 20          .  .  JF=J+1
 21          .  .  CALL MINUS(VD(1,J),VD(1,JF),AM)
 22          .  .  T1=.5*AK*VDOT(W,AM)
 23          .  .  WHEN (ABS(T1).LT.DEL) T1=1.0
 25          .  .  ELSE T1=SIN(T1)/T1
 27          .  .  CALL PLUS(VD(1,J),VD(1,JF),RM)
 29          .  .  T2=.5*AK*VDOT(W,RM)
 30          .  .  SUM=SUM+T1*VDOT(P,AM)*CMPLX(COS(T2),SIN(T2))
             .  ...FIN
 31          .  SUM=SUM/(ROOTPI*T)
 33          .  RETURN
             ...FIN
 34
 35          DO (J=3,NC)
 36          .  JF=J-1
 37          .  CALL MINUS(VD(1,1),VD(1,J),AM)
 38          .  CALL MINUS(VD(1,1),VD(1,JF),RM)
 39          .  Q=TRIPLE(AM,RM,N)
 40          .  SUM=SUM-CMPLX(0.0,Q)
             ...FIN
 41          SUM=SUM*AK/(2.*ROOTPI)
 43          RETURN
 44          END

        (FLECS 77  VERSION 22.38)

   MODULE CONTAINS  NO MINOR ERRORS
   MODULE CONTAINS  NO MAJOR ERRORS
```

```
        C   SPOTTER.FLX

        C   CONTAINS A DIFFERENT VERSION OF PATCHES.  THIS VERSION IS A BRUTE
        C   FORCE ASSEMBLY OF VERTICES, SOME OF WHICH HAVE TO BE DELETED BE-
        C   CAUSE SOME "INSIDE" POINTS COINCIDE WITH SOME INTERSECTIONS.

   1           SUBROUTINE PATCHES(NA,NB,NC,VA,VB,VC,SHADOW,YES)
   2           DIMENSION VA(2,20),VB(2,20),VC(2,20),ANG(20)
   3           LOGICAL PAB(20),PBA(20),SHADOW,TEST,YES
   4           DIMENSION A(2),B(2),C(2),EA(2,20),EB(2,20)
   5           DEL=0.00001
   6           PI=3.141592654

        C   FIRST, FORM THE EDGE VECTORS.

   7           DO (J=1,NA)
   8           .  JF=J+1
   9           .  IF (J.EQ.NA) JF=1
  10           .  CALL PMINUS(VA(1,J),VA(1,JF),EA(1,J))
               ...FIN
  11           DO (K=1,NB)
  13           .  KF=K+1
  14           .  IF (K.EQ.NB) KF=1
  15           .  CALL PMINUS(VB(1,K),VB(1,KF),EB(1,K))
               ...FIN
  16
        C   GOTTA ESTABLISH THE PROPER SENSE FOR WALKING AROUND THE POLYGONS,
        C   BECAUSE HALF THE TIME THEY'RE PASSED IN THE REVERSED SENSE.

  17           DO(J=1,NA)
  18           .  J2=J+1
  19           .  IF(J.EQ.NA) J2=1
  20           .  Q=PCROSS(EA(1,J),EA(1,J2))
  21           .  WHEN(J.EQ.1) SENSE=SIGN(1.,Q)
  23           .  ELSE
  24           .  .  SENSE2=SIGN(1.,Q)
  25           .  .  IF(SENSE.NE.SENSE2)
  26           .  .  .  YES=.FALSE.   ! Impossible quadralateral.  Go ahead and return with
  27           .  .  .  NC=NB         ! VB being unshadowed.
  28           .  .  .  DO(K=1,NC)
  29           .  .  .  .  DO(I=1,3) VC(I,K)=VB(I,K)
  31           .  .  .  ...FIN
  32           .  .  .  RETURN
               .  .  ...FIN
  34           .  ...FIN
  35           ...FIN
  36

        C   ESTABLISH WHETHER THE VERTICES OF VA ARE INSIDE OR OUTSIDE VB.

  37           DO (J=1,NA)
  38           .  PAB(J)=.FALSE.
  39           .  DO(K=1,NB)
```

```
40             .  .  CALL PMINUS(VB(1,K),VA(1,J),C)
41             .  .  Q=SENSE*(EB(2,K)*C(1)-EB(1,K)*C(2))
42             .  .  IF(Q.GT.0.) PAB(J)=.TRUE.
               .  ...FIN
43             ...FIN
44
         C   AND NOW DO THE SAME FOR VERTICES OF VB IN VA.

45             DO (K=1,NB)
46             .  PBA(K)=.FALSE.
47             .  DO(J=1,NA)
48             .  .  CALL PMINUS(VA(1,J),VB(1,K),C)
49             .  .  Q=SENSE*(EA(2,J)*C(1)-EA(1,J)*C(2))
50             .  .  IF(Q.GT.0.) PBA(K)=.TRUE.
               .  ...FIN
51             ...FIN
52
53             YES=.TRUE.
54             DO(K=1,NB)
55             .  IF(PBA(K)) YES=.FALSE.
               ...FIN
56             IF(YES.AND.SHADOW) RETURN        ! Scatterer is completely shadowed

         C   ALL INSIDE VERTICES BECOME VERTICES OF THE COMMON AREA VC.

58             L=0
59             DO (J=1,NA)
60             .  UNLESS(PAB(J))
61             .  .  L=L+1
62             .  .  DO (I=1,2) VC(I,L)=VA(I,J)
64             .  ...FIN
65             ...FIN
66             DO (K=1,NB)
68             .  WHEN(SHADOW) TEST=PBA(K)
70             .  ELSE TEST=.NOT.PBA(K)
72             .  IF(TEST)
74             .  .  L=L+1
75             .  .  DO (I=1,2) VC(I,L)=VB(I,K)
77             .  ...FIN
78             ...FIN
79
         C   ALL INTERSECTIONS ARE ALSO VERTICES OF THE COMMON AREA.

80             DO (J=1,NA)
81             .  DO (K=1,NB)
82             .  .  CALL PMINUS(VA(1,J),VB(1,K),C)
83             .  .  T1=PCROSS(EA(1,J),EB(1,K))
84             .  .  T2=PCROSS(C,EA(1,J))
85             .  .  T3=PCROSS(C,EB(1,K))
86             .  .  IF (ABS(T1).NE.0.0)
87             .  .  .  P=T2/T1
88             .  .  .  Q=T3/T1
89             .  .  .  UNLESS(P.LT.0.0.OR.P.GT.1.0.OR.Q.LT.0.0.OR.Q.GT.1.0)
90             .  .  .  .  L=L+1
91             .  .  .  .  DO (I=1,2) VC(I,L)=VA(I,J)+Q*EA(I,J)
93             .  .  .  ...FIN
```

```
 94         .  .  ...FIN
 95         .  ...FIN
 96         ...FIN
 97
 98         NC=L

      C   BUT THERE WILL BE DUPLICATIONS, SO LET'S WEED 'EM OUT.

 99         L=0
100         DO (J=1,NC)
101         .  L=L+1
102         .  DO (I=1,2) VC(I,L)=VC(I,J)
104         .  DO (K=J,NC)
106         .  .  IF (J.NE.K)
107         .  .  .  CALL PMINUS(VC(1,K),VC(1,J),C)
108         .  .  .  Q=PSIZE(C)
109         .  .  .  IF (Q.LE.DEL)
110         .  .  .  .  L=L-1
111         .  .  .  .  GO TO 150
            .  .  .  ...FIN
112         .  .  ...FIN
113         .  ...FIN
114 150     .  CONTINUE
            ...FIN
116         NC=L

      C   THE GET THE SHORTENED LIST OF VERTICES IN THE PROPER ORDER, FIRST
      C   FIND THE CENTROID OF THE COMMON AREA.

118         C(1)=0.0
119         C(2)=0.0
120         DO (J=1,NC)
121         .  C(1)=C(1)+VC(1,J)
122         .  C(2)=C(2)+VC(2,J)
            ...FIN
123         UNLESS(NC.LE.0)
125         .  C(1)=C(1)/NC
126         .  C(2)=C(2)/NC
            ...FIN
127
      C   NOW CALCULATE THE ANGLE SUBTENDED BY THE VECTOR BETWEEN A GIVEN
      C   VERTEX AND THE CENTROID AND THE VECTOR BETWEEN THE LAST VERTEX AND
      C   THE CENTROID.

128         CALL PMINUS(VC(1,NC),C,A)
129         DO (L=1,NC)
130         .  CALL PMINUS(VC(1,L),C,B)
131         .  P=PCROSS(A,B)
132         .  Q=A(1)*B(1)+A(2)*B(2)
133         .  WHEN(Q.EQ.0.0)
            .  .  CONDITIONAL
134         .  .  .  (P.LT.0.) ANG(L)=-0.5*SENSE*PI
136         .  .  .  (P.EQ.0.) ANG(L)=0.0
138         .  .  .  (P.GT.0.) ANG(L)=0.5*SENSE*PI
            .  .  ...FIN
140         .  ...FIN
```

```
141         .  ELSE ANG(L)=SENSE*ATAN2(P,G)
143         ...FIN
144
        C   FINALLY, ALL WE HAVE TO DO IS USE THE ANGLES TO REARRANGE THE VER-
        C   TICES IN THE PROPER ORDER.

145         DO (J=1,NC)
146         .  DO (K=J,NC)
147         .  .  IF (K.NE.J)
148         .  .  .  IF (ANG(J).GT.ANG(K))
149         .  .  .  .  DO (I=1,2)
150         .  .  .  .  .  G=VC(I,J)
151         .  .  .  .  .  VC(I,J)=VC(I,K)
152         .  .  .  .  .  VC(I,K)=G
            .  .  .  .  ...FIN
153         .  .  .  .  G=ANG(J)
155         .  .  .  .  ANG(J)=ANG(K)
156         .  .  .  .  ANG(K)=G
            .  .  .  ...FIN
157         .  .  ...FIN
158         .  ...FIN
159         ...FIN
160
        C   AND RETURN VC TO THE CALLING PROGRAM.

161         RETURN
162         END

        (FLECS 77  VERSION 22.38)

   MODULE CONTAINS  NO MINOR ERRORS
   MODULE CONTAINS  NO MAJOR ERRORS
```

```
-----------------------------------------
      C      SHRIMRAT STRADDLE Point-in-polygon routine.
      C      CALLING PROGRAM SENDS THE X-Y COORDINATES OF THE POINT TO BE
      C      TESTED. SUBROUTINE RETURNS 1 IF POINT IS INSIDE POLYGON, -1
      C      IF POINT IS A VERTEX OF THE POLYGON (AND THUS ON THE BOUNDARY),
      C      AND 0 IF OUTSIDE THE POLYGON.
      C      CALLING PROGRAM MUST CLOSE POLYGON BY SETTING (NPTS+1)st VALUE TO
      C      THE PARAMETER M = NPTS+1, WHERE NPTS IS THE # NODES IN THE POLYGO
      C
 1          SUBROUTINE PIP (XO, YO, RESULT, POLY, NN)
 2          INTEGER RESULT, M, NPTS, K, J
 3          REAL X(21), Y(21), XO, YO, LAMBDA, XX(21), YY(21), POLY(2,20)
 4          REAL RHO, DELTA

 5          RESULT=1

 6          M=NN+1
 7          DO(J=1,M)
 8          .  X(J)=POLY(1,J)
 9          .  Y(J)=POLY(2,J)
            ...FIN
10          X(M)=POLY(1,1)
12          Y(M)=POLY(2,1)

      C      TRANSLATE VERTICES OF THE POLYGON SO (XO, YO) IS LOCATED AT
      C      THE ORIGIN. IF ANY TRANSLATED VERTEX IS ZERO, THE POINT IN
      C      QUESTION IS A VERTEX.

13          NPTS=M-1
14          DO 10 I=1,M
15          XX(I)=X(I)-XO
16          YY(I)=Y(I)-YO
17          IF (XX(I)) 10,9,10
18 9        IF (YY(I)) 10,70,10
19 10       CONTINUE

20          K=0
      C      INITIALIZE INTERSECTION COUNT

21          DO 50 J=1,NPTS
22          IF (YY(J)) 11,30,12
23 11       IF (YY(J+1)) 50,31,13
24 12       IF (YY(J+1)) 13,31,50
25 13       IF (XX(J)) 14,31,15
26 14       IF (XX(J+1)) 50,50,18
27 15       IF (XX(J+1)) 18,45,45
28 18       DELTA=YY(J)-YY(J+1)
29          IF (DELTA) 16,50,16
30 16       LAMDA=-YY(J+1)/DELTA
31          RHO=(YY(J)*XX(J+1)-XX(J)*YY(J+1))/DELTA
32          IF (RHO) 50,20,20
33 20       IF (LAMDA) 50,25,25
34 25       IF (LAMDA-1.) 45,50,50
```

```
      C         HANDLE SPECIAL CASES
 35 30     IF (XX(J)) 50,60,60
 36 31     IF (XX(J+1)) 50,60,60
 37 45     K=K+1
 38 50     CONTINUE

 39        IF (MOD(K,2)) 65,60,65
 40 60     RESULT=0
 41 65     RETURN
 42 70     RESULT=-1
 43        RETURN

 44        END

          (FLECS 77  VERSION 22.38)

   MODULE CONTAINS  NO MINOR ERRORS
   MODULE CONTAINS  NO MAJOR ERRORS
```

```
----------------------------------------

          C     IMAGE.FLX

    1             SUBROUTINE IMAGE(A,N,VECT)
    2             DIMENSION A(3),VECT(3)
    3             REAL N(3)
    4             D=2.0*VDOT(N,A)
    5             DO (I=1,3) VECT(I)=A(I)-D*N(I)
    7             RETURN
    9             END

          (FLECS 77  VERSION 22.38)

      MODULE CONTAINS  NO MINOR ERRORS
      MODULE CONTAINS  NO MAJOR ERRORS
```

```
-----------------------------------------
          C     PROJECT.FLX

    1             SUBROUTINE PROJECT(A,B,N,PROJ)
    2             REAL N(3)
    3             DIMENSION A(3),B(3),C(3),D(3),PROJ(3)
    4             CALL VCROSS(A,B,C)
    5             CALL VCROSS(N,C,D)
    6             P=VDOT(N,B)
    7             DO (I=1,3) PROJ(I)=D(I)/P
    9             RETURN
   11             END

          (FLECS 77  VERSION 22.38)

      MODULE CONTAINS  NO MINOR ERRORS
      MODULE CONTAINS  NO MAJOR ERRORS
```

```
--------------------------------------

C       TRIPLE.FLX

        FUNCTION TRIPLE(A,B,C)
        DIMENSION A(3),B(3),C(3),D(3)
        CALL VCROSS(A,B,D)
        TRIPLE=VDOT(C,D)
        RETURN
        END

    (FLECS 77  VERSION 22.38)

MODULE CONTAINS  NO MINOR ERRORS
MODULE CONTAINS  NO MAJOR ERRORS
```

```
      C       PLUS.FLX

  1         SUBROUTINE PLUS(A,B,C)
  2         DIMENSION A(3),B(3),C(3)
  3         DO (I=1,3) C(I)=A(I)+B(I)
  5         RETURN
  7         END

      (FLECS 77  VERSION 22.38)

MODULE CONTAINS  NO MINOR ERRORS
MODULE CONTAINS  NO MAJOR ERRORS
```

```
--------------------------------------

     C       MINUS.FLX

1            SUBROUTINE MINUS(A,B,C)
2            DIMENSION A(3),B(3),C(3)
3            DO (I=1,3) C(I)=B(I)-A(I)
5            RETURN
7            END

        (FLECS 77  VERSION 22.38)

   MODULE CONTAINS  NO MINOR ERRORS
   MODULE CONTAINS  NO MAJOR ERRORS
```

```
      ----------------------------------------

      C     PCROSS.FLX

  1          FUNCTION PCROSS(A,B)
  2          DIMENSION A(2),B(2)
  3          PCROSS=A(1)*B(2)-A(2)*B(1)
  4          RETURN
  5          END

      (FLECS 77  VERSION 22.38)

 MODULE CONTAINS  NO MINOR ERRORS
 MODULE CONTAINS  NO MAJOR ERRORS
```

```
-----------------------------------------

      C       PMINUS.FLX

  1           SUBROUTINE PMINUS(A,B,C)
  2           DIMENSION A(2),B(2),C(2)
  3           DO (I=1,2) C(I)=B(I)-A(I)
  5           RETURN
  7           END

        (FLECS 77  VERSION 22.38)

     MODULE CONTAINS  NO MINOR ERRORS
     MODULE CONTAINS  NO MAJOR ERRORS
```

```
    --------------------------------------

    C       PSIZE.FLX

1           FUNCTION PSIZE(A)
2           DIMENSION A(2)
3           PSIZE=SQRT(A(1)*A(1)+A(2)*A(2))
4           RETURN
5           END

     (FLECS 77  VERSION 22.38)

MODULE CONTAINS  NO MINOR ERRORS
MODULE CONTAINS  NO MAJOR ERRORS
```

```
--------------------------------------

1         BLOCK DATA
2         COMPLEX EYE, IK
3         REAL NTM
4         COMMON/EXTRA/TWOPI, ROOTPI, EYE, IK, PI, PIO4
5         COMMON/PHASE/RDD, RDI, RII
6         COMMON/DATUM/ CNVT, FTM, NTM, PIO2
7         DATA CNVT, FTM, NTM/. 01745329, . 3048, 1851. 965/
8         DATA PI, TWOPI, ROOTPI, PIO4, EYE/3. 141594, 6. 283185, 1. 772454, . 785398,
         &(0. , 1. )/, PIO2/1. 570796327/
9         END

            (FLECS 77  VERSION 22. 38)

      MODULE CONTAINS  NO MINOR ERRORS
      MODULE CONTAINS  NO MAJOR ERRORS
```

```
        ---------------------------------------

            C         VECTOR LIBRARY

       1              SUBROUTINE MATMU(A,B,C,L,M,N)
       2              DIMENSION A(L,M),B(M,N),C(L,N)
       3    C         MATRIX MULTIPLICATION
       4              DO(I=1,L)
       5              .  DO(J=1,N)
       6              .  .  DO(K=1,M)
       7              .  .  .  WHEN(K.EQ.1) C(I,J)=A(I,K)*B(K,J)
       9              .  .  .  ELSE C(I,J)=C(I,J)+A(I,K)*B(K,J)
      11              .  .  ...FIN
      12              .  ...FIN
      13              ...FIN
      14              RETURN
      16              END

            (FLECS 77  VERSION 22.38)

        MODULE CONTAINS  NO MINOR ERRORS
        MODULE CONTAINS  NO MAJOR ERRORS
```

```
          ------------------------------------------------------------------

      SUBROUTINE VCROSS(A,B,C)
      DIMENSION A(3),B(3),C(3)
C     C = A X B
C     VECTOR OR CROSS PRODUCT
      C(1)=A(2)*B(3)-A(3)*B(2)
      C(2)=A(3)*B(1)-A(1)*B(3)
      C(3)=A(1)*B(2)-A(2)*B(1)
      RETURN
      END

          (FLECS 77  VERSION 22.38)

     MODULE CONTAINS  NO MINOR ERRORS
     MODULE CONTAINS  NO MAJOR ERRORS
```

```
---------------------------------------

      FUNCTION VDOT(A,B)
      DIMENSION A(3),B(3)
      VDOT=A(1)*B(1)+A(2)*B(2)+A(3)*B(3)
      RETURN
      END

  (FLECS 77  VERSION 22.38)

MODULE CONTAINS  NO MINOR ERRORS
MODULE CONTAINS  NO MAJOR ERRORS
```

```
---------------------------------------

      SUBROUTINE NORMLZ(A,B)
      DIMENSION A(3),B(3)
      D=VMAG(A)
      B(1)=A(1)/D
      B(2)=A(2)/D
      B(3)=A(3)/D
      RETURN
      END

 (FLECS 77  VERSION 22.38)

MODULE CONTAINS  NO MINOR ERRORS
MODULE CONTAINS  NO MAJOR ERRORS
```

```
      FUNCTION VMAG(A)
      DIMENSION A(3)
      VMAG=SQRT(A(1)*A(1)+A(2)*A(2)+A(3)*A(3))
      RETURN
      END
```

(FLECS 77 VERSION 22.38)

MODULE CONTAINS NO MINOR ERRORS
MODULE CONTAINS NO MAJOR ERRORS

TECHNICAL REPORT
PROJECT NO. A–2986

RADAR GLINT MODEL METHODOLOGY

By

R. B. Rakes and M. M. Horst

Prepared for

ROCKWELL INTERNATIONAL
MISSILE SYSTEMS DIVISION
COLUMBUS, OHIO 43216

May 1982

GEORGIA INSTITUTE OF TECHNOLOGY

A Unit of the University System of Georgia
Engineering Experiment Station
Atlanta, Georgia 30332

Technical Report

Georgia Tech/EES Project A-2986

RADAR GLINT MODEL METHODOLOGY

By

R. B. Rakes and M. M. Horst

Prepared for
Rockwell International
Missile Systems Division
Columbus, Ohio 43216

under Agreement V161-SA-113203

Prepared by

GEORGIA INSTITUTE OF TECHNOLOGY
Engineering Experiment Station
Atlanta, Georgia 30332

May 1982

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SECTION 1
INTRODUCTION

The Georgia Tech Radar Glint model is a set of subroutines to predict seeker track errors due to radar glint from vehicular targets. The model is intended for use as a "plug-in" module into existing missile seeker programs, although it may be used in a stand-alone mode. This Radar Glint model, written in FLECS (an extended version of Fortran 77), was developed entirely at Georgia Tech and implemented on a VAX 11/780 computer. A working version of the model is also available for a SYSTEMS 32/7780 computer. The purpose of this technical manual is to provide the basic equations and algorithms used by the Radar Glint model. For information on implementing this model as well as a complete program listing, refer to the Software Documentation manual titled "Radar Glint Model" [1].

The Radar Glint model supports a missile model by simulating a monopulse tracking radar employed on the missile. The primary outputs of the model are the azimuth (yaw) and elevation (pitch) track errors which are used as inputs to the guidance portion of the missile simulation. One of the main advantages of this program is that the target of the missile is modeled as an extended target (i.e., as a large collection of individual scatterers, for the calculation of the radar cross section) as opposed to a single point target with a constant cross section. This allows for the effects of glint to be modeled more accurately. Another advantage of this computer model is that the effects of multipath and clutter from the ground plane are also included. As an added bonus, the program returns the location of the missile's impact point on the target.

The Radar Glint model was developed from two earlier Georgia Tech computer programs: The multipath/clutter routines for the TAC/ZINGER program [2] and the radar cross section (RCS) model known as CROSS [3]. The multipath/clutter model, described in Section 2, produces track errors for a monopulse radar, but is restricted to a single point target. The CROSS model, described in Section 3, was originally designed to calculate the RCS of ships, but has since been adapted to find the near field RCS of other three dimensional targets, including land as well as sea targets.

SECTION 2
TRACK ERROR MODEL

For a typical monopulse system with four equal beams, the track errors in elevation (ε_e) and azimuth (ε_a) are proportional to the difference channel divided by the sum channel:

$$\varepsilon_a = P_a \frac{D_a}{S} \quad (1)$$

$$\varepsilon_a = P_e \frac{D_e}{S} \quad (2)$$

where the subscripts denote azimuth and elevation. The proportionality constants, P_e and P_a, can be found by noting that, for a single target and small errors, the error signal should be exactly the negative of the off-axis angle [2].

Denoting the one-way voltage pattern by $f(\theta_e, \theta_a)$, where θ_e and θ_a are the angles off boresight in elevation and azimuth, the sum and difference signals can be expressed in terms of the four voltage patterns:

$$\begin{aligned} S = {} & f(\theta_e + \beta_e \theta_q, \theta_a - \beta_a \theta_q) + f(\theta_e - \beta_e \theta_q, \theta_a - \beta_a \theta_q) \\ & + f(\theta_e + \beta_e \theta_q, \theta_a + \beta_a \theta_q) + f(\theta_e - \beta_e \theta_q, \theta_a + \beta_a \theta_q) \end{aligned} \quad (3)$$

$$\begin{aligned} D_e = {} & -f(\theta_e + \beta_e \theta_q, \theta_a - \beta_a \theta_q) + f(\theta_e - \beta_e \theta_q, \theta_a - \beta_a \theta_q) \\ & - f(\theta_e + \beta_e \theta_q, \theta_a + \beta_a \theta_q) - f(\theta_e - \beta_e \theta_q, \theta_a + \beta_a \theta_q) \end{aligned} \quad (4)$$

$$\begin{aligned} D_a = {} & f(\theta_e + \beta_e \theta_q, \theta_a + \beta_a \theta_q) + f(\theta_e - \beta_e \theta_q, \theta_a - \beta_a \theta_q) \\ & - f(\theta_e + \beta_e \theta_q, \theta_a + \beta_a \theta_q) - f(\theta_e - \beta_e \theta_q, \theta_a + \beta_a \theta_q) \end{aligned} \quad (5)$$

where $\beta_i \theta_q$ is the squint angle (i.e., the angle between the center of each beam and each boresight axis), and θ_q was taken to be equal to 0.3 for each beam.

For the present application, the antenna pattern $f(\theta_e,\theta_a)$ for each beam was taken to be:

$$f(\theta_e,\theta_a) = \left(\frac{3}{2+\Delta}\right)\left[\frac{\Delta \sin u}{u} + 2(1-\Delta)\left(\frac{\sin u}{u^3} - \frac{\cos u}{u^2}\right)\right]$$

$$\Delta = 0.1818 \qquad\qquad (6)$$

$$u = 2.7831\sqrt{\left(\frac{\theta_e}{\theta_e}\right)^2 + \left(\frac{\theta_a}{\theta_a}\right)^2}$$

β_e, β_a are the 3 dB beamwidths.

For a single point target, the expressions for azimuth and elevation error are:

$$\varepsilon = P \cdot \mathrm{Re}\left[\frac{D\, S^T \sqrt{\sigma_T} + D_c \sqrt{\sigma_c}}{S^R S^T \sqrt{\sigma_T} + S_c \sqrt{\sigma_c}}\right] \qquad (7)$$

where the subscripts a and e for azimuth and elevation have been omitted, and Re means the real part of the complex quantity in brackets. The superscripts R and T denote received and transmitted one-way voltage patterns:

D = Difference pattern for received signal

$$= D_D + \sum_i D_i \rho_i$$

S^T = Sum pattern for transmitted signal

$$= S_D^T + \sum_i S_i^T \rho_i$$

D_c = Difference pattern for clutter signal

S^R = Sum pattern for received signal

$$= S_D^R + \sum_i S_i^R \rho_i$$

S_c = Sum pattern for clutter signal

D_D, S_D^T, S_R^R = Direct path pattern

D_i, S_i^T, S_i^R = Indirect pattern for i^{th} facet

$\sqrt{\sigma_T}$ = Free space scattering length of the point target

$\sqrt{\sigma_c}$ = Clutter scattering length

$\sum_i$ = Sum over ground facets between target and radar

ρ_i = Combination of specular and diffuse reflection coefficients for i^{th} facet.

Table 1 presents the various signals to be included in the summation in order to compute the angular errors. The signal amplitudes, phases, and directions are identified by type in the table.

Note that to generate ε_e and ε_a as functions of time, the diffuse and clutter terms are random variables and the expressions given in the multipath and clutter sections above give the rms value of the ρ_{di} and the average of σ_c. Instantaneous values of these variables are generated for use in the above expressions using bivariate Gaussian distributions having standard deviations ρ_{di} (the diffuse reflection coefficient from facet i) and $\sqrt{\sigma_c}$. This generator produces the correct amplitude and phase distributions for the radar variables.

Equation 7 above represents the track errors in azimuth and elevation for a single point target. For multiple spatially separated point targets, $DS^T \sqrt{\sigma_T}$ and $S^R S^T \sqrt{\sigma_T}$ in Equation (7) are replaced by the phasor sums of similar terms for each point target.

The difficulty arises when the target is not a point target or a collection of point targets, but rather is an extended target or a collection of extended targets. The extension from one to many targets carried over, but the relatively simple form of Equation 7 is further complicated by the replacement of the free space scattering length of the target, $\sqrt{\sigma_T}$, by the combination of the free space, image, and the diplane scattering lengths,

TABLE 1. THE VARIOUS SIGNALS ARRIVING AT THE ANTENNA

PATH	REFLECTION COEFFICIENT (ρ_i)		ANGLE ERROR DIRECTION	
	Magnitude	Phase	Elevation	Azimuth
1. Direct	1	reference	$\theta_{te} - \theta_{se}$	$\theta_{ta} - \theta_{sa}$
2. Diffuse	ρ_{di}	uniform	$-(\theta_{se} + \theta_{1i})$	$\theta_{ta} - \theta_{sa} + \eta_i$
3. Specular	ρ	$\kappa\delta_o$	$-(\theta_{se} + \theta_{1i})$	$\theta_{ta} - \theta_{sa} + \eta_i$
4. Clutter	1	uniform	$-(\theta_{se} - \bar{\theta}_1)$	η_c

where

θ_{se}, θ_{sa} = radar main axis elevation and azimuth angles, respectively

θ_{te}, θ_{ta} = target elevation and azimuth angle

θ_{1i} = depression angle to facet 1

$\bar{\theta}_1$ = average depression angle to clutter cell

η_i = apparent azimuthal location of i^{th} scattering center [2] with each being multiplied by the appropriate reflection coefficients, and all cross terms being carried along. Specifically, for a single extended target, such as one triangular flat plate element:

$$DS^T \sqrt{\sigma_T} \rightarrow D_D S_D^T \sqrt{\sigma_f} + \sum_i \rho_i \sqrt{\sigma_{di}} \; (D_D S_i^T + S_D D_i^T) + \sum_{k,\ell} D_k S_\ell^T \rho_k \rho_\ell \sqrt{\sigma_{k\ell}} \tag{8}$$

$$S^R S^T \sqrt{\sigma_T} \rightarrow S_D^R S_D^T \sqrt{\sigma_f} + \sum_i \rho_i \sqrt{\sigma_{di}} \; (S_D^T S_i^R + S_i^T S_D^R) + \sum_{k,\ell} S_k^R S_\ell^T \rho_k \rho_\ell \sqrt{\sigma_{k\ell}} \tag{9}$$

Where $\sqrt{\sigma_f}$ = free space scattering length of element

$\sqrt{\sigma_{di}}$ = diplane scattering length of element for i^{th} indirect path

$\sqrt{\sigma_{k\ell}}$ = image scattering length of element for indirect paths k and ℓ.

Figure 1 illustrates the multipath geometry, showing the four ways in which the signal can travel from the radar to the target and back, resulting in three terms for target scattering length. The diplane scattering length is assumed to be the same for the direct-indirect and the indirect-direct paths.

Equations 8 and 9 apply in the case of a single extended scatterer. For a large extended target described as a collection of scatterers, each of the terms in Equations 8 and 9 must be calculated for each scatterer and the phasor sum must be computed. The angle error is then found as the real part of the complex sum.

Although a bit cumbersome, formulations such as Equations 8 and 9 are nonetheless straightforward to code on a digital computer. However, the number of calculations to be performed by the computer rapidly escalates to unmanageable proportions. For diffuse reflection with n_g ground reflection points, the total number of possible paths n_t is $(n_g + 1)^2$. With tank target files numbering thousands of scatterers, the program cannot afford the computation time to calculate each term for every scatterer and ten to twenty ground reflection points for the diffuse reflection.

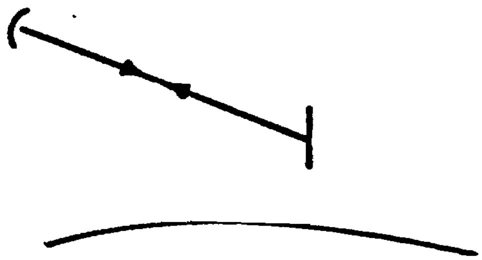
Therefore the following approximations were made to the exact equations:

1. Restrict the number of ground facets to ten between missile starting point and target. Then, as the missile crosses over a new facet on its way toward the target, eliminate that facet from the calculations.

2. Rather than calculating indirect incidence angles to each scatterer on the target for each of the ten facets, select ten scatterer heights and precalculate indirect incidence angles for the array of ten ground facets and ten scatterer heights. Then use the indirect incidence angle for the height closest to the height of the center of each scatterer in RCS calculations. Figure 2 illustrates some of the possible paths described by this array.

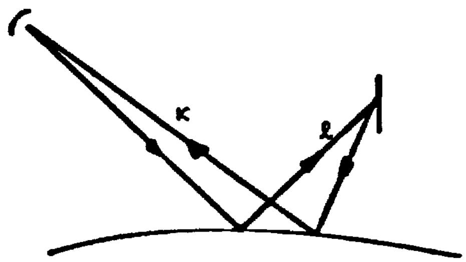
3. Pre-calculate the specular and diffuse reflection coefficient for each combination of ground facet and scatterer height, and store in the array with the indirect incidence angles. This array is updated every time the missile crosses over a new ground facet.

4. Re-calculate scatterer RCS every N steps of the missile fly-out model, where N is the number of facets currently between the missile and the target.



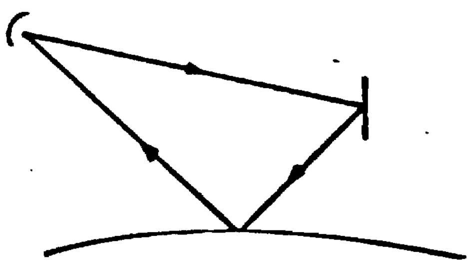
a. **Direct-Direct Free Space Path**

$X_D S_D^T \sqrt{\sigma_f}$



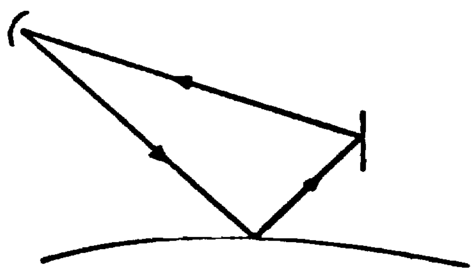
b. **Indirect-Indirect Image Paths κ and ℓ**

$X_\kappa S_\ell^T \rho_\kappa \rho_\ell \sqrt{\sigma_{\kappa\ell}}$



c. **Direct-Indirect Diplane Path for Facet i**

$\rho_i \sqrt{\sigma_{di}}\ S_D^T X_i$



d. **Indirect-Direct Diplane Path For Facet i**

$\rho_i \sqrt{\sigma_{di}}\ S_i^T X_D$

Figure 1. **Multipath geometry, showing the origin of the terms in Equations** 8 **and** 9**, where X is either D or S^R.**

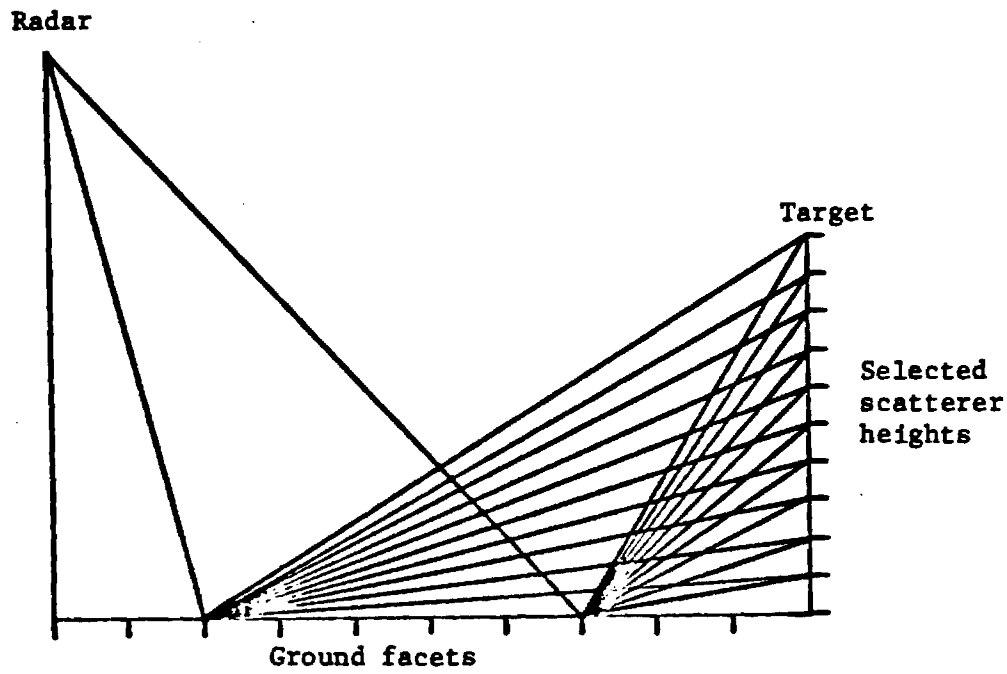


Figure 2. **Some of the possible paths. For n_g ground facets and n_h scatterer heights, an ($n_g \times n_h$) array containing specular and diffuse reflection coefficients and local incidence angles for all paths is pre-calculated once for every update of missile position, and the results stored for access by the scatterer RCS and multipath routines.**

SECTION 3
CROSS COMPUTER MODEL

CROSS (Coherent Radar cross sections Of Surface Ships) is a computer program developed and implemented at Georgia Tech. This model was developed in particular to calculate the effective radar cross section (RCS) of ships, but is general enough to handle most extended targets in a wide variety of scenarios. CROSS uses many of the features developed in its predecessor CROW (Coherent Radar cross section Over Water) [4], which was used to calculate the RCS of submarine masts and other small targets on or above the sea surface. The program CROSS has the advantage over CROW in that it can handle a complete three-dimensional target stored in one data file rather than a separate data file for each aspect of the target. This is because CROSS has a sophisticated hidden surface algorithm not found in CROW.

CROSS finds the total RCS of a target by calculating the RCS return for each individual scatterer and adding the results either coherently or incoherently. The program allows the target to move with respect to the radar in any straight line path on the earth's surface. The target can be incremented through any combination of yaw, pitch, and roll in any order. The model takes into account the curvature of the earth, multipath from the sea surface, and variable sea states. The types of scatterer it can presently handle are: triangular flat plates, truncated cone frusta, ellipsoids, dihedrals, and trihedrals.

There are two possible ways in which the RCS returns from the individual scatterers can be summed: coherently or incoherently. The CROSS program can handle either method depending on the user's preference. A coherent summation is one in which the phase angles of the individual scatterers are retained. It is accomplished by the program as

$$\sigma = \left| \sum_{i=1}^{n} W_i \sqrt{\sigma_i} \right|^2 \qquad (10)$$

where σ is the total cross section, n is the number of scatterers, $\sqrt{\sigma_i}$ is the complex scattering length for an individual scatterer, and W_i is a weighting factor for the scatterer. Since $\sqrt{\sigma_i}$ is computed as a complex number, the phase is automatically included. The weighting factor W_i is a real number between zero and one that allows for the handling of imperfectly reflecting surfaces such as dielectrics or radar absorbent material (RAM). If an incoherent summation is desired, the expression for σ is

$$\sigma = \sum_{i=1}^{n} |W_i \sqrt{\sigma_i}|^2 \tag{11}$$

where the square of the modulus is found before the addition as opposed to the coherent summation where the modulus is not found until after the sum is completed.

3.1. PHYSICAL OPTICS EQUATIONS

The basic physical optics formula used in deriving the bistatic scattering equation is

$$\sqrt{\sigma} = \frac{\pm ik}{\sqrt{\pi}} \exp[ik\vec{r}_o \cdot (\hat{i} - \hat{s})]\ I$$
$$I = \int_A \hat{n} \cdot \hat{h}_i \times \hat{e}_r \exp[ik\vec{r} \cdot (\hat{i} - \hat{s})]da \tag{12}$$

where $k = 2\pi/\lambda$ is the free space wave number of the incident wave, $\hat{i}$ and $\hat{s}$ are unit vectors aligned along the directions of propagation of the incident and scattered waves, $\hat{h}_i$ is the incident magnetic field unit vector, and $\hat{e}_r$ is the reflected electric field unit vector. The position vector $\vec{r}_o$ runs from some fixed origin to an auxiliary origin in or on the scatterer (such as the midpoint of a frustum or flat plate), and the position vector $\vec{r}$ is attached to the auxiliary origin and sweeps over the surface of integration A. The unit vector $\hat{n}$ is an outward normal erected on the surface element da and the integration is performed only over the scatterer surfaces that are illuminated by the incident plane wave. Only $\hat{n}$ and $\vec{r}$ vary over this surface, with $\hat{h}_i \times \hat{e}_r$ and $(\hat{i} - \hat{s})$ being independent of the variable of integration. The phase convention $e^{i\omega t}$ is assumed throughout.

3.1.1 FLAT PLATES

The above integral has a closed form analytic solution for the simple geometric shapes implemented as scatterer types in CROSS. For a triangular flat plate, the result is

$$\sqrt{\sigma} = \frac{\hat{n} \cdot \hat{e}_r \times \hat{h}_i}{\sqrt{\pi}\, T} e^{(ik\vec{r}_o \cdot \vec{w})} \left\{ \hat{p} \cdot \vec{a}\, e^{(ik\vec{r}_a \cdot \vec{w})} \left[\frac{\sin(\frac{1}{2} k\vec{a} \cdot \vec{w})}{\frac{1}{2} k\vec{a} \cdot \vec{w}} \right] + \hat{p} \cdot \vec{b}\, e^{(ik\vec{r}_b \cdot \vec{w})} \left[\frac{\sin(\frac{1}{2} k\vec{b} \cdot \vec{w})}{\frac{1}{2} k\vec{b} \cdot \vec{w}} \right] + \hat{p} \cdot \vec{c}\, e^{(ik\vec{r}_c \cdot \vec{w})} \left[\frac{\sin(\frac{1}{2} k\vec{c} \cdot \vec{w})}{\frac{1}{2} k\vec{c} \cdot \vec{w}} \right] \right\} \tag{13}$$

where $\vec{w} = \hat{i} - \hat{s}$ (14)

$$\hat{p} = \frac{\hat{n} \times \vec{w}}{|\hat{n} \times \vec{w}|} \tag{15}$$

$$T = |\hat{n} \times \vec{w}| \tag{16}$$

The vector $\hat{n}$ represents the unit normal to the plate surface. Its direction is defined by a right-handed (counterclockwise) ordering of the vertices. Thus,

$$\hat{n} = \frac{-\hat{a} \times \vec{c}}{|\hat{a} \times \vec{c}|} \tag{17}$$

where $\hat{a}$, $\hat{b}$, and $\hat{c}$ are unit vectors aligned along the three edges of the plate as illustrated in Figure 3. The vectors $\vec{a}$, $\vec{b}$, and $\vec{c}$ are defined as

$$\begin{aligned} \vec{a} &= a\,\hat{a} \\ \vec{b} &= b\,\hat{b} \\ \vec{c} &= c\,\hat{c} \end{aligned} \tag{18}$$

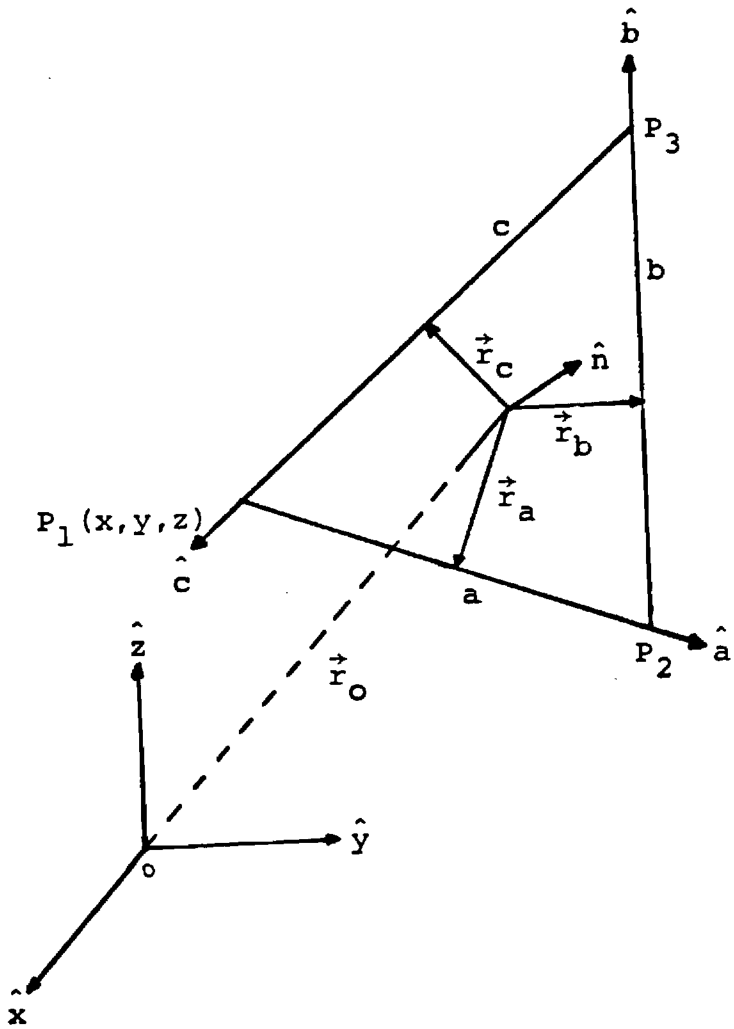


Figure 3. Triangular flat plate geometry.

where a, b, and c are the lengths of the three edges, respectively. The vector $\vec{r}_o$ is the position vector of the centroid of the plate with respect to the main origin of the target and is found only in the plane wave phase term $\exp(ik\,\vec{r}_o \bullet \vec{w})$ in (4). The vectors $\vec{r}_a$, $\vec{r}_b$, and $\vec{r}_c$ are position vectors of the midpoints of the edges with respect to the centroid of the plate. The variable T in the denominator of (13) is the projection of $\vec{w}$ onto the plane of the plate. When T = 0, a specular condition occurs where the contribution of all three edges at a far field point are in phase, but (13) becomes singular for this case. Thus, a check is made for the case when T = 0 where $\sqrt{\sigma}$ is now found by

$$\sqrt{\sigma} = \frac{\hat{n} \bullet \hat{e}_r \times \hat{h}_i}{\sqrt{\pi}}\, e^{ik\vec{r}_o \bullet \vec{w}}\, A \tag{19}$$

and A is just the area of the triangle found by

$$A = \frac{1}{2}\, |\vec{a} \times \vec{b}| \tag{20}$$

3.1.2 CONE FRUSTA

The closed form solution of Equation (12) for a truncated cone frustum is

$$\sqrt{\sigma} = -is\sqrt{\frac{2ka}{T}}\; \hat{n}_o \bullet \hat{h}_i \times \hat{e}_r \exp[ik\vec{r}_o \bullet (\hat{i} - \hat{s})] \exp[-i\,(kaT - \pi/4)]G \tag{21}$$

$$\text{where}\quad G = F\left(1 - \frac{i\,\tan\,\tau}{kQa}\right) + \frac{i\,\cos[\frac{1}{2}\,kQ\ell]\tan\,\tau}{kQa} \tag{22}$$

and

$$F = \frac{\sin[\frac{1}{2}\,kQ\ell]}{\frac{1}{2}\,kQ\ell} \tag{23}$$

Figure 4 is an illustration of the geometry of the truncated cone frustum. The points P_1 and P_2 are the Cartesian coordinates (with respect to the target origin 0) of the endpoints of the frustum along the central axis.

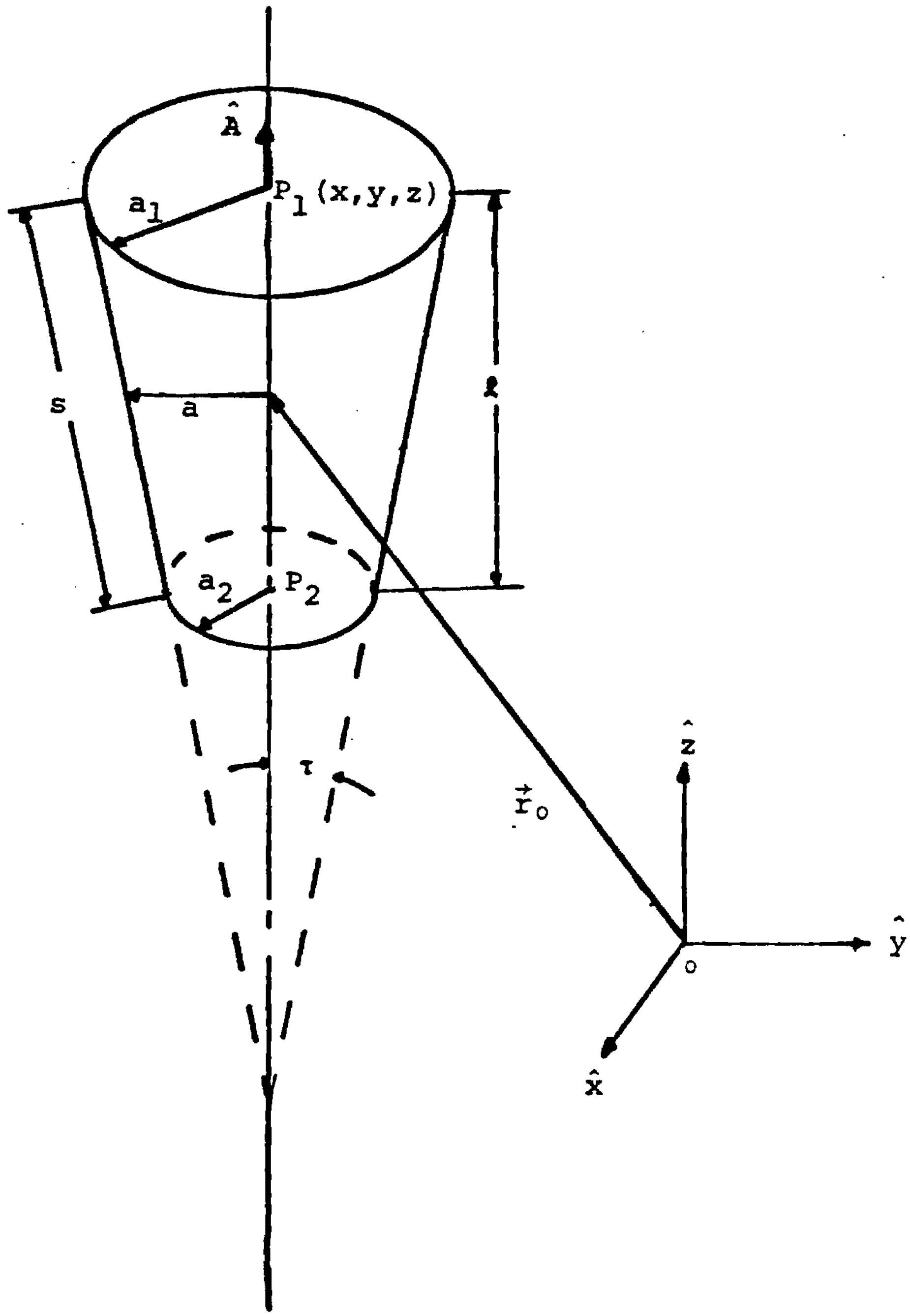


Figure 4. Truncated cone frustum geometry.

The radii at the two ends are a_1 and a_2. The axial length of the frustum is designated by ℓ, while the slant length is s. The cone half-angle is τ, but τ need not be found explicitly since tan τ is the only form in which τ appears in (22). Thus, tan τ is found by

$$\tan \tau = (a_1 - a_2)/\ell \qquad (24)$$

The symbol a in Equations (21) through (23) represents the mean radius of the frustum. It is found by averaging the radii of the two ends; a_1 and a_2. The vector $\vec{r}_o$ is the position vector of the center of the frustum with respect to the target origin. The quantity Q in Equations (22) and (23) is defined by

$$Q = P + T \tan \tau \qquad (25)$$

where $P = (\hat{i} - \hat{s}) \cdot \hat{A}$ (26)

and $T = |\hat{A} \times (\hat{i} - \hat{s})|$ (27)

The unit vector $\hat{A}$ is oriented along the main axis of the frustum and points from the small end toward the large end.

In the case where $a_1 = a_2$, the frustum degenerates to a cylinder. For this case, the imaginary components of G in (22) are omitted, otherwise they would be singular. Thus, for a cylinder, G = F. Another special case is the cone, which is found when $a_2 = 0$. Equations (21) through (23) can handle this case without adjustments.

3.1.3 DIHEDRALS AND TRIHEDRALS

A major analytical undertaking in the expansion of ship modeling techniques was the development of a procedure for predicting the return from arbitrary dihedrals and trihedrals. The radar return from re-entrant right-angled corner is large and persists over wide viewing angles, hence the corner is a dominant scatterer. However, echo area reductions of 20 dB or more can be achieved if the faces of the corner can be tilted away from perpendicularity. This non-perpendicular case is modeled in more or less routine fashion. [5]

Despite the reference to "arbitrary" corners, there is an implicit restriction: the model accounts for multiple internal interactions in which no trihedral face participates more than once. Thus, although the prescription works for up to three internal reflections, a triple reflection involving only two faces is not taken into account. This can occur only if some of the angles between faces are acute, hence the theory is applicable only to obtuse, but otherwise arbitrary, trihedral reflectors. There is no restriction on the use of acute angles, of course, but the model will not accurately include all the interactions.

The model has these salient features:

1. a marriage of geometrical optics (ray tracing) and physical optics approximations;
2. a physical optics prescription for the bistatic scattering from a perfectly conducting polygonal plate; and
3. a procedure for describing or identifying the common area shared by a pair of overlapping polygons.

The approach is to allow all interactions between the faces to follow the laws of geometric optics (GO), except for the final one, and to apply the physical optics (PO) prescription only then. In essence, the faces act like mirrors for all except the final reflection, which is treated by PO methods instead of GO. The procedure then allows the use of the standard bistatic far field PO scattering formula for flat surfaces.

Some care must be exercised in applying this concept. The direction followed by a ray when reflected from a flat surface can easily be determined by reversing the normal component of the incident ray — but one must also reverse the normal component of the incident magnetic field. The images of both the direction of propagation and of the magnetic field must be used in the bistatic PO scattering expression for a polygonal plate.

Extending Equation (13) from triangles to M-sided polygons,[6]

$$\sqrt{\sigma} = \frac{\hat{n}\cdot\hat{e}_r \times \hat{h}_i}{\sqrt{\pi}\ T} e^{ik\vec{r}_o\cdot\vec{w}} \sum_{m=1}^{M} \hat{p}\cdot\vec{a}_m e^{ik\vec{r}_m\cdot\vec{w}} \left[\frac{\sin(\frac{1}{2}k\vec{a}_m\cdot\vec{w})}{\frac{1}{2}k\vec{a}_m\cdot\vec{w}}\right] \qquad (28)$$

In equation (28),

σ = bistatic radar cross section of the plate,

$\hat{n}$ = unit normal to the plate surface,

$\hat{e}_r$ = unit vector along the electric vector of a far field receiver,

$\hat{h}_i$ = unit vector along the incident magnetic polarization,

$\vec{r}_o$ = position vector of the origin of the coordinate system in which the plate vertex positions are described,

$\vec{w}$ = $\hat{i} - \hat{s}$

$\hat{i}$ = unit vector along the direction of incidence,

$\hat{s}$ = unit vector along the direction from the origin to the far field receiver,

$\vec{a}_m$ = a vector describing the length and orientation of the m^{th} edge of the plate; the edge vectors must be arranged tip-to-tail around the perimeter of the polygon,

$\vec{r}_m$ = position vector of the midpoint of the m^{th} edge,

T = length of the projection of $\vec{w}$ onto the plane of the plate,

$\hat{p}$ = $\hat{n} \times \vec{w}/|\hat{n} \times \vec{w}|$ = unit vector in the plane of the plate perpendicular to $\vec{w}$,

M = number of plate edges.

The summation in (28) has the dimension of a length and includes the discrete contribution of each plate edge. T = 0 defines a specular condition for which the contributions of all the edges at a far field point are in phase, but (28) becomes singular in the specular direction. However, the expression reduces to a simpler form for specular scattering, namely

$$\sqrt{\sigma} = -ikA \frac{\hat{n}\cdot\hat{e}_r \times \hat{h}_i}{\sqrt{\pi}} e^{ik\vec{r}_o \cdot \vec{w}} \tag{29}$$

where A is the geometric area of the plate. For numerical purposes, it is better to switch from (28) to (29) when T is small but finite, instead of at precisely T = 0. In the model, the switch is made whenever $|T| < 10^{-2}$, and this finite switching point has a negligible effect on the predicted scattering.

Figure 5 illustrates the tracing of the reflections of incident beams from one face to another. If a face is fully illuminated by a beam or by the incident wave, only that part of the beam intercepted by the face is reflected, as shown in Figure 5(a). If it were not for the presence of the third face (shown in outline by the dashed line), the full reflection would be as shown. But, in actual fact, the third face will prevent the rightmost third of the reflected beam from ever reaching the plane of the third face. Hence the portion of the second face receiving a reflection of the incident wave from face 1, is shown in Figure 5(b).

The image direction $\tilde{i}$ of the incident wave is the effective direction of incidence for this patch of surface, and if the scattering direction is specified, along with the coordinates of the vertices of the illuminated patch, the scattering due to the interaction can be calculated by the use of Equation (28) or (29). Since two faces are involved in the far field scattering, this is called a "double-bounce" contribution to the echo.

A second reflection can occur in which face 2 reflects the beam it receives onto the plane of face 3, as shown in Figure 5(c). As in the reflection from face 1, the reflected direction from face 2 is found by reversing the normal component of the wave propagation direction. Thus the

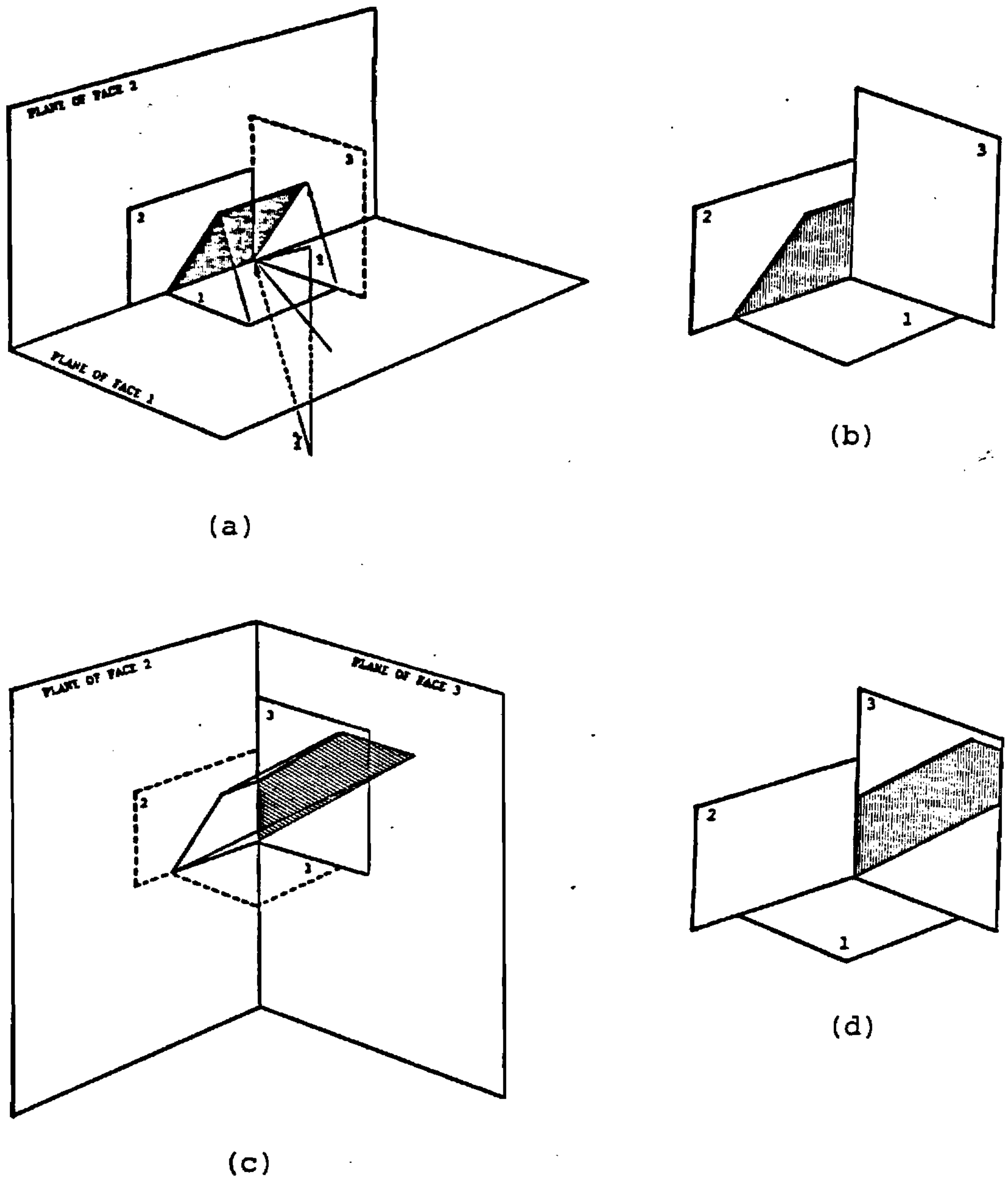


Figure 5. The shaded areas represent: (a) and (b) the portion of the incident wave intercepted by face 1 and reflected onto the plane of face 2; (c) and (d) the portion of face 2 illuminated by a reflection off face 1, and reflected onto the plane of face 3.

doubly imaged direction of propagation of the incident wave is used, imaged first in the plane of face 1, which image is then imaged in the plane of face 2. The doubly imaged direction is such as to cast the illuminated patch (shown shaded in Figure 5(c) onto the plane of face 2).

However, face 3 does not intercept the entire doubly-reflected beam. The rightmost tip of the patch will be clipped off because it extends past the physical boundaries of the third face. Thus, the actual size and shape of the "active" surface patch is shown in Figure 5(d). Again, a knowledge of the effective direction of the wave that generated the patch, along with the spatial positions of the patch vertices, allows the calculation of the far field scattering. In this case, the contribution would be a "triple-bounce" term, since three faces participated in the scattering.

This is one of six possible ways the three faces of a trihedral corner can participate in the far field scattering. The reflection of the incident wave onto face 3 by face 1 in the Figure 5(a) and the other combinations were not taken into consideration. The following list includes the six possible permutations for an obtuse trihedral corner:

1 2 3	2 3 1	3 1 2
2 1 3	3 2 1	1 3 2

These include six double-bounce and six triple-bounce contributions.

Finally, there are three single-bounce contributions to tally, and these do not involve interactions between the faces. They are the returns from the three faces when illuminated by the incident wave. The vertex positions and the directions of incidence and scattering can be used immediately in the PO formulas (28) or (29). The single-bounce scattering contributions are important for the backscattering case only when one of the faces is within a few degrees of its orientation for specular scattering. Nevertheless, the model continues to include the single-bounce scattering even when the face orientation is well away from the specular orientation. Figure 6 illustrates the definitions of dihedrals and trihedrals for input to the CROSS model.

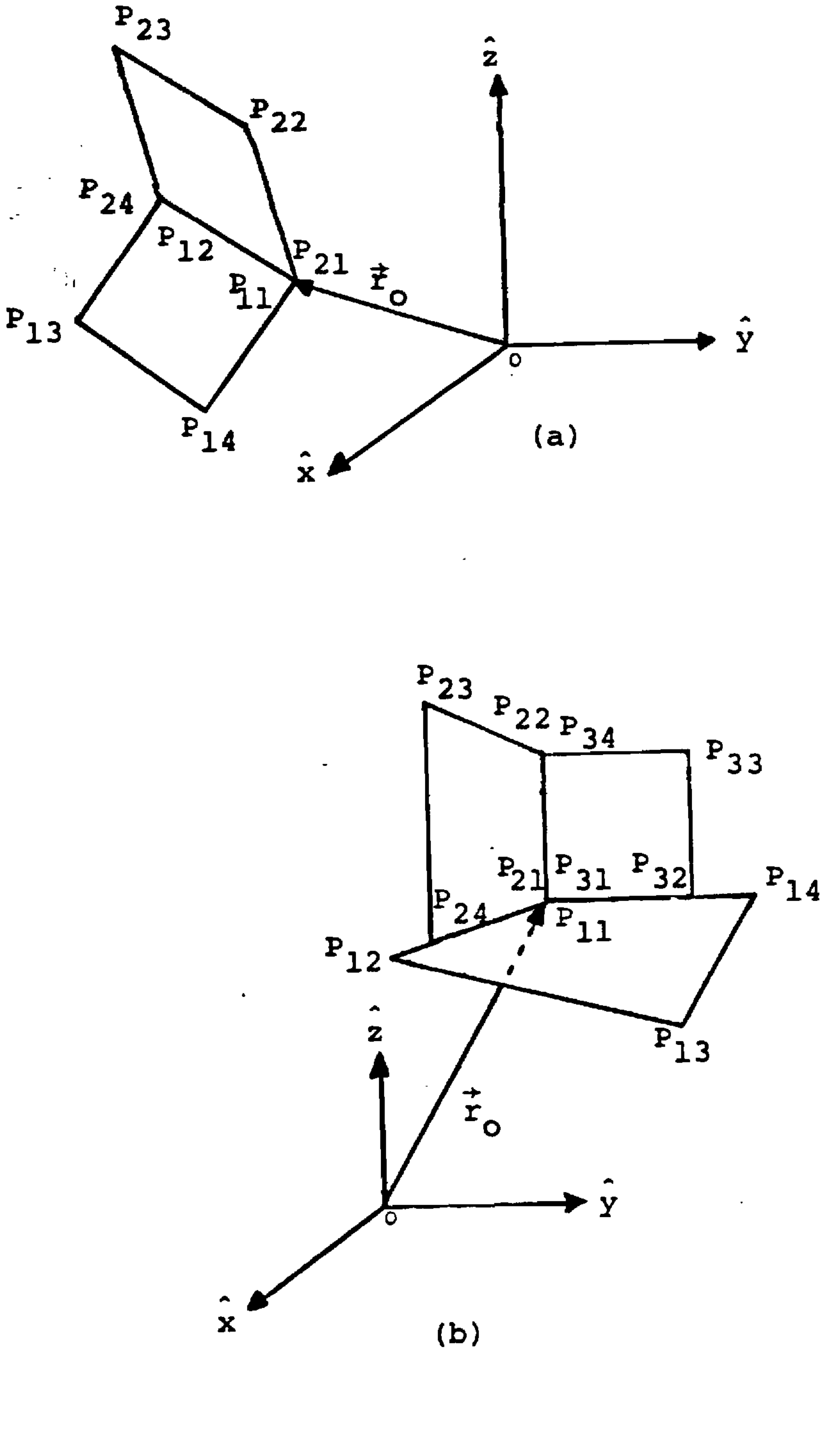


Figure 6. (a) Dihedral and (b) trihedral geometry.

3.1.4 ELLIPSOIDS

The geometrical-optics formulation for the radar cross section of an ellipsoid is given by

$$\sigma = \pi R_1 R_2, \tag{30}$$

where R_1 and R_2 are the two principal radii of curvature in orthogonal directions at the specular point on the ellipsoid's surface. Note that the product R_1R_2 is the reciprocal of the "Gaussian curvature". Gaussian curvature at a point on a surface is defined as the product of the principal curvatures, and the principal curvatures are the reciprocals of the principal radii of curvature [7]. Thus, without calculating R_1 and R_2 separately, the radar cross section of an ellipsoid can be computed as

$$\sigma = \frac{\pi}{\kappa} \tag{31}$$

where κ is the Gaussian curvature at the specular point. Because it is easier to calculate κ directly than to find R_1 and R_2 separately, the formulation for the ellipsoid in Equation (31) simplifies the RCS computation.

The first step in finding the Gaussian curvature at a point on an ellipsoid is to describe the ellipsoidal surface as a vector function of two parameters ϕ and θ. For the ellipsoid defined by

$$\left(\frac{x}{A}\right)^2 + \left(\frac{y}{B}\right)^2 + \left(\frac{z}{C}\right)^2 = 1$$

This can be done by describing the ellipsoid in spherical coordinates:

$$\vec{x} = (A \sin\phi \cos\theta,\ B \sin\phi \sin\theta,\ C \cos\phi). \tag{32}$$

The Gaussian curvature at the specular point is then described in terms of the first and second order derivatives at the specular point. Specifically,

$$\kappa = \frac{LN - M^2}{EG - F^2} \tag{33}$$

where

$L = \vec{n} \cdot \vec{V}_{11}$

$M = \vec{n} \cdot \vec{V}_{12}$

$N = \vec{n} \cdot \vec{V}_{22}$

$\vec{n}$ = the unit normal to the surface at the specular point $= (\vec{V}_1 \times \vec{V}_2)/|\vec{V}_1 \times \vec{V}_2|$

$E = \vec{V}_1 \cdot \vec{V}_1$

$F = \vec{V}_1 \cdot \vec{V}_2$

$G = \vec{V}_2 \cdot \vec{V}_2$

and

$$\vec{V}_1 = \frac{d\vec{x}}{d\phi} = (A \cos\phi \cos\theta,\ B \cos\phi \sin\theta,\ -C \sin\phi)$$

$$\vec{V}_2 = \frac{d\vec{x}}{d\theta} = (-A \sin\phi \cos\theta,\ B \sin\phi \cos\theta,\ 0)$$

$$\vec{V}_{12} = \frac{d^2 \vec{x}}{d\phi d\theta} = (-A \cos\phi \sin\theta,\ B \cos\phi \cos\theta,\ 0)$$

$$\vec{V}_{11} = \frac{d^2 \vec{x}}{d\phi^2} = (-A \sin\phi \cos\theta,\ -B \sin\phi \sin\theta,\ -C \cos\phi)$$

$$\vec{V}_{22} = \frac{d^2 \vec{x}}{d\theta^2} = (-A \sin\phi \cos\theta,\ -B \sin\phi \sin\theta,\ 0)$$

The derivative vectors $\vec{V}_1$, $\vec{V}_2$, $\vec{V}_{12}$, $\vec{V}_{11}$, and $\vec{V}_{22}$ are all evaluated at the specular point whose coordinates are known (and therefore ϕ and θ are known). Once the derivative vectors and the normal vector are defined, the Gaussian curvature is easily computed from (33) using the vector manipulation library subroutines. Figure 7 illustrates the geometry for ellipsoid scatterers.

3.2. GEOMETRICAL CONSIDERATIONS

The geometry relating the target to the radar in a spherical earth environment can become quite complicated, so to provide a scheme that is both general and practical, the following assumptions are made:

* Flat earth
* Target constrained to lie on the earth's surface
* Radar fixed but target allowed to move
* Cartesian coordinate system.

The primary reason for staying with Cartesian coordinates is ease of programming as well as the mathematics. A library of FORTRAN functions and subroutines exists to help simplify standard vector operations (dot product, cross product, etc.), but they are only defined for Cartesian coordinates.

The origin of the initial coordinate system is located on the earth's surface. The x and y axes are tangential to the earth's surface and the z axis normal to it. This coordinate system (as well as all others in this report), is "right-handed." The radar is fixed on the z-axis at a height h_a above the origin.

Since the target is constrained to lie on the earth's surface, only two coordinates are necessary to define its initial position. The two coordinates used here are range (R_a) and azimuth (ϕ). The range is defined as the distance between the origin and the target as measured along the earth's surface. The azimuth is the angle between the x-axis and the target measured counterclockwise. (See Figure 8.)

To express the coordinates of the target in x,y,z coordinates, an initial range vector $\vec{R}^i$ to the target can be set up.

$$\vec{R}^i = \begin{pmatrix} R_a \cos \phi \\ R_a \sin \phi \\ 0 \end{pmatrix} \qquad (34)$$

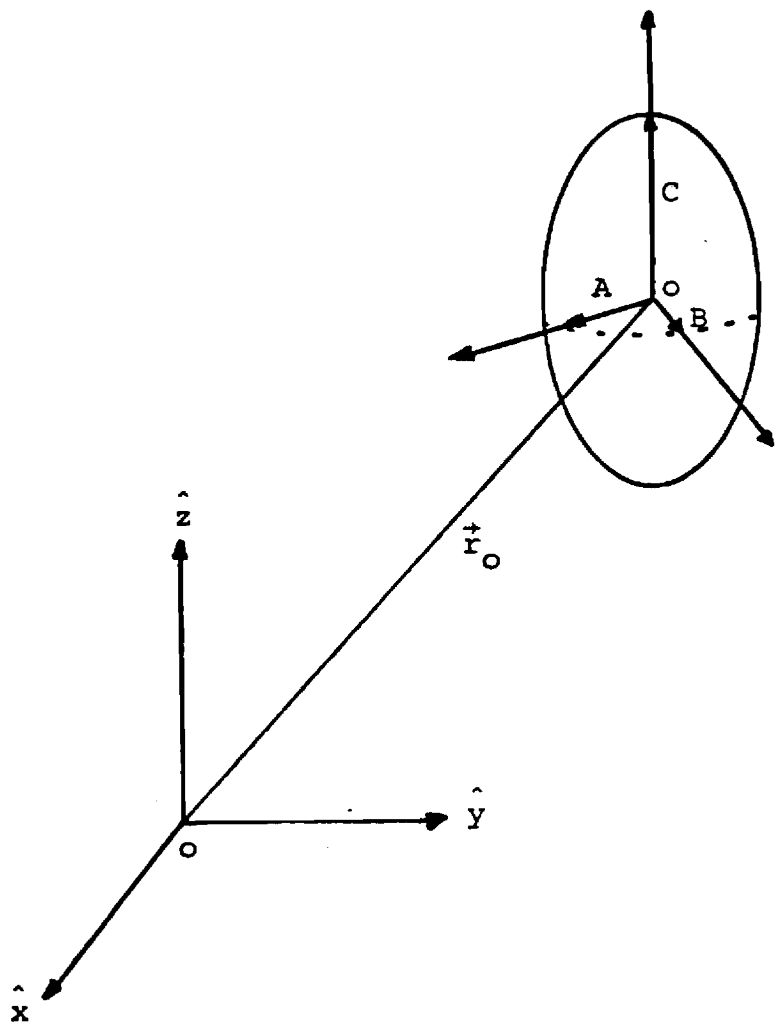


Figure 7. Ellipsoid geometry.

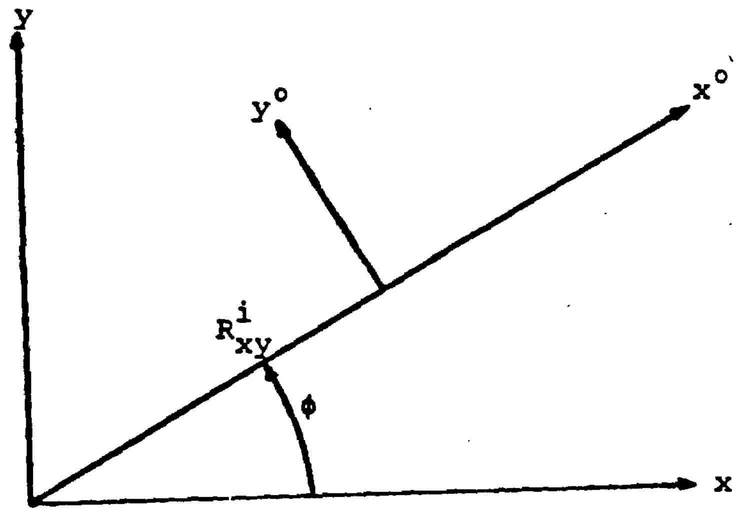


Figure 8. Relative positions of initial coordinate systems.

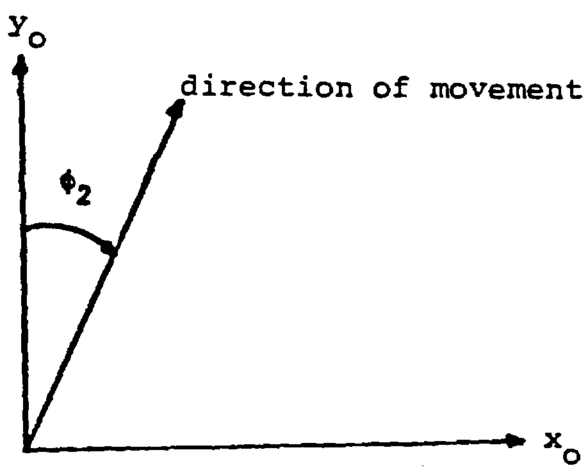


Figure 9. Definition of "bearing".

To facilitate computations, transformations are made from the initial coordinate system (fixed earth) to the fixed target coordinate system rather than conversely. This greatly reduces the number of vectors to be transformed since there are many more target scatterers than position vectors (radar position, etc.) in the initial coordinate system.

An intermediate coordinate system is defined at the initial target position (x^o, y^o, z^o) such that the z^o axis is normal to the earth's surface, and the y^o axis is orthogonal to the displacement vector $\vec{R}^i$, as in Figure 8. The rotation matrix to transform to this frame of reference is defined as

$$\mathrm{Rot}_{\phi} = \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{35}$$

Now, for an arbitrary vector $\vec{r}$ in xyz, a translation in xyz must occur before rotation, so

$$\vec{r}^{\,o} = \mathrm{Rot}_{\phi}\,(\vec{r} - \vec{R}^i) \tag{36}$$

or

$$\vec{r}^{\,o} = \tilde{B}\,(\vec{r} - \vec{R}^i) \tag{37}$$

where

$$\tilde{B} = \mathrm{Rot}_{\phi} \tag{38}$$

To allow for motion of the target, a "path" is defined by defining a direction and a distance (along the earth's surface, of course). Let ϕ_2 be a "bearing" of the target such that ϕ_2 is measured clockwise from the y^o axis. (See Figure 9.) The associated rotation matrix is

$$\mathrm{Rot}\ \phi_2 = \begin{pmatrix} \cos\phi_2 & -\sin\phi_2 & 0 \\ \sin\phi_2 & \cos\phi_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{39}$$

$$\tilde{C} = \text{Rot } \phi_2 \; \tilde{B} \tag{40}$$

The target can now be allowed to move a range increment ΔR_a between successive RCS predictions in the new direction. A transformation to each new position of the target is as follows

$$\Delta\vec{R} = \begin{pmatrix} 0 \\ \Delta R_a \\ 0 \end{pmatrix} \tag{41}$$

Now,

$$\vec{r}^N = \text{Rot } \phi_2 \text{ Rot } \phi \; (\vec{r} - \vec{R}^i) - \Delta\vec{R}$$
$$\vec{r}^N = \text{Rot } \phi_2 \; \vec{r}^o - \Delta\vec{R} \tag{42}$$

For successive movements of the target, the program sets $\vec{r}^o = \vec{r}^N$ and uses Equation (42) to transform to the new coordinates of the target where ϕ_2 is the new "bearing." If the direction of motion remains unchanged between successive range steps, it uses

$$\vec{r}^N = \vec{r}^o - \Delta\vec{R} \tag{43}$$

instead of (42) after the initial calculation of $\vec{r}^N$.

Now that the target location and the related transformations have been specified, the target orientation needs to be defined. The most logical scheme for specifying the orientation is to use yaw, pitch, and roll. Here, yaw, pitch, and roll are defined as counterclockwise rotations about the Z^N, X^N, and Y^N axes, respectively. Since the order of rotations is important, the computer program allows for optional orders of rotation (pitch before yaw, roll before pitch, etc.). The transformation matrices are defined below.

$$\tilde{Y} = \begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{44}$$

$$\tilde{P} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & \sin\beta \\ 0 & -\sin\beta & \cos\beta \end{pmatrix} \tag{45}$$

$$\tilde{R} = \begin{pmatrix} \cos\gamma & 0 & -\sin\gamma \\ 0 & 1 & 0 \\ \sin\gamma & 0 & \cos\gamma \end{pmatrix} \tag{46}$$

Thus, the final transformation of a vector $\vec{r}^N$ to the fixed target frame is

$$\vec{r}' = \tilde{R} \; \tilde{P} \; \tilde{Y} \; \vec{r}^N \tag{47}$$

(in any order)

To calculate the RCS of a target, the two unit incident vectors (direct and indirect) must be specified. To find the direct incident vector $\hat{i}'$, the vector denoting the position of the radar antenna must be transformed to target coordinates. Let $\vec{A}$ represent the antenna coordinates in the initial frame. Thus,

$$\vec{A} = (0, 0, h_a)$$

where h_a is the height of the antenna above the surface.

$$\vec{A}' = \tilde{R} \, \tilde{P} \, \tilde{Y} \, \vec{A}^N \tag{48}$$

and $\vec{A}^N$ is defined by either transformation (42) or (43), whichever is appropriate. Now,

$$\hat{i}' = \frac{\vec{r}_o' - \vec{A}'}{|\vec{r}_o' - \vec{A}'|} \tag{49}$$

where $\vec{r}_o$ is the position of the centroid of a particular scatterer in the target frame.

The indirect incident vector $\hat{i}_2'$ is not difficult to find since a flat earth is assumed and an image technique may be used (See Figure 10). Since the plane in which both $\hat{i}'$ and $\hat{i}_2'$ lie must be perpendicular to the earth's surface, then $\hat{i}_2'$ is found as follows

$$\tilde{Q} = \tilde{R}\ \tilde{P}\ \tilde{Y} \tag{50}$$

and $$\tilde{Q}^{-1} = \tilde{Y}^{-1}\ \tilde{P}^{-1}\ \tilde{R}^{-1} = \tilde{Y}^T\ \tilde{P}^T\ \tilde{R}^T \tag{51}$$

(Note: Since rotation matrices are orthogonal, the inverse is simply the transpose of the matrix.)

$$\vec{r}_o^{\,n} = \tilde{Q}^{-1}\ \vec{r}_o' \tag{52}$$

Now the image of the scatterer can be formed by simply negating the z coordinate of scatterer's position vector:

$$\vec{r}_o^{\,i} = (x_o^n,\ y_o^n,\ -z_o^n) \tag{53}$$

$$\hat{i}_2^{\,i} = \frac{\vec{r}_o^{\,i} - \vec{A}^n}{|\vec{r}_o^{\,i} - \vec{A}^n|} \tag{54}$$

Finally, the z coordinate of $\hat{i}_2^{\,i}$ must be negated and the resulting vector rotated back into the target's (primed) coordinate system.

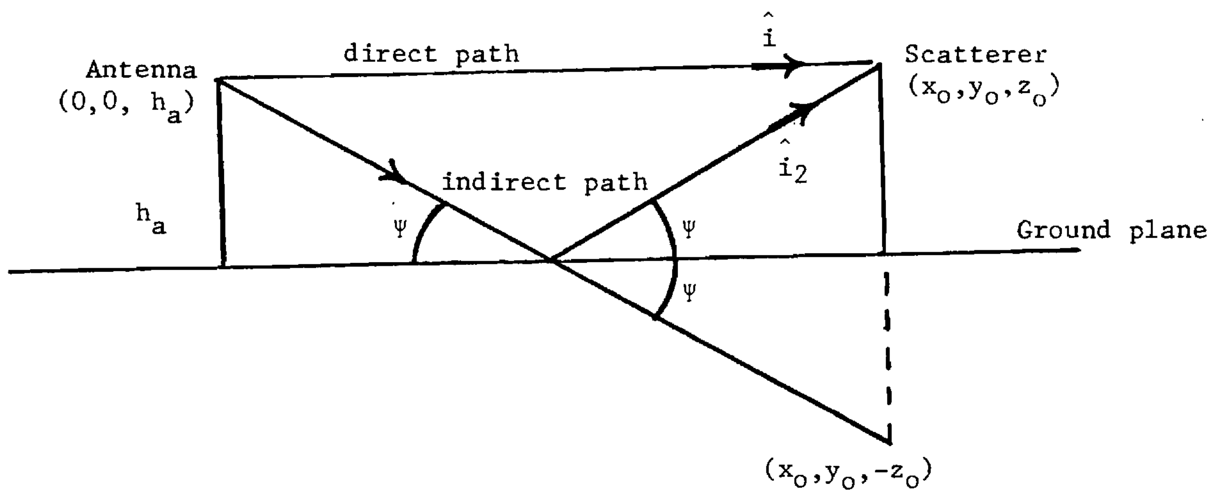


Figure 10. Multipath geometry illustrating the direct and indirect incident vectors.

$$\hat{i}_2^n = (i_x^i,\ i_z^i,\ -i_z^i) \tag{55}$$

$$\hat{i}_2^1 = \tilde{Q}\ \hat{i}_2^n \tag{56}$$

3.3. MULTIPATH

Equation (12) is the physical optics integral for free space scattering. To handle the multipath due to the earth's surface, the return from a scatterer can be represented as the sum of three contributions, one to the target as if the surface were not present (the free space contribution), another due to the target image, and the third due to the diplane effect. The image term must be multiplied by the square of the reflection coefficient of the earth's surface, because this contribution involves a double bounce. The diplane term must be multiplied by twice the reflection coefficient because only one bounce off the earth's surface occurs, but two distinct propagation paths exist. For each of the types of scatterers, the total scattering can be represented as the sum

$$\sqrt{\sigma} = \sqrt{\sigma_f} + \rho^2 \sqrt{\sigma_i} + 2\rho \sqrt{\sigma_d} \tag{57}$$

where ρ is the complex reflection coefficient of the earth's surface and $\sqrt{\sigma_f}$, $\sqrt{\sigma_i}$, and $\sqrt{\sigma_d}$ are the free space, image, and diplane contributions respectively. All three contributions are found from Equation (12) depending on the scatterer type. Note that the equation contains the scattering vector $\hat{i} - \hat{s}$. The free space term $\sqrt{\sigma_f}$ is found by setting $\hat{i}$ equal to the direct incident vector $\hat{i}'$ defined in Equation (49) and by setting $\hat{s} = -\hat{i}'$. For the diplane term, $\hat{i} = \hat{i}'$ as above, but $\hat{s} = -\hat{i}_2'$ where $\hat{i}_2'$ is the indirect incident vector from the earth's surface defined in Equation (56). Similarly, the image term $\sqrt{\sigma_i}$ is found by setting $\hat{i} = \hat{i}_2'$ and $\hat{s} = -\hat{i}_2'$.

3.4. NEAR FIELD APPROXIMATION

In its present form, Equation (12) assumes the far field approximation that planes of constant phase are present over the entire target. CROSS allows the user to choose a near field approximation if desired. That is, the phase term for each scatterer can be calculated separately by using the distance from the radar to the centroid of the scatterer. This simulates a spherical wave front over the entire target, although the return from each scatterer is still calculated as if it had a plane wave over its surface. This is accomplished by replacing the relative phase term $e^{ik\vec{r}_o \cdot (\hat{i} - \hat{s})}$ found in Equation (12) with the near field phase term $e^{ik\Delta r}$, where Δr is the difference between the length of the vector from the radar to the target's origin and the length of the propagation path for the scatterer of interest.

3.5. REFLECTION FROM THE EARTH'S SURFACE

The complex reflection coefficient ρ is a function of the incident angle at the earth's surface (Ψ), the surface roughness, and the complex permittivity of the reflecting surface which the program requires the as an input. The reflection coefficient ρ is expressed as the product of the Fresnel reflection coefficient for smooth surfaces [10] and the Rayleigh roughness factor [11]. The Fresnel reflection coefficient is polarization dependent. The angle of reflection Ψ can be found from the indirect incident vector $\hat{i}_2^n$ defined in equation (55).

$$\Psi = \pi/2 - \hat{i}_2^n \quad \hat{z} \tag{58}$$

For horizontal polarization

$$r = \frac{\sin \Psi - \sqrt{\varepsilon - \cos^2 \Psi}}{\sin \Psi + \sqrt{\varepsilon - \cos^2 \Psi}} \tag{59}$$

and for vertical polarization

$$r = \frac{\varepsilon \sin\Psi - \sqrt{\varepsilon - \cos^2 \Psi}}{\varepsilon \sin\Psi + \sqrt{\varepsilon - \cos^2 \Psi}} \tag{60}$$

Now, the Rayleigh roughness factor is

$$\rho_s = \left\{ \exp \quad \frac{-1}{2} \left[\frac{(4\pi \ \sigma_H \sin \Psi)}{\lambda}^{2} \right] \right\} \tag{61}$$

where σ_H is the standard deviation of the wave height. Thus,

$$\rho = r\rho_s \tag{62}$$

SECTION 4

REFERENCES

1. R. B. Rakes, "Radar Glint Model," Software Documentation Report on Rockwell International Purchase Order V161-5A-113203 Georgia Institute of Technology, Engineering Experiment Station, April 1982, UNCLASSIFIED.

2. S. P. Zehner and M. T. Tuley, "Development and Validation of Multipath and Clutter Models for TAC ZINGER in Low Altitude Scenarios," Final Report on Contract F49620-78-C-0121, March 1979, UNCLASSIFIED.

3. M. M. Horst, et al., "Ship RCS Predictions (U)," Final Report on Contract Noo167-82-M-0737, Georgia Institute of Technology, Engineering Experiment Station, January 1982, SECRET.

4. M. T. Tuley, M. M. Horst, and K. B. Langseth, "Radar Modeling Studies (U)," Task 7 Final Report on APL Subcontract 600403, Georgia Institute of Technology, Engineering Experiment Station, July 1978, SECRET, XGDS.

5. E. F. Knott, "A Tool for Predicting the Radar Cross Section of a Trihedral Corner," Paper presented at IEEE Southeast Conf. '81, Huntsville, Alabama, 6-8 April 1981, Published in IEEE Publication 81-CH-1650-1, pp. 17-20.

6. W. B. Gordon, "Far-field Approximation of the Kirchoff-Helmholtz Representation of Scattered Fields," IEEE Transactions on Antennas and Propagation, AP-23, July 1975, pp. 590-592.

7. W. Gellert, H. Kustner, M. Hellwich, and H. Kastner, Editors, The VNR Concise Encyclopedia of Mathematics, pp. 565-571, Van Nostrand Reinhold Company, New York, 1977.

8. H. Flanders, R. Korfhage, and J. Price, A Second Course in Calculus, p. 501, Academic Press, New York, 1974.

9. E. Shotland and R. Rollin, "The Complex Reflection Coefficient Over a Smooth Sea in the Micro and Millimeter-Wave Bands for Linear and Circular Polarization," The John Hopkins University, APL, March 1976.

10. J.A. Stratton, Electromagnetic Theory, McGraw-Hill Book Company, New York, 1941, pp. 492-511.

11. P. Beckman and A. Spizzichino, The Scattering Of Electromagnetic Waves From Rough Surfaces, (New York: The McMillan Company, 1963), pp. 80-81.

P2966

ENGINEERING EXPERIMENT STATION
GEORGIA INSTITUTE OF TECHNOLOGY • ATLANTA, GEORGIA 30332

21 June 1982

Mr. Carl Bates
Rockwell International
Missile Systems Division
P. O. Box 1259
Columbus, Ohio 43216

Dear Carl:

Enclosed are the results of the Simulation Verification data run by the Georgia Tech Radar Glint model. These include Track Error and RCS predictions for both variation with range and variation with aspect. If any problem is encountered in attempting to match these results with your program, please feel free to call me anytime.

Sincerely,

R. Bruce Rakes
Research Scientist I

swc

Enclosure - Simulation Verification data results

TRACK ERROR/RCS Benchmark

Variation of Aspect

Elevation = 15°, Slant Range = 679'
σ_h = 0.1 ft., σ^o = -17.5 dB
ϵ_r = (2.44, 0.00267)

Track Error AZ (deg)	Track Error El (deg)	RCS	
-1. 4917186E-02	7. 8193314E-02	-11. 66653	
0. 1012685	3. 8093068E-02	-4. 729362	Initial Az = -2°
-7. 6000090E-03	0. 5996329	25. 54455	Final Az = 2°
-4. 2801026E-02	0. 2332899	4. 329116	Az Step = 1°
7. 4208438E-02	0. 1547721	1. 707733	
0. 6000569	0. 4268321	10. 19216	
0. 9245617	0. 2220081	24. 26637	Initial Az = 88°
0. 2524590	0. 3098174	14. 04780	Final Az = 92°
-0. 7222527	0. 3191296	30. 79765	Az Step = 1°
-0. 7957122	-4. 8491564E-02	2. 880258	
9. 8208282E-03	0. 1101521	-8. 688541	
-5. 5280502E-04	9. 2625953E-02	-11. 51835	Initial Az = 133°
-8. 6806461E-02	0. 2581288	-0. 2211581	Final Az = 137°
-9. 7268699E-03	0. 1596832	-4. 905655	Az Step = 1°
-2. 2283677E-02	0. 2067052	3. 348537	
7. 9131819E-02	0. 1051137	-12. 86676	
0. 2964405	0. 1294527	2. 982919	Initial Az = 178°
1. 8847820E-04	0. 1760707	21. 70618	Final Az = 182°
-0. 3032177	0. 1816531	4. 863090	Az Step = 1°
-1. 6459955E-02	0. 2171304	-6. 635852	
3. 3235546E-02	0. 2351657	3. 629490	
-1. 0232148E-02	-9. 0830155E-02	-14. 17316	Initial Az = 223°
-2. 4799721E-03	5. 3917691E-02	-15. 70564	Final Az = 227°
8. 8533349E-03	-7. 2386903E-03	-7. 138330	Az Step = 1°
8. 4184855E-03	-0. 1683839	-8. 843243	

Variation with Aspect

Elevation = 15°,   Slant Range = 679 ft.

Track Error Az (deg)	Track Error EL (deg)	RCS	
-1.3064086E-03	1.2878230E-02	-12.90181	
-1.1134183E-02	0.1332418	-5.838790	Initial Az = 313°
-7.7350549E-02	8.8500634E-02	-9.296549	Final Az = 317°
-1.0944321E-02	-0.1152363	-13.06730	Az Step = 1°
1.6181894E-02	0.2733363	-2.269258	
-0.3047307	-0.5423833	-3.486520	
2.1760428E-02	3.6029760E-02	-10.50823	Initial Az = 44.8°
6.0479820E-02	8.2971193E-02	-9.157353	Final Az = 45.3°
1.8063786E-02	-3.7682410E-05	-13.66778	Az Step = 0.1°
-6.6842869E-02	-0.2566604	-8.927942	
2.9440064E-02	-0.3291941	-6.914072	
0.6000569	0.4268321	10.19216	
0.9245617	0.2220081	24.26637	Initial Az = 88°
0.2524590	0.3098174	14.04780	Final Az = 92°
-0.7222527	0.3191296	30.79765	Az Step = 1°
-0.7957122	-4.8491564E-02	2.880258	
0.1240494	0.2301475	15.43640	
0.3590895	0.1475474	5.057254	Initial Az = 269.8°
2.8979480E-02	0.1421916	14.16871	Final Az = 270.2°
-0.2459276	0.2557647	11.63514	Az Step = 0.1°
-0.2160972	0.2142867	17.62584	

Variation with Aspect

Elevation = 45°, Slant Range = 928 ft.

Track Error Az (deg)	Track Error El (deg)	RCS	
-1.9274877E-02	-2.2689503E-02	-2.542281	
3.4068629E-02	-0.1611469	-4.411059	Initial Az = -2°
2.4577057E-02	0.6706424	20.59393	Final Az = 2°
-3.5952434E-02	-0.2544481	-19.27741	Az Step = 1°
-6.2617145E-02	-4.2861851E-04	-3.577843	

4.2926678E-03	-0.4959759	15.35856	Initial Az = 88°
-0.1564836	0.3106360	22.00022	Final Az = 92°
0.1287092	0.2740144	10.95899	Az Step = 1°
1.5138185E-02	0.2336436	7.872255	
2.5987023E-02	0.2988815	-0.7816853	

-7.4495883E-03	-1.5718155E-03	-31.37795	
3.6409266E-02	-0.1131416	-8.609474	Initial Az = 133°
-4.1325737E-02	-3.5774961E-02	-19.34916	Final Az = 137°
1.6667228E-02	1.3446214E-02	-16.36296	Az Step = 1°
8.1879966E-02	0.1833812	18.65425	

-0.2119371	8.6204693E-02	-11.61060	
-2.2932965E-02	-0.3513575	3.498105	Initial Az = 178°
5.9584049E-03	-0.1978338	27.88261	Final Az = 182°
-0.3124326	-0.4548050	-3.274962	Az Step = 1°
9.1297589E-02	9.5918082E-02	5.607147	

-1.0909708E-02	6.9532908E-02	-6.188841	
-3.4034748E-02	1.9587083E-02	-13.90251	Initial Az = 223°
-0.2595626	-0.5096098	-4.639563	Final Az = 227°
-2.8910710E-02	-1.9107087E-02	-24.74454	Az Step = 1°
-5.1068950E-02	-2.8811533E-02	-12.33971	

Variation with Aspect

Elevation = 45° , Slant Range = 928 ft.

Track Error AZ (deg)	Track Error EL (deg)	RCS	
5.0766252E-02	1.8673576E-02	-5.516499	
1.9176574E-02	1.9454479E-02	-16.16605	Initial AZ = 313°
1.5347627E-02	7.6911777E-02	-4.118290	Final AZ = 317°
-4.5549266E-02	2.1738037E-02	-2.678010	AZ Step = 1°
4.5443572E-02	3.0895209E-02	-10.36890	

-1.7437069E-02	8.5160486E-02	-0.4634165	
-3.9793797E-02	5.9714567E-02	-5.484707	Initial AZ = 44.8°
8.1778161E-04	-7.9572191E-03	-13.96869	Final AZ = 45.3°
-4.3971192E-02	0.1123019	-0.4220569	AZ Step = 0.1°
2.5749382E-02	-2.5159303E-02	-16.97661	
-5.9530873E-02	6.3213855E-02	-5.524306	

4.2926678E-03	-0.4959759	15.35856	
-0.1564836	0.3106360	22.00022	Initial AZ = 88°
0.1287092	0.2740144	10.95899	Final AZ = 92°
1.5138185E-02	0.2336436	7.872255	AZ Step = 1°
2.5987023E-02	0.2988815	-0.7816853	

0.1507022	0.1921148	9.461764	
-0.2697877	0.8945019	16.23875	Initial AZ = 269.8°
-4.7635950E-02	0.3799199	20.14245	Final AZ = 270.2°
-8.7587938E-02	0.3708959	14.52001	AZ Step = 0.1°
-0.1651377	0.3518490	18.60176	

Variation with Range

Elevation = 15 degrees

Slant Range (feet)	Track Error AZ (deg)	Track Error EL (deg)	RCS	
1020.000	5.0061331E-03	0.4993497	20.81643	
850.0000	4.4283173E-03	0.4867148	26.09033	
680.0000	-7.3807300E-03	0.5950443	25.58356	
510.0000	2.8371245E-03	1.024202	24.70302	Az = 0 deg
340.0000	-3.6155514E-03	0.4436491	27.82073	
170.0000	-1.5099181E-02	-8.8675983E-02	27.13674	
1020.000	0.2612238	0.1562295	15.85933	
850.0000	-0.8101477	0.3949793	7.254593	
680.0000	0.2563402	0.2943910	13.20210	
510.0000	0.1019797	0.2819632	15.98870	Az = 90 deg
340.0000	0.3291698	0.4264002	13.34318	
170.0000	0.5785391	0.1595181	15.77669	
1020.000	8.4236858E-04	9.2937738E-02	27.35988	
850.0000	-4.2908260E-04	0.1205537	25.24064	
680.0000	4.2360471E-04	0.1758326	21.67067	
510.0000	4.3997159E-03	0.6104326	15.25537	Az = 180 deg
340.0000	4.2263950E-05	0.2516498	26.40305	
170.0000	1.1080918E-04	0.3964351	25.99878	
1020.000	0.1729786	0.2754012	14.20288	
850.0000	-0.5209087	6.9691449E-02	2.496394	
680.0000	2.1545710E-02	0.1604084	14.58973	
510.0000	7.6774158E-02	5.5852821E-03	15.08606	Az = 270 deg
340.0000	0.1917440	0.1952392	11.34719	
170.0000	-0.4512213	-0.1868914	7.818293	

Variation with Range

Elevation = 45 degrees

Slant Range (feet)	Track Error AZ (deg)	Track Error EL (deg)	RCS	
1392.000	-2.5184453E-03	0.3971903	25.36010	
1160.000	-1.2370380E-02	0.3866917	24.46605	
928.0000	2.9494504E-02	0.6621591	20.74256	
696.0000	-0.1362364	-3.4903526E-02	16.60512	Az = 0 deg
464.0000	-0.1028221	-0.1756750	23.66253	
232.0000	1.5592365E-02	-0.2130625	26.46243	
1392.000	0.1009403	9.7273618E-02	14.88155	
1160.000	9.8801769E-02	0.2273987	12.57011	
928.0000	0.1165720	0.2748711	10.85851	
696.0000	0.1066830	0.2244116	15.91226	Az = 90 deg
464.0000	-0.1799550	0.2849002	5.193667	
232.0000	-0.1177549	-1.1346363E-02	10.58290	
1392.000	3.5835656E-03	-0.1465807	36.12967	
1160.000	-1.8315621E-03	-0.1992521	31.75100	
928.0000	5.3459797E-03	-0.1968702	27.73532	
696.0000	3.2525361E-03	-0.3056526	29.42298	Az = 180 deg
464.0000	2.6337963E-03	-0.5881231	26.01520	
232.0000	1.2147897E-03	-0.7972733	16.99524	
1392.000	-0.4080007	0.7293573	14.04557	
1160.000	-0.4243019	1.738803	11.91391	
928.0000	-4.8317187E-02	0.3796228	20.20825	
696.0000	-6.2007017E-02	0.2290223	9.183858	Az = 270 deg
464.0000	-0.1811789	0.5283466	15.89808	
232.0000	-0.2259777	-0.5519359	7.652482	