Technical Note

1970-6

DEANE: A Computer Aid for Ballistic Missile Defense Analysis

D. L. McCraith

4 November 1970

Prepared for the Office of the Chief of Research and Development, Department of the Army, under Electronic Systems Division Contract F19628-70-C-0230 by

Lincoln Laboratory

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Lexington, Massachusetts

Approved for public release; distribution unlimited.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

DEANE: A COMPUTER AID FOR BALLISTIC MISSILE DEFENSE ANALYSIS

D. L. McCRAITH

Group 31

TECHNICAL NOTE 1970-6

4 NOVEMBER 1970

LEXINGTON MASSACHUSETTS

The work reported in this document was performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology. The work is sponsored by the Office of the Chief of Research and Development, Department of the Army; it is supported by the Advanced Ballistic Missile Defense Agency under Air Force Contract F19628-70-C-0230.

ABSTRACT

DEANE is a special-purpose computer language designed for use in a time-shared environment by a ballistic missile defense systems analyst. In essence, it is a sophisticated calculator whose modular design allows the user to request basic computations in a convenient fashion. The interpretation of the computational results is not made by DEANE under the pretense of being a simulator, but is left to the user.

This report is a user's manual describing the computations available and giving examples of how they may be ordered to solve typical problems. Also presented is a description of the logical and/or mathematical foundations of the computations.

Accepted for the Air Force
Joseph R. Waterman, Lt. Col., USAF
Chief, Lincoln Laboratory Project Office

CONTENTS

DEANE: A COMPUTER AID FOR BALLISTIC MISSILE DEFENSE ANALYSIS

I. INTRODUCTION

DEANE is a special-purpose computer language designed to aid in certain aspects of ballistic missile defense systems analysis. It is designed to help solve problems which involve understanding primarily geometric and kinematic relationships. DEANE is intended to operate in a time-shared environment, but it can also be executed under a batch monitor.

The earth is modeled as a sphere which may be either stationary or rotating. Trajectories are initially calculated to be Keplerian; however, there is an option for replacing the re-entry part of the trajectory with values attained by integration of the equations of motion using a real atmosphere and a finite ballistic coefficient for the re-entry object. These Keplerian trajectories may be determined from either launch site and impact site specification or by specification of ejection velocity components from a base trajectory. Axisymmetric objects may be given a cross section as a function of aspect angle by specification of a table, and their orientation may be modeled to correspond to either constant rate inplane tumbling or a spin-stabilized situation with coning.

With the exception of the parameters necessary to calculate the signal-to-noise ratio, each sensor is modeled only for geometric purposes by its position, azimuth boresight, and a limiting range, elevation, and azimuth. Each interceptor farm is modeled by its position and what type of interceptor is stationed there, the interceptor type referring to a particular set of flyout characteristics as specified by a table of flyout times vs tangent range and altitude.

Within the model defined above, DEANE can perform various computations. These computations are just that, leaving any interpretation of the results to be made by the user, not by DEANE. Typical examples would be: (a) calculating ideal observables on a trajectory over some time interval from specified sensors; (b) calculating when an object is first visible to a specific sensor as determined by the trajectory and a limiting range, azimuth, and elevation associated with that sensor; (c) calculating the earliest possible intercept and the intercept with the latest launch from some set of farms, subject to various restrictions such as a minimum intercept altitude, a maximum commit altitude, and the requirement that the launch must occur at least so many seconds after any one of a set of specified radars has been able to detect the object; and (d) determining a ground footprint of interceptor coverage subject to restrictions similar to those above.

Also available are "cookie-cutting" functions for investigating the deployment of sensors, farms, and Standard Metropolitan Statistical Areas (SMSA's). Typical calculations would be to determine which SMSA's are within a certain distance d_1 of any farm and within a certain distance d_2 of any sensor, or to determine which farms or sensors are within a certain distance of specified SMSA's.

As DEANE is intended for use in a time-shared environment, significant consideration was given to making the specification of program options as convenient for the user as possible. The resulting solution was that all the user input to DEANE is in the form of a command, or keyword, followed by the necessary input data needed by that particular command. Accordingly, DEANE can be considered as either a program or a special-purpose language, the latter being preferred for purposes of explanation, understanding, and usage.

II. SYSTEM OVERVIEW

An oversimplified view of a computer will help understand how DEANE works. We may consider the computer to be some organized collection of memory cells and a set of hardware that can perform various arithmetic and logical operations on these cells. Each cell is made accessible to the arithmetic/logical hardware by specifying a bit pattern address which at the user level is equivalent to specifying a mnemonic. Whatever is in one of these cells is unchanged each time the contents are read for computational purposes, and the contents are changed only when something else is written into the cell. Accordingly, there are two kinds of commands in DEANE: (a) data defining commands which cause a sequence of numbers to be read and stored in appropriate cells, overwriting the previous contents; and (b) action commands which initiate the desired calculations using the present values resident in the appropriate cells. Data defining commands are the only means by which certain values (e.g., trajectory launch site) can be changed. Action commands either produce output or determine values for variables which need be known only to the computer (e.g., the argument of the trajectory apogee).

A statement in DEANE consists of a command followed by a sequence of numbers and/or mnemonics. This data sequence varies in length and meaning for the different commands. The leading keyword of each statement must not be in quotes and, except where noted, the entire statement must be entered on one 72-space line. Either one comma and/or any number of blanks may be used to separate the entries within a statement. The units for the input for all the commands are identical; time is in seconds, distance is in kilometers, speed is in kilometers/second, angles are in degrees, angular rates are in degrees/second, latitude is in degrees north, and longitude is in degrees east.

To help clarify the above comments and to give insight into how computations can be achieved using the entire instruction set, Fig. II-1 is an example of some of the commands and how they might be used.

```
                                                          -3-13045
launch 38.0   110.0  ,    ,   foe
impact 40.8,-75.0 5. friend
parameter   0   1
kepler
  TOTAL FLIGHT TIME IS  2070.48 SECONDS.
altitude 70.0
  THE OBJECT IS AT ALTITUDE    70.00 AT TIME   -26.43.
parameter 1,2,10
kepler
  TOTAL FLIGHT TIME IS  1710.14 SECONDS.
altitude 70
  THE OBJECT IS AT ALTITUDE    70.00 AT TIME   -51.19.
stop
T=3.01/6.96 13:52:58
```

Fig. II-1. Example of things to come.

The first statement fixes the site named 'FOE', located at 38.0° north latitude, 110.0° east longitude, and 0 kilometers height as the launch site. The second statement specifies that impact is to be at 5.0 kilometers over 40.8° north latitude and −75.0° east longitude, and this site is given the name 'FRIEND'. Statement three indicates that a minimum energy trajectory is desired for a nonrotating earth, and the statement 'KEPLER' then causes calculation of the parameters specifying that trajectory and the line of output that follows. The next command produces the line of output indicating that the object is at 70.0 km altitude 26.43 seconds before impact. 'PARAMETER 1, 2, 10' sets the parameters for future trajectory definitions to be a

rotating earth with a specified impact velocity angle of 10°. 'KEPLER' then determines the trajectory using the launch site 'FOE', the impact site 'FRIEND', a rotating earth, and an impact velocity angle of 10°. 'ALTITUDE 70.0' then produces the following line of output, and 'STOP' terminates the execution.

In this example the commands 'LAUNCH', 'IMPACT', and 'PARAMETER' are data-defining commands. 'KEPLER' is an action command which determines a set of values which need be known only to the computer and produces one line of output. 'ALTITUDE' is an action command which serves only to produce output.

The complete list of instructions is presented in Secs.III through VII. The description of each command is separated into three sections: USE, which describes why the command is included in DEANE and how it might be used; RESPONSE, which describes the computer's response to the command; and ERROR, which describes syntactic and/or semantic limitations on the command.

For all the commands, the error message 'ILLEGAL INPUT INVOLVING THE SYMBOL "x" ' will be printed when DEANE's input routine recognizes an illegal representation for a number in a statement. The error message 'SOME INPUT VALUE IS UNACCEPTABLE. PLEASE RETYPE THIS LINE.' will be printed when the value of some input is syntactically unacceptable. Usually, all input up to the point of infraction is accepted, and all input including and past the illegal number is not accepted. The user response should be to retype the entire command in which the error occurred.

III. TRAJECTORY SPECIFICATION COMMANDS

LAUNCH lat long ht mnemonic

USE – This specifies the launch site of a trajectory.

RESPONSE – The launch site is set to 'lat' degrees north latitude, 'long' degrees east longitude, and 'ht' kilometers above the earth. Any 'mnemonic' entered after 'ht' is ignored by DEANE, but it may serve the user for purposes of identification. If 'ht' and 'mnemonic' are omitted, 'ht' is set to zero.

ERROR – None.

IMPACT lat long ht mnemonic

USE – This specifies the impact site of the Keplerian trajectory.

RESPONSE – The impact site is set to 'lat' degrees north latitude, 'long' degrees east longitude, and 'ht' kilometers above the earth. Any 'mnemonic' entered after 'ht' is ignored by DEANE, but it may serve the user for purposes of identification. If 'ht' and 'mnemonic' are omitted, 'ht' is set to zero.

ERROR – None.

PARAMETER rotate type value1 value2

USE – This command specifies the information additional to the launch and impact sites necessary to allow calculation of a Keplerian trajectory.

RESPONSE – If 'rotate' equals one, the earth is modeled as rotating; otherwise it is modeled as fixed. There are four different types of parameters which are indexed by 'type' and given values via 'value1'. The association is given in Table III-1.

TABLE III-1
INTERPRETATION OF DATA FOR COMMAND 'PARAMETER'

'type'	Interpretation of 'value1'
1	None – minimum energy calculation.
2	Angle between local horizontal and velocity vector at impact.
3	Velocity magnitude at impact. Since this leads to two possible trajectories, a positive value yields the lofted trajectory choice and a negative value yields the depressed trajectory choice.
4	Apogee height in kilometers.

If the trajectory specified is an orbit, 'value2' represents the number of complete orbits desired before satisfying the impact conditions.

ERROR – 'type' must be an integer between 1 and 4, and 'rotate' and 'value2' must be integers.

KEPLER and KEPLERP

USE – At any time, the user has access to only one trajectory for purposes of performing computations. This command provides one means of attaining such a working trajectory.

RESPONSE – The Keplerian trajectory as specified via the most recent values from 'LAUNCH', 'IMPACT', and 'PARAMETER' is determined. 'KEPLERP' will cause printing of various characteristics about the trajectory (longitude of the ascending node, inclination, argument of apogee, eccentricity, semi-latus rectum, semi-major axis, flight ground range, total time of flight, and the velocity, azimuth, and flight path angle at launch and impact). For 'KEPLER', only the total time of flight is printed. At the successful completion of either of these commands, the working trajectory is the one calculated, overwriting any previous working trajectory.

ERROR – There are several situations where the data specified via 'PARAMETER' are not compatible with the launch and impact sites. If the velocity impact angle is too large, a message will be printed and a trajectory with an apogee of 25,000 km is substituted. If the impact velocity is too small, a trajectory with minimum launch velocity is substituted and a note is printed to that effect. Should the impact velocity be too large, a message is printed and an appropriate substitution is made for the impact velocity. If computational problems arise, the trajectory calculation is abnormally terminated and a message to that effect is printed.

Example 1

Of recurring interest in many analyses is the relation of re-entry angle to ground range. Using only the DEANE commands introduced thus far, we can determine the re-entry angle as a function of trajectory ground range for a given burnout velocity. A sequence of commands to do this might be:

```
impact 48.0 −80.0 0.0
parameter 0 3 7.2
launch 70.0 120.0 200.0
keplerp
launch 67.0 120.0 200.0
keplerp
launch 64.0 120.0 200.0
keplerp
.
.
.
launch 19.0 120.0 200.0
keplerp
parameter 0 3 −7.2
launch 70.0 120.0 200.0
keplerp
launch 67.0 120.0 200.0
keplerp
.
.
.
```

The launch velocity, ground range, and impact angle printed as a result of 'KEPLERP' then specify the relationship of interest. (By conservation of energy, constant impact altitudes and velocities with constant launch altitudes give constant launch velocities.) Figure III-1 shows

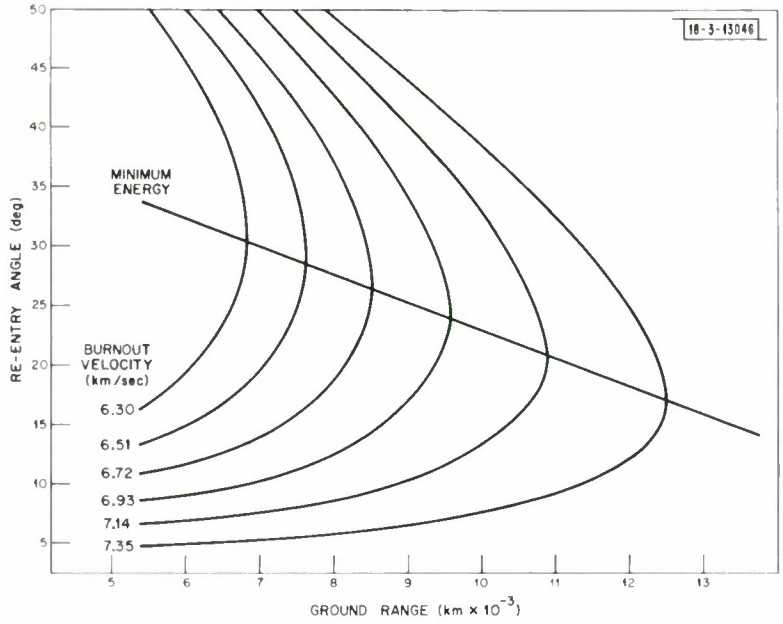

Fig. III-1. Re-entry angle vs ground range as calculated by DEANE.

such curves for several burnout velocities, from which the analyst could make observations about possible re-entry angles for a particular booster and ground range or about the effects of changing the payload weight and burnout velocity and hence range of re-entry angles.

EJECT time, delvx, delvy, delvz, alt

USE – This command provides another means of obtaining a working trajectory to model decoy ejection, tank displacement, or deorbiting.

RESPONSE – At the time 'time' before impact, the present working trajectory's velocity vectors are incremented by 'delvx', 'delvy', and 'delvz' km/sec and calculations then determine a new Keplerian trajectory. The impact point, velocity, angle, and time to impact of the resulting trajectory are printed, where impact is taken to be at the altitude 'alt'. The coordinate system used for specifying the velocity increments is seen in Fig. III-2; $\bar{x}$ is the inplane tangent to the old trajectory at the time 'time', $\bar{y}$ is the inplane normal to $\bar{x}$, and $\bar{z}$ is the right-hand normal to $\bar{x}$ and $\bar{y}$.

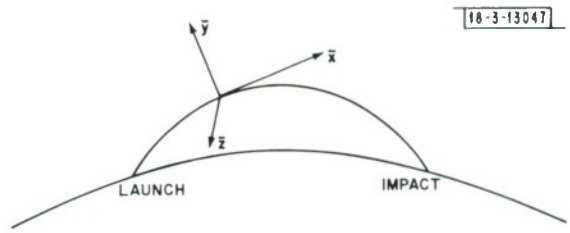

Fig. III-2. Coordinate system for input to 'EJECT'.

ERROR – If the new trajectory never reaches the altitude 'alt', an arbitrary acceptable altitude for impact is chosen, and a message is printed to this effect.

R_I_AXIS time, lat, long, alt

USE – This command aids the user in selecting input values for the command 'EJECT'.

RESPONSE – The vector (in the coordinate system sketched in Fig. III-2) along which small velocity increments at time 'time' will produce impact at 'lat' latitude, 'long' longitude, at a height 'alt' over the earth is determined and printed. Also printed are the magnitude and components of the minimum magnitude velocity increment necessary to achieve this displacement. The calculations are based on linear approximations and a nonrotating earth. They are thus valid only for desired impact points near that of the present trajectory.

ERROR – Because of the nature of the computation, if the working trajectory never reaches the altitude 'alt', the command is terminated.

Example 2

Suppose we desire to model a decoy whose impact is 0.5° (about 55 km) uprange from the impact point for a base trajectory. The sequence of DEANE commands with their output (seen in Fig. III-3) would achieve this. The first four commands specify and cause calculation of a base trajectory. The 'R_I_AXIS' command indicates the displacement can be achieved at time −1840 with a velocity increment along the vector (−.010, .022, 0)+c(−.207, 2.86, 0) where c can be arbitrarily chosen. Choosing the minimum velocity magnitude from 'R_I_AXIS' as input for 'EJECT' causes calculation of this trajectory which does indeed impact 0.5° uprange of the base trajectory.

-3-13048

```
launch 42.1 83.0 200.0
Impact 48.0 -97.0 0.0
parameter  0 2 20.0
kepler
  TOTAL FLIGHT TIME IS  1859.50 SECONDS.
r_i_axis -1840 48.5 -97.0 0.0
  DELVX = -1.010E-02 + -2.070E-01 TIMES C.
  DELVY =  2.215E-02 +  2.865E 00 TIMES C.
  DELVZ =  6.481E-06 +  2.821E-07 TIMES C.
  FOR MINIMUM VELOCITY MAGNITUDE OF  8.474E-03
    DELVX = -8.452E-03, DELVY = -6.107E-04, DELVZ =  6.478E-06
eject -1840 -8.452e-3 -6.107e-4 6.478e-6
  IMPACT AT  -0.00 KM ABOVE  48.501 DEG LAT AND  -97.000 DEG LONG
    WITH A SPEED OF  7.2771 KM/SEC AND AT AN ANGLE OF  20.01 DEGREES.
    FLIGHT TIME FROM VELOCITY INCREMENT IS 1829.37.
```

Fig. III-3. DEANE commands for Example 2.

To extend this example, the minimum velocity magnitude and associated change in flight time can be calculated as a function of the inplane displacement by repeating the 'R_I_AXIS' and 'EJECT' sequence for several displacements. The results of doing this for a 10,000-km ground range trajectory are seen in Fig. III-4 where, for all three re-entry cases, ejection is about 20 sec after 200-km altitude.

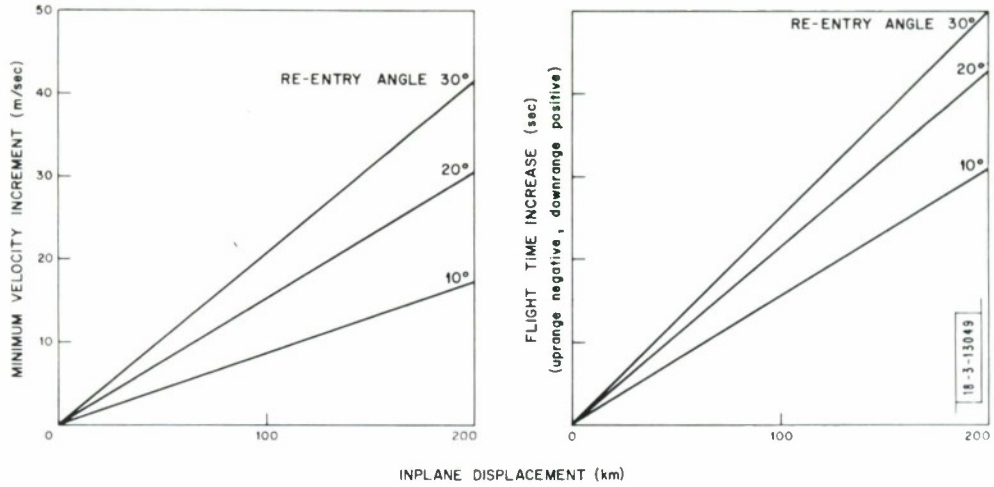

Fig. III-4. Minimum velocity increment and flight time increase vs inplane displacement as calculated by repetition of 'R_I_AXIS' and 'EJECT'.

BETA_TABLE β_1, h_1, β_2, h_2, β_3, h_3, β_4

USE – This command defines a ballistic coefficient (β) history as a function of altitude. This is necessary input for achieving trajectories whose re-entry portion is the result of integration.

RESPONSE – The approximation to the β history is defined by the piecewise step function shown in Fig. III-5. The β_i units are taken as pounds per square foot. If any 'h_i, β_{i+1}' pairs are omitted in the data sequence, 'h_i' is assumed to be zero.

ERROR – 'h_{i+1}' must be less than 'h_i'.

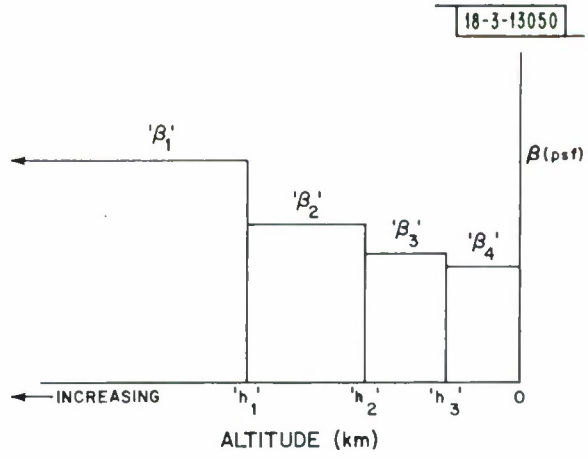

Fig. III-5. Interpretation of input data for 'BETA_TABLE'.

INTEGRATE alt_1, alt_2

USE – This is the third means by which working trajectories are attained, and it is intended to provide better modeling of the re-entry portion of a trajectory.

RESPONSE – The re-entry part of the present working trajectory is replaced by values attained from integrating the equations of motion using a real (tabular) atmosphere and

```
launch 42.1 83.0 200.0
impact 48.0 -97.0 3.0
parameter 1 2 30.0
keplerp
      LONG. OF NODE =   77.67 DEG., INCLINATION =  94.78 DEG., ARGUMENT OF APOGEE =  85.50 DEG.
      ECCENTRICITY = 0.5147, SEMI-LATUS RECTUM = 4109.09 KM, SEMI-MAJOR AXIS =  5589.93 KM.
      FLIGHT GROUND RANGE =  9958.13 KM, TOTAL FLIGHT TIME =  2312.32 SECONDS.
      LAUNCH VELOCITY =  7.0667, LAUNCH AZIMUTH =   -6.45, LAUNCH GAMMA = 29.42
      IMPACT VELOCITY =  7.3269, IMPACT AZIMUTH = -172.85, IMPACT GAMMA = 30.00
beta_table 600.0
integrate 100 3
  IMPACT AT   2.99 KM ABOVE 48.007 DEG LAT AND  -97.038 DEG LONG
    WITH A SPEED OF  0.7073 KM/SEC AND AT AN ANGLE OF  33.91 DEGREES.
    TIME TO   2.99 KM ALTITUDE FROM 100.0 KM INCREASED BY    9.55 SECONDS DUE TO SLOWDOWN.
parameter 1 2 10.0
kepler
  TOTAL FLIGHT TIME IS  1530.27 SECONDS.
integrate 100 3.0
  IMPACT AT   2.99 KM ABOVE 48.355 DEG LAT AND  -97.138 DEG LONG
    WITH A SPEED OF  0.2883 KM/SEC AND AT AN ANGLE OF  50.08 DEGREES.
    TIME TO   2.99 KM ALTITUDE FROM 100.0 KM INCREASED BY   43.72 SECONDS DUE TO SLOWDOWN.
```

Fig. III-6. DEANE commands for Example 3.

the ballistic coefficient history specified via 'BETA_TABLE'. The integration starts at 'alt_1' km altitude and terminates at 'alt_2' km altitude with the initial state vector determined from the Keplerian specification of the working trajectory. As drag can significantly alter the impact point of such a trajectory from the vacuum impact, the new impact point, speed at impact, and impact velocity angle with the local horizontal are printed. Also printed is the time increase between 'alt_1' and 'alt_2' km altitude due to the integrated trajectory.

ERROR – If the integrated trajectory's time to 'alt_2' km from 'alt_1' km altitude is greater than 200 sec, the integration is terminated before completion and a message is printed to that effect.

Example 3

The slowdown time is often of interest in analyses and can be calculated using the two commands presented above. Figure III-6 shows a set of DEANE commands and their output which indicate that for a 10,000-km ground range trajectory and a constant β of 600 there are 9.6 sec of slowdown between 100- and 3-km altitude for a 30° re-entry trajectory. For a 10° re-entry trajectory, there are 43.7 sec of slowdown. By executing 'INTEGRATE' with several different constant β's and three different re-entry angles, the relationships shown in Fig. III-7 may be determined.

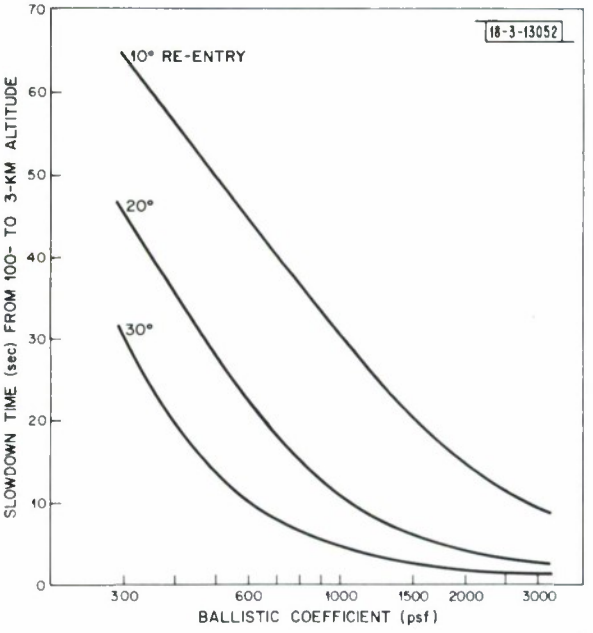

Fig. III-7. Re-entry slowdown time vs ballistic coefficient as calculated by repetition of 'BETA_TABLE' and 'INTEGRATE'.

IV. TRAJECTORY OBSERVATION COMMANDS

PRINT φ_1 φ_2 $\varphi_3 \cdots$ φ_r END

USE – This command allows the user to specify which sensor observables are to be printed.

RESPONSE – The mnemonics 'φ_1, φ_2, φ_3, $\cdots$ φ_r' specify what radar observables are to be printed when the request for these values is made. This command may extend over more than one line and is terminated by the mnemonic 'END'. The list of mnemonics and their interpretations is given in Table IV-1, and the default set for this command is the top 11 in this table.

TABLE IV-1
INTERPRETATION OF INPUT MNEMONICS FOR 'PRINT'

Mnemonic	Interpretation
TIME	time in seconds
RANGE	sensor range in kilometers
RDOT	sensor range rate in km/sec
RDDOT	sensor range acceleration in km/sec^2
ELE	elevation in degrees
ELEDOT	elevation rate in deg/sec
AZIMUTH	azimuth in degrees
AZIDOT	azimuth rate in deg/sec
ALTITUDE	altitude
GRANGE	ground range from sensor in kilometers
VAANGLE	velocity aspect angle in degrees
BAANGLE	body aspect angle in degrees
SIGMA	body cross section in square meters
S/N	signal/noise ratio in decibels
LAT	latitude of object in degrees north
LONG	longitude of object in degrees east

ERROR – If any mnemonic entered is not in Table IV-1, a message to that effect will be printed and that mnemonic will be ignored. If too many mnemonics are listed (as limited by 132 characters for the output) a message will be printed and the command will be terminated.

POINTING lat, long, alt, rate, angle

USE – This command sets up the parameter values for modeling spin stabilized orientation with coning to allow computation of the body aspect angle.

RESPONSE – A vector in inertial space pointing toward 'lat' degrees north latitude and 'long' degrees east longitude on the earth's surface from the trajectory point specified by 'alt' is established as the axis of orientation. If 'alt' is positive, the trajectory point

is 'alt' km altitude on the impact side of apogee; if it is negative, the point is |'alt'| km altitude on the launch side of apogee. The object is then given a coning rate of 'ratc' dcg/sec at an angle of 'angle' degrees with the axis.

ERROR – None.

TUMBLE rate

USE – This sets up the parameter value for modeling inplane tumbling to allow computation of the body aspect angle.

RESPONSE – The body axis is established in the trajectory plane and is given a tumbling rate of 'rate' dcg/sec.

ERROR – The body orientation may be specified either by 'POINTING' or by 'TUMBLE', but not by both. Whichever is more recent will determine which is used.

SIGMA_TABLE i, number

USE – This command is the means by which a cross-section table for an axisymmetric body is defined to allow computation of cross section by linear interpolation.

RESPONSE – This command puts DEANE in a mode to receive a list of 'number' $(\Theta_j, \sigma(\Theta_j))$ pairs which define cross-section table 'i' starting on the next input line. These pairs should be entered just as a sequence of numbers Θ_1, $\sigma(\Theta_1)$, Θ_2, $\sigma(\Theta_2)$, ... and they may extend over several input lines. $\sigma(\Theta_j)$ is interpreted as square meters. This table naturally overwrites any previous cross-section table of the same index 'i'.

ERROR – There is an upper limit of 20 pairs and 'number' must be an integer. The list must be ordered beginning with 0° and terminating with $\sigma(180°)$. The angles must be between 0° and 180°, and 'i' must be between 1 and 5.

RV_TYPE i

USE – This command specifies which of the cross-section tables defined is to be used for attaining cross sections.

RESPONSE – The cross-section table to be used is noted as cross-section table 'i'.

ERROR – 'i' must be an integer between 1 and 5.

RADAR_TYPE n, a, b, c, Θ, f, E, L, T

USE – For determining the S/N sensor, parameters need to be known; and for determining the geometric limits as to what the sensor can see (for purposes of launching interceptors, for example), additional parameters need to be specified.

'RADAR_TYPE' achieves this by associating these values with a number indicating a radar type, and the user can then associate a radar type with each radar in a deployment.

RESPONSE – Radar type 'n' is defined to be limited by 'a' kilometers range, ±'b' degrees azimuth off azimuth boresight, 'c' degrees lower elevation limit, and to have a beamwidth of 'Θ' degrees, frequency 'f' megahertz, 'E' joules radiated energy per pulse, 'L' decibel loss, and an operating temperature of 'T' degrees Kelvin. (A filled aperture is assumed for the S/N calculation.) Type 1 is reserved for a default radar

having 5500-km range, ±180° azimuth, 1° horizon, 1° beamwidth, 500-MHz frequency, 125,000 joules/pulse, 4-dB loss, and an operating temperature of 1000°K.

ERROR – 'n' must be an integer between 2 and 5.

NEW_SENSORS

USE – This command allows a list of sensors to be defined.

RESPONSE – DEANE enters a mode to receive a list of sensor mnemonics, latitudes, longitudes, heights, azimuth boresights, and types. Each set of data defining a sensor is entered on one line (in the order above) and the list is terminated by typing the mnemonic 'END'. Default value for the height is 0.0 km, for the boresight is 0.0°, and for the type is 1 if the numerical entries on a line are exhausted before satisfying all the values. Any previous sensor list is destroyed at the beginning of this command.

ERROR – The 'type' must be an integer between 1 and 5. If the 'mnemonic' entered for the addition is already being used for another sensor, an error message will say so and that input line will be ignored. Limits of 12 characters for the mnemonics and a total of 40 sensors are enforced.

USE_SENSORS

USE – This command selects which sensors from the total sensor list are to be used either for purposes of observables printout, for aiding in conducting an intercept, or for "cookie-cutting" functions. It also allows for some degree of editing the list of sensors.

RESPONSE – DEANE enters a mode to receive a list of sensor mnemonics, one per line. This list is terminated by entering the mnemonic 'END'. If the mnemonic is found in the existing sensor list, that sensor is marked as in use. If a previously undefined mnemonic is followed by a latitude, longitude, and, optionally, height, boresight, and type, the sensor so specified will be added to the existing sensor list without changing any of the previous entries, and this sensor will be marked as in use. If a previously defined sensor mnemonic is followed by a latitude, longitude, and, optionally, height, boresight, and type, the said values for the named sensor will be changed to the values entered and the sensor will be marked as in use. If the mnemonic entered is 'ALL', all the sensors in the sensor list will be marked as in use and the command will be terminated.

ERROR – There is a limit of 12 characters for each mnemonic and a total of 40 sensors. Any 'type' entered must be an integer between 1 and 5. If the mnemonic is not found in the sensor list and is not followed by defining data, a comment to that effect is printed.

TIMES t_1 t_2 dt

USE – This is the command which produces the printout of the radar observables.

RESPONSE – For each sensor listed in 'USE_SENSORS', values for the quantities listed in 'PRINT' are printed using the working trajectory. Each line of output corresponds to a specific time, and the time interval covered by the printout is from 't_1' to 't_2' in increments of 'dt'. The calculation of the body aspect angle is determined by either

-3-13053

```
launch 70 120 300
impact 39 -77 300
parameter 1 4 304 1
kepler
  TOTAL FLIGHT TIME IS  6475.12 SECONDS.
  ORBIT ATTAINED WITH PERIOD =  5406.176
print time lat long altitude end
times 0 -210 -50
  NO SENSORS ARE IN USE.
use_sensors
washington 39 -77
end
times 0 -210 -50
  OUTPUT FOR SENSOR WASHINGTON
    TIME  ALTITUDE      LAT      LONG
     0.0     299.99    39.00   -77.00
   -50.0     300.68    42.30   -76.51
  -100.0     301.31    45.61   -75.98
  -150.0     301.88    48.91   -75.42
  -200.0     302.39    52.21   -74.81
times 6275.1 6576 50
  OUTPUT FOR SENSOR WASHINGTON
    TIME  ALTITUDE      LAT      LONG
  6275.1     302.39    52.21   -74.81
  6325.1     301.88    48.91   -75.42
  6375.1     301.31    45.61   -75.98
  6425.1     300.69    42.31   -76.51
  6475.1     299.99    39.00   -77.00
  6525.1     299.24    35.70   -77.47
  6575.1     298.43    32.39   -77.91
```

Fig. IV-1. DEANE commands for Example 4.

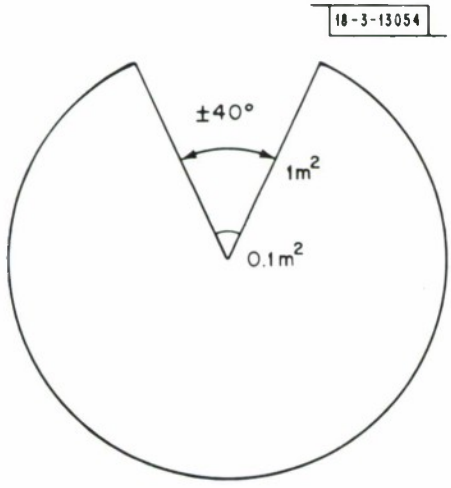

Fig. IV-2. RV cross-section model for Example 5.

'TUMBLE' or 'POINTING', depending on which one is more recent. The body cross section is obtained by a linear interpolation in the cross-section table of type 'RV_TYPE'. The signal-to-noise ratio is determined using the parameters describing the radar type of each sensor in use as specified via 'RADAR_TYPE'. For this command, the times may be positive or negative. A minus time is time to impact, and a positive time is time after launch for 'KEPLER' and 'INTEGRATE' and time after ejection for 'EJECT'. 't_1' and 't_2' may stand in any comparative relationship.

ERROR – This command is terminated if either of the following situations exists:

(1) Either BAANGLE, SIGMA, or S/N is requested by 'PRINT' and orientation is determined by 'POINTING', but the altitude set in 'POINTING' is greater than apogee.

(2) Either SIGMA or S/N is requested by 'PRINT' but the cross-section table type specified by 'RV_TYPE' has not been defined.

The output for any particular sensor is not generated if the parameters for that radar type have not yet been specified via 'RADAR_TYPE' and S/N is requested by 'PRINT'.

Example 4

To see how the positive time convention is used, consider the set of DEANE commands shown in Fig. IV-1 where the first four commands define an orbital trajectory which passes over the impact point after one complete period. The 'PRINT' command states what is to be printed, and a first attempt at 'TIMES' yields an error message. Even though all the data requested is sensor independent, some sensor must be defined as in use. This is done via the next three lines where the editing feature of 'USE_SENSORS' is used to define and mark as in use a sensor at the impact point. The execution of 'TIMES 0 -210 -50' gives the requested output. The next execution of 'TIMES' gives equivalent information but in a different time order and with the positive time convention. The time 6475.1 is equivalent to time 0, the time 6275.1 is equivalent to time -200, etc. Note that data beyond the "impact" point may be obtained by using the positive time convention as seen in this second execution of 'TIMES'. This use of positive time is the motivation for printing the flight time for 'KEPLER' and 'EJECT'.

Example 5

Probably the most frequent type of calculation desired is simply the trajectory observables for a specified sensor deployment and trajectory geometry. Consider the deployment of two acquisition radars ~1000 km apart near the northern border of the continental United States and an RV cross-section model similar to that shown in Fig. IV-2. (The radar range at which detection is made depends on which cross section is seen by the radars.) The question is: "Can such an RV on a minimum energy trajectory from China targeted at one radar be oriented such that detection by either radar will be made on the smaller cross section and, hence, later in time than if detection could be made on the larger cross section?" Assume that the radar has a 1° horizon limit and the sensitivity and detection logic are such that detection will occur at 2000-km range on the small cross section.

The set of DEANE commands and their associated output seen in Fig. IV-3 indicate that such an orientation does exist. The first six lines in the figure define the sensors and the next four lines define the trajectory. The cross-section table is then defined and associated with the RV via 'RV_TYPE'. Interesting observables are specified in 'PRINT' and then the orientation is

```
new_sensors
radar1 49  -110
radar2 49  -95
end
use_sensors
all
impact 49 -110
launch 30 100
parameter 1 1
kepler
  TOTAL FLIGHT TIME IS  2041.67 SECONDS.
sigma_table 1 4
0 .1 40 .1 41 1.0 180 1.0
rv_type 1
print time range ele azimuth altitude grange baangle vaangle sigma end
pointing 49 -102 1000
times -600 0 100
  OUTPUT FOR SENSOR RADAR1
    TIME      RANGE       ELE   AZIMUTH   ALTITUDE    GRANGE   VAANGLE   BAANGLE        SIGMA
  -600.0    3791.81      0.76    -17.98    1085.57   3398.49     21.33      9.46    1.000E-01
  -500.0    3231.42      3.94    -17.64     967.44   2897.79     17.86      9.11    1.000E-01
  -400.0    2644.47      7.14    -17.32     824.08   2378.37     14.42      9.89    1.000E-01
  -300.0    2029.47     10.39    -17.03     655.50   1835.29     11.01     11.62    1.000E-01
  -200.0    1384.83     13.70    -16.75     461.77   1262.78      7.65     13.98    1.000E-01
  -100.0     708.91     17.09    -16.50     243.11    653.86      4.45     16.77    1.000E-01
     0.0       0.02     12.25    -20.14       0.0       0.02      8.19     10.75    1.000E-01
  OUTPUT FOR SENSOR RADAR2
    TIME      RANGE       ELE   AZIMUTH   ALTITUDE    GRANGE   VAANGLE   BAANGLE        SIGMA
  -600.0    4163.77     -2.50    -21.39    1085.57   3770.54     25.04      5.91    1.000E-01
  -500.0    3623.52      0.18    -23.83     967.44   3290.21     23.39      8.18    1.000E-01
  -400.0    3065.87      2.62    -27.28     824.08   2801.38     23.17     12.29    1.000E-01
  -300.0    2496.85      4.60    -32.48     655.50   2306.78     25.46     18.13    1.000E-01
  -200.0    1932.27      5.51    -40.97     461.77   1817.96     32.08     26.90    1.000E-01
  -100.0    1417.92      3.64    -56.27     243.11   1373.68     46.75     42.06    1.000E 00
     0.0    1091.86     -4.91    -84.33       0.0    1093.20     76.09     70.29    1.000E 00
```

Fig. IV-3. DEANE commands for Example 5.

defined in 'POINTING'. Printout from 'TIMES' then shows that first detection will be made on the small cross section at 2000-km range by the radar under attack.

ALTITUDE alt

USE – This command makes available to the user the time at which the object on the working trajectory is at a certain altitude. It is intended as a convenience for selecting times of interest for input to 'TIMES'.

RESPONSE – The time at which the object on the working trajectory is at altitude 'alt' is printed. If 'alt' is positive, this is taken at the impact side of apogee; if 'alt' is negative, this is taken as |'alt'| kilometers altitude on the launch side of apogee.

ERROR – If |'alt'| is greater than the trajectory apogee or if |'alt'| is lower than the trajectory's minimum altitude, a message to this effect is printed and the command is terminated.

VISIBLE mnemonic

USE – This command makes available to the user the earliest and latest times at which a sensor can view a trajectory. Like 'ALTITUDE', this is intended as a convenience for selecting input times for 'TIMES'.

RESPONSE – 'mnemonic' is taken as the name of a sensor in the sensor list and it accordingly has a position, boresight, and a limiting azimuth, elevation, and range as specified through the sensor type. This is sufficient information to calculate the times at which this sensor can first and last view an object on the working trajectory. If the sensor can view the trajectory, these times and the associated ranges, azimuths, and elevations are printed. If the sensor cannot view the trajectory, a comment is printed to indicate this. If 'mnemonic' is 'ALL', the above information is printed for all the sensors marked as in use.

ERROR – If 'mnemonic' is not found in the sensor list and is not 'ALL', a message will so indicate and the command will be terminated. Also, if the corresponding sensor parameters are undefined, a message will so indicate.

Example 6

To see how 'VISIBLE' can be used, in Example 5 we needed observables printout over some time interval. With the trajectory and radar geometry of Example 5, 'VISIBLE' may be used as seen in Fig. IV-4 to indicate that the RV first crosses the radar horizon at −592 sec. Thus, −600 is a good time to start requesting observables. If the radar had a 2° horizon instead of the 1° default horizon, we would have to define this via 'RADAR_TYPE' as seen in Fig. IV-4. . 'VISIBLE' then indicates a horizon crossing at −560 sec.

```
impact 49 -110
launch 30 100
parameter 1 1
kepler
  TOTAL FLIGHT TIME IS  2041.67 SECONDS.
use_sensors
radar1 49 -110
radar2 49 -95
end
visible radar1
  FIRST VISIBLE AT TIME  -592.45 WITH RANGE =  3750.37, AZIMUTH =  -17.95, AND ELEVATION =    1.00.
   LAST VISIBLE AT TIME     0.0  WITH RANGE =     0.02, AZIMUTH =  -20.14, AND ELEVATION =   12.25.
visible radar2
  FIRST VISIBLE AT TIME  -467.59 WITH RANGE =  3444.52, AZIMUTH =  -24.81, AND ELEVATION =    1.00.
   LAST VISIBLE AT TIME   -56.61 WITH RANGE =  1239.45, AZIMUTH =  -66.59, AND ELEVATION =    1.00.
use_sensors
radar1 49 -110 0 0 2
radar2 49 -95 0 0 2
end
radar_type 2 5500 180 2
visible all
  FOR SENSOR RADAR1
  FIRST VISIBLE AT TIME  -560.89 WITH RANGE =  3575.70, AZIMUTH =  -17.84, AND ELEVATION =    2.00.
   LAST VISIBLE AT TIME     0.0  WITH RANGE =     0.02, AZIMUTH =  -20.14, AND ELEVATION =   12.25.
  FOR SENSOR RADAR2
  FIRST VISIBLE AT TIME  -426.66 WITH RANGE =  3216.01, AZIMUTH =  -26.23, AND ELEVATION =    2.00.
   LAST VISIBLE AT TIME   -70.32 WITH RANGE =  1291.15, AZIMUTH =  -63.02, AND ELEVATION =    2.00.
```

Fig. IV-4. DEANE commands for Example 6.

V. TRAJECTORY INTERCEPTION COMMANDS

FLYOUT_TABLE i, dela, delr, numa, numr

USE – This command provides the means for defining interceptor flyout characteristics.

RESPONSE – The flyout reach table to be entered should resemble that shown in Fig. V-1, where the entries are the flyout times in seconds required to make an intercept at the specified tangent range from the farm and at the specified altitude above the earth. For (tangent range, altitude) pairs the interceptor cannot reach, 1.0E7 should be entered for the flyout time. The interceptor type is 'i', 'dela' is the table altitude increment in kilometers, 'delr' is the table tangent range increment in kilometers, 'numa' is the

18-3-13057

TANGENT RANGE \ ALTITUDE	0	dela	2 × dela	(numa−1) × dela
0	$t_{0,0}$	$t_{0,1}$	$t_{0,2}$	$t_{0,numa-1}$
delr	$t_{1,0}$	$t_{1,1}$	$t_{1,2}$	$t_{1,numa-1}$
2 × delr				
(numr−1) × delr	$t_{numr-1,0}$			$t_{numr-1,numa-1}$

Fig. V-1. General form of interceptor reach table.

number of altitude entries, and 'numr' is the number of tangent range entries. After reading these numbers, DEANE expects a sequence of 'numa' × 'numr' numbers starting on the next input line to correspond to the table entries. The order of the entries is row by row, i.e., the sequence would be $t_{0,0}$, $t_{0,1}$, ..., $t_{0,numa-1}$, $t_{1,0}$, ... The sequence may extend over any number of input lines, and it overwrites any previous flyout table of type 'i'.

ERROR – 'i' must be an integer between 1 and 5, and 'numa' and 'numr' must be integers no greater than 30.

NEW_FARMS

USE – This command allows a list of interceptor farms to be defined.

RESPONSE – DEANE enters a mode to receive a list of farm mnemonics, latitudes, longitudes, heights, populations, and types. Each set of data defining a farm is entered on one line in the order designated above, and the list is terminated by typing the mnemonic 'END'. Default value for the height is 0 km, for the population (i.e., number of interceptors at the farm) is 0, and for the type is 1 if the numerical entries of a line are exhausted before satisfying all the values. Any previous farm list is destroyed at the beginning of this command.

ERROR – The 'type' must be an integer between 1 and 5. If the 'mnemonic' entered for the addition is already being used for another farm, an error message will say so and

that input line will be ignored. Limits of 12 characters for mnemonics and a total of 40 farms are enforced.

USE_FARMS

USE – This command selects which farms from the total farm list are to be used as the origin of interceptors. It also permits editing the farm list to some extent.

RESPONSE – DEANE enters a mode to receive a list of farm mnemonics, one per line, and this list is terminated by entering the mnemonic 'END'. If the mnemonic is found in the farm list, that farm is marked as in use. Should an undefined mnemonic be followed by a latitude, longitude, and, optionally, height, population, and type, the farm so specified will be added to the farm list without changing any of the previous entries and it will be marked as in use. If a defined farm is followed by a latitude, longitude, and, optionally, height, population, and type, the said values for that farm will be changed to the values entered and the farm will be marked as in use. If the mnemonic entered is 'ALL', all the farms in the farm list will be marked as in use and the command will be terminated.

ERROR – There is a limit of 12 characters for each mnemonic and a total of 40 farms. Any 'type' entered must be an integer between 1 and 5. If the mnemonic is not found in the farm list and is not followed by defining data, a comment to that effect is printed.

TRACKING_SENSORS

USE – This command defines which sensors are to be used for modeling tracking to intercept.

RESPONSE – DEANE enters a mode to receive a list of sensor mnemonics, one per line, terminated by the mnemonic 'END'. Each sensor entered is then noted as a site from which tracking can be done.

ERROR – If any mnemonic entered is not in the sensor list, a message to that effect is printed and that mnemonic is ignored.

RESTRICT low, high, accuracy, time1, time2

USE – This command specifies values for restrictions to be placed on intercepts.

RESPONSE – 'low' is the value of the minimum intercept altitude, 'high' is the altitude below which the object must be when the interceptor is launched (i.e., maximum commit altitude), and 'accuracy' is the time accuracy desired in determining the interceptor launch times. 'time1' is the required time delay before the interceptor launch after the first possible detection by any sensor marked as in use, and 'time2' is the length of the time interval immediately prior to the intercept during which the object must be visible to some sensor being used for tracking.

ERROR – All entries must be positive, and 'low' must be less than 'high'.

INTERCEPT

USE – This command provides information concerning intercepts of an object on the working trajectory.

RESPONSE – The times of the intercepts with the earliest intercept time and with the latest interceptor launch time are printed together with the farms used and the intercept altitudes and ranges from the farms. The data defining acceptable intercepts are specified via 'RESTRICT', 'USE_SENSORS', 'RADAR_TYPE', 'USE_FARMS', 'FLYOUT_TABLE', and 'TRACKING_SENSORS'. 'USE_FARMS' defines a set of farms from which interceptors can be launched; the type of interceptor at a farm references a set of flyout characteristics specified via 'FLYOUT_TABLE', which then define the performance of interceptors launched from that farm. Intercepts must occur at or before the minimum intercept altitude specified in 'RESTRICT', and interceptor launches must occur at or after the maximum commit altitude set in 'RESTRICT'. If there are no sensors marked as in use and there are no tracking sensors defined, the above constitute the only restrictions on intercepts. If there are some sensors in use, before any interceptor launch occurs some sensor of those marked in use via 'USE_SENSORS' must have been able to detect the object as limited by the elevation, azimuth, and detection range associated with those sensors via 'RADAR_TYPE'. The earliest acceptable interceptor launch is then 'time1' sec after this radar detection where 'time1' is set in 'RESTRICT'. This allows modeling the time needed to insure acquisition in the surveillance pattern plus the time needed to establish a track sufficiently good to permit committing an interceptor. If there are some sensors being used for tracking as specified in 'TRACKING_SENSORS', one of these sensors must be able to track to intercept as modeled by being able to view the object during the 'time2' sec just prior to the intercept, where 'time2' is set in 'RESTRICT'.

If no acceptable intercepts can be made, a 4-bit code is printed to give insight into the situation. The first (leftmost) bit is 0 if the earliest launch is determined by the specified maximum commit altitude, and it is 1 if the acquisition/track restriction limits the earliest launch. The next bit is 0 if the lowest intercept altitude is determined by the minimum intercept altitude set in 'RESTRICT', and is 1 if it is determined by the tracking to intercept restriction. The third bit is 0 if an interceptor cannot physically reach the trajectory between the two limits set above. If one interceptor can reach the trajectory but the flight time is too long to be an acceptable intercept, the third bit is 1. If the final bit is 0, this means that no intercept can be made with intercept above the lowest acceptable altitude and launch after the earliest acceptable time; if the final bit is 1, such intercepts can be made but are not acceptable due to a lack of the required visibility modeling track to intercept. For example, the code "0000" would indicate that the interceptor does not have enough performance to make any intercepts within the launch-intercept altitude band specified in 'RESTRICT'; the code "1010" would indicate that the minimum intercept altitude set in 'RESTRICT' specifies the lowest acceptable intercept altitude and intercepts can be made above this altitude, but none can be made which have a launch after the earliest acceptable launch time as determined by the acquisition/track restriction.

ERROR – If no farms are in use or the flyout characteristics for any farm in use are not defined, a message will so indicate and the command will be terminated. If radar restrictions are included and the radar parameters for any such radar are not defined, a message will be printed to this effect and the command will be terminated.

Example 7

Consider the case of a defense site in the central United States which consists of a radar and an interceptor farm. The radar must provide its own surveillance and the threat of interest is from China with an assumed RV cross section such that the radar can detect it at 4000-km range. The radar has a 1.5° horizon and, because of a narrow threat corridor, we assume it has full azimuth coverage. (This assumption will speed execution.) It is further assumed that 60-sec time delay after first possible geometric detection will give guaranteed detection and enough initial tracking to permit an interceptor launch. The (idealized) exoatmospheric interceptor at the site has the reach curves shown in Fig. V-2, and it has a minimum intercept altitude of 75 km due to warhead considerations. What we want to know is, assuming threats on the radar, when can intercepts be made?

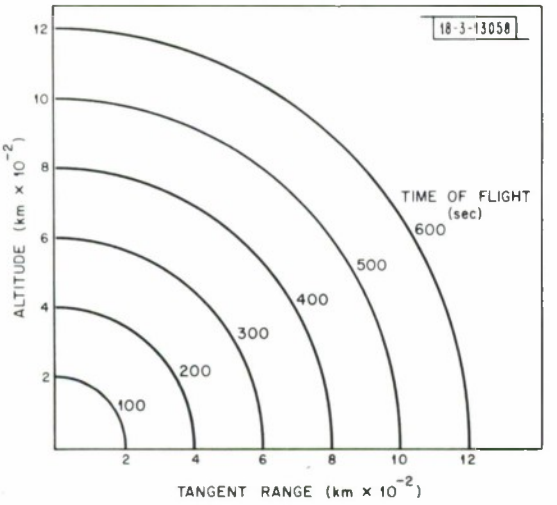

Fig. V-2. Interceptor reach curves for Example 7.

The DEANE commands seen in Fig. V-3 provide the answers to this question. (The "#" is used as a break character to mark the start of a new DEANE line for processing while staying on the same teletype line.) The first line in Fig. V-3 defines the radar deployment with pertinent characteristics, and the second line defines the farm deployment. The next five lines define the interceptor performance, and then it is specified that there are no tracking to intercept radars. (In this geometry, the tracking to intercept visibility requirement would not be a limiting factor so we neglect it to speed execution.) Now the restrictions are set. The minimum intercept altitude is set to 75 km and the maximum commit altitude is set very large so that the radar detection will be limiting. A 2-sec accuracy is sufficient, and the time delay of 60 sec is entered.

With all this necessary data defined, the next line defines and calculates a minimum energy trajectory impacting at the site. 'INTERCEPT' causes the next four lines of output. Because of the geometry here, the latest launch gives the latest intercept which is at 75-km altitude. Execution of 'VISIBLE' shows that the earliest intercept corresponds to interceptor launch 60 sec after first geometric visibility. It is interesting to note that the latest launch occurs before the earliest intercept so that the system could not attempt one intercept, evaluate the intercept, and then launch a second interceptor (i.e., this system has no shoot-look-shoot capability against a minimum energy trajectory). The trajectory is then changed to 10° re-entry and 'INTERCEPT' says that no intercepts can be made and the code is "1010" (discussed in command definition for 'INTERCEPT'). This is understandable for two reasons: first, detection will be horizon limited

```
use_sensors # radar1 37 -97 0 0 2 # end # radar_type 2 4000 180 1.5
use_farms # farm1 37 -97 0 0 1 # end
flyout_table 1 200 200 7 7
  0 100 200 300 400 500 600 100 141 224 316 412 510 1.0e7
200 224 283 360 447 540 1.0e7 300 316 360 425 500 580 1.0e7
400 412 447 500 570 1.0e7 1.0e7 500 510 540 580 1.0e7 1.0e7 1.0e7
600 1.0e7 1.0e7 1.0e7 1.0e7 1.0e7 1.0e7
tracking_sensors # end
restrict 75 10000 2.0 60.0 0
impact 37 -97 # launch 38 110 200 # parameter 1 1 # kepler
  TOTAL FLIGHT TIME IS  2142.81 SECONDS.
intercept
  EARLIEST INTERCEPT AT  -106.3 SECS AT AN ALTITUDE OF  254.8 AND A RANGE OF  738.5 FROM FARM FARM1
    LAUNCH AT  -499.1 SECS CORRESPONDING TO A COMMIT ALTITUDE OF  981.9.
  LATEST LAUNCH AT  -150.9 SECS CORRESPONDING TO A COMMIT ALTITUDE OF  354.5 KM.
    INTERCEPT AT   -30.3 SECS AT AN ALTITUDE OF   75.0 AND A RANGE OF  211.2 FROM FARM FARM1
visible radar1
  FIRST VISIBLE AT TIME  -562.26 WITH RANGE =  3674.75, AZIMUTH =  -12.75, AND ELEVATION =    1.50.
   LAST VISIBLE AT TIME    -0.01 WITH RANGE =     0.06, AZIMUTH =  -14.28, AND ELEVATION =   12.26.
parameter 1 2 10.0 # kepler
  TOTAL FLIGHT TIME IS  1760.53 SECONDS.
intercept
  NO INTERCEPTS ARE ACCEPTABLE.  CODE IS 1010.
parameter 1 2 30.0 # kepler
  TOTAL FLIGHT TIME IS  2778.66 SECONDS.
Intercept
  EARLIEST INTERCEPT AT  -121.8 SECS AT AN ALTITUDE OF  443.1 AND A RANGE OF  793.7 FROM FARM FARM1
    LAUNCH AT  -577.6 SECS CORRESPONDING TO A COMMIT ALTITUDE OF 1717.8.
  LATEST LAUNCH AT  -107.7 SECS CORRESPONDING TO A COMMIT ALTITUDE OF  394.2 KM.
    INTERCEPT AT   -19.8 SECS AT AN ALTITUDE OF   75.0 AND A RANGE OF  129.5 FROM FARM FARM1
```

Fig. V-3. DEANE commands for Example 7.

-3-13060

```
use_sensors
radar1 37 -97 0 0 2
end
radar_type 2 4000 180 1.5
use_farms
farm1 37 -97 0 0 1
end
flyout_table 1 200 200 7 7
  0 100 200 300 400 500 600
100 141 224 316 412 510 1.0e7
200 224 283 360 447 540 1.0e7
300 316 360 425 500 580 1.0e7
400 412 447 500 570 1.0e7 1.0e7
500 510 540 580 1.0e7 1.0e7 1.0e7
600 1.0e7 1.0e7 1.0e7 1.0e7 1.0e7 1.0e7
tracking_sensors
end
restrict 75 10000 2.0 60.0 0
launch 38 83 200
parameter 0 2 20
contour 10 180 25 -20 0
  THE CENTER IS  37.000 LAT AND  -97.000 LONG.
    PSI       RANGE        LAT        LONG     CODE
    0.0       425.0      40.82      -97.00     1010
   10.0       425.0      40.76      -96.12     1010
   20.0       450.0      40.79      -95.17     1010
   30.0       450.0      40.47      -94.34     1010
   40.0       475.0      40.22      -93.41     1010
   50.0       525.0      39.94      -92.29     1010
   60.0       575.0      39.44      -91.20     1010
   70.0       625.0      38.73      -90.23     1010
   80.0       725.0      37.86      -88.86     1010
   90.0       825.0      36.64      -87.75     1010
  100.0       950.0      35.07      -86.71     1010
  110.0      1025.0      33.39      -86.62     0000
  120.0      1100.0      31.62      -86.94     0000
  130.0      1200.0      29.68      -87.50     0000
  140.0      1375.0      27.19      -88.10     0000
  150.0      1700.0      23.47      -88.74     0000
  160.0      2150.0      18.64      -90.14     0000
  170.0      2625.0      13.69      -92.90     0000
  180.0      3125.0       8.91      -97.00     0000
```

Fig. V-4. DEANE commands for Example 8.

and will occur later in time than for the minimum energy case; second, 75-km altitude occurs farther uprange for a 10° trajectory than for a minimum energy trajectory. We thus have farther to go in less time than in the minimum energy case. The trajectory is now changed to 30° re-entry and we note that we do have a shoot-look-shoot capability with the earliest intercept at −122 sec and the latest launch at −108 sec.

CONTOUR delpsi, psi, dr, tau, alt, beta

USE – This command produces a table defining footprints of interceptor coverage.

RESPONSE – A contour of ground coverage is printed in the form of an angle and ground range from some center point, which is also printed. The corresponding latitudes and longitudes are printed as is a code indicating the limiting factor for that point of the contour. The limitations are the same as those discussed for the command 'INTERCEPT' and have the same meaning. The interpretation of the contour is that if the impact point is 'alt' km above a point interior to the contour (using the launch site set in 'LAUNCH' and trajectory parameters set in 'PARAMETER'), the threatening object can be intercepted using the same intercept restrictions discussed in 'INTERCEPT'. Similarly, a point exterior to the contour cannot be defended under the same restrictions. If 'beta' is specified and is nonzero, the trajectories used to obtain the contour are Keplerian with the re-entry portion obtained by integration using a real atmosphere and the ballistic coefficient history specified in 'BETA_TABLE'. If the contour is desired for strictly Keplerian trajectories, 'beta' should be omitted from the input.

The table generated is in angular increments of 'delpsi' deg from zero to 'psi' deg with a radial accuracy of 'dr' km. If 'tau' is less than zero, the contour is for single intercepts; but if 'tau' is greater than zero, the contour is for shoot-look-shoot with 'tau' sec between the first intercept and the second launch.

ERROR – The errors are the same as discussed in 'INTERCEPT'. If the contour's north-south dimension is less than 'dr', a message will so indicate and the command will be terminated. A word of warning is in order regarding the interpretation of the results of this command due to the algorithm used. The algorithm finds one point of the curve and then traces the rest assuming the contour is a single valued function in a polar coordinate system with the origin at the center point determined. No test is made to find undefended areas interior to the contour.

If 'beta' is entered, it must be an integer.

Example 8

Consider an extension of the problem posed in Example 7. With the same defense site and characteristics, we want to know what the defended footprint is. Use of 'CONTOUR' in the sequence of DEANE commands seen in Fig. V-4 provides this information for a 20° re-entry trajectory neglecting tracking to intercept considerations. The first 18 lines in Fig. V-4 are identical with the first 9 lines in Fig. V-3, but the launch point has been changed and a nonrotating earth is used. This is done so that the resulting footprint will be symmetric about the farm longitude, and only half the footprint need be calculated. It is interesting to interpret the codes printed in Fig. V-4. To the side and uprange of the site, the footprint is limited by the radar acquisition

(CODE = 1010). To the rear, the footprint is limited only by the reach capabilities of the interceptor (CODE = 0000). The results of 'CONTOUR' seen in Fig. V-4 are plotted in Fig. V-5 as are similar results for 10° and 30° re-entry trajectories.

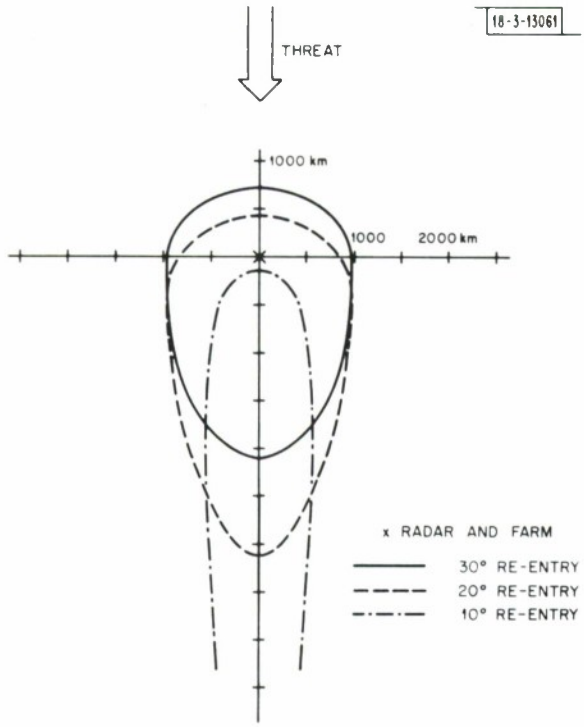

Fig. V-5. Footprints for Example 8 and for 10° and 30° re-entry trajectories.

Example 9

To demonstrate the endoatmospheric footprint capability with an integrated trajectory, consider an (idealized) interceptor which can fly at 2 km/sec with a maximum time of flight of 60 sec. Sited with a radar, we assume that the radar's performance is described by a maximum commit altitude and that the radar's field of view does not geometrically limit intercepts. The threat of interest here is at minimum energy with a β of 1500 down to 30 km, and a β of 500 from 30- to 0-km altitude. With a minimum intercept altitude of 6 km and a maximum commit altitude of 70 km, the DEANE commands in Fig. V-6 yield the contour which defines the defended region. By inspecting the codes, note that the contour is not limited by the interceptor reach but by the commit and minimum intercept altitudes. This footprint is plotted in Fig. V-7, with footprints generated for 50- and 30-km commit altitudes. Also plotted for interest is the footprint for a maximum commit altitude of 30 km and a minimum intercept altitude of 3 km. This footprint illustrates the fact that there is significantly more slowdown at lower altitudes than at the higher altitudes, thus providing almost as much coverage as for commit after 70 km and intercept by 6 km.

-3-13062

```
flyout_table 5 20 20 7 7
0 10 20 30 40 50 60
10 14.1 22.4 31.6 41.2 51 1.0e7
20 22.4 28.3 36 44.7 54 1.0e7
30 31.6 36 42.5 50 58 1.0e7
40 41.2 44.7 50 57 1.0e7 1.0e7
50 51 54 58 1.0e7 1.0e7 1.0e7
60 1.0e7 1.0e7 1.0e7 1.0e7 1.0e7 1.0e7
use_farms
farm1 37 -97 0 0 5
end
use_sensors
end
tracking_sensors
end
beta_table 1500 30 500
parameter 0 1
launch 38 83 200
restrict 6 70 0.5 0 0
contour 10 180 1 -20 0 1
  THE CENTER IS  37.000 LAT AND  -97.000 LONG.
    PSI       RANGE        LAT        LONG     CODE
    0.0        76.0      37.68      -97.00     0010
   10.0        77.0      37.68      -96.85     0010
   20.0        77.0      37.65      -96.70     0010
   30.0        77.0      37.60      -96.56     0010
   40.0        78.0      37.54      -96.43     0010
   50.0        79.0      37.45      -96.31     0010
   60.0        80.0      37.36      -96.22     0010
   70.0        81.0      37.25      -96.14     0010
   80.0        82.0      37.12      -96.09     0010
   90.0        83.0      37.00      -96.07     0010
  100.0        84.0      36.87      -96.07     0010
  110.0        85.0      36.74      -96.10     0010
  120.0        86.0      36.61      -96.17     0010
  130.0        87.0      36.50      -96.25     0010
  140.0        88.0      36.39      -96.37     0010
  150.0        89.0      36.31      -96.50     0010
  160.0        89.0      36.25      -96.66     0010
  170.0        90.0      36.20      -96.83     0010
  180.0        90.0      36.19      -97.00     0010
```

Fig. V-6. DEANE commands for Example 9.

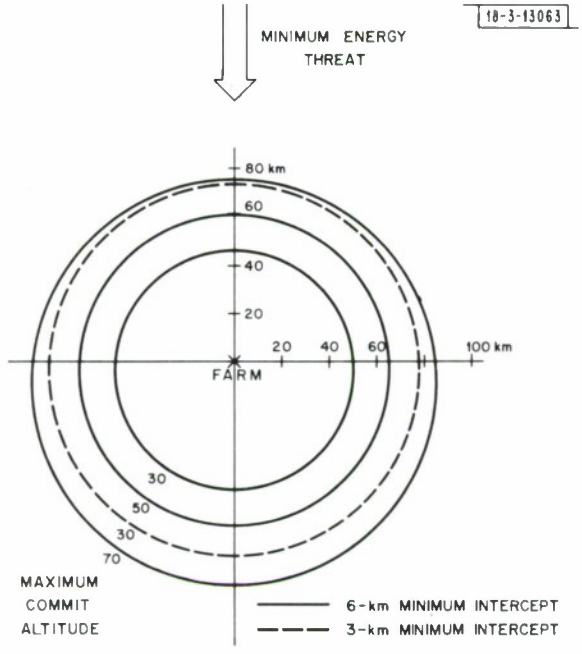

Fig. V-7. Footprint for Example 9 and three other endoatmospheric cases.

VI. "COOKIE-CUTTING" COMMANDS

SITES_ABOUT_SMSA φ d rank1 rank2

USE – This command allows for seeing what deployment sites (farms or radars) are within a certain distance of certain SMSA's (Standard Metropolitan Statistical Areas).

RESPONSE – For SMSA's of 'rank1' through 'rank2', the rank, name, and population of the SMSA are printed under which are printed the name of each site, the azimuth of the SMSA from the site, and its distance to the SMSA, if that distance is not greater than 'd'. If 'φ' is 'FARMS', the farm deployment specified by 'USE_FARMS' is used; if 'φ' is 'RADARS', the sensor deployment specified by 'USE_SENSORS' is used for the calculation.

ERROR – If 'φ' is 'FARMS' and no farms are in use or if 'φ' is 'RADARS' and no radars are in use, a message will so indicate and the command will be terminated. 'rank1' must be less than or equal to 'rank2', and the range of ranks is from 1 to 228. 'φ' must be either 'FARMS' or 'RADARS'.

SMSAS_ABOUT_SITE φ name d

USE – This allows for seeing what SMSA's are within a certain distance of one particular site.

RESPONSE – For each SMSA within distance 'd' of the site 'name', the SMSA rank, name, population, distance to the site, and azimuth from the site are printed. If 'φ' is 'FARM', 'name' references a farm; if 'φ' is 'RADAR', 'name' references a sensor.

ERROR – If the site 'name' is not in the list of farms or sensors (as indicated by 'φ'), a message will so indicate and the command will be terminated. 'φ' must be either 'FARM' or 'RADAR'.

INSIDE_SITE_RING φ d_1 d_2 rank1 rank2

USE – This command considers an entire farm or radar deployment and determines which SMSA's are in a certain range of distances from any site.

RESPONSE – For SMSA's of 'rank1' through 'rank2', if the distance from the SMSA to the closest site in use is greater than 'd_1' but not greater than 'd_2' the SMSA rank, name, and population will be printed with the name of the closest site, that distance, and the azimuth of the SMSA from the site. If 'φ' is 'FARMS', the farm deployment specified by 'USE_FARMS' is used; if 'φ' is 'RADARS', the sensor deployment specified by 'USE_SENSORS' is used for the calculation.

ERROR – If no sites are marked in use (depending on 'φ'), a message will so indicate and the command will be terminated. 'd_1' must be less than 'd_2', 'rank1' must be less than or equal to 'rank2', and the range of ranks is 1 to 228. 'φ' must be either 'FARMS' or 'RADARS'.

CUM_DISTRIBUTION φ rank1 rank2

USE – This command considers a deployment of farms or radars and prints the normalized cumulative population distribution as a function of distance from the closest site.

RESPONSE – The rank, name, and population of SMSA's 'rank1' through 'rank2' are printed in order of increasing distance from the closest site in use. Also printed for each SMSA is the name of the closest site, the distance to that site, the total population within that distance, and this population normalized to the total population in the SMSA's of 'rank1' through 'rank2'. If 'φ' is 'FARMS', the farm deployment specified by 'USE_FARMS' is used; if 'φ' is 'RADARS', the sensor deployment specified by 'USE_SENSORS' is used for the calculation.

ERROR – If no sites are in use (depending on 'φ'), a message will so indicate and the command will be terminated. 'rank1' must be less than or equal to 'rank2', with the range of ranks from 1 to 228. 'φ' must be either 'FARMS' or 'RADARS'.

INSIDE_BOTH d_1 d_2 rank1 rank2

USE – This command considers a deployment of both farms and radars and determines what SMSA's are within a certain distance of a farm and within a certain distance of a radar.

RESPONSE – For SMSA's of 'rank1' through 'rank2', if that SMSA is within a distance 'd_1' of any farm in use and is also within a distance 'd_2' of any radar marked in use, that SMSA rank, name, and population are printed.

ERROR – If no farms are in use or no radars are in use, a message will so indicate and the command will terminate. 'rank1' must be less than or equal to 'rank2', with the range of ranks being from 1 to 228.

NOT_INSIDE_BOTH d_1 d_2 rank1 rank2

USE – This command is the logical complement of 'INSIDE_BOTH'.

RESPONSE – For SMSA's of 'rank1' through 'rank2', if the distance from the SMSA to the nearest farm in use is greater than 'd_1' or if the distance to the nearest radar in use is greater than 'd_2', the SMSA rank, name, and population are printed. Under this are printed the names and distances to the closest farm and the closest radar if these distances are greater than 'd_1' and 'd_2', respectively. Note that by setting 'd_1' and 'd_2' to zero, the result is a listing of SMSA's followed by the name and distance to the closest farm in use and the name and distance to the closest radar in use.

ERROR – If no farms are in use or no radars are in use, a message will so indicate and the command will be terminated. 'rank1' must not be greater than 'rank2', with the range of ranks being from 1 to 228.

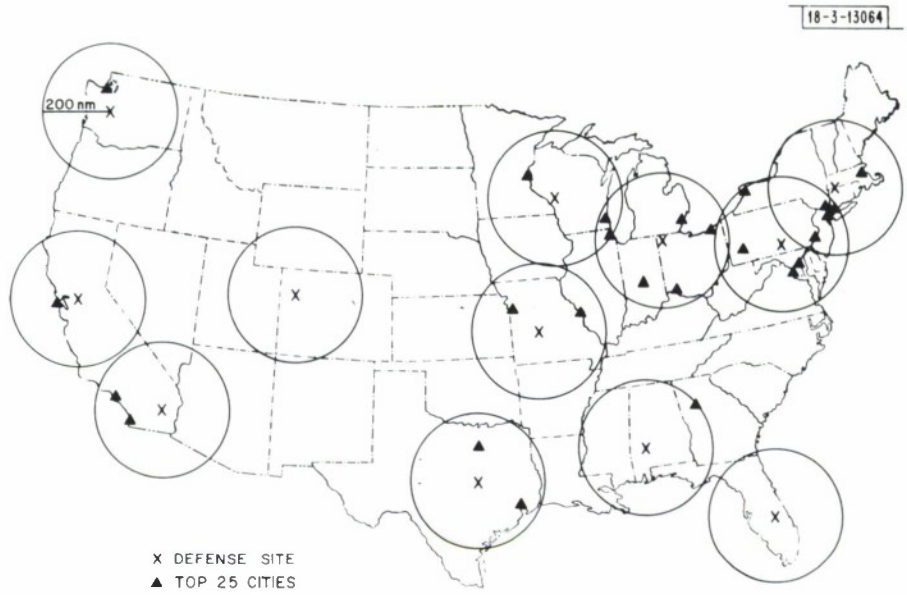

Fig. VI-1. Population defense radar deployment for Example 10.

Example 10

Consider the 12-site population defense radar deployment shown in Fig. VI-1. The DEANE commands seen in Fig. VI-2 provide data for making comments about this deployment. 'USE_SENSORS' in Fig. VI-2 defines the radars preparatory to the cookie-cutting commands. If a radar's coverage extends to 360 km (200 nmi) from the site, then 'SITES_ABOUT_SMSA' shows that three of the top five SMSA's have double coverage. (For example, New York City is at a distance of 172 km and an azimuth of −158° from RADAR2, and at a distance of 268 km and an azimuth of 73° from RADAR8.) To see what SMSA's are capable of defense from RADAR12, we execute 'SMSAS_ABOUT_SITE'. The coverage for the top ten SMSA's is found by executing 'INSIDE_SITE_RING', and we note that all the top ten SMSA's are within 150 nmi (270 km) from a site and none are closer than 65 nmi to a SMSA. 'CUM_DISTRIBUTION' gives coverage information for the next fifteen largest SMSA's, and we see that all but two are within 150 nmi of a site and all are within 200 nmi of a site.

```
use_sensors
radar1 47 -122 # radar2 42.1 -73.2 # radar3 33.8 -115.4
radar4 31.5 -97.3 # radar5 41.3 -84.7 # radar6 32.6 -87.8
radar7 28.1 -80.9 # radar8 40 -77 # radar9 38 -121
radar10 44.1 -90.8 # radar11 38.5 -93 # radar12 40 -108
end
sites_about_smsa radars 360 1 5
    1  NEW YORK,NY              10694632.
         RADAR2          171.79  -157.9
         RADAR8          267.72    72.9
    2  CHICAGO,ILL               6220913.
         RADAR5          260.72   -75.8
         RADAR10         353.92   134.4
    3  LOS ANGELES,CAL           6038771.
         RADAR3          259.56   -84.3
    4  PHILADELPHIA,PA           4342897.
         RADAR2          296.04  -146.4
         RADAR8          161.32    94.7
    5  DETROIT,MICH.             3762360.
         RADAR5          164.86    47.1
smsas_about_site radar radar12  360
 RANK  NAME                    POPULATION  DISTANCE  AZIMUTH
   27  DENVER,COLO               929383.    257.66     95.2
   63  SALT LAKE CITY,UTAH       447795.    342.32    -74.6
  164  COLORADO SPRGS,COLO       143742.    301.60    114.5
  188  PUEBLO,COLO               118707.    347.71    122.3
  196  PROVO,UTAH                106991.    313.21    -83.7
    5      TOTAL                1746618.

inside_site_ring radars 0 10000 1 10
 RANK  NAME                    POPULATION  CLOSEST SITE  DISTANCE  AZIMUTH
    1  NEW YORK,NY             10694632.  RADAR2          171.79   -157.9
    2  CHICAGO,ILL              6220913.  RADAR5          260.72    -75.8
    3  LOS ANGELES,CAL          6038771.  RADAR3          259.56    -84.3
    4  PHILADELPHIA,PA          4342897.  RADAR8          161.32     94.7
    5  DETROIT,MICH.            3762360.  RADAR5          164.86     47.1
    6  SAN FRANCISCO,CAL        2648762.  RADAR9          130.37   -101.9
    7  BOSTON,MASS              2595481.  RADAR2          176.36     80.8
    8  PITTSBURGH,PA            2405435.  RADAR8          259.39    -78.3
    9  SAINT LOUIS,MO           2104669.  RADAR11         227.94     86.9
   10  WASHINGTON,DC            2076610.  RADAR8          120.50    180.0
   10      TOTAL               42890464.
cum_distribution radars 11 25
 RANK  NAME                    POPULATION  CLOSEST SITE  DISTANCE     CUM      NORMAL
   21  SEATTLE,WASH              1107213.  RADAR1           69.57   1107213.  0.0563
   12  BALTIMORE,MD              1803745.  RADAR8           84.02   2910958.  0.1481
   22  KANSAS CITY,MO            1092545.  RADAR11         146.97   4003503.  0.2037
   20  DALLAS,TEXAS              1119410.  RADAR4          150.36   5122913.  0.2606
   19  PATTERSON,NJ              1186873.  RADAR2          158.36   6309786.  0.3210
   13  NEWARK,NJ                 1689420.  RADAR2          172.78   7999206.  0.4069
   23  SAN DIEGO,CAL             1033011.  RADAR3          201.69   9032217.  0.4595
   25  INDIANAPOLIS,IND           944475.  RADAR5          212.43   9976692.  0.5075
   14  MINNEAPOLIS,MINN          1482030.  RADAR10         218.53  11458722.  0.5829
   18  CINCINNATI,OHIO           1268479.  RADAR5          249.04  12727201.  0.6474
   11  CLEVELAND,OHIO            1909483.  RADAR5          252.77  14636684.  0.7446
   17  MILWAUKEE,WIS             1278850.  RADAR10         258.94  15915534.  0.8096
   15  HOUSTON,TEXAS             1418323.  RADAR4          265.37  17333856.  0.8818
   16  BUFFALO,NY                1306957.  RADAR8          356.82  18640800.  0.9483
   24  ATLANTA,GA                1017188.  RADAR6          357.79  19657984.  1.0000
```

Fig. VI-2. DEANE commands for Example 10.

VII. CONVENIENCE COMMANDS

INPUT φ

USE – This command allows for specifying input devices other than the default device. It is useful for reading large or frequently used files of DEANE commands from storage. Such a file should end with 'INPUT 5' if it is desired to return control to the default input device.

RESPONSE – (This is implementation dependent and is described for Lincoln Laboratory's implementation of IBM's CP/CMS.) If 'φ' is an integer, it becomes the input device; otherwise, 'φ' is assumed to be the name of a data file of filetype 'file', and the input device is set to 4 and is defined to correspond to file 'φ'. Reading always starts at the top of the specified input file.

ERROR – If 'φ' is an integer, it may not be 0, 4, 6, or any number greater than 12. If 'φ' is a filename and is not known to the account, an error message is printed and the command is ignored.

OUTPUT φ

USE – This provides the feature for writing output onto a data file so that the file offline can then be printed.

RESPONSE – (This is implementation dependent and is described for Lincoln Laboratory's implementation of IBM's CP/CMS.) If 'φ' is an integer, it becomes the output device, otherwise, 'φ' is assumed to be the name of a data file of filetype 'file', and the output device is set to 33 and is defined to correspond to file 'φ'. Printing always begins at the end of the existing data in the specified file.

ERROR – If 'φ' is an integer, it may not be 0, 4, 5, or any number greater than 12. If 'φ' is a filename and is not known to the account, a message is printed to this effect, the file is created, and execution continues.

LIST_SENSORS

USE – This command affords the user the opportunity to obtain a listing of all the sensors in the sensor list, together with their positions, boresights, and types.

RESPONSE – The listing described above is produced.

ERROR – None.

LIST_FARMS

USE – This command allows the user to obtain a listing of all the farms in the farm list, together with their positions, populations, and types.

RESPONSE – The listing described above is produced.

ERROR – None.

DUMP φ

USE – The two commands above allow the user to view the contents of the sensor and farm lists during execution. This command permits the user to interrogate the contents

of any other input data either for purposes of debugging, review, or completeness in the output.

RESPONSE – If 'φ' is any data defining command other than 'NEW_FARMS' or 'NEW_SENSORS', output with identification will be produced giving the values associated with that command. If 'φ' is 'ALL', values associated with every data defining command will be printed with identification.

ERROR – If 'φ' is neither 'ALL' nor a DEANE command, a message will be printed to that effect.

COMMENT φ

USE – This command allows the user to enter a line in the output for purposes of identification, clarification, or documentation.

RESPONSE – The character string 'φ' is reproduced on the output device.

ERROR – None.

SPECIAL

USE – This command provides an escape into the code for the DEANE interpreter.

RESPONSE – Control passes to a Fortran subroutine named SPECIAL which the user must have written, compiled, and loaded with the DEANE interpreter. As the interpreter is written in Fortran, all the data and subroutines in the DEANE interpreter are accessible via this command if the user understands the coding of the interpreter.

ERROR – None.

STOP

USE – This command provides the normal exit from DEANE.

RESPONSE – Control returns to the CMS environment from DEANE.

ERROR – None.

Example 11

All the data which can be specified by data defining commands have initial values at the beginning of a session. If the first command is 'DUMP ALL', as seen in Fig. VII-1, these initial values are printed. In printing the β table, * 's result because the initial value is a constant β of 10^7 which is too large to fit in the field provided. To see the result for dumping just one command, 'DUMP PARAMETER' is executed as seen in Fig. VII-1.

```
dump all
  THE QUANTITIES BEING PRINTED FOLLOW:
       TIME
       RANGE
       RDOT
       RDDOT
       ELE
       ELEDOT
       AZIMUTH
       AZIDOT
       ALTITUDE
       GRANGE
       VAANGLE
  USING INPUT DEVICE 5.
  USING OUTPUT DEVICE 6.
  LAUNCH LAT =  38.000, LAUNCH LONG =  110.000, LAUNCH HEIGHT =    50.00
  IMPACT LAT =  42.100, IMPACT LONG =  -87.800, IMPACT HEIGHT =     5.00
  THE OBJECT TUMBLING RATE IS  30.00 DEGREES/SEC.
  ORIENTATION IS DETERMINED BY THE COMMAND 'TUMBLE'.
  USING RV_TYPE 1.
  WHEN AT ALTITUDE   70.00 SPIN STABILIZED AXIS POINTS AT (LAT,LONG) = ( 42.100, -87.800).
  CONING RATE IS    0.0  DEGREES/SEC AT AN ANGLE OF    0.0  DEGREES.
  ORIENTATION IS DETERMINED BY THE COMMAND 'TUMBLE'.
  MIN INTERCEPT ALT =  10.0 KM, MAX COMMIT ALT =   100.0 KM, TIME ACCURACY =  1.00 SECONDS.
  INITIAL ACQUISITION/TRACK TIME =   0.0, AND TRACK TO INTERCEPT TIME =   0.0.
  FIXED SPHERICAL EARTH.
  USING A STRICTLY KEPLERIAN TRAJECTORY.
  TYPE 1 TRAJECTORY, I.E. MINIMUM ENERGY.
  NO FARMS ARE IN USE.
  RADAR TYPE 1, LIMITING RANGE = 5500.00, LIMITING AZIMUTH = 180.00, LIMITING ELEVATION =   1.00
    BEAMWIDTH =   1.00, FREQUENCY =   500., ENERGY = 125000., LOSS =   4., AND TEMP =  1000.
  I  BETA(I)     H(I)
  1  *******      0.0
  NO SENSORS ARE IN USE.
  THERE ARE NO TRACKING SENSORS.
dump parameter
  FIXED SPHERICAL EARTH.
  USING A STRICTLY KEPLERIAN TRAJECTORY.
  TYPE 1 TRAJECTORY, I.E. MINIMUM ENERGY.
stop
T=3.03/7.14 15:03:08
```

Fig. VII-1. DEANE commands for Example 11.

VIII. COMMAND SUMMARY

The dependence of the results of executing a command on the results of executing previous commands is seen in Table VIII-1. The vertical column contains 'USE_SENSORS', 'USE_FARMS', 'TRACKING_SENSORS', and all the action commands; the horizontal entries are the data defining commands and 'working trajectory' which refers to the data defining the working trajectory as calculated by execution of either 'KEPLER', 'KEPLERP', 'EJECT', or 'INTEGRATE'. Reading horizontally for any present command, an 'x' indicates that the present command does depend on the data defined by the corresponding horizontal entry, a '*' indicates that the present command may or may not depend on the associated horizontal entry, and a blank indicates that there is no dependence.

For example, executing 'INTERCEPT' depends on 'OUTPUT' for specification of where the results are to be printed, 'USE_SENSORS' for specification of the acquisition sensors, 'FLYOUT_TABLE' for specification of interceptor performance characteristics, 'RESTRICT' for system parameters limiting intercepts, 'USE_FARMS' for specification of an interceptor deployment, 'TRACKING_SENSORS' for definition of the tracking radars, and the 'working trajectory' for defining the object to be intercepted. Data from 'RADAR_TYPE' will be needed to define limiting range, elevation, and azimuth for acquisition and tracking sensors only if some radar restriction is included, i. e., it is not needed if there are no sensors in use and there are no tracking sensors defined.

TABLE VIII-1

MATRIX SHOWING DEPENDENCE OF ONE COMMAND ON EXECUTION OF OTHER COMMANDS

Present Command \ Previously Defined Data	LAUNCH	IMPACT	PARAMETER	BETA_TABLE	PRINT	POINTING	TUMBLE	SIGMA_TABLE	RV_TYPE	RADAR_TYPE	NEW_SENSORS	USE_SENSORS	FLYOUT_TABLE	NEW_FARMS	USE_FARMS	TRACKING_SENSORS	RESTRICT	INPUT	OUTPUT	'working trajectory'
KEPLER	x	x	x																x	
KEPLERP	x	x	x																x	
EJECT																			x	x
R_I_AXIS																			x	x
INTEGRATE				x															x	x
USE_SENSORS											x	x							*	
TIMES					x	*	*	*	*	*		x							x	x
ALTITUDE																			x	x
VISIBLE										x	*	*							x	x
USE_FARMS											x	x							*	
TRACKING_SENSORS											*	*							*	
INTERCEPT										*		x	x		x	x	x		x	x
CONTOUR	x		x	*						*		x	x		x	x	x		x	
SITE_ABOUT_SMSA												*			*				x	
SMSAS_ABOUT_SITE											*	*		*	*				x	
INSIDE_SITE_RING												*			*				x	
CUM_DISTRIBUTION												*			*				x	
INSIDE_BOTH												x			x				x	
NOT_INSIDE_BOTH												x			x				x	
LIST_SENSORS											*	*							x	
LIST_FARMS														*	*				x	
DUMP	*	*	*	*	*	*	*	*	*	*		*	*		*	*	*	*	*	
COMMENT																			x	
SPECIAL																				
STOP																				

IX. MATHEMATICS AND LOGIC

A. Basic Coordinate System

It proves convenient to work in an inertial Cartesian coordinate system whose origin is coincident with the center of a spherical earth of radius Re. The time convention adopted is that t = 0 is the impact time. Time is referenced as time before impact, and it has a negative value. The orientation of the earth relative to the inertial frame is defined to be such that at t = 0 the x-axis passes through 0° latitude and 0° longitude. This system is sketched in Fig. IX-1. The position of a point at latitude φ, longitude Θ, and height h above the earth is thus given by

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = (Re + h) \begin{bmatrix} \cos\varphi \cos(\Theta + \omega_g t) \\ \cos\varphi \sin(\Theta + \omega_g t) \\ \sin\varphi \end{bmatrix}$$

where ω_g is the angular rate of the earth's rotation in inertial space.

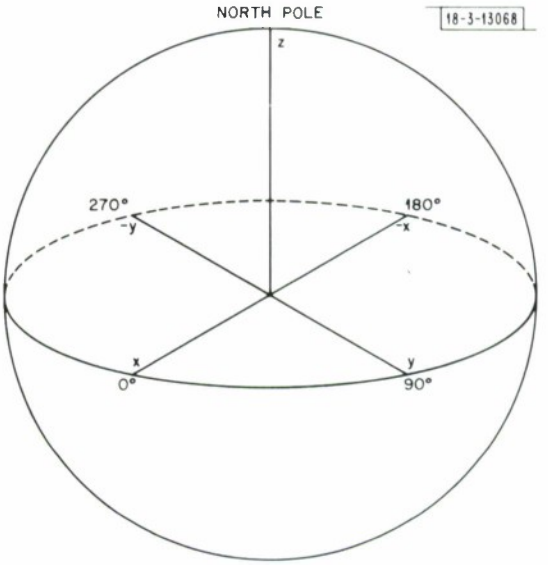

Fig. IX-1. Geocentric inertial Cartesian coordinate system at impact time.

B. KEPLER

The missile is assumed to move in an inverse-square-law central gravitational field for its entire flight.† That is, air drag, rocket thrust, lunar and solar gravity, asphericity of the earth's gravitational field, and the earth's orbital motion are neglected. Thus, the trajectory calculated is an arc of an ellipse in inertial space.

To see what needs to be known to specify motion on a trajectory, refer to Fig. IX-2. The orientation of the trajectory plane with respect to the inertial system is defined by Ω (longitude of the ascending node) and i (inclination). Motion in the trajectory plane describes an ellipse and the spatial relationship is

$$r_m = \frac{p}{1 - \epsilon \cos\nu} \qquad (1)$$

† The analysis in this subsection was performed by W. C. Kellogg.

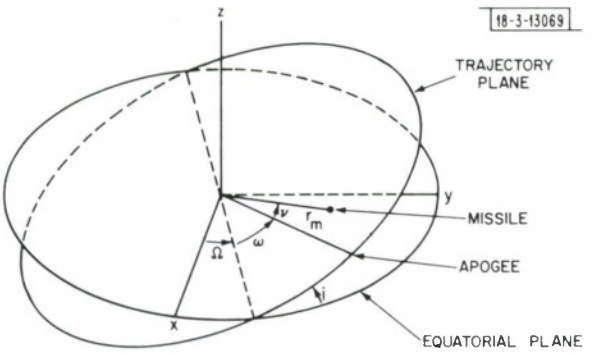

Fig. IX-2. Keplerian trajectory parameters.

where r_m is the distance from the center of the earth, p is the semi-latus rectum, and ϵ is the eccentricity. The angle ν (true anomaly) is measured from apogee and is related to time via Kepler's equations. The angle ω (argument of apogee) is that in the trajectory plane from the ascending node to apogee. It is positive if apogee is in the northern hemisphere, and negative if apogee is in the southern hemisphere. With these definitions, the inertial position of a projectile is given by

$$\vec{r}_m = \begin{bmatrix} x_m \\ y_m \\ z_m \end{bmatrix} = r_m \begin{bmatrix} \cos(\omega + \nu)\cos\Omega - \sin(\omega + \nu)\sin\Omega\cos i \\ \cos(\omega + \nu)\sin\Omega + \sin(\omega + \nu)\cos\Omega\cos i \\ \sin(\omega + \nu)\sin i \end{bmatrix} . \tag{2}$$

To get velocity components in the inertial frame, differentiation of Eq. (1) gives

$$\dot{r}_m = -\frac{r_m^2}{p}\,\epsilon\dot{\nu}\sin\nu$$

but for the inverse-square-law central force we have

$$r_m^2\dot{\nu} = \sqrt{pG}$$

where G is the universal gravitational constant times the mass of the earth. Thus,

$$\dot{r}_m = -\sqrt{\frac{G}{p}}\,\epsilon\sin\nu \tag{3}$$

$$\dot{\nu} = \frac{\sqrt{pG}}{r_m^2} . \tag{4}$$

The velocity may now be calculated simply as the time derivative of Eq. (2) yielding

$$\dot{\vec{r}}_m = \begin{bmatrix} \dot{x}_m \\ \dot{y}_m \\ \dot{z}_m \end{bmatrix} = \frac{\dot{r}_m}{r_m}\begin{bmatrix} x_m \\ y_m \\ z_m \end{bmatrix} + r_m\dot{\nu}\begin{bmatrix} -\sin(\omega + \nu)\cos\Omega - \cos(\omega + \nu)\sin\Omega\cos i \\ -\sin(\omega + \nu)\sin\Omega + \cos(\omega + \nu)\cos\Omega\cos i \\ \cos(\omega + \nu)\sin i \end{bmatrix} . \tag{5}$$

Thus, to define a Keplerian trajectory, we need to find values for the constants p, ϵ, Ω, i, and ω and to determine the functional relationship between ν and t.

1. Values of Constants

G = Universal gravitational constant times mass of earth

$= 0.3986094 \times 10^{6}\ km^3/sec^2$

ω_g = Angular rate of earth in inertial space

$= 0.72921 \times 10^{-4}\ rad/sec$

Re = Radius of spherical earth

$= 6375.2\ km$

2. The Spherical Triangle

In order to determine the plane of the orbit and to compute the distance flown by the missile, it is useful to consider the spherical triangle shown in Fig. IX-3 with vertices at the launch point, the target, and the North Pole.† The sides of this triangle are the launch point and target meridians and the intersection of the trajectory plane with the earth's surface.

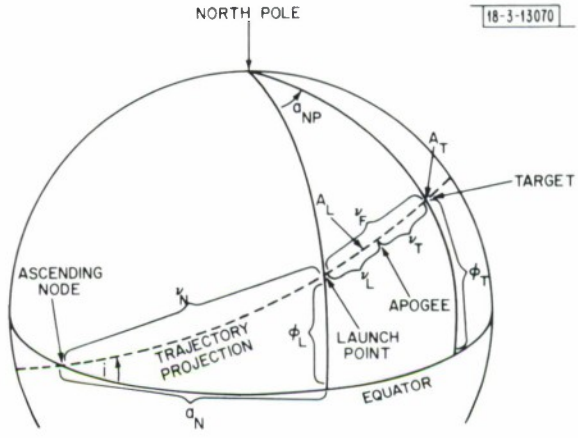

Fig. IX-3. The spherical triangle.

The lengths of the sides meeting at the North Pole are the complements of the launch latitude φ_L and the target latitude φ_T, and are known from input. The North Pole angle a_{NP} is the difference between the target east longitude Θ_T and the launch point east longitude Θ_L, plus the angle through which the earth rotates during the flight.

a. The North Pole Angle a_{NP}

The angle a_{NP} is given by:

$$a_{NP} = \Theta_T - \Theta_L + \omega_g t_F$$

where Θ_L and Θ_T are the launch and target longitudes, ω_g is the angular rate of the earth, and t_F is the time of flight. The difference in longitudes is used as an initial value, and the corresponding time of flight is calculated. This time of flight leads to a new value of a_{NP}, which

† The analysis here relies heavily on spherical trigonometry. The same results can be derived more easily using vector analysis in the inertial Cartesian frame. Since the subroutine which performs these computations is based on the trigonometric analysis, that is the analysis discussed.

leads to a new time of flight, etc. Iteration ceases if the following Cauchy-derived criteria are met: let

t_f = time of flight calculated from this iteration

t'_f = time of flight used in this iteration

t''_f = time of flight used in last iteration

(t_f, t'_f, t''_f are three consecutive calculations of the time of flight; before iteration begins, the time of flight is initialized to zero). Then we have converged if

(1) the last two times are nearly equal: $|t_f - t'_f| < 0.1$ sec, or

(2) the last two times are somewhat close, and solution has begun to diverge: $|t_f - t'_f| < 0.5$, and $|t_f - t'_f| > |t'_f - t''_f|$.

If neither (1) nor (2) above is met within 12 passes, an error message is printed and the calculation is aborted.

In the fixed-earth case where $\omega_g = 0$, the time of flight has no influence on the calculations. Thus, only one pass is taken and iteration ceases after the first set of calculations.

The angle a_{NP} is a signed quantity: it is positive if the target at impact time is east of the position of the launch point meridian at launch time.

b. The Flight Path Arc ν_F

From the law of cosines for spherical triangles,

$$\cos \nu_F = \sin \varphi_L \sin \varphi_T + \cos \varphi_L \cos \varphi_T \cos a_{NP}$$

we derive

$$\sin^2 \left(\frac{\nu_F}{2}\right) = \sin^2 \left(\frac{\varphi_L - \varphi_T}{2}\right) + \cos \varphi_L \cos \varphi_T \sin^2 \left(\frac{a_{NP}}{2}\right) \tag{6}$$

and

$$\cos^2 \left(\frac{\nu_F}{2}\right) = \sin^2 \left(\frac{\varphi_L + \varphi_T}{2}\right) + \cos \varphi_L \cos \varphi_T \cos^2 \left(\frac{a_{NP}}{2}\right) \quad .$$

We combine these two equations with

$$\sin \nu_F = 2 \sin \left(\frac{\nu_F}{2}\right) \cos \left(\frac{\nu_F}{2}\right)$$

to get

$$\nu_F = \tan^{-1} \left(\frac{\sin \nu_F}{\cos \nu_F}\right) \quad .$$

This is superior to the usual method of calculating $\sin \nu_F$

$$\sin \nu_F = (1 - \cos^2 \nu_F)^{1/2}$$

for angles of ν_F close to 90°.

c. Missile Headings A_L at Launch and A_T at Impact

The four-quadrant angles A_L and A_T specify the angles from north to the projections on the earth's surface of the missile velocity vectors at launch and impact, respectively. If the projection points east, A = 90°; if it points south, A = 180°; if it points west, A = 270°.

From the law of sines, we have

$$\sin A_L = \frac{\sin a_{NP}}{\sin \nu_F} \cos \varphi_T$$

$$\sin A_T = \frac{\sin a_{NP}}{\sin \nu_F} \cos \varphi_L$$

and from the law of cosines,

$$\cos A_L = \frac{\sin \varphi_T - \cos \nu_F \sin \varphi_L}{\cos \varphi_L \sin \nu_F}$$

$$\cos A_T = \frac{\sin \varphi_L - \cos \nu_F \sin \varphi_T}{\cos \varphi_T \sin \nu_F} \quad .$$

Accordingly,

$$A_L = \tan^{-1} \frac{\sin a_{NP} \cos \varphi_L \cos \varphi_T}{\sin \varphi_T - \cos \nu_F \sin \varphi_L}$$

$$A_T = \tan^{-1} \frac{\sin a_{NP} \cos \varphi_L \cos \varphi_T}{-\sin \varphi_L + \cos \nu_F \sin \varphi_T} \quad .$$

d. The Inclination i

The ascending node is the node at which the trajectory passes from the southern to the northern hemisphere. The direction in which the missile moves along the trajectory is specified by the inclination i from the equatorial plane to the trajectory plane. To an observer in space above the North Pole, $0 \leqslant i < 90°$ for a counter-clockwise trajectory, and $90° < i \leqslant 180°$ for a clockwise trajectory.

Since i is an angle of a right spherical triangle,

$$\cos i = \cos \varphi_L \sin A_L$$

and substituting for $\sin A_L$ gives

$$\cos i = \frac{\sin a_{NP} \cos \varphi_L \cos \varphi_T}{\sin \nu_F} \quad .$$

whence

$$i = \tan^{-1} \frac{\sqrt{\sin^2 \nu_F - (\sin a_{NP} \cos \varphi_L \cos \varphi_T)^2}}{\sin a_{NP} \cos \varphi_L \cos \varphi_T}$$

since, by convention, $\sin i \geqslant 0$.

e. Longitude of the Ascending Node Ω

Since no sidereal time reference is given in this program, it is necessary to establish arbitrarily that the longitude of the node will be given as earth longitude at impact time. Thus, the longitude Ω of the ascending node is given by

$$\Omega = \Theta_T - a_N - a_{NP}$$

where Θ_T is the target longitude, a_{NP} is the converged value of the North Pole angle, and a_N (as indicated in Fig. IX-3) is the equatorial arc directed from the ascending node to the foot of the launch point meridian.

From the law of sines,

$$\sin a_N = \frac{\sin A_L \sin \varphi_L}{\sin i}$$

and, because we are in a right triangle,

$$\cos a_N = \frac{\cos A_L}{\sin i}$$

so that substituting for $\tan A_L$ gives

$$a_N = \tan^{-1} \frac{\sin a_{NP} \sin \varphi_L \cos \varphi_L \cos \varphi_T}{\sin \varphi_T - \cos \nu_F \sin \varphi_L} \quad .$$

f. Argument of Apogee ω

The angle ω is the arc along the trajectory projection from the ascending node to apogee, positive if apogee is in the northern hemisphere, negative if it is in the southern hemisphere. In terms of quantities in Fig. IX-3, it is given by $\omega = \nu_N - \nu_L$ because if apogee occurs between launch and impact, $\nu_L < 0$ by convention. The angle ν_L comes out of the time-of-flight iteration. The angle ν_N is derived from standard right-triangle formulas

$$\sin \nu_N = \frac{\sin \varphi_L}{\sin i}$$

$$\cos \nu_N = \operatorname{ctn} A_L \operatorname{ctn} i \quad .$$

Substituting from the equations for $\cos i$ and $\cos A_L$ gives

$$\nu_N = \tan^{-1} \frac{\sin \nu_F \sin \varphi_L}{\sin \varphi_T - \cos \nu_F \cos \varphi_L} \quad .$$

3. Inplane Trajectory Parameters

Figure IX-4 illustrates the relevant quantities of the inplane analysis, from which we are to determine the parameters of the ellipse, position of apogee, time of flight, and launch and impact velocity.

Apogee will usually occur between launch and impact; but it need not if launch and impact elevations are very different, as, for example, if the missile were launched against a surface target from a satellite in orbit. The true anomaly ν, specifying the position of the missile in

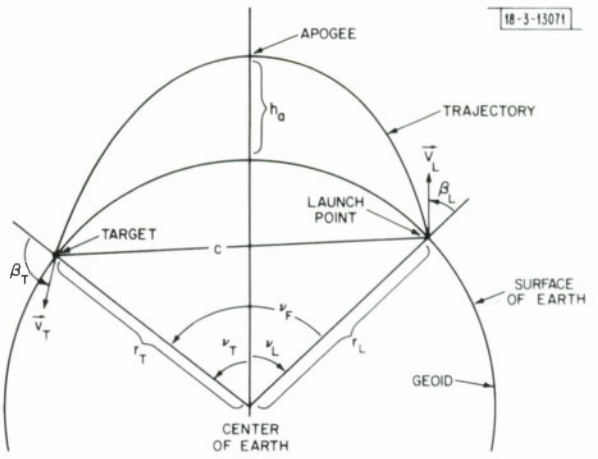

Fig. IX-4. Inplane trajectory geometry.

its trajectory, is measured from apogee and is taken positive in the sense in which the missile moves. The direction of the orbit about the earth's axis is specified by the inclination i.

From the way in which the true anomaly ν is defined, we have

$$\nu_F = \nu_T - \nu_L \tag{7}$$

where ν_L is the true anomaly at launch, ν_T is the true anomaly at impact, and ν_F is the great circle arc traversed by the missile from launch to impact.

Inputs to the inplane analysis are r_L and r_T (the radii at launch and impact), ν_F, and one other datum. From these, we need to compute the semi-latus rectum p; the eccentricity ϵ; the true anomalies ν_L and ν_T at launch and impact, which determine the position of apogee; the time of flight t_F; and the launch and impact velocities $\vec{V}_L$ and $\vec{V}_T$. All but the velocities must be determined inside the iterative loop for determining t_F and a_{NP}.

The chordal distance c from the launch point to the impact point is an important parameter. The basic equation

$$c^2 = r_L^2 - 2r_L r_T \cos \nu_F + r_T^2$$

may be altered to use the result of Eq. (6) for added accuracy when $\nu_F \doteq 0$

$$c^2 = (r_L - r_T)^2 + 4r_L r_T \sin^2 \left(\frac{\nu_F}{2}\right) \quad .$$

Besides the quantities ν_L, ν_T, and ν_F, the fourth datum may be one of four types:

(a) A minimum launch velocity (energy) trajectory.

(b) β_T, impact velocity angle, measured from local horizontal.†

(c) $|\vec{V}_T|$, magnitude of impact velocity.

(d) h_a, apogee height.

† The representation of the angles shown in Fig. IX-4 has them measured from an upward directed local vertical, and this convention is used in the analysis and in the actual computer code. For DEANE input and output, however, the angles are measured from the local horizontal. Thus, β_L (analysis) $\equiv 90 - \beta_L$ (input and output) and β_T (analysis) $\equiv \beta_T$ (input and output) + 90.

Before developing the algorithms for these four types of solutions, it is useful to have an expression for the velocity in the trajectory plane. We have

$$\vec{V} = \dot{r}\vec{r} + r\dot{\nu}\vec{\nu}$$

where $\vec{r}$ and $\vec{\nu}$ are unit vectors in the radial and angular directions, respectively. Substituting from Eqs. (1), (3), and (4), we have

$$\vec{V} = \sqrt{\frac{G}{p}}\,[-\epsilon \sin\nu\,\vec{r} + (1 - \epsilon\cos\nu)\,\vec{\nu}] \quad . \tag{8}$$

a. Minimum Impact Velocity Solution

If we take the magnitude of Eq. (8), substitute $\nu = \nu_T$, we get

$$V_T^2 = \frac{G}{p}\,(1 - 2\epsilon\cos\nu_T + \epsilon^2)$$

and minimize it (by method of Lagrange multipliers) with two constraints from Eq. (1) at launch and impact, respectively,

$$p = r_L\,[1 - \epsilon\cos(\nu_T - \nu_F)] \tag{9}$$

$$p = r_T\,[1 - \epsilon\cos\nu_T] \quad . \tag{10}$$

We arrive at the basic equation for the minimum energy case:

$$\sin\nu_F(1 + \epsilon^2) = 2\epsilon(\sin\nu_T - \sin\nu_L) \quad . \tag{11}$$

We then eliminate p from Eqs. (9) and (10) to get

$$\epsilon(r_T\cos\nu_T - r_L\cos\nu_L) = r_T - r_L \quad . \tag{12}$$

Thus, from Eqs. (11) and (12), we may solve for ν_L in terms of ν_F, r_T, r_L, and ϵ:

$$\sin\nu_L = \frac{r_T\,[(1 + \epsilon^2)\cos\nu_F - 2] + r_L(1 - \epsilon^2)}{2\epsilon(r_T + r_L)\,(1 - \cos\nu_F)}\,\sin\nu_F$$

$$\cos\nu_L = \frac{r_T\,[(1 + \epsilon^2)\sin^2\nu_F - 2(1 - \cos\nu_F)] + 2r_L(1 - \cos\nu_F)}{2\epsilon(r_T + r_L)\,(1 - \cos\nu_F)} \quad .$$

We can eliminate ν_L from the two equations above, leaving a quadratic in ϵ^2. The useful solution of the quadratic is

$$\epsilon^2 = \frac{3 - \cos\nu_F - 2(1 - \cos\nu_F)\,\dfrac{r_T + r_L}{e}}{1 + \cos\nu_F} \quad .$$

If $\nu_F = 180°$, the original quadratic may be reduced with the result

$$\epsilon = \frac{r_T - r_L}{r_T + r_L} \quad .$$

If $\epsilon < 0$ ($\nu_F = 180°$ and $r_L > r_T$), then we set $\nu_L = 0$ and apogee is placed at launch. In any case, when $\epsilon = 0$, we set $\nu_L = -\nu_F/2$ and the trajectory is circular.

Having determined ϵ, we now may determine ν_L. By using the chordal distance c,

$$\sin \nu_L = \frac{-(c + r_T \cos \nu_F - r_L) \sin \nu_F}{\epsilon c(1 + \cos \nu_F)}$$

$$\cos \nu_L = \frac{c - r_T(1 - \cos \nu_F)}{\epsilon c} \quad .$$

From Eq. (8), the actual minimum velocities are

$$\nu_T^2 = \frac{2G(c - r_T + r_L \cos \nu_F)}{r_T r_L(1 + \cos \nu_F)}$$

$$\nu_L^2 = \frac{2G(c - r_L + r_T \cos \nu_F)}{r_T r_L(1 + \cos \nu_F)} \quad .$$

For the critical cases, $\nu_F = 180°$ or $\epsilon = 0$, the velocities are

$$\nu_T^2 = \frac{2Gr_L}{r_T(r_T + r_L)}$$

$$\nu_L^2 = \frac{2Gr_T}{r_L(r_T + r_L)} \quad .$$

If we substitute the expression for $\cos \nu_L$ in Eq. (1), we get an expression for the semi-latus rectum

$$p = \frac{r_L r_T}{c}(1 - \cos \nu_F) \quad .$$

The angles between the velocity vector and the upward directed verticals at launch and impact (β_L and β_T) are determined from Eq. (8). For any angle ν,

$$\beta = \tan^{-1} \frac{\epsilon \cos \nu - 1}{\epsilon \sin \nu} \quad . \tag{13}$$

At launch, we substitute for $\cos \nu_L$ and $\sin \nu_L$ in Eq. (13) to get:

$$\beta_L = \tan^{-1} \frac{r_T \sin \nu_F}{c + r_T \cos \nu_F - r_L} \quad .$$

Similarly, for impact, we use Eq. (11) to obtain expressions for $\sin \nu_T$ and $\cos \nu_T$, and get

$$\beta_T = \tan^{-1} \frac{r_L \sin \nu_F}{-c - r_L \cos \nu_F + r_T} \quad .$$

b. Flight Path Angle at Impact Specified

If the angle β_T is specified, we can obtain from Eq. (8) [see Eq. (13)]

$$\tan\beta_T = \frac{\epsilon \cos\nu_T - 1}{\epsilon \sin\nu_T} \tag{14}$$

which can be solved for ϵ in terms of β_T and the unknown angle ν_T. This expression for ϵ may be substituted into Eq. (1) to yield

$$p = r_L\left(1 - \frac{\cos\nu_L}{\cos\nu_T - \sin\nu_T \tan\beta_T}\right)$$

and

$$p = r_T\left(1 - \frac{\cos\nu_T}{\cos\nu_T - \sin\nu_T \tan\beta_T}\right)$$

for launch and impact, respectively. We equate these two expressions for p and use Eq. (7) to get an equation whose only unknown is ν_T,

$$[(r_L - r_T)\tan\beta_T + r_L \sin\nu_F]\sin\nu_T - r_L(1 - \cos\nu_F)\cos\nu_T = 0$$

$$\nu_T = \tan^{-1}\frac{r_L(1 - \cos\nu_F)}{(r_L - r_T)\tan\beta_T + r_L \sin\nu_F} \quad .$$

Equation (14) yields ϵ, which may be outside the interval (0, 1) if an improper value of β_T is input. The semi-latus rectum p can be obtained from Eq. (1). β_L is obtained by substituting ν_L in Eq. (13).

c. Impact Velocity Specified

To solve the case in which $|\overline{V}_T|$ is specified, we first substitute from Eq. (1) into Eq. (8) to get

$$V_T^2 = \frac{G(\epsilon^2 - 2\epsilon\cos\nu_T + 1)}{r_T(1 - \epsilon\cos\nu_T)} \quad .$$

If we let ρ be the ratio of impact velocity to escape velocity,

$$\rho^2 = \frac{V_T^2}{V_E^2} = \frac{r_T V_T^2}{2G}$$

then, solving the equation for V_T^2 for $\epsilon\cos\nu_T$ gives

$$\epsilon\cos\nu_T = \frac{\epsilon^2 + (1 - 2\rho^2)}{2(1 - \rho^2)} \tag{15}$$

We then substitute $\nu_L = \nu_T - \nu_F$ in Eq. (12), and the expression for $\epsilon \cos\nu_T$ from Eq. (15) in that result, which yields

$$\epsilon \sin\nu_T = \frac{[\epsilon^2 + (1 - 2\rho^2)](r_T - r_L \cos\nu_F) - 2(1 - \rho^2)(r_T - r_L)}{2(1 - \rho^2) r_L \sin\nu_F} \quad . \tag{16}$$

We can now combine Eqs. (15) and (16) to obtain a quadratic in ϵ^2 of the form,

$$c^2\epsilon^4 - 2B\epsilon^2 + (D^2 + B^2)\frac{1}{c^2} = 0$$

which has solutions

$$\epsilon^2 = \frac{B + D}{c^2} \tag{17}$$

and

$$\epsilon^2 = \frac{B - D}{c^2} \tag{18}$$

where c is the chordal distance from launch to impact, and

$$B = 2[r_L \sin\nu_F(1 - \rho^2)]^2 - 2(r_T + r_L) r_L(1 - \cos\nu_F)(1 - \rho^2) + c^2$$

$$D = 2(1 - \rho^2) r_L(1 - \cos\nu_F) \cdot \{r_L(1 - \cos\nu_F)[\rho^2 r_L(1 + \cos\nu_F)$$

$$+ 2\rho(r_T - r_L \cos\nu_F) - r_L(1 - \cos\nu_F)]\}^{1/2} \quad .$$

The angle ν_T may be computed from Eqs. (15) and (16), and the semi-latus rectum p from Eq. (1). The velocity at launch, $\vec{V}_L$, is determined by substituting ν_L for ν in Eq. (8). Finally, the flight path angles β_L and β_T may be determined from the velocity Eq. (13).

For certain values of $|V_T|$, either no trajectory exists, or more than one exist. The program tests for all possible cases as follows.

First, a minimum energy trajectory is solved, and the minimum velocity V_{min} is computed. If $|V_T| < V_{min}$, no trajectory exists, so we set $V_T = V_{min}$, and a message is printed.

Next, ρ is computed, and tested. If $\rho \geq 1$, the given velocity V_T is greater than the escape velocity V_E. The trajectory is thus hyperbolic and the solution is not within the scope of DEANE. So, we set $V_T = 0.99\, V_E$, $\rho^2 = 0.9801$, and print a message to that effect.

For values of V_T between V_{min} and V_e, there will be two trajectories [corresponding to the two values of ϵ^2 given in Eqs. (17) and (18)], one below and one above the minimum energy trajectory. The one below is chosen [Eq. (18)] if the velocity is specified with a minus sign, and the one above [Eq. (17)] is chosen if the velocity is specified with a positive sign.

d. Apogee Height h_a Specified

The radius to apogee r_A is given by $r_A = h_a + Re$. At apogee, $\cos\nu = 1$, so that

$$r_A = \frac{p}{1 - \epsilon} \quad . \tag{19}$$

It is convenient to introduce the ratio

$$\rho = \frac{r_L}{r_A} \quad .$$

Combining Eq. (19) with Eq. (1) at $\nu = \nu_L$ yields

$$\epsilon = \frac{1 - \rho}{1 - \rho \cos \nu_L} \quad . \tag{20}$$

When substituted in Eq. (1) at both launch and impact, this value of ϵ yields two expressions for p in terms of ν_L which must be the same for the correct value of ν_L:

$$p = r_L \left[1 - \frac{(r_A - r_L) \cos \nu_L}{r_A - r_L \cos \nu_L}\right]$$

$$p = r_T \left[1 - \frac{(r_A - r_L) \cos \nu_T}{r_A - r_L \cos \nu_L}\right] \quad .$$

Equating these and substituting $\nu_T = \nu_F + \nu_L$, we obtain

$$r_L - r_T = [r_L - \rho r_T - (1 - \rho)\, r_T \cos \nu_F] \cos \nu_L + [r_T(1 - \rho) \sin \nu_F] \sin \nu_L \quad . \tag{21}$$

For convenience, the quantities in square brackets in Eq. (21) are labeled B and C, respectively. Then, we define angles ψ_S and ψ, given by

$$\sin \psi_S = \frac{r_L - r_T}{\sqrt{B^2 + C^2}}$$

$$\sin \psi = \frac{B}{\sqrt{B^2 + C^2}}$$

$$\cos \psi = \frac{C}{\sqrt{B^2 + C^2}} \quad .$$

Then, Eq. (21) becomes

$$\sin \psi_S = \sin (\nu_L + \psi)$$

$$\nu_L = \psi_S - \psi$$

after which we obtain ϵ from Eq. (20) and p from Eq. (19).

e. Velocities and Time of Flight

The calculation of p, ϵ, ν_L, and ν_T is carried out as described above by the method appropriate for the input data. Equation (14) and an analogous one for β_L give the impact and launch angles, respectively. An equation obtained by taking the magnitude of Eq. (8) provides the magnitudes of the velocities:

$$V = \sqrt{\frac{G}{p}\,(1 - 2\epsilon \cos \nu + \epsilon^2)} \quad .$$

To calculate the time of flight, it is convenient to introduce a new angle E called the eccentric anomaly, given by

$$\tan \frac{E}{2} = \sqrt{\frac{1+\epsilon}{1-\epsilon}} \tan \frac{\nu}{2} \quad . \tag{22}$$

The eccentric anomaly is zero at apogee like the true anomaly. The time t since apogee is given by Kepler's equation of time:

$$nt = E + \epsilon \sin E \tag{23}$$

where n is the mean anomaly, given by

$$n = \sqrt{\frac{G}{a^3}}$$

and a is the semi-major axis:

$$a = \frac{p}{1-\epsilon^2} \quad .$$

Equations (22) and (23) are derived in standard references on orbital mechanics. In order to obtain the total time of flight, we evaluate Eq. (22) for $\nu = \nu_L$ and $\nu = \nu_T$ in order to get values of E for launch and impact; compute the corresponding times t_L and t_T from Eq. (23); and then subtract $t_F = t_T - t_L$. To relate the true anomaly ν to time for the convention that t = 0 at impact, Eq. (23) is modified to become

$$n(t + t_T) = E + \epsilon \sin E \tag{24}$$

from which E may be calculated by a Newton-Raphson iteration for any value of t. The true anomaly then follows from Eq. (22) as

$$\nu = 2 \tan^{-1}\left(\sqrt{\frac{1-\epsilon}{1+\epsilon}} \tan \frac{E}{2}\right) \quad . \tag{25}$$

C. EJECT

The problem here is to take velocity increments in the coordinate system shown in Fig. IX-5, add them to the velocity vector at point P to get a new state vector in the inertial coordinate system, and then calculate the trajectory parameters from this new state vector.

Let ($\Delta V_{\bar{x}}$, $\Delta V_{\bar{y}}$, $\Delta V_{\bar{z}}$) be the input velocity increments and let the state vector at P [as known from the old trajectory via Eqs. (24), (25), (2), and (5)] be (x, y, z, V_x, V_y, V_z). In the

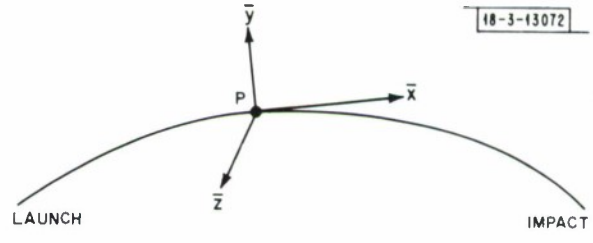

Fig. IX-5. Coordinate system for input to 'EJECT'.

inertial system, the vector $\bar{x}$ is merely the velocity vector at P implying

$$\bar{x} = \frac{1}{V}\begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix}$$

where $V = \sqrt{V_x^2 + V_y^2 + V_z^2}$. The vector $\bar{z}$ is the vector normal to the old trajectory plane and is given by

$$\bar{z} = \begin{bmatrix} -\sin i \sin\Omega \\ \sin i \cos\Omega \\ -\cos i \end{bmatrix} .$$

Of course, $\bar{y} = \bar{z} \times \bar{x}$. The velocity increment in the inertial system is then

$$\begin{bmatrix} \Delta V_x \\ \Delta V_y \\ \Delta V_z \end{bmatrix} = \begin{bmatrix} \bar{x}_x & \bar{y}_x & \bar{z}_x \\ \bar{x}_y & \bar{y}_y & \bar{z}_y \\ \bar{x}_z & \bar{y}_z & \bar{z}_z \end{bmatrix} \begin{bmatrix} \Delta V_{\bar{x}} \\ \Delta V_{\bar{y}} \\ \Delta V_{\bar{z}} \end{bmatrix} \tag{26}$$

giving the new state vector

$$(x,\ y,\ z,\ V_x + \Delta V_x,\ V_y + \Delta V_y,\ V_z + \Delta V_z) \equiv (x,\ y,\ z,\ \dot{x},\ \dot{y},\ \dot{z}) \quad .$$

Straightforward application of trajectory relationships now gives

$$a^{-1} = \frac{2}{r_m} - \frac{S^2}{G} \tag{27}$$

$$\epsilon = \left\{1 - \frac{1}{G}\left[(x\dot{y} - y\dot{x})^2 + (x\dot{z} - z\dot{x})^2 + (y\dot{z} - z\dot{y})^2\right] a^{-1}\right\}^{1/2} \tag{28}$$

$$i = \cos^{-1}\left\{(x\dot{y} - y\dot{x})\left[(x\dot{y} - y\dot{x})^2 + (x\dot{z} - z\dot{x})^2 + (y\dot{z} - z\dot{x})^2\right]^{1/2}\right\} \tag{29}$$

$$p = a(1 - \epsilon^2) \tag{30}$$

where $r_m = \sqrt{x^2 + y^2 + z^2}$, $s = \sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2}$, a is the semi-major axis, ϵ is the eccentricity, i is the inclination, and p is the semi-latus rectum.

At the point P, we can calculate ν from Eq. (1) as

$$\nu = \cos^{-1}\left(\frac{r_m - p}{r_m \epsilon}\right) \quad .$$

Equations (2) and (5) give

$$\sin(\omega + \nu) = z/\sin i$$

$$\cos(\omega + \nu) = \frac{r_m \dot{z} + z\sqrt{\frac{G}{p}}\,\epsilon \sin\nu}{\sqrt{Gp}\,\sin i}$$

so the argument of apogee ω is given by

$$\omega = \tan^{-1}\left[\frac{\sin(\omega + \nu)}{\cos(\omega + \nu)}\right] - \nu \quad .$$

To get the longitude of the ascending node Ω, we have from Eq. (2)

$$\frac{x}{r} = \cos(\omega + \nu)\cos\Omega - \sin(\omega + \nu)\sin\Omega\cos i$$

$$\frac{y}{r} = \cos(\omega + \nu)\sin\Omega + \sin(\omega + \nu)\cos\Omega\cos i$$

which is two equations in two unknowns – $\cos\Omega$ and $\sin\Omega$. It readily follows that after solving these equations

$$\Omega = \tan^{-1}\left(\frac{\sin\Omega}{\cos\Omega}\right) \quad .$$

It remains to fix up the time convention. Using Eq. (1) and the given impact altitude, ν_1 (the true anomaly at impact) can be determined. Similarly, the angle ν_2 at ejection can be calculated. Equation (22) gives the associated eccentric anomalies E_1 and E_2. It follows from Eq. (24) that

$$t_1 \equiv 0 = \frac{E_1 + \epsilon \sin E_1}{n} - t_T$$

defines the time from apogee to impact t_T. The time from ejection to impact is thus

$$t_F = \frac{E_2 + \epsilon \sin E_2}{n} - t_T \quad .$$

Let t_0 be the time for an object on the base trajectory to travel from the ejection point to impact. The difference $t_F - t_0$ must be saved and taken into account for the rotating earth case. Impact is still defined as time zero, but with a nonzero time difference the x-axis no longer points through 0° longitude at time zero. Thus, if impact is at (Re + alt) (u_x, u_y, u_z), the impact latitude and longitude are given by

$$\text{impact latitude} = \sin^{-1}(u_z)$$

$$\text{impact longitude} = \tan^{-1}\left(\frac{u_y}{u_z}\right) - \omega_g(t_F - t_0) \quad .$$

The velocity and impact angle may be calculated, knowing the trajectory parameters, from Eqs. (8) and (13) taken at impact.

D. R_I_AXIS

This problem is the inverse of the problem for EJECT. Given an impact latitude φ, longitude Θ, and altitude A, and given an ejection time, we want the velocity increments which give that impact. We assume the earth does not rotate and that changes in the trajectory parameters are small so that linear approximations (via Taylor expansions) are applicable. (As things turn out with the linear approximation, there are a set of velocity increments which satisfy the impact conditions and the answer will be a vector instead of a point.) The problem is solved in the inertial coordinate system and then transformed to the coordinate system of Fig. IX-5.

At the new impact point, we have

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = (Re + A) \begin{bmatrix} \cos\varphi \cos\Theta \\ \cos\varphi \sin\Theta \\ \sin\varphi \end{bmatrix} .$$

For the base trajectory at the same altitude, we can get ν_A from Eq. (1), and the position of that point in the inertial frame is

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} = (Re + A) \begin{bmatrix} \cos(\omega + \nu_A)\cos\Omega - \sin(\omega + \nu_A)\sin\Omega\cos i \\ \cos(\omega + \nu_A)\sin\Omega + \sin(\omega + \nu_A)\cos\Omega\cos i \\ \sin(\omega + \nu_A)\sin i \end{bmatrix} . \tag{31}$$

Defining $\alpha = \omega + \nu_A$ and since $dr = 0$ at the impact altitude, differentiation of Eq. (31) gives

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} + B \begin{bmatrix} d\alpha \\ d\Omega \\ di \end{bmatrix} \tag{32}$$

where

$$B = \begin{bmatrix} \partial x/\partial\alpha & \partial x/\partial\Omega & \partial x/\partial i \\ \partial y/\partial\alpha & \partial y/\partial\Omega & \partial y/\partial i \\ \partial z/\partial\alpha & \partial z/\partial\Omega & \partial z/\partial i \end{bmatrix}_{\nu=\nu_A}$$

What we need now is the matrix which expresses the relation in Eq. (33) below, since combining that with Eq. (32) will specify $d\vec{V}$:

$$\begin{bmatrix} d\alpha \\ d\Omega \\ di \end{bmatrix}_{\substack{\text{at impact} \\ \nu=\nu_A}} = C \begin{bmatrix} dV_x \\ dV_y \\ dV_z \end{bmatrix}_{\substack{\text{at ejection} \\ \nu=\nu_0}} . \tag{33}$$

Toward this end, consider the point of ejection where the spatial coordinates are identical for the base and new trajectories. Differentiation of Eq. (29) at ejection time gives

$$di = \begin{bmatrix} \partial i/\partial V_x & \partial i/\partial V_y & \partial i/\partial V_z \end{bmatrix} \begin{bmatrix} dV_x \\ dV_y \\ dV_z \end{bmatrix}_{\nu=\nu_0} \tag{34}$$

Define $\beta = \omega + \nu_0$, where ν_0 is the true anomaly of the base trajectory at the time of ejection. Differentiation of Eq. (2) gives

$$dz = 0 \Longrightarrow d\beta = -\tan\beta \operatorname{ctn} i \, di$$

$$dx = 0 \Longrightarrow d\Omega = \left(\frac{\sin^2\beta \cos\Omega \cos i + \sin\beta \cos\beta \sin\Omega}{\cos^2\beta \sin\Omega \sin i + \sin\beta \cos\beta \cos\Omega \sin i \cos i} \right) di \quad . \tag{35}$$

The relation remaining to complete Eq. (32) is that between $d\alpha$ and $d\beta$. Differentiating Eq. (1) yields

$$d\nu = \frac{dp + r \cos\nu \, d\epsilon}{\epsilon r \sin\nu}$$

and differentiating Eqs. (27), (28), and (30) at $\nu = \nu_0$ gives dp and $d\epsilon$ in terms of $d\vec{V}$ so

$$d\nu = \left[\partial\nu/\partial V_x \quad \partial\nu/\partial V_y \quad \partial\nu/\partial V_z \right] \begin{bmatrix} dV_x \\ dV_y \\ dV_z \end{bmatrix}_{\nu=\nu_0} .$$

Thus, evaluation of $d\nu$ at $\nu = \nu_0$ and $\nu = \nu_A$ allows

$$d\alpha = d(\omega + \nu_A) = d\omega + d\nu_A + d\nu_0 - d\nu_0 = d\beta + d\nu_A - d\nu_0 \quad . \tag{36}$$

Equations (34), (35), and (36) now completely specify the relation in Eq. (33), and we have

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} - \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} = BC \begin{bmatrix} dV_x \\ dV_y \\ dV_z \end{bmatrix}_{\nu=\nu_0} . \tag{37}$$

Since $d\Omega$ is a constant times di (implying the matrix BC is singular), the solution to Eq. (37) for $d\vec{V}$ is a line and not a point. We seek a solution to Eq. (37) of the form

$$\begin{aligned} dV_x &= a_x + kb_x \\ dV_y &= a_y + kb_y \\ dV_z &= a_z + kb_z \end{aligned} \tag{38}$$

and it is convenient to consider Eq. (39) below where D = BC:

$$\begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} = D \begin{bmatrix} dV_x \\ dV_y \\ dV_z \end{bmatrix}_{\nu=\nu_0} . \tag{39}$$

By setting $dV_z = 0$, (d_x, d_y, d_z) equal to the left-hand side of Eq. (37), and solving Eq. (39) yields a_x and a_y with $a_z = 0$ and $k = 0$ in Eq. (38). Setting $(d_x, d_y, d_z) = \vec{0}$, $dV_z = 1$, and solving Eq. (39) gives b_x and b_y with $b_z = 1$ in Eq. (38) completing the solution for the vector of velocity increments in the inertial frame. Transforming these results into the coordinate system of Fig. IX-5 using the inverse (transpose) of the matrix in Eq. (26) gives

$$\Delta\overline{V} = \begin{bmatrix} \Delta V_{\bar{x}} \\ \Delta V_{\bar{y}} \\ \Delta V_{\bar{z}} \end{bmatrix} = \begin{bmatrix} \bar{x}_x & \bar{x}_y & \bar{x}_z \\ \bar{y}_x & \bar{y}_y & \bar{y}_z \\ \bar{z}_x & \bar{z}_y & \bar{z}_z \end{bmatrix} \begin{bmatrix} a_x + kb_x \\ a_y + kb_y \\ k \end{bmatrix} = \begin{bmatrix} a_{\bar{x}} \\ a_{\bar{y}} \\ a_{\bar{z}} \end{bmatrix} + k \begin{bmatrix} b_{\bar{x}} \\ b_{\bar{y}} \\ b_{\bar{z}} \end{bmatrix}$$

To find the velocity increment with minimum magnitude

$$\Delta\overline{V}^2 = (a_{\bar{x}} + kb_{\bar{x}})^2 + (a_{\bar{y}} + kb_{\bar{y}})^2 + (a_{\bar{z}} + kb_{\bar{z}})^2$$

setting the derivative of $\Delta\overline{V}^2$ with respect to k equal to zero gives

$$k = -\frac{a_{\bar{x}}b_{\bar{x}} + a_{\bar{y}}b_{\bar{y}} + a_{\bar{z}}b_{\bar{z}}}{b_{\bar{x}}^2 + b_{\bar{y}}^2 + b_{\bar{z}}^2}$$

from which $\Delta\overline{V}$ may be evaluated.

E. INTEGRATE

In the inertial coordinate system, the equations of motion are

$$\ddot{x} = -\frac{x}{r^3} G - \frac{1}{2} \alpha\rho V\dot{x}$$

$$\ddot{y} = -\frac{y}{r^3} G - \frac{1}{2} \alpha\rho V\dot{y}$$

$$\ddot{z} = -\frac{z}{r^3} G - \frac{1}{2} \alpha\rho V\dot{z} \qquad (40)$$

where $r = \sqrt{(x^2 + y^2 + z^2)}$, $v = \sqrt{(\dot{x}^2 + \dot{y}^2 + \dot{z}^2)}$, α is $1/\beta$, ρ is the atmospheric density, and G is as before. These can be integrated numerically by defining

$$\vec{u} = (x, y, z, \dot{x}, \dot{y}, \dot{z})$$

$$\vec{f} = \frac{d\vec{u}}{dt} = (\dot{x}, \dot{y}, \dot{z}, \ddot{x}, \ddot{y}, \ddot{z})$$

so that the differential equation (40) becomes

$$\vec{f} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -\frac{G}{r^3} & 0 & 0 & -\frac{1}{2}\alpha\rho V & 0 & 0 \\ 0 & -\frac{G}{r^3} & 0 & 0 & -\frac{1}{2}\alpha\rho V & 0 \\ 0 & 0 & -\frac{G}{r^3} & 0 & 0 & -\frac{1}{2}\alpha\rho V \end{bmatrix} \vec{u} \quad .$$

Using the predictor-corrector scheme of Milne, we have the predictor

$$\vec{u}_p = \vec{u}_{p-4} + \frac{1}{3}\Delta t(8\vec{f}_{p-1} - 4\vec{f}_{p-2} + 8\vec{f}_{p-3})$$

and the corrector

$$\vec{u}_p = \vec{u}_{p-2} + \frac{1}{3}\Delta t(\vec{f}_p + 4\vec{f}_{p-1} + \vec{f}_{p-2}) \quad .$$

To initialize the iteration of these equations, the quantities $\vec{u}_{-1}$ and $\vec{f}_{-1}$ are taken at the altitude where the integration begins from the Keplerian specification of the working trajectory. The quantities $\vec{u}_{-2}$, $\vec{f}_{-2}$, $\vec{u}_{-3}$, $\vec{f}_{-3}$, and $\vec{u}_{-4}$ are similarly calculated where the spatial and velocity components are taken from Eqs. (2) and (5), and the accelerations are computed from Eq. (40) with $\alpha = 0$.

The results for the state vector $\vec{u}$ are made accessible by storing them in a table. References to the integrated part of the working trajectory are then made by linear interpolation in this table, while references to the Keplerian part are made via Eqs. (2) and (5) as before.

As in the case for EJECT, additional consideration must be given to the time convention. Consider Fig. IX-6 where the integration begins at alt_1 km and terminates at alt_2 km. The time from alt_1 to alt_2 is t_k sec for the Keplerian base trajectory, and t_i sec for the integrated trajectory with $|t_i| \geq |t_k|$. With a rotating earth, this time delay must be included for transforming longitude into the Cartesian frame or from the Cartesian frame into longitude. Also, in referencing the Keplerian part of the trajectory when the re-entry portion is the result of integration, $|t_i - t_k|$ must be considered. For the point labeled t_0 in Fig. IX-6, Eqs. (2) and (5) must be evaluated at the true anomaly ν corresponding to a time $t = t_0 + |t_i - t_k|$, where t_0 is the time before impact and $t_0 < 0$.

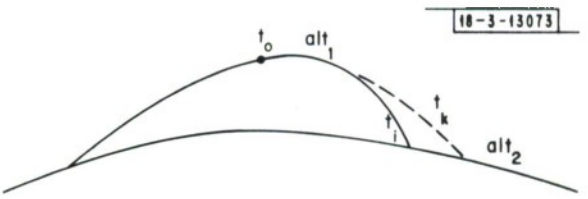

Fig. IX-6. Trajectory and time change from 'INTEGRATE'.

F. PRINT

Given a sensor at latitude φ, longitude Θ, and height h above the earth, and given a time t_0, the problem is to determine the quantities listed in Table IV-1.

1. Time Considerations

We have to be concerned about time in two systems – inertial Cartesian and rotating earth. Making the conditional assignments in Eqs. (41), (42), and (43) below, we have the position of the sensor in inertial space as specified in Eq. (44).

$$t_0 = \begin{cases} t_0 & \text{if } t_0 < 0 \\ t_0 - \text{time of flight} & \text{if } t_0 > 0 \end{cases} \tag{41}$$

$$\Delta t_1 = \begin{cases} 0 & \text{if working trajectory not from 'EJECT'} \\ \text{flight time increase due to 'EJECT' otherwise} \end{cases} \tag{42}$$

$$\Delta t_2 = \begin{cases} 0 & \text{if working trajectory not from 'INTEGRATE'} \\ \text{flight time increase due to 'INTEGRATE' otherwise} \end{cases} \tag{43}$$

$$\vec{r}_s = (Re + h)\,\vec{u}_s$$

$$\vec{u}_s = [\cos\varphi\,\cos(\Theta + \omega_g t'_0),\ \cos\varphi\,\sin(\Theta + \omega_g t'_0),\ \sin\varphi] \tag{44}$$

where

$$t'_0 = t_0 + \Delta t_1 + \Delta t_2 \quad .$$

Calculation of the missile's state vector is similarly conditionally dependent on time and the nature of the working trajectory. If $t_0 > -\Delta t_2$, the state vector is found by linear interpolation in the table of stored integration results. Otherwise, it is taken from Eqs. (2) and (5). (As noted at the end of Sec. IX-E, if the re-entry portion is obtained from 'INTEGRATE', Eqs. (2) and (5) are evaluated at $t_0 + \Delta t_2$ not at t_0 for $t_0 < -\Delta t_2$.) At any rate, we get a state vector Eq. (45) in the inertial system, and the problem is now one of combining Eqs. (44) and (45):

$$\vec{r}_m = (x, y, z)$$

$$\dot{\vec{r}}_m = (\dot{x}, \dot{y}, \dot{z}) \quad . \tag{45}$$

2. Range, Azimuth, and Elevation

The radar range is simply given by

$$r_o = |\vec{r}_o| = |\vec{r}_m - \vec{r}_s| \quad .$$

It is useful to define a unit vector $\vec{u}_r$ along the line of sight, and also two other unit vectors $\vec{u}_E$ and $\vec{u}_N$ which point east and north from the sensor and are in the plane tangent to a spherical earth (of radius Re + h) at the sensor site. The vectors $\vec{u}_s$, $\vec{u}_E$, and $\vec{u}_N$ then define a right-handed coordinate system. These definitions are:

$$\vec{u}_r = \frac{\vec{r}_m - \vec{r}_s}{r_o}$$

$$\vec{u}_E = [-\sin(\Theta + \omega_g t'_0),\ \cos(\Theta + \omega_g t'_0),\ 0]$$

$$\vec{u}_N = [-\cos(\Theta + \omega_g t'_0)\sin\varphi,\ -\sin(\Theta + \omega_g t'_0)\sin\varphi,\ \cos\varphi] \tag{46}$$

The azimuth a from north and the elevation e are now given by

$$a = \tan^{-1}\left(\frac{\sin a}{\cos a}\right) = \tan^{-1}\left(\frac{\vec{u}_r \cdot \vec{u}_E}{\vec{u}_r \cdot \vec{u}_N}\right)$$

$$e = \sin^{-1}(\vec{u}_r \cdot \vec{u}_s) \quad .$$

3. Time Derivatives of Range, Azimuth, and Elevation

It must be remembered that $\vec{u}_s$, $\vec{u}_E$, and $\vec{u}_N$ rotate with the earth. To obtain the velocity seen by an observer in the rotating coordinate system, we use the operator relation

$$\left.\frac{d}{dt}\right|_{\text{rotating}} = \left.\frac{d}{dt}\right|_{\text{fixed}} - \vec{\omega} \times \tag{47}$$

which is derived in texts of rigid-body mechanics. In Eq. (47), $\vec{\omega}$ is $(0, 0, \omega_g)$. The missile velocity $\vec{v}$ as observed by the sensor is then given by

$$\vec{v} = \dot{\vec{r}}_o - \vec{\omega} \times \vec{r}_o = (\dot{x}_m + \omega_g y_m,\ \dot{y}_m - \omega_g x_m,\ \dot{z}_m) \quad . \tag{48}$$

In order to obtain time derivatives of scalar range, azimuth, and elevation, two more unit vectors are introduced. $\vec{u}_a$ is a unit vector in the horizontal plane pointing in the direction of increasing a, and $\vec{u}_e$ points in the direction of increasing e and is normal to the unit vector $\vec{u}_r$ of Eq. (46) in the plane containing $\vec{u}_r$ and the unit vector $\vec{u}_s$:

$$\vec{u}_a = \vec{u}_E \cos a - \vec{u}_N \sin a$$

$$\vec{u}_e = -\vec{u}_E \sin a \sin e - \vec{u}_N \cos a \sin e + \vec{u}_s \cos e$$

where $\vec{u}_E$, $\vec{u}_N$, and $\vec{u}_s$ are the unit vectors at the sensor site previously defined.

In terms of this set, we have

$$\vec{v} = \dot{r}_o \vec{u}_r + r_o \dot{a} \cos e\, \vec{u}_a + r_o \dot{e}\, \vec{u}_e \tag{49}$$

where $\dot{r}_o$ is the Doppler velocity observed by the sensor. Thus, the equations needed to convert $\vec{v}$ of Eq. (49) into time derivatives of radar observables are

$$\dot{r}_o = \vec{v} \cdot \vec{u}_r$$

$$\dot{a} = \frac{\vec{v} \cdot \vec{u}_a}{r_o \cos e}$$

$$\dot{e} = \frac{\vec{v} \cdot \vec{u}_e}{r_o} \quad . \tag{50}$$

4. Range Acceleration

The easiest way to get the range acceleration $\ddot{r}_o$ is to differentiate Eq. (50), giving

$$\ddot{r}_o = \dot{\vec{v}} \cdot \vec{u}_r + \vec{v} \cdot \dot{\vec{u}}_r \quad .$$

Differentiation of Eq. (48) by application of Eq. (47) gives

$$\dot{\vec{v}} = [\ddot{x}_m + \omega_g(2\dot{y}_m - \omega_g x_m),\ \ddot{y}_m - \omega_g(2\dot{x}_m + \omega_g y_m),\ \ddot{z}_m]$$

where $(\ddot{x}_m, \ddot{y}_m, \ddot{z}_m)$ is defined in Eq. (40), and differentiation of Eq. (46) gives

$$\dot{\vec{u}}_r = \frac{(\dot{x} + \omega_g y_s,\ \dot{y} - \omega_g x_s,\ \dot{z})}{|\vec{r}_o|} - \frac{\vec{u}_r}{|\vec{r}_o|^2}\left[\vec{r}_o \cdot \dot{\vec{r}}_m + \omega_g(x_m y_s - y_m x_s)\right]$$

5. Altitude and Ground Range

The altitude above the earth is simply

$$\text{alt} = |\vec{r}_m| - \text{Re} \quad .$$

The ground range at time t is defined as the distance measured along the earth's surface from the sensor to a point on the earth directly beneath the missile, so

$$\text{ground range} = \text{Re} \cos^{-1} \frac{\vec{r}_m \cdot \vec{u}_s}{|\vec{r}_m|} \quad .$$

6. Velocity and Body Aspect Angles

The angle between the line of sight and the missile's velocity vector is simply

$$\text{velocity aspect angle} = \cos^{-1}\left(\frac{-\dot{\vec{r}}_m \cdot \vec{u}_r}{|\dot{\vec{r}}_m|}\right) \quad .$$

The body aspect angle, however, depends on whether orientation is specified via 'POINTING' or 'TUMBLE'.

a. Orientation Specified via 'TUMBLE'

If the tumbling rate is ω_T and we assume random initiation of the tumbling, all we need is a vector which traces out a circle in the trajectory plane. This is easily defined in the inertial coordinate system by using Eq. (2) with $\omega + \nu$ replaced by $\omega_T t$. Thus,

$$\vec{p}_T = (\cos\omega_T t \cos\Omega - \sin\omega_T t \sin\Omega \cos i,$$

$$\cos\omega_T t \sin\Omega + \sin\omega_T t \cos\Omega \cos i,\ \sin\omega_T t \sin i)$$

and

$$\text{body aspect angle} = \cos^{-1}(\vec{p}_T \cdot \vec{u}_r) \quad .$$

b. Orientation Specified via 'POINTING'

The altitude, latitude, and longitude specified in 'POINTING' determine a vector $\vec{a}$ in the inertial coordinate system. This vector is simply found by turning the altitude into time, finding the coordinates of the missile $\vec{r}_m$ and the coordinates of the latitude and longitude $\vec{r}_p$ at this time so that

$$\vec{a} = \frac{\vec{r}_p - \vec{r}_m}{|\vec{r}_p - \vec{r}_m|} \equiv (\alpha, \beta, \gamma) \quad .$$

Referring to Fig. IX-7 we need to get the vector $\vec{b}$ which is due to spinning at an angle ψ from $\vec{a}$ and with a rate ω_s. This determines $\vec{c} = \vec{a} + \vec{b}$ so that the aspect angle is

$$\text{body aspect angle} = \cos^{-1}\left(-\vec{u}_r \cdot \frac{\vec{c}}{|\vec{c}|}\right) \quad .$$

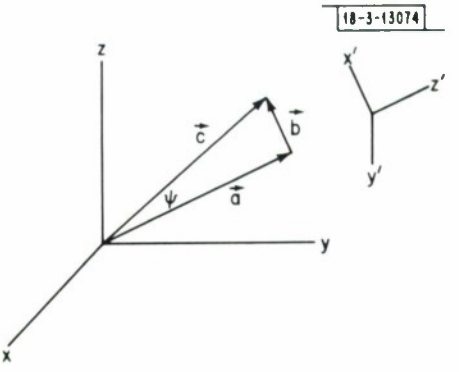

Fig. IX-7. Coordinate systems used for finding 'PRINT' mnemonic 'POINTING'.

To express $\vec{b}$, define the right-handed orthogonal coordinate system (x', y', z') such that

$$z' = (\alpha, \beta, \gamma) \equiv \vec{a}$$

$$y' = (-\beta, \alpha, 0)/\sqrt{\alpha^2 + \beta^2}$$

$$x' = y' \times z' = [\alpha\gamma, \beta\gamma, -(\alpha^2 + \beta^2)]/\sqrt{\alpha^2 + \beta^2} \quad . \tag{51}$$

The length of $\vec{b}$ is $\tan\psi$, and it rotates in the (x', y') frame such that

$$\vec{b}_{x', y', z'} = (\tan\psi \cos\omega_s t, \tan\psi \sin\omega_s t, 0) \quad .$$

Transforming $\vec{b}$ into inertial frame coordinates using Eq. (51), we have

$$\vec{b}_{x, y, z} = \frac{1}{\sqrt{\alpha^2 + \beta^2}} \begin{bmatrix} \alpha\gamma & -\beta \\ \beta\gamma & \alpha \\ -(\alpha^2 + \beta^2) & 0 \end{bmatrix} \begin{bmatrix} \tan\psi & \cos\omega_s t \\ \tan\psi & \sin\omega_s t \end{bmatrix}$$

7. Cross Section and S/N

The cross section is found by interpolation in the cross-section table specified by 'RV_TYPE' using the angle determined as the body aspect angle.

With this cross-section σ, the S/N is evaluated as

$$S/N = 10 \log_{10}\left(\frac{E\sigma}{R^4 T \Theta^4 f^2}\ 3.734 \times 10^{21}\right) - L$$

where E, T, Θ, f and L are defined via 'RADAR_TYPE', and R is the range in kilometers.

8. Latitude and Longitude

To find the latitude and longitude directly beneath the missile, consideration must be given to time as discussed in Sec. IX-F-1. Taking the inverse of the relation between a point on the rotating earth and the inertial frame which is expressed in Eq. (41), we have

$$\text{latitude} = \sin^{-1}\left(\frac{z_m}{|\vec{r}_m|}\right)$$

$$\text{longitude} = \tan^{-1}\left(\frac{y_m}{x_m}\right) - \omega_g t'_0$$

where t'_0 is defined in Eq. (44).

G. VISIBLE

We are given a working trajectory and a sensor at latitude φ, longitude Θ, height h with an azimuth boresight δ. It is not easy to find the times at which a projectile on the trajectory is first visible and is last visible as limited by range R, elevation E, and azimuth scan $\pm$ A in an analytic fashion, but it can be done. A search is certainly less aesthetic than application of three numerical nonlinear equation solvers, but it proves to be speedier in terms of execution time. Accordingly, a search is used to perform the calculation, with some analysis applied to determine starting points.

We wish to find two times at which to start searches for visibility – one time for the early visibility, and one for the late visibility. To do this, we assume that the whole trajectory is Keplerian and work with the true anomaly ν as an equivalent to time. Initially, the two points are ν_L and ν_T, corresponding to the launch and impact. Assuming that E = 0, we can find the two points where the plane specified by the sensor position and E = 0 intersects the trajectory, implying associated values of ν_1 and ν_2 with $\nu_1 < \nu_2$.† A similar computation on the range restriction leads to the problem of finding (or approximating) the intersections of an ellipse and a sphere. This implies values of ν_3 and ν_4 with $\nu_3 < \nu_4$.† To start the search for first visibility, the time corresponding to $\nu_e = \max(\nu_L, \nu_1, \nu_3)$ is chosen. Similarly, to start the search for last visibility, the time corresponding to $\nu_l = \min(\nu_T, \nu_2, \nu_4)$ is chosen.

†In degenerate cases, i.e., one or no intersections, the two anomalies are set to ν_L and ν_T, respectively.

1. Analysis on Elevation

The plane of 0° elevation is given by

$$\frac{x_o}{(Re + h)^2} x + \frac{y_o}{(Re + h)^2} y + \frac{z_o}{(Re + h)^2} z = 1 = \alpha x + \beta y + \gamma z \tag{52}$$

where

$$(x_o, y_o, z_o) = (Re + h)(\cos\varphi \cos\Theta, \cos\varphi \sin\Theta, \sin\varphi) \quad .$$

Introducing a Cartesian coordinate system in the trajectory plane such that

$$\xi = r_m \cos\nu$$

$$\eta = r_m \sin\nu$$

we have the relationship

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\text{inertial}} = T \begin{bmatrix} \xi \\ \eta \end{bmatrix}$$

where

$$T = \begin{bmatrix} \cos\omega \cos\Omega - \sin\omega \sin\Omega \cos i & -\sin\omega \cos\Omega - \cos\omega \sin\Omega \cos i \\ \sin\Omega \cos\omega + \sin\omega \cos\Omega \cos i & -\sin\omega \sin\Omega + \cos\omega \cos\Omega \cos i \\ \sin\omega \sin i & \cos\omega \sin i \end{bmatrix} .$$

Substituting in Eq. (52) gives

$$(\alpha, \beta, \gamma) \begin{bmatrix} x \\ y \\ z \end{bmatrix} = (\alpha, \beta, \gamma)\, T \begin{bmatrix} \xi \\ \eta \end{bmatrix} = 1$$

or

$$\Lambda\xi + \psi\eta = 1$$

where

$$\Lambda = \alpha T_{11} + \beta T_{21} + \gamma T_{31}$$

$$\psi = \alpha T_{21} + \beta T_{22} + \gamma T_{32}$$

$$\xi = \frac{p \cos\nu}{1 - \epsilon \cos\nu}$$

$$\eta = \frac{p \sin\nu}{1 - \epsilon \cos\nu} \quad . \tag{53}$$

To solve Eq. (53) for ν, we substitute $\sin\nu = \sqrt{1 - \cos^2\nu}$ and get a quadratic whose solution is

$$\cos\nu = \frac{\epsilon + \Lambda p \pm \psi p \sqrt{\epsilon^2 + \Lambda^2 p^2 + 2\Lambda p\epsilon + \psi^2 p^2 - 1}}{\epsilon^2 + \Lambda^2 p^2 + 2\Lambda p\epsilon + \psi^2 p^2} \tag{54}$$

From Eq. (53),

$$\sin\nu = \frac{1 - (\epsilon + \Lambda p)\cos\nu}{\psi p}$$

so that

$$\nu = \tan^{-1}\left(\frac{\sin\nu}{\cos\nu}\right) \quad .$$

Since Eq. (54) gives two values of $\cos\nu$, two values for ν are obtained.

2. Analysis on Range

The intersection of a sphere of radius R centered at the sensor and the trajectory plane is a circle. To describe this circle, the equation of the trajectory plane is

$$A_o x + B_o y + C_o z = 1$$

where

$$(A_o, B_o, C_o) = (-\sin\Omega \sin i,\ \cos\Omega \sin i,\ -\cos i) \quad .$$

With (x_o, y_o, z_o) given by Eq. (52), the distance from the sensor to the trajectory plane is

$$d = \left|A_o x_o + B_o y_o + C_o z_o\right|$$

so the radius of the circle is

$$g = \sqrt{R^2 - d^2} \quad .$$

The center of the circle is at (ξ_o, η_o) as expressed in Eq. (55) below

$$k = A_o x_o - B_o y_o - C_o z_o$$

$$x_1 = kA_o + x_o$$

$$y_1 = kB_o + y_o$$

$$z_1 = kC_o + z_o$$

$$\begin{bmatrix} \xi_o \\ \eta_o \end{bmatrix} = \begin{bmatrix} \cos\omega\cos\Omega & \cos\omega\sin\Omega & \frac{\sin\omega}{\sin i} \\ -\sin\omega\cos\Omega & -\sin\omega\sin\Omega & \frac{\cos\omega}{\sin i} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \tag{55}$$

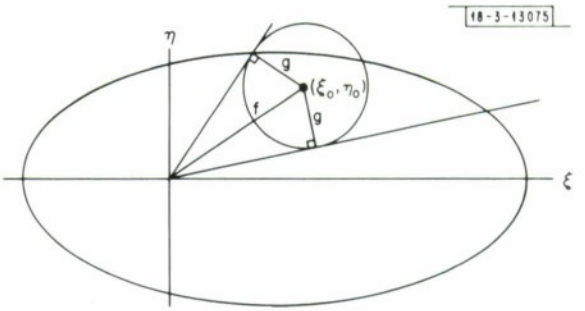

Fig. IX-8. Geometry for bounding time from range limits in 'VISIBLE'.

In the trajectory plane, we have the situation shown in Fig. IX-8. Being satisfied to only bound the angles, we have Eq. (56) which gives values for ν_3 and ν_4:

$$f = \sqrt{\xi_o^2 + \eta_o^2}$$

$$\nu_o = \tan^{-1}(\eta_o/f,\ \xi_o/f)$$

$$\nu_3 = \nu_o - \sin^{-1}\left(\frac{g}{f}\right)$$

$$\nu_4 = \nu_o + \sin^{-1}\left(\frac{g}{f}\right) \quad . \tag{56}$$

3. The Search

It is assumed that the period of visibility is continuous in time. The idea of the search is then to:

(a) Using t_e = function (ν_e) as a starting point, find a time t_e when the missile is visible by using a gross search.

(b) Push t_e as far backward in time as possible with a fine search to get the earliest visibility.

(c) Using t_l = function (ν_l) as a starting point, find a time t_l when the missile is visible by using a gross search.

(d) Push t_l as far forward in time as possible with a fine search to get the latest visibility.

The logic for steps (a) and (b) above is shown in Fig. IX-9, and the logic for steps (c) and (d) is similar.

H. INTERCEPT

Here the basic problem is one of defining a fast, complete, logically complex algorithm. The structure of the algorithm is shown in Fig. IX-10. The actual computer code is slightly more subtle because of logic to speed execution, but it is basically that seen in Fig. IX-10.

I. CONTOUR

As for 'INTERCEPT', the problem here is to define a fast, complete, and logically complex algorithm. The basic structure of the algorithm is shown in Fig. IX-11, although the actual code is much more complicated in an attempt to speed execution.

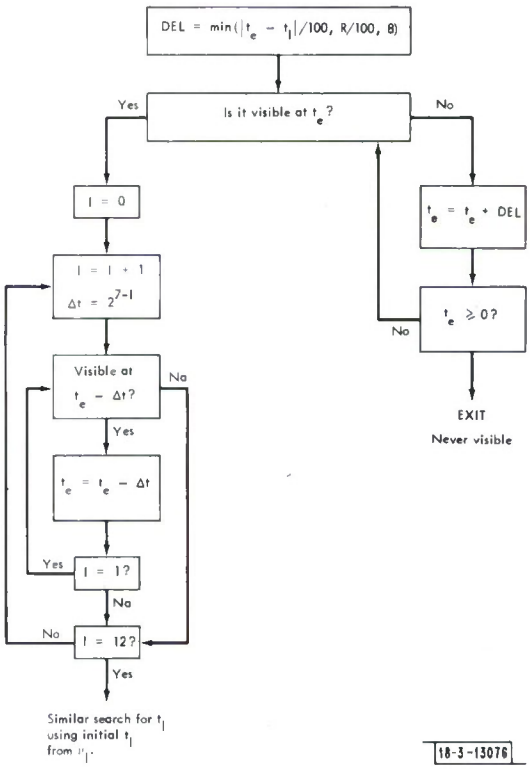

Fig. IX-9. Search logic for 'VISIBLE'.

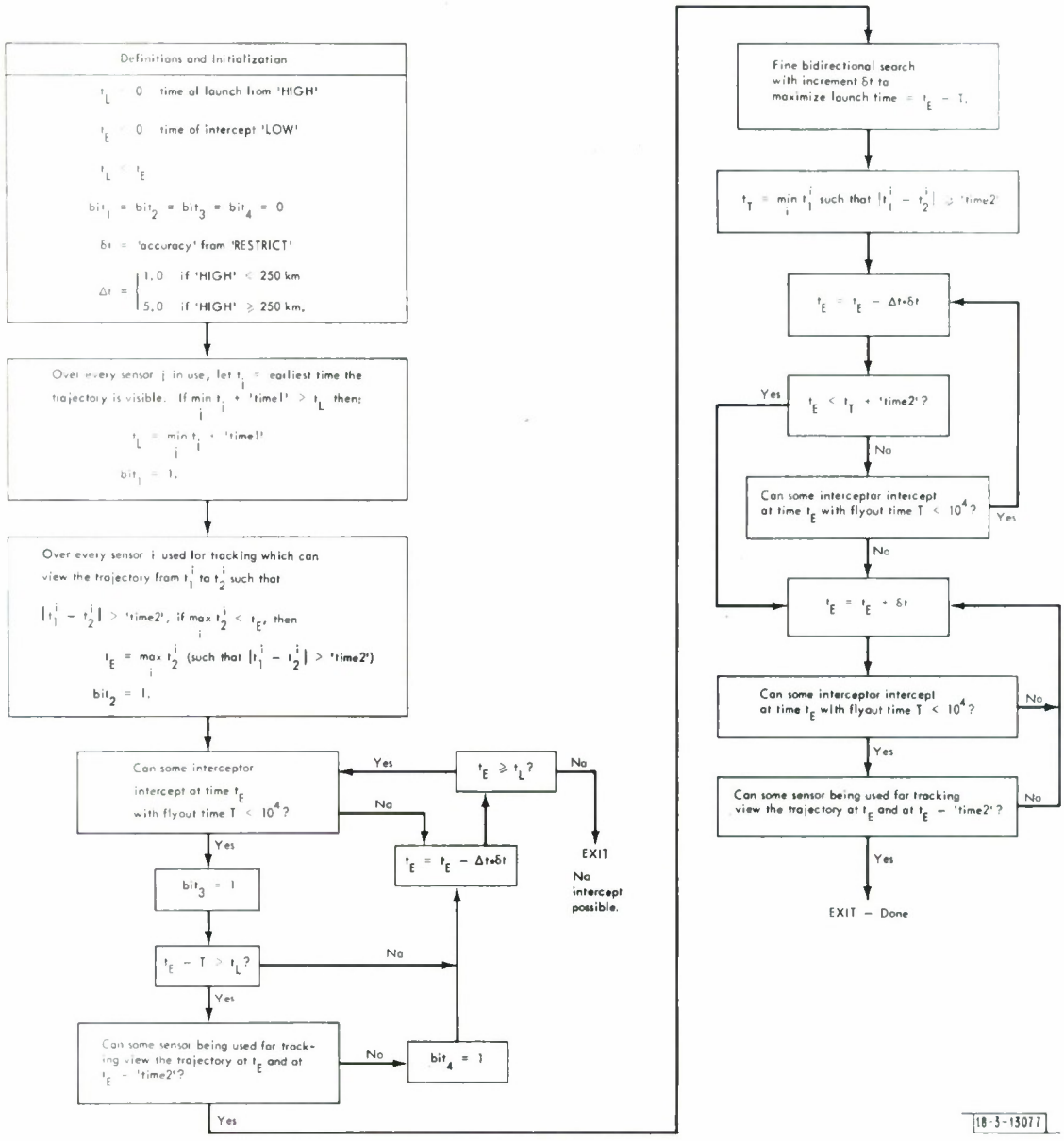

Fig. IX-10. Basic logic for 'INTERCEPT'.

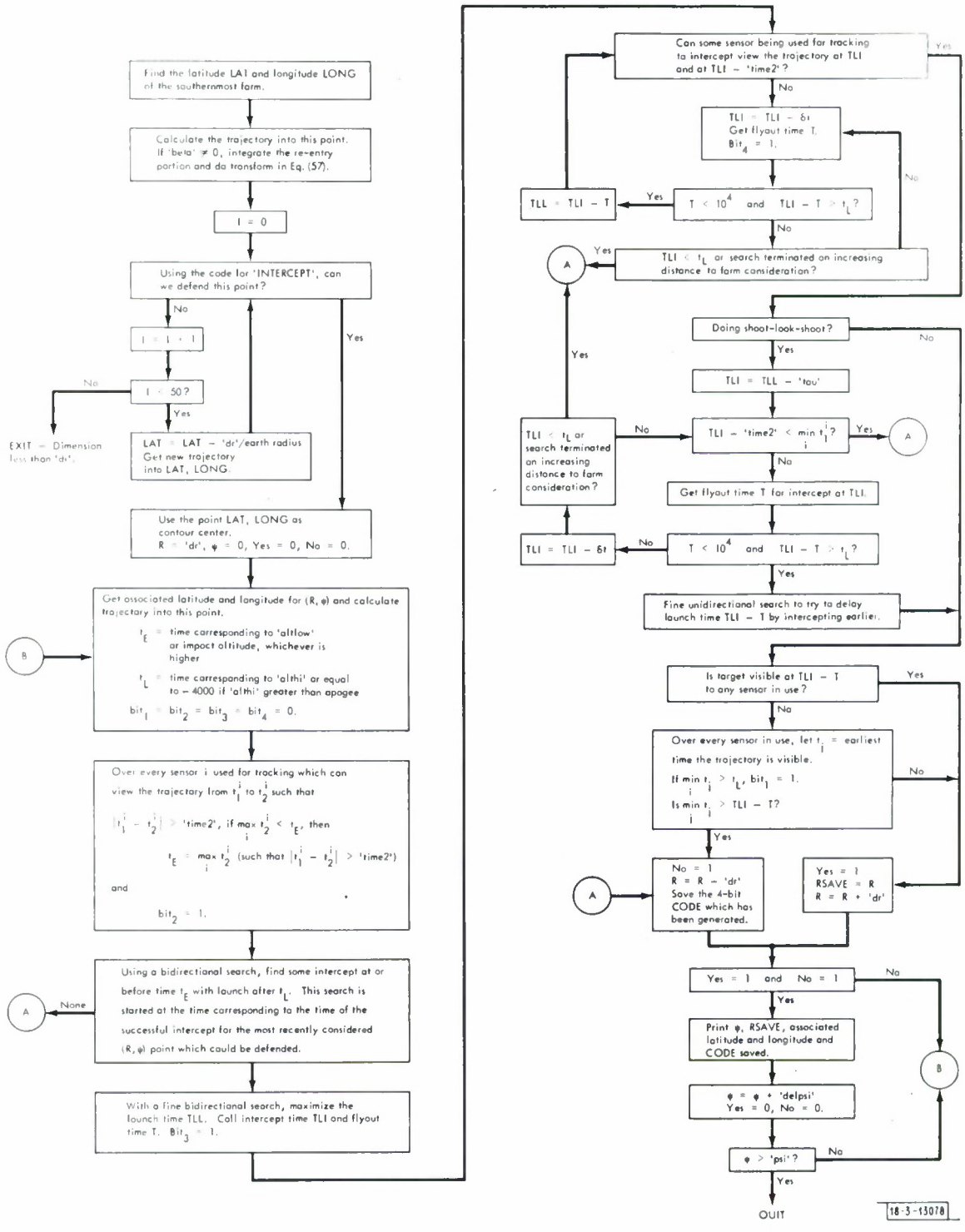

Fig. IX-11. Basic logic for 'CONTOUR'.

Note in Fig. IX-11 that many trajectories are calculated. If the contour is for vacuum trajectories, each trajectory is calculated using the mathematics for 'KEPLER'. However, if the contour is to be generated including atmospheric slowdown, an integrated re-entry portion is calculated once, transformed, and then stored in a table. These data are then transformed to correspond to different impact points (for different trajectories), and a Keplerian exoatmospheric portion is calculated and tagged on to the re-entry portion. To see the transformations, assume the earth is nonrotating and let

$$\text{launch point} = \bar{p}_o = (x_o, y_o, z_o)$$

$$\text{actual impact point} = \bar{p}_1 = (x_1, y_1, z_1)$$

for the integrated trajectory calculated using the mathematics for 'INTEGRATE'. To store these data, define

$$\bar{p}_z = \frac{\bar{p}_1 \times \bar{p}_o}{|\bar{p}_1 \times \bar{p}_o|}$$

$$\bar{p}_x = \frac{\bar{p}_1}{|\bar{p}_1|}$$

$$\bar{p}_y = \bar{p}_z \times \bar{p}_x \quad .$$

The trajectory plane is the $\bar{p}_x$, $\bar{p}_y$ plane, with $\bar{p}_x$ pointing through the impact point.

Performing the transformation (57) below on (x, y, z) and (V_x, V_y, V_z) gives a

$$\bar{\alpha}_p = \begin{bmatrix} p_{xx} & p_{xy} & p_{xz} \\ p_{yx} & p_{yy} & p_{yz} \\ p_{zx} & p_{zy} & p_{zz} \end{bmatrix} \bar{\alpha}_{\text{inertial}} \tag{57}$$

four-element $(p_x, p_y, V_{px}, V_{py})$ table to store. To make these data correspond to a different impact point $\bar{p}_2 = (x_2, y_2, z_2)$, define

$$\bar{q}_z = \frac{\bar{p}_2 \times \bar{p}_o}{|\bar{p}_2 \times \bar{p}_o|}$$

$$\bar{q}_x = \bar{p}_2$$

$$\bar{q}_y = \bar{q}_z \times \bar{q}_x \quad .$$

For each point stored, we perform Eq. (58) below to get (x, y, z) from (p_x, p_y, 0) and (V_x, V_y, V_z) from (V_x, V_y, 0).

$$\bar{\alpha}_{\text{inertial}} = \begin{bmatrix} q_{xx} & q_{yx} & q_{zx} \\ q_{xy} & q_{yy} & q_{zy} \\ q_{xz} & q_{yz} & q_{zz} \end{bmatrix} \bar{\alpha}_p \quad . \tag{58}$$

In Fig. IX-11, there are several references to getting flyout times. There is some logic coded here, not shown in the figure, which terminates searches for intercepts. If, in two successive calls for flyout times, say t_1 and t_2 (with $t_2 < t_1$), neither intercept is acceptable (because of reach or flyout time) and the range to the trajectory from each farm in use is increasing from time t_1 to time t_2, then no acceptable intercepts will be found by searching further back in time and the search is terminated.

ACKNOWLEDGMENTS

The author wishes to thank all the staff members in the Radar Concepts Group of Lincoln Laboratory for their comments about and interest in the definition of this computer program. Dr. W. C. Kellogg, presently at Boston University, provided consultation on certain mathematical issues pertaining to his past work at Lincoln on trajectory calculation and observation. Dr. J. R. Sklar of the Systems Evaluation Group at Lincoln provided an insight into useful mathematics of trajectory translation which reduce that particular computation time.

DOCUMENT CONTROL DATA – R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Lincoln Laboratory, M.I.T.	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP None

3. REPORT TITLE

DEANE: A Computer Aid for Ballistic Missile Defense Analysis

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Technical Note

5. AUTHOR(S) *(Last name, first name, initial)*

McCraith, Douglas L.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
4 November 1970	74	None

8a. CONTRACT OR GRANT NO. F19628-70-C-0230 b. PROJECT NO. 7X263304D215 c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) Technical Note 1970-6 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* ESD-TR-70-339

10. AVAILABILITY/LIMITATION NOTICES

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
None	Office of the Chief of Research and Development, Department of the Army

13. ABSTRACT

DEANE is a special-purpose computer language designed for use in a time-shared environment by a ballistic missile defense systems analyst. In essence, it is a sophisticated calculator whose modular design allows the user to request basic computations in a convenient fashion. The interpretation of the computational results is not made by DEANE under the pretense of being a simulator, but is left to the user.

This report is a user's manual describing the computations available and giving examples of how they may be ordered to solve typical problems. Also presented is a description of the logical and/or mathematical foundations of the computations.

14. KEY WORDS

computer language
computer programs
ballistic missile defense systems analysis
Keplerian trajectories