5 RUN-TIME MODELS - FLIGHT PROCESSING

5.1 OVERVIEW

Flight Processing (FP) provides the EADSIM with two basic functions: the movement of platforms and the reporting of the updated state (position, velocity, and in some cases orientation and status data). Both functions occur for each simulation time step.

Processing for each time step begins when the engagement records are received from the C3I model. Each engagement record contains the platform identifiers and commands that slave the actions of the FP model to the battle management and engagement modeling functions of the C3I model. The platform identifiers are used to determine which platform is taking the action and against which platform the action is being taken. The command portion of the record identifies the action to be taken.

The FP model provides all of the modeling of platform movement. It models ground platform movement, tactical ballistic missile, explicit SAM 3-DOF and constant velocity missiles, and airplane and helicopter flight. The actions identified through the C3I interface determine the current movement mode of each platform. For example, a vector command from the C3I model causes the FP model to vector the aircraft to the current position of the target platform.

The FP model reports the updated status of each platform to the other run-time models. The C3I model uses this information for the current position and velocity of the platforms. The Detection and Propagation models use this information for both the current state and the status of the platforms (e.g., determination of dead and inactive platforms).

5.2 INPUTS

As with all of the run-time models, the FP model is driven by its inputs, which can be broken out by their two sources. Input data files are developed by the user through Scenario Generation, while the other input is received from the C3I model. The EADSIM may be run in several configurations. The FP model is required in all these configurations for useful results. FP will run without the dynamic interaction of the combatants.

5.2.1 Data Files

The data files are the method by which the user establishes the scenario and all the parameters associated with it. Since the FP model performs only a portion of the processing, it does not use all of the inputs. The FP model reads the scenario file specified at the time of the simulation execution. From this file, the FP model

uses the laydown file names, paths for element files, map file name, output data file names, and the process flags.

The laydown file names are used to locate the laydown files, which provide the data specific to each platform in the scenario. The laydown files provide FP with the waypoints, flight leaders, system types, user-defined targets, initial flight modes for each platform in the laydown, and satellite initialization data.

The paths to the element files determine which set of user-defined elements will be used. The FP model uses only the aircraft, weapon, and system elements. The aircraft elements provide the data for the modeling of each airplane and helicopter type. The airplane data include maximum speed, weight, maximum gravitational force, thrust, wing area, wing span, altitude, and turn parameters for each aircraft type. The helicopter data include number of blades, blade length, tip velocity, coefficients of lift and drag, weights, and available power.

The weapon elements provide information for flight modeling of each missile type flown by the FP model. Only weapons of type cruise and ballistic are flown by this model. Cruise missiles are modeled from the velocity, minimum range, and maximum range only. The multistage ballistic missile model requires both guidance phase and flight section data to define a missile system. Guidance phase data consist of pitch angle, pitch rate, and gravity turn option as functions of time, guidance mode, and identification of iteration guidance parameter. Flight section data consist of thrust, fuel mass, vehicle mass, tabled coefficients of lift and drag as functions of Mach number, nozzle area, reference area, and burn rate, all as functions of flight section cut-off time. Also required to define the missile flyout capability are minimum and maximum ranges. The system elements identify the weapons on the platform and whether the platform is an air or a ground platform.

The map file provides FP with the DTED file to be used and the extent of data in the DTED file (i.e., lower left and upper right corners of the data). The DTED file provides the terrain data used to move ground platforms, fly aircraft in a terrain-following mode, and determine impact of missiles and aircraft with the ground.

5.2.2 Engage Command Reception

The other input to the FP model is the dynamic engagement command reception. Data comes into the FP model from the C3I model. The majority of the information is processed as engagement records containing an engagement action and identification of two platforms. The platforms identify the platform taking the action and the platform against which the action is taken. In other cases, the engagement records contain an engagement action, a platform identifier, and a value to be set for the platform.

The flight mode for the platform taking the action is set based on the specified engagement action. In setting the flight mode, all other information needed by FP is also set.

5.3 INITIALIZATION

Initial processing within FP sets up the FP model to run with the other models and to interact in a consistent manner with them. The FP model reads the scenario file specified at execution time into the data structures, setting the paths for all files used by FP. The inter-process connection is then established between FP and the other models specified in the scenario file. The data files discussed in Subsection 5.2.1 are then read into the FP model.

Once FP is initialized, it moves all platforms to the time of the first exchange of data with the other models, the scenario interval. Because all models know the initial platform positions, the initial positions are never reported to the other models.

5.4 NAVIGATION

EADSIM has the ability to optionally model the effects of imperfect navigation. Imperfect navigation results from secondary navigation error when a GPS position and velocity solution are not available and from GPS error when a GPS solution is available. EADSIM currently supports modeling imperfect navigation for aircraft and precision guided munitions (PGMs). All other entities in EADSIM use perfect navigation. PGMs are modeled in EADSIM by complex weapons, and captive platforms. Entity navigation depends on the entity's perceived position and perceived velocity. When an entity's GPS receiver cannot acquire a position and velocity solution it is forced to rely solely on its secondary navigation. Secondary navigation systems such as inertial navigation systems (INS) suffer from a drift error. Due to this error the entity tends to drift off course at an increasing rate with respect to the amount of time that it has been without GPS. This drift causes the entity's perceived position and velocity to differ from its true position and velocity. When the entity's GPS receiver is able to produce a position and velocity solution it suffers from GPS errors which also cause the perception to differ from truth, although at a much lower magnitude. When PGMs are launched there is an associated "handoff error" that is optionally applied to the initial perceived position of the PGM. If there is no navigation device or the navigation device is selected to accumulate error only, the airframe will always fly according to truth.

There are several considerations that impact imperfect navigation. The first consideration is GPS connectivity. GPS connectivity models a GPS receiver's ability to acquire and track enough GPS satellite transmitter signals used to compute a perceived position and velocity solution in the presence of noise and jamming. The

next consideration is the navigation error. Navigation error includes secondary navigation error and GPS error. Navigation error is used to compute the entity's perceived position and velocity. At this time, perceived position and velocity are used solely to accomplish imperfect navigation of aircraft and PGMs. The imperfect knowledge of location and speed is not used to impact battle management and handover of errored target track locations.

The magnitude of the GPS error as a function of connected satellite geometry resulting in Dilution of Precision (DOP) is not explicitly modeled in EADSIM. The purpose of the GPS module is for mission level planning and not engineering level analysis of GPS. The GPS Interface and Analysis Tool (GIANT) software is an engineering level model that provides a more detailed analysis of the GPS errors. In EADSIM the GPS error is modeled using user defined random number distributions.

5.4.1 Perception Data

Perception includes an entity's perceived position, velocity, and orientation. Perception is updated for every platform in the scenario at each integration interval in flight processing (FP). The perception update includes the calculation of the navigation position and velocity errors, perceived position, perceived velocity, and perceived orientation. If a navigation element is not associated with a platform then the perception is set equal to the truth for that platform and perfect navigation occurs. If the accumulate error only option is selected then the perception is also set equal to the truth for this platform and the error is accumulated.

EADSIM maintains a copy of each platform for each of the four models including C3I, FP, Detection, and Propagation. A copy of the truth position, velocity, and orientation is maintained for each of these models. The truth data is updated at each scenario interval if the platform status has changed. The perception data is updated regardless of whether the platform status has changed. A pointer to the perception data may be stored on each platform for each model. Perception data is only used in the C3I and FP models therefore the pointer is only set for these models. During initialization the C3I perception pointer for each platform is set to point to the FP perception pointer. C3I and FP thus use common memory for perception data. When the perception data is updated in FP, it is also updated in C3I due to the common memory usage as shown in ***Figure 5.4-1***.

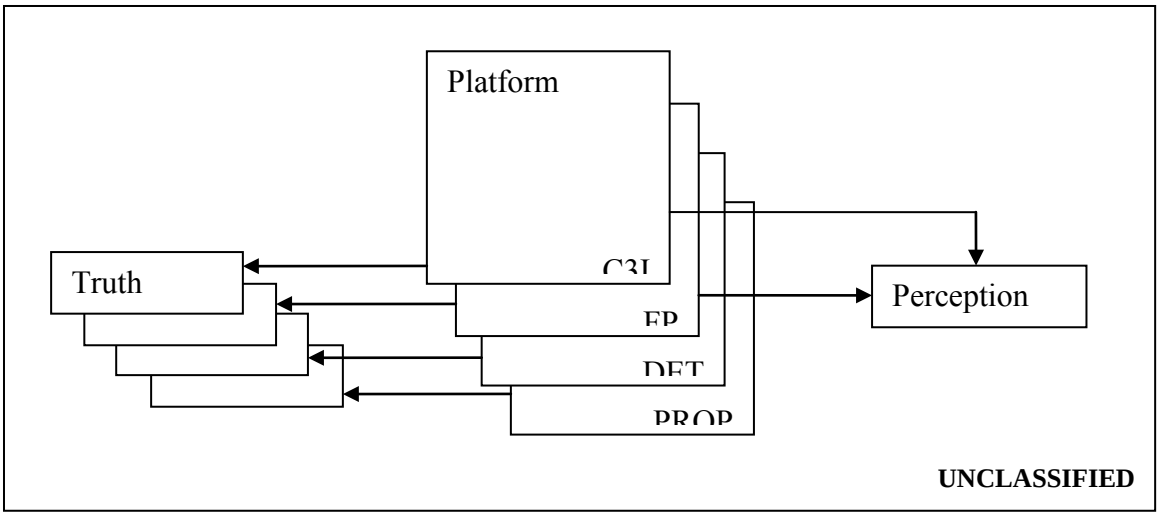

Figure 5.4-1 Perception data storage

5.4.2 Navigation Error

Navigation error is produced by the GPS receiver when a perceived position and velocity solution is available from the GPS and by secondary navigation when it is not. A perceived position and velocity solution is not available from the GPS while the receiver is in the Not Connected overall connectivity state. The solution is available in all other states. Navigation error is computed at each integration time step in each applicable flight mode in FP in order to update perception.

5.4.2.1 GPS Navigation Error

GPS navigation error is characterized by user defined random number distributions in the XY and Z directions. There are three different levels of error modeled for GPS navigation in EADSIM as shown in ***Table 5.4-1***. Separate distributions are defined for perceived position and perceived velocity for each level of error. State 5 alone has two levels of error modeled to enable the receiver's ability to account for the ionospheric correction provided by the L2 signal when it is available. State 3 does not have the ability to account for ionospheric correction.

Table 5.4-1 Levels of GPS Navigation Error

State 5 Dual	Position	Velocity
State 5 Single	Position	Velocity
State 3	Position	Velocity

The XY direction is defined as the local horizontal direction and the Z is defined as the local up direction. The distribution defined for the XY distribution is used to generate the error in the east and north directions. The distribution in the horizontal plane is the bivariate normal distribution with the input XY error

representing the sigma independently applied in the X and Y directions. If randomness has been eliminated from the scenario run, the errors will be zero in all directions.

5.4.2.2 Secondary Navigation Error

EADSIM has the ability to model any secondary navigation method whose error can be described by a drift rate as a function of time using up to a ninth order polynomial. Secondary navigation can be used by itself as an entity's only form of navigation or as a backup for GPS navigation. While using secondary navigation an entity's position and velocity errors increase at a pre-determined drift rate. When used with GPS the drift rate is a function of the amount of time since the GPS receiver was in one of the connected states. If secondary navigation is the only form of navigation the drift rate is a function of the amount of time it has been used for navigation which is time since activation of the platform. The most common form of secondary navigation used as backup for GPS is the Inertial Navigation System (INS). An INS uses the time history of acceleration measurements to determine the position and velocity of an entity.

Secondary navigation error is characterized in EADSIM by ninth order curve-fit polynomials with user-defined coefficients which describe the secondary navigation error growth in the XY and Z directions.

$$\sigma = C_9(\Delta t)^9 + C_8(\Delta t)^8 + C_7(\Delta t)^7 + C_6(\Delta t)^6 + C_5(\Delta t)^5 + C_4(\Delta t)^4 + C_3(\Delta t)^3 + C_2(\Delta t)^2 + C_1(\Delta t)^1 + C_0$$

Where

σ	-	Standard deviation of error or sigma
C_n	-	Coefficients entered by user
Δt	-	The length of time using secondary navigation

The position error is defined by the user by entering the coefficients to the above polynomial and the velocity error coefficients are computed by the user interface and displayed on the window for convenience. The velocity error is used for computing the perceived position and velocity. The velocity error is the first time derivative of the position error.

$$\sigma_{vel} = \dot{\sigma}_{pos} = 9 \cdot C_9(\Delta t)^8 + 8 \cdot C_8(\Delta t)^7 + 7 \cdot C_7(\Delta t)^6 + 6 \cdot C_6(\Delta t)^5 + 5 \cdot C_5(\Delta t)^4 + 4 \cdot C_4(\Delta t)^3 + 3 \cdot C_3(\Delta t)^2 + 2 \cdot C_2(\Delta t) + C_1$$

The zero coefficient is C_1, the first coefficient is $2 \cdot C_2$, and so on.

The acceleration error is used for computing the perceived position and velocity. The acceleration error sigma is computed by taking the time derivative of the velocity polynomial.

$$\sigma_{acc} = \dot{\sigma}_{vel} = 72 \cdot C_9 (\Delta t)^7 + 56 \cdot C_8 (\Delta t)^6 + 42 \cdot C_7 (\Delta t)^5 + 30 \cdot C_6 (\Delta t)^4 + 20 \cdot C_5 (\Delta t)^3 + 12 \cdot C_4 (\Delta t)^2 + 6 \cdot C_3 (\Delta t) + 2 \cdot C_2$$

Where

σ_{acc}	-	Standard deviation of acceleration error
$\dot{\sigma}_{vel}$	-	Time derivative of velocity error polynomial
C_n	-	Position error coefficients entered by user
Δt	-	The length of secondary navigation time

The XY direction is defined as the local horizontal direction and the Z is defined as the local up direction. The sigma defined for the XY distribution is the sigma used to generate the error in the east and north (X and Y) directions. The random aspects of the secondary navigation drift are assumed to behave as bias errors. When a platform first starts to navigate using secondary navigation a one time random draw is made in the X, Y, and Z directions from a zero mean unit sigma normal distribution to obtain the sigma multiplier. A single sigma is defined for both the X and Y directions but two random draws are made, one in each direction, to obtain the sigma multipliers. The sigma multiplier is then used along with the computed sigma for the current time to compute the error in each of the east, north, and up (X, Y, and Z) directions. The sigma is defined in the XY and Z directions. The sigma in the XY direction represents the sigma of a bivariate normal distribution in the east and north directions. If randomness has been eliminated from the scenario run, the errors will be zero in all directions. The position error is computed in each direction as follows.

$$P_X = \rho_{posX} \cdot \sigma_X$$
$$P_Y = \rho_{posY} \cdot \sigma_Y$$
$$P_Z = \rho_{posZ} \cdot \sigma_Z$$

Where

$\vec{P}$	-	Position error in ENU coordinates
$\vec{\rho}_{pos}$	-	Position sigma multiplier
$\vec{\sigma}_{pos}$	-	Position sigma computed from polynomial

The velocity error is computed as follows.

$$V_X = \rho_{velX} \cdot \sigma_{velX}$$

$$V_Y = \rho_{velY} \cdot \sigma_{velY}$$

$$V_Z = \rho_{velZ} \cdot \sigma_{velZ}$$

Where

$\vec{V}$ - Velocity error in ENU coordinates

$\vec{\rho}_{vel}$ - Velocity sigma multiplier

$\vec{\sigma}_{vel}$ - Velocity sigma computed from velocity error polynomial

The acceleration error is computed as follows.

$$A_X = \rho_{accX} \cdot \sigma_{accX}$$

$$A_Y = \rho_{accY} \cdot \sigma_{accY}$$

$$A_Z = \rho_{accZ} \cdot \sigma_{accZ}$$

Where

$\vec{A}$ - Acceleration error in ENU coordinates

$\vec{\rho}_{acc}$ - Sigma multiplier

σ_{acc} - Sigma computed from acceleration error polynomial

5.4.3 Perceived Position

The perceived position is defined in ECEF coordinates. The perceived position is initialized upon activation of the platform using the navigation position error. A local ENU coordinate system centered at the platform position is established for computation of the navigation error. This perceived position is computed relative to the truth position at the end of each integration interval. The ENU navigation position error vector is computed as described in section 5.4.2. The rotation matrix used for the transformation between the ECEF and ENU coordinate systems is computed using the platform truth position.

$$\vec{P}_{err} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}$$

The position error in ECEF coordinates is then computed as follows:

$$\vec{P} = \left[\vec{R}\right]^T \left[\vec{P}_{err}\right]$$

Where

$\vec{P}$ - Position error vector in ECEF coordinates

$\vec{R}$ - Rotation matrix from ECEF to ENU

$\vec{P}_{err}$ - ENU navigation position error vector

The perceived position is then computed using the position error vector:

$$\vec{P}_p = \vec{P}_t + \vec{P}$$

Where

$\vec{P}_p$ - Entity perceived position

$\vec{P}_t$ - Entity truth position

$\vec{P}$ - Position error vector in ECEF coordinates

The perceived position is updated at each integration time step during flight processing. If the entity is using GPS then the perception update algorithm is the same as the initialization algorithm. But if the entity is using backup navigation the update computation is different. This is due to a direct time correlation between the velocity and position error components. To compute the position error the velocity error is first computed in ENU coordinates using the method described in section 5.4.2.2. The velocity error is then summed with the initial error to obtain the total velocity error. The initial error is the GPS error at the time when the entity started using backup navigation, the initial error generated from the user-defined initial error distributions for cold handoff for a PGM, or the handoff error for a PGM. The handoff errors are used if they are not zero and the hot handoff option is selected. If the cold handoff option is selected, the user-defined initial error distributions for cold handoff are used. All handoff errors are set to zero if a PGM acquires GPS after handoff. If an entity goes from using GPS navigation to using backup navigation then the GPS error at the time when the entity started using backup navigation is used as the initial error.

$$\vec{V}_{err} = \vec{V}_{init} + \vec{V}_{entity}$$

Where

$\vec{V}_{err}$ - Total velocity error

$\vec{V}_{init}$ - Initial error including cold handoff, GPS, and handoff

$\vec{V}_{entity}$ - Entity backup navigation error

The velocity error in ECEF coordinates is then computed as follows:

$$\vec{V} = \left[\vec{R}\right]^T \left[\vec{V}_{err}\right]$$

Where

$\vec{V}$ - Velocity error vector in ECEF coordinates

$\vec{R}$ - Rotation matrix from ECEF to ENU

$\vec{V}_{err}$ - ENU navigation velocity error vector

The acceleration error is then computed in ENU coordinates as described in section 5.4.2.2. The acceleration error in ECEF coordinates is then computed as follows:

$$\vec{A} = \left[\vec{R}\right]^T \left[\vec{A}_{err}\right]$$

Where

$\vec{A}$ - Acceleration error vector in ECEF coordinates

$\vec{R}$ - Rotation matrix from ECEF to ENU

$\vec{A}_{err}$ - ENU navigation acceleration error vector

The change in position from the previous integration time step to the current is then computed.

$$\Delta P = CurrP_t - PrevP_t$$

Where

ΔP - Change in position from previous integration time step to current

$CurrP_t$ - Current truth position

$PrevP_t$ - Truth position from previous integration time step.

The perceived position is then computed using the change in position, previous perceived position, velocity error, acceleration error, and the integration time step.

$$\vec{P}_p = \vec{P}_{p0} + \Delta\vec{P} + \vec{V} \cdot t + \frac{1}{2}\vec{A} \cdot t^2$$

Where

$\vec{P}_p$ - Entity perceived position

$\vec{P}_{p0}$ - Perceived position from previous integration time step

ΔP - Change in position from previous integration time step to current

$\vec{V}$ - Velocity error vector in ECEF coordinates

t - Integration time step

$\vec{A}$ - Acceleration error vector in ECEF coordinates

5.4.4 Perceived Velocity

The perceived velocity is defined in ECEF coordinates. The perceived velocity is initialized at the start of the scenario using the navigation velocity error. A local ENU coordinate system centered at the platform position is established for computation of the navigation error. The rotation matrix used for the transformation between the ECEF and ENU coordinate systems is computed using the platform position. The ENU navigation velocity error vector is computed as described in section 5.4.2. The velocity error is then rotated into ECEF coordinates as follows.

$$\vec{V} = \left[\vec{R}\right]^{T}\left[\vec{V}_{err}\right]$$

Where

$\vec{V}$ - Velocity error vector in ECEF coordinates

$\vec{R}$ - Rotation matrix from ECEF to ENU

$\vec{V}_{err}$ - ENU navigation velocity error vector

The perceived velocity is then computed using the velocity error vector:

$$\vec{V}_p = \vec{V}_t + \vec{V}$$

Where

$\vec{V}_p$ - Entity initial perceived velocity

$\vec{V}_t$ - Entity truth velocity

$\vec{V}$ - Velocity error vector in ECEF coordinates

The perceived velocity is updated at each integration time step during flight processing. If the entity is using GPS error then the perception update algorithm is the same as the initialization algorithm. But if the entity is using backup navigation the update computation is different. This is due to a direct time correlation between the position, velocity, and acceleration error components. The acceleration error is computed in ENU coordinates as described in section 5.4.2.2. The acceleration error is then rotated into ECEF coordinates.

$$\vec{A} = \left[\vec{R}\right]^{T}\left[\vec{A}_{err}\right]$$

Where

$\vec{A}$ - Acceleration error vector in ECEF coordinates

$\vec{R}$ - Rotation matrix from ECEF to ENU

$\vec{A}_{err}$ - ENU navigation acceleration error vector

The change in velocity from the last integration time step to the current is then computed.

$$\Delta\vec{V} = VCurr_t - V\,Prev_t$$

Where

$\Delta\vec{V}$ - Change in velocity from the last integration time step to the current

$VCurr_t$ - Current truth velocity

$V\,Prev_t$ - Truth velocity at last integration time step

The perceived velocity is then computed.

$$\vec{V}_p = \vec{V}_{p0} + \Delta\vec{V} + \vec{A} \cdot t$$

Where

$\vec{V}_p$ - Perceived velocity

$\vec{V}_{p0}$ - Perceived velocity at previous integration time step

$\Delta\vec{V}$ - Change in velocity from the last integration time step to the current

$\vec{A}$ - Acceleration error vector in ECEF coordinates

t - Integration time step

5.4.5 Use of Perception

Currently perception is only used for navigation of aircraft and precision guided munitions (PGMs). PGMs are modeled as captive platforms and complex weapons with navigation elements associated at the system level. If the Propagation option is not selected on the Edit Scenario Processing Options window no errors are accumulated or applied. In this case all platforms in the scenario fly using perfect navigation. The Propagation model is necessary for GPS Connectivity calculations.

5.5 AIRCRAFT FLIGHT MODELING

There are two major aspects to the flight modeling in FP. The first is the determination of where the aircraft should fly. This decision is handled through computation of the direction vector and is affected by navigation errors. Navigation error causes the platform to have a false sense of position and velocity. This false

sense of position and velocity causes the pilot or auto navigation system to apply false corrections to the vehicle's course. Navigation is only errored on platforms with Navigation elements associated at the system level with the Apply Error to Navigation option selected. Navigation error is only available to aircraft, captive platforms, and complex weapons using an airframe. While the 3-DOF SAM is a complex weapon, it does not use the airframe modeling.

If the Accumulate Error only option is selected for a host platform the error will be accumulated for handoff to a captive platform or complex weapon with which a navigation element is associated.

If the Accumulate Error only option is selected for a captive platform or complex weapon the weapon will fly to truth and then apply error upon impact. The handoff error will be saved at the time of launch to aid in endgame miss distance computations.

The second aspect of flight modeling involves the actual algorithms used by the flight model for the aircraft flight. The equations describing the aircraft flight are essentially the same for all flight modes.

There are four flight modes that the user can define: Waypoints, Wingman, Scramble, and Airborne Refueling. For Waypoints and Wingman modes, an aircraft will remain in these initial modes until the end of the scenario unless the platform's ruleset initiates another flight mode. There are six modes that the rulesets schedule: Engage, Drag, Return to Base (RTB), Avoid, Vector, and Maneuver. Each of these flight modes are discussed in subsequent subsections.

5.5.1 Computation of the Direction Vector

The direction vector is used to determine the direction in which to move the platform. Calculation of the direction vector depends, in part, on whether the platform is flying according to truth or perception. This is determined prior to calculation of the direction vector for each flight mode.

If the platform is flying according to truth, the truth direction vector is first calculated according to the following equation:

$$\vec{D}ir_T = \vec{D}_T - \vec{L}_T$$

Where

$\vec{D}ir_T$ - True direction vector (m)

$\vec{D}_T$ - True destination waypoint position (m)

$\vec{L}_T$ - True aircraft current position (m).

The perceived direction vector is then calculated based on whether the platform is updating perception. This is determined prior to calculation of the direction vector for each flight mode.

On the other hand, if the platform is flying according to perception, the perceived direction vector is calculated first. This direction vector is based on the platform's perception of his own location, his perception of true North, his perception of up, and his true destination. The true destination depends on the current flight mode of the platform. Waypoint flight mode for example uses the next waypoint location as the destination. First the perceived direction vector, from the platform's perception of its position to the true position of its destination, is computed.

$$\vec{D}ir_P = \vec{D}_T - \vec{L}_P$$

Where

$\vec{D}ir_P$ - Perceived direction vector (m)

$\vec{D}_T$ - True destination waypoint position (m)

$\vec{L}_P$ - Perceived aircraft current position (m).

The above direction vector accounts for the platform's perception of its own location. Now the platform's perception of true North and up must be accounted for. The platform's perceived position and perceived orientation are used to rotate the perceived direction vector into Body Frame coordinates. The function ECEFToBody described in Methodology Manual section B10.2.4 is used to obtain the rotation matrix and the function Rotate1 described in Methodology Manual section B10.3.7 is used to perform the actual rotation. Then the platform's true position and orientation are used to rotate the body frame direction vector back into ECEF coordinates. Again, the function ECEFToBody is used to obtain the rotation matrix, and then the Rotate2 function described in Methodology Manual section B10.3.11 is used to perform the actual rotation. This yields the true direction vector which is then used throughout the remainder of the flight algorithms.

Supposing the truth direction vector has been calculated and it is now necessary to determine the correlated perceived direction vector, the function RotateDirVec can be used. This function first calls ECEFToBody with the truth position and truth orientation vector to acquire the appropriate rotation matrix. Rotate1 is then used to transfer the truth direction vector into body coordinates. Next, ECEFToBody is called with the perceived position and perceived orientation vector. Finally, Rotate2 is used to get the perceived direction vector in ECEF coordinates. The same methodology can be employed in reverse to yield the truth direction vector from the perceived direction vector.

5.5.2 Aircraft Acceleration Calculations

Both airplanes and helicopters are flown using a fractional update interval on a modified 3-DOF model of motion. If the scenario is running in an HLA federation and the ABT update rate is less than 1.0 second, the aircraft will be flown using the specified update time. Otherwise, a 0.5 second update time is used. The modeling accounts for the effects of gravity and atmosphere when calculating the acceleration force on the aircraft. The forces of drag, gravity, and thrust are summed to calculate the resulting acceleration acting on both types of aircraft.

For airplanes, thrust and fuel consumption data may be specified for use in military (MIL) power mode as well as afterburner (AB) power mode. For signature purposes, an airplane is considered to be in AB mode when 1) its speed is greater than the specified afterburner speed or 2) when it is flying a leg of an intercept profile which specifies that afterburner be used or 3) when the computed desired thrust for the current interval is greater than the available MIL power thrust and AB mode is available. There is also an option to specify hi-fidelity flight performance data. This includes user-specified tables for thrust and fuel flow as a function of altitude and mach. To characterize turn-G, hi-fidelity G-Available tables may be specified as a function of altitude, mach, weight, zero-lift drag coefficient (Cd_0), and specific power (P_s).

Like airplanes, helicopters are flown using a modified 3-DOF model of motion, which accounts for drag and weight forces when computing the required thrust to allow for acceleration along the direction vector. Unlike airplanes, however, thrust is required to balance both weight and drag since helicopters have a limited lift generation capability. Also, helicopter thrust is limited by both the maximum producible thrust by the rotor disk and by the amount of power available to generate that thrust.

5.5.2.1 Airplane Acceleration

Airplane drag is computed as a function of air density and lift. The air density is computed using pressure curve fit and temperature gradient data derived from the 1962 Standard Atmosphere data. See Subsection 5.6.6 for more information on the density calculations.

Once known, the air density, ρ, and the velocity, V, of the aircraft are used to compute the dynamic pressure, Q, as follows:

$$Q = 0.5\rho V^2$$

where

Q - dynamic pressure ($kg/msec^2$)
ρ - air density (kg/m^3)
V - aircraft velocity (m/sec).

The coefficient of force Coef, used for calculating lift and drag, is determined by:

$$Coef = Q \times S$$

where

Coef - coefficient of force (N)
Q - dynamic pressure ($kg/msec^2$)
S - aircraft wing area (m^2).

The lift coefficient, C_L, is calculated next. The value used for gravitational force of a turn, G, will be one if the aircraft is flying straight; otherwise, it will be the command gravitational force. If the hi-fidelity option is not selected for an airplane, the commanded turn G is limited by the user specified maximum G.

If the aircraft is modeled using the hi-fidelity airplane options, commanded turn G is limited by the current G-Available for the aircraft. G-Available tables specify turn-G available at full MIL power and optionally at full AB power. G-Available tables are specified as a function of any combination of: MSL altitude, mach, weight, zero-lift drag coefficient (Cd_0), and specific power (P_s).

When using MSL altitude and mach lookup criteria, G-Available is specified based on the aircraft's level flight envelope. ***Figure 5.5-1*** shows a notional flight envelope. For altitude and mach values within the flight envelope, G-Available values are specified. For altitude and mach values outside the flight envelope, zero values are populated into the G-Available table.

Figure 5.5-1 shows top speeds and a ceiling altitude achievable at MIL power. The figure also overlays a larger envelope that shows top speeds and a ceiling altitude achievable at MAXAB power. If AB is not available, the G-Available table based on the MIL power flight envelope is used exclusively. In this case, the G-Available table should be sized (in altitude and mach) to contain the entire MIL power flight envelope.

If AB is available, MIL and maximum AB G-Available tables based on the Maximum AB power flight envelope are used. In this case, the G-Available tables for MIL and MAXAB should be sized (in altitude and mach) to contain the entire MAXAB power flight envelope.

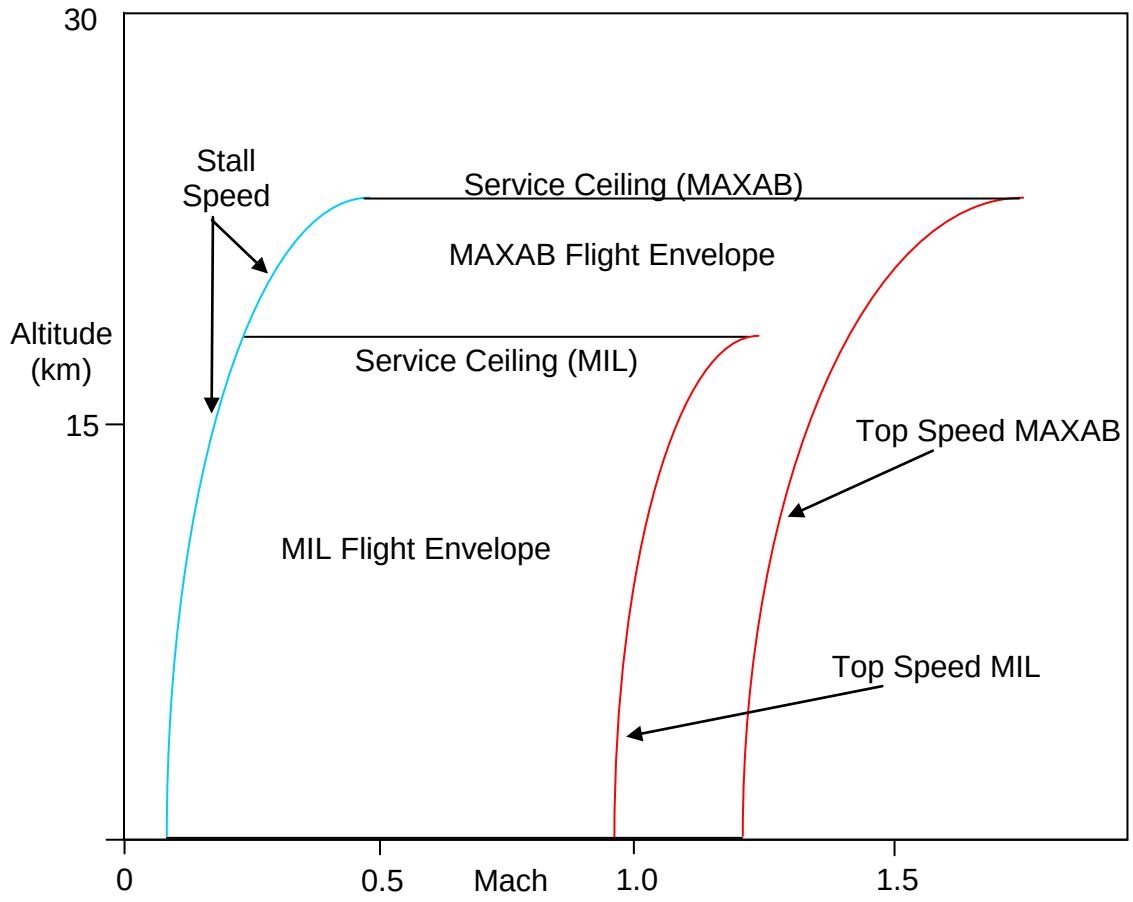

Figure 5.5-1 Notional Flight Envelope

The aircraft's current G-Available is updated each integration time step by performing a lookup based on the user specified G-Available criteria. The P_s criterion value is computed by the following equation;

$$P_s = \frac{(ThrustAvailable - Drag)}{Weight} * V$$

where

P_s	-	aircraft's current specific power (m/s)
ThrustAvailable	-	aircraft's maximum available thrust force (N)
Drag	-	aircraft's current drag force (N)
Weight	-	aircraft's current total weight, including weapons load (N)
V	-	aircraft's current speed (m/s) .

The aircraft's current mach is computed as a function of speed of sound at current altitude.

$$\text{Mach} = \frac{CurrentSpeed}{SpeedofSound}$$

where

Mach - aircraft's current mach
CurrentSpeed - aircraft's current velocity magnitude/speed (m/s)
SpeedofSound - speed of sound at current altitude (m/s).

The table lookup for G-Avail begins by computing the values for each user specified lookup criterion. G-Available tables in flight manuals are typically specified as a function of all five possible criteria; MSL altitude, mach, weight, Cd_0, and P_s.

When defining G-Available tables, the criteria set should, as a minimum, altitude and mach. G-Available tables characterize turn-G available based on the current values for each specified lookup criterion. For each table location that is within the aircraft's performance envelop, a non-zero G-Available value is specified. A zero G-Available value indicates a region outside the aircraft's performance envelop.

The process of limiting turn-G begins by determining G-Available for the aircraft's current conditions. If both altitude and mach criteria are specified, adjustments may be made to the altitude and/or mach lookup values to lookup a turn-G value that is within the aircraft's flight envelop. If the aircraft's current mach is off the G-Available table, i.e., either below the minimum or above the maximum mach value specified in the table, the lookup algorithm adjusts the current mach to be on the table. If the current mach is less than the minimum mach value specified for the table, mach is set to the minimum value. If the current mach is greater than the maximum mach value specified for the table, mach is set to the maximum value. A similar adjustment is made for altitude if the aircraft's current altitude is off the G-Available table. If an aircraft is flying within the mach/altitude boundaries specified for the table, no adjustments are made to the current mach/altitude lookup values. The lookup algorithm then evaluates the G-Available value for the current/adjusted mach/altitude table location. If the G-Available value is zero, this indicates a region outside the performance envelop and the lookup algorithm begins to reduce altitude while holding mach constant until a non-zero G-Available value is identified, the lowest altitude value for the table is reached, or the aircraft's defined floor altitude is reached. If a non-zero G-Available value is not identified by holding mach constant and reducing altitude, then altitude is reset to its original "adjusted onto table" value and the lookup algorithm then reduces the mach value by one column. With mach and altitude reset, the

lookup algorithm then continues by evaluating the G-Available values and reducing altitude until the entire table is evaluated or a non-zero value is identified. Finally, the maximum G-Available at full MIL thrust is determined to be the G-Available value at the current/adjusted mach/altitude table location. It should be noted that G-Available lookup criteria other than mach and altitude may be specified. However, only mach and/or altitude may be adjusted during the lookup to account for the aircraft's flight envelop.

The EADSIM flight model applies the airframe's service altitude limit as a ceiling to all aircraft flight. When operating with the hi-fidelity aircraft option and the lookup requires a reduction in altitude, the ceiling will be reduced to match the altitude determined from the table lookup.

$$C_L = (G * W) / Coef$$

where

C_L - coefficient of lift
G - turn gravitational force
W - aircraft total weight (N)
Coef - coefficient of force (N).

To calculate drag effects, the aspect ratio of the aircraft wings must first be calculated:

$$AR = b^2/S$$

where

AR - aspect ratio
b - aircraft wing span (m)
S - aircraft wing area (m^2).

Then the zero lift drag coefficient, Cd_0, is computed by:

$$Cd_0 = (T_{max}/Coef) - AR\ C_L^2$$

where

Cd_0 - zero lift drag coefficient
T_{max} - user-specified military maximum thrust of the aircraft (N) for non-hifidelity airplanes, or the maximum military thrust at the specified cruise altitude and max speed for hi-fidelity aircraft .
Coef - coefficient of force at the specified maximum speed and cruise altitude (N)
AR - aspect ratio
C_L - coefficient of lift at aircraft's empty weight

Next, the induced drag coefficient, Cd_i, is computed by:

$$Cd_i = C_L^2 / (\pi AR)$$

where

Cd_i	-	induced drag coefficient
C_L	-	coefficient of lift
AR	-	aspect ratio.

Finally, the effect of drag is computed from:

$$D = Coef (Cd_0 + Cd_i)$$

where

D	-	drag (N)
Coef	-	coefficient of force (N)
Cd_0	-	zero lift drag coefficient
Cd_i	-	induced drag coefficient.

If the aircraft has fuel, the desired acceleration for the internal FP model time step is computed as:

$$a = \frac{\left(V_{Cmd} - V\right)}{dt}$$

where

a	-	desired acceleration (m/sec^2)
V_{Cmd}	-	commanded speed (m/sec)
V	-	aircraft velocity (m/sec)
dt	-	integration time step (sec).

The required thrust is then computed by the following formula:

$$T = ma + D + W\cos\phi$$

where

T	-	thrust required to achieve commanded speed (N)
m	-	aircraft mass (kg)
a	-	desired acceleration (m/sec^2)
D	-	drag force (N)
W	-	aircraft weight (N)

ϕ - angle between velocity vector and local vertical (rads).

The term Wcos ϕ accounts for the effects of gravity along the direction of the velocity vector.

The required thrust is then limited. For aircraft not using the hi-fidelity option, the required thrust is limited to a value between zero and the aircraft's maximum available thrust. The aircraft's maximum available thrust is the user-specified maximum MIL thrust or maximum AB thrust if AB is available for the aircraft.

If the aircraft is modeled using hi-fidelity flight performance tables, required thrust is limited based on the user-specified thrust tables. Thrust and fuel flow tables are specified as a function of MSL altitude and mach for IDLE and MIL power levels and optionally for maximum AB power. The IDLE thrust provides a lower limit on available thrust at given altitudes and mach values while MIL and MAXAB thrust tables provide upper limits on the available thrust at given altitudes and mach values. The IDLE thrust table characterizes engine thrust available at IDLE throttle setting as a function of altitude and mach. The IDLE fuel flow table characterizes fuel consumption rates at IDLE throttle setting as a function of altitude and mach. The MIL thrust table characterizes engine thrust available at full MIL throttle setting as a function of altitude and mach. The MIL fuel flow table characterizes fuel consumption rates at full MIL throttle setting as a function of altitude and mach. The optional maximum AB thrust table characterizes engine thrust available at maximum AB throttle setting as a function of altitude and mach. The maximum AB fuel flow table characterizes fuel consumption rates at maximum AB throttle setting as a function of altitude and mach.

Figure 5.5-1 shows top speeds and a ceiling altitude achievable at MIL power. The figure also overlays a larger envelope that shows top speeds and a ceiling altitude achievable at MAXAB power. If AB is not available, tables based on the MIL power flight envelope are used. In this case, the thrust and fuel flow tables for IDLE and MIL should be sized to contain the entire MIL power flight envelope. Furthermore, the IDLE and MIL thrust and fuel flow tables should use the same number of rows and columns and the same altitude/mach values in each row/column.

If AB is available, tables based on the maximum AB power flight envelope are used. In this case, the thrust and fuel flow tables for IDLE, MIL and maximum AB should be sized to contain the entire maximum AB power flight envelope. Furthermore, the IDLE, MIL and maximum AB thrust and fuel flow tables should use the same number of rows and columns and the same altitude/mach values in each row/column.

For each mach/altitude location in the table that is within the aircraft's performance envelop, a non-zero thrust value is specified. A zero thrust value in a

thrust table indicates a mach/altitude region outside the aircraft's flight envelope. Lookups are performed using these tables to determine available engine thrust for the aircraft's current mach and altitude. The available engine thrust is then used to limit required thrust.

The process of limiting required thrust begins by determining the available MIL thrust for the aircraft's current mach/altitude, via the MIL thrust table. If the aircraft's current mach is off the thrust table, i.e., either below the minimum or above the maximum mach value specified in the table, the lookup algorithm adjusts the current mach to be on the table. If the current mach is less than the minimum mach value specified for the table, mach is set to the minimum value. If the current mach is greater than the maximum mach value specified for the table, mach is set to the maximum value. A similar adjustment is made for altitude if the aircraft's current altitude is off the thrust table. If an aircraft is flying within the mach/altitude boundaries specified for the thrust table, no adjustments are made to the current mach/altitude lookup values. The lookup algorithm then evaluates the thrust value for the current/adjusted mach/altitude table location. If the thrust value is zero, this indicates a region outside the performance envelop and the lookup algorithm begins to reduce altitude while holding mach constant until a non-zero thrust value is identified, the lowest altitude value for the table is reached, or the aircraft's defined floor altitude is reached. If a non-zero thrust value is not identified by holding mach constant and reducing altitude, then altitude is reset to its original "adjusted onto table" value and the lookup algorithm then reduces the mach value by one column. With mach and altitude reset, the lookup algorithm then continues by evaluating the thrust values and reducing altitude until the entire table is evaluated or a non-zero thrust value is identified. Once a non-zero thrust value is identified, the mach/altitude values associated with the thrust value are stored. Finally, the maximum available MIL thrust is the thrust value at the current/adjusted mach/altitude table location.

Once the MIL thrust lookup has been completed, the current/adjusted mach/altitude values are used to retrieve the IDLE thrust value from the IDLE thrust table for the current/adjusted mach/altitude lookup values. If the required thrust is less than the IDLE thrust value, the required thrust is set to the IDLE thrust value.

If AB mode is available, the available maximum AB thrust is determined via the maximum AB thrust table using the same lookup algorithm as described for the MIL thrust lookup. If the required thrust is greater than the available maximum AB thrust, required thrust is limited to the maximum AB thrust determined by the lookup algorithm. If AB mode is not available and the required thrust is greater than the MIL thrust, required thrust is limited to the MIL thrust determined by the lookup algorithm.

Another important consideration for hi-fidelity aircraft is establishing the current throttle setting. Throttle setting reflects the computed required/limited thrust as a percentage between IDLE and MIL, or, MIL and maximum AB if AB mode is available. Throttle setting is established by a linear interpolation. The following examples describe how throttle setting is computed. If the required thrust is at the midpoint between the IDLE and MIL thrust values, throttle setting is computed to be 50% MIL. If the required thrust was limited to IDLE thrust, throttle setting is 1% MIL. If the required thrust was limited to MIL thrust, throttle setting is 100% MIL. If AB mode is available and the required thrust is at the midpoint between MIL thrust and MAX AB thrust, throttle setting is 50% maximum AB. If AB mode is available and the required thrust is limited to maximum AB thrust, throttle setting is 100% MAX AB.

Once the throttle setting is established, it is used to interpolate fuel flow rate values between IDLE and MIL or MIL and maximum AB in the afterburner region. The current/adjusted mach/altitude values from the MIL thrust lookup are used to retrieve the IDLE and MIL fuel flow rate values from the IDLE and MIL fuel flow tables. If AB mode is being used, the current/adjusted mach/altitude values from the MAX AB thrust lookup are used to retrieve the MAX AB fuel flow rate value from the MAX AB fuel flow table. Throttle setting is then applied using a linear interpolation to determine the fuel flow rate associated with the computed/limited required thrust.

Once the required thrust is computed and the fuel flow rate is determined, the aircraft's weight is updated.

For aircraft not using the hi-fidelity option, weight is updated by:

$$W = W - (TSFC \times T \times dt)$$

where

W - aircraft weight (N)
TSFC - thrust specific fuel consumption (1/sec)
T - available thrust (N)
dt - integration time step (sec).

For hi-fidelity aircraft, weight is updated by:

$$W = W - (FuelFlowRate \times g_0 \times dt)$$

where

W - aircraft weight (N)
FuelFlowRate – fuel flow rate obtained from the hi-fidelity table lookup (kg/sec)

g_0 - gravitational constant, 9.80665 (m/sec/sec)
dt - integration time step (sec).

If the aircraft is out of fuel, the thrust is set to zero, and the aircraft's weight is set to the aircraft's empty weight.

The actual acceleration is determined by summing forces along the velocity vector using the calculated thrust and dividing by the initial aircraft mass, as follows:

$$A = (T - D - W\cos\phi) / m$$

where

A - actual acceleration (m/sec^2)
T - aircraft thrust (N)
D - aircraft drag (N)
W - aircraft weight (N)
ϕ - angle between velocity vector and local vertical (rads)
m - aircraft mass (kg).

After determining the aircraft's acceleration, the commanded maneuvers dictate the flight path. These commanded maneuvers result in the establishment of a vector in the direction of desired flight. Depending on the flight mode, this destination vector is either a vector relative to the location of another platform or a vector derived from a desired direction of flight. There are two types of flight paths that are flown to achieve the desired direction: a straight line and a curve.

5.5.2.2 Helicopter Acceleration

Like airplanes, helicopters are flown using a simple 3-DOF flight model, which accounts for drag and weight forces when computing the required thrust to allow for acceleration along the direction vector. Unlike airplanes, however, thrust is required to balance both weight and drag since helicopters have a limited lift generation capability. Also, thrust for helicopters is limited by both the maximum producible thrust by the rotor disk and by the amount of power available to generate that thrust.

Air density is first computed at the helicopter's current altitude using a curve fit to the 1962 Standard Atmosphere data. See Subsection 5.6.6 for a detailed description of the air density computation. Next, the angle, θ, between the desired direction of flight and the ENU vertical is computed utilizing DblComputePitch from Appendix B10.

$$\theta = \pi/2 - \theta$$

where

θ - angle between direction vector and ENU vertical (rads)

Next, the acceleration magnitude required to change the helicopter from its current velocity to the commanded speed is computed:

$$a = \frac{(V_{Cmd} - V)}{dt}$$

where

a - desired acceleration (m/sec^2)
V_{Cmd} - commanded speed (m/sec)
V - aircraft velocity (m/sec)
dt - integration time step (sec).

The component of the helicopter's velocity in the ENU vertical direction is then computed:

$$V_v = \left(\vec{V} \bullet \vec{U}_P\right)$$

where

V_v - velocity component in ENU vertical (m/sec)
$\vec{V}$ - helicopter velocity vector (m)
$\vec{U}_P$ - Unit ECEF up vector (m) from DblECEFtoUp in Appendix B10.

The velocity component in the ENU horizontal direction is found from the vertical velocity:

$$V_H = \sqrt{V_{Mag}{}^2 - V_v{}^2}$$

where

V_H - velocity component in ENU horizontal (m/sec)
V_{mag} - magnitude of the velocity vector (m/sec)
V_v - velocity component in ENU vertical (m/sec).

Next, the solidity of the helicopter rotor disk is computed according to:

$$\sigma = \frac{(N_B \times C)}{(\pi \times R_B)}$$

where

σ - solidity = ratio of blade area to rotor disk area
N_B - number of rotor blades

C - blade chord (m)
R_B - blade radius (m).

Tip speed ratio is computed by:

$$\mu = \frac{V_H}{V_T}$$

where

μ - tip speed ratio
V_H - velocity component in ENU horizontal (m/sec)
V_T - tip speed (m/sec).

Using air density, solidity, and tip speed ratio, the drag contribution of the rotor disk is now computed:

$$D_{Prof} = \frac{1}{8} \times \sigma \times \overline{C}d_o \times \rho \times \pi \times R_B^2 \times V_T^2 \times (4.3 \times \mu^2)$$

where

D_{Prof} - profile drag of rotor disk (N)
σ - solidity of rotor disk
Cd_o - -profile drag coefficient
ρ - air density (kg/m^3)
R_B - blade radius (m)
V_T - tip speed (m/sec)
μ - tip speed ratio.

Parasitical drag is next computed using the horizontal and vertical velocities in the ENU directions:

$$D_{HP} = \frac{1}{2} \times \overline{C}d_o \times \rho \times A_F \times V_H^2$$

$$D_{VP} = \frac{1}{2} \times \overline{C}d_o \times \rho \times A_F \times V_V^2$$

where

D_{HP} - parasitical drag in ENU horizontal (N)
D_{VP} - parasitical drag in ENU vertical (N)
Cd_o - parasitic drag coefficient
ρ - air density (kg/m^3)
A_F - fuselage wet area (m^2)

V_H - velocity component in ENU horizontal (m/sec)
V_V - velocity component in ENU vertical (m/sec).

Thrust required to accelerate along the direction vector to achieve the commanded speed is then computed in the direction vector local x and z directions, where the x-axis is aligned with the direction vector and the z-axis is orthogonal to the direction vector:

$$T_H = a \times m + D_{HP} \times \sin(\theta) + D_{Prof} \times \sin(\theta) + W \times \cos(\theta) + D_{VP} \times \cos(\theta)$$

$$T_V = W \times \sin(\theta) + D_{VP} \times \sin(\theta) - D_{HP} \times \cos(\theta) - D_{Prof} \times \cos(\theta)$$

where

T_H - thrust along direction vector (N)
a - desired acceleration (m/s^2)
m - helicopter mass (kg)
D_{HP} - parasitical drag in ENU horizontal (N)
D_{Prof} - profile drag of rotor disk (N)
W - helicopter weight (N)
D_{VP} - parasitical drag in ENU vertical (N)
T_V - thrust perpendicular to direction vector (N).

T_V is the thrust which acts to balance the forces that allow the helicopter to fly along the direction vector direction of flight. If negative, it is set to 0.0. T_H offsets the forces acting along the direction vector and supplies the net force for accelerating in the desired direction. It can be negative since the helicopter thrust can be oriented forward or backwards, depending on the manipulation of the rotor disk. Using these thrust components, the total thrust required is computed:

$$T_{Req} = \sqrt{T_H^{\,2} + T_V^{\,2}}$$

where

T_{Req} - total thrust required
T_H - thrust along direction vector (N)
T_V - thrust perpendicular to direction vector (N).

The first limit on thrust is the maximum thrust producible by the main rotor disk. The maximum thrust producible is computed by:

$$T_{Max} = \frac{1}{6} \times N_B \times C \times \rho \times V_T^{\,2} \times R_B \times C_L$$

where

T_{max} - maximum producible rotor thrust (N)
N_B - number of rotor blades
C - blade chord (m)
ρ - air density (kg/m^3)
V_T - tip speed (m/sec)
R_B - blade radius (m)
C_L - blade coefficient of lift.

If the required thrust exceeds the maximum thrust, thrust is set to the maximum thrust. The thrust components are adjusted to maintain the vertical force balance and to allow the remaining thrust to act in the desired direction. Otherwise, thrust is set to the required thrust. Once thrust has been found, the induced velocity due to thrust generation is computed:

$$V_I = \sqrt{\frac{T}{2 \times \pi \times R_B^{\ 2} \times \rho}}$$

where

V_I - induced velocity (m/sec)
T - min (T_{req}, T_{max}) = thrust (N)
R_B - blade radius (m)
ρ - air density (kg/m^3).

Next, the power required to overcome drag effects is computed. Profile power is computed by:

$$P_{Prof} = \frac{1}{8} \times \sigma \times C\bar{d}_0 \times \rho \times \pi \times R_B^{\ 2} \times V_T^{\ 3} \times (1 + 4.3 \times \mu^2)$$

where

P_{Prof} - power required to overcome profile drag (W)
σ - solidity of rotor disk
Cd_0 - profile drag coefficient
ρ - air density (kg/m^3)
R_B - blade radius (m)
V_T - tip speed (m/sec)
μ - tip speed ratio.

Parasite powers required to overcome drag in the ENU horizontal and vertical directions are computed by:

$$P_{HP} = D_{HP} \times V_H$$

$$P_{VP} = D_{VP} \times V_V$$

where

P_{HP} - power required for horizontal parasite drag (W)
D_{HP} - parasitical drag in ENU horizontal (N)
V_H - velocity component in ENU horizontal (m/sec)
P_{VP} - power required for vertical parasite drag (W)
D_{VP} - parasitical drag in ENU vertical (N)
V_V - velocity component in ENU vertical (m/sec).

Using the thrust, induced velocity, and computed powers, the total power required to generate this thrust and overcome drag effects is next computed:

$$P_{Req} = T \times V_I + P_{Prof} + P_{HP} + P_{VP} + T \times V_V$$

where

P_{Req} - total power required (W)
T - thrust (N)
V_I - induced velocity (m/sec)
P_{HP} - power required for horizontal parasite drag (W)
P_{VP} - power required for vertical parasite drag (W)
V_V - velocity component in ENU vertical (m/sec).

If the power required is greater than the available power, power is set to the available power, and thrust and acceleration must both be recomputed. Thrust is computed by:

$$T = \left(\frac{P_{Avail} - P_{Prof} - P_{HP} - P_{VP}}{V_I + V_V} \right)$$

where

T - thrust (N)
P_{Avail} - total available power (W)
P_{HP} - power required for horizontal parasite drag (W)
P_{VP} - power required for vertical parasite drag (W)
V_I - induced velocity (m/sec)
V_V - velocity component in ENU vertical (m/sec).

This thrust calculation assumes that the induced velocity corresponds to the amount produced by the resultant thrust. Since this is not the case, an iterative loop is used to recompute the induced velocity for the resultant thrust. First, induced velocity for the thrust is computed by:

$$V_I = \sqrt{\frac{T}{2 \times \pi \times R_B^{\,2} \times \rho}}$$

where

V_I	-	induced velocity (m/sec)
T	-	power limited thrust (N)
R_B	-	blade radius (m)
ρ	-	air density (kg/m^3).

Thrust is then recomputed using the power available equation and this induced velocity. Repeating this procedure eventually yields the power limited thrust which corresponds to the correct induced velocity. This thrust is the total available for helicopter flight. Once total thrust has been determined, the thrust component in the desired direction is found by holding the direction vector vertical component constant to balance the vertical forces:

$$T_H = \pm\sqrt{T^{\,2} - T_V^{\,2}}$$

where

T_H	-	thrust in direction vector horizontal (N)
T	-	power limited thrust (N)
T_V	-	thrust in direction vector vertical (N).

The horizontal thrust component is set to positive if thrust is required to offset the horizontal forces to provide for the desired acceleration. It is set to negative if thrust is required to add to the forces for the desired acceleration. If T_V is greater than T, T_V is set to T and T_H is set to 0.0.

Whether the total thrust is power limited or not, if it is less than the weight of the helicopter and the desired direction of flight involves a climb, the direction vector is reset to force the helicopter to descend straight down. T_H is set to the total thrust and θ is set to π. This allows the recalculation of the desired acceleration in the correct direction.

Next, the weight of the helicopter is reduced by the amount of fuel spent in this time step:

$$W = W - SFC \times P \times Dt \times 9.80665$$

where

W	-	helicopter weight (N)
SFC	-	power specific fuel consumption (kg/sec/W)
P	-	power (W)

Dt - integration time step (sec)
9.80665 - acceleration due to gravity (m/sec^2).

Finally, the acceleration in the desired direction is recomputed to account for the possible change in the horizontal component of thrust:

$$a = \frac{[T_H - D_{HP} \times \sin(\theta\ ) - D_{Prof} \times \sin(\theta) - W \times \cos(\theta) - D_{VP} \times \cos(\theta)]}{m}$$

where

a - desired acceleration (m/sec^2)
T_H - thrust along direction vector (N)
D_{HP} - parasitical drag in ENU horizontal (N)
D_{Prof} - profile drag of rotor disk (N)
W - helicopter weight (N)
D_{VP} - parasitical drag in ENU vertical (N)
m - helicopter mass (kg).

The resultant acceleration magnitude is then applied along the direction vector when the helicopter state vectors are updated.

5.5.3 Aircraft Direction of Flight Propagation

Both airplanes and helicopters are propagated by applying their accelerations, described in Subsection 5.5.2, along their respective directions of flight. Application of acceleration depends upon whether the airplane or helicopter is in straight flight (the velocity vector is currently aligned with its direction vector) or curved flight.

5.5.3.1 Airplane Straight-Line Flight

In determining a flight path, each of the flight modes establishes this destination vector. A platform will use a straight-line flight path if the angle between the platform's current velocity vector and the destination vector is less than 0.8 degrees. For this case, the assumption is made that only slight modifications to the current flight path are required to maintain a course in the desired direction. The platform's roll is set to zero. The current magnitude of the velocity is reoriented to be along the given destination vector. The desired acceleration is also oriented along this same vector. The position and velocity of the aircraft are then updated using the current position, the reoriented velocity, and the acceleration along the destination vector:

$$\vec{x}_n = \vec{x}_0 + \dot{\vec{x}}_0\, dt + 0.5\, \ddot{\vec{x}}\, dt^{\ 2}$$

$$\vec{x}_n = \vec{\dot{x}}_0 + \vec{\ddot{x}}dt$$

where

$\vec{x}_0$ - current position
$\vec{\dot{x}}_0$ - reoriented velocity
$\vec{\ddot{x}}$ - acceleration along destination vector.

If the velocity of the aircraft is less than the user-specified minimum speed for the aircraft, the destination vector is modified before the state is updated. If the aircraft still has fuel, the aircraft is assumed to still have thrust. The destination vector is set equal to the difference between the unit vector of the destination vector and the up vector at the position, causing the aircraft to descend as it gathers speed. For the case where the aircraft has no fuel, the destination vector is oriented straight down. The modified destination vector is then used as the direction of the velocity and the acceleration to compute the new position and velocity of the aircraft.

5.5.3.2 Airplane Curved Flight

A platform will use a curved flight path if the angle between the platform's velocity vector and the destination vector is equal to or greater than 0.8 degrees and the minimum of the commanded turn G, airframe max G, and the G-Available computed from the Hi-Fi model is non-zero. If the minimum of the commanded turn G, airframe max G, and the G-Available computed from the Hi-Fi model is zero, then the aircraft is flown using straight-line flight as described in section 5.5.3.1. This flight pattern is also controlled by the given destination vector.

The lower limits of the velocity are tested to verify that the aircraft is not below minimum speed. If the velocity is below the minimum speed, the destination vector is modified to point straight down. The acceleration of gravity is applied in the direction of the destination vector, ignoring the effects of drag. The position and velocity are updated using the current velocity vector and the acceleration of gravity as:

$$\vec{x}_n = \vec{x}_0 + \vec{\dot{x}}_0\, dt + 0.5\, \vec{\ddot{x}}\, dt^{\,2}$$

$$\vec{\dot{x}}_n = \vec{\dot{x}}_0 + \vec{\ddot{x}}dt.$$

Roll is assumed to be zero for this case.

5.5.3.2.1 Maximum Allowable Turn Rate

The required radius for the turn will be computed one of two ways, based on user specification. The radius can be computed using the maximum allowable turn

rate, based on the aircraft's speed, or it can be limited by a user-specified bank angle. If hi-fidelity airplane flight is specified, both turn-G (G_T) and max-G (G_{max}) in the following section is limited by the airplane's current G-Available.

When the aircraft is configured to turn at a specific bank angle, the turn G is determined from the bank angle input.

$$G_T = \tan(\phi_{bank})$$

where

ϕ_{bank}	-	Bank angle (deg)
G_T	-	Max Turn G

If the aircraft's speed is currently greater than the corner speed, the turn radius is computed as:

$$R_T = (V^2)/G_T$$

where

R_T	-	turn radius (m)
V	-	aircraft velocity (m/s)
G_T	-	turn acceleration = g-force x 9.80665 (m/sec^2).

If the current speed is less than the corner speed, then the aircraft's turn rate is computed from the quadratic curve fit to velocity described in Subsection 8.8. If the turn rate is positive, then the radius of the turn is then computed as magnitude of the velocity divided by the turn rate. If the turn rate is negative, the aircraft's turn rate is set to .01 and the turn radius will be set to the velocity divided by 0.01. The user is warned of the bad turn parameters in the log file for flight processing.

5.5.3.2.2 Turn Center

A platform will use a curved flight path by calculating the Turn Center if the platform is between a Start and End Turn waypoint. The turn radius specified on the Start Turn will be used until the aircraft reaches the End Turn waypoint. Turns are performed such that the radius of the resulting ground track is constant throughout the turn though altitude may be changed during the turn due to terrain following or to Route waypoints that are defined along the turn.

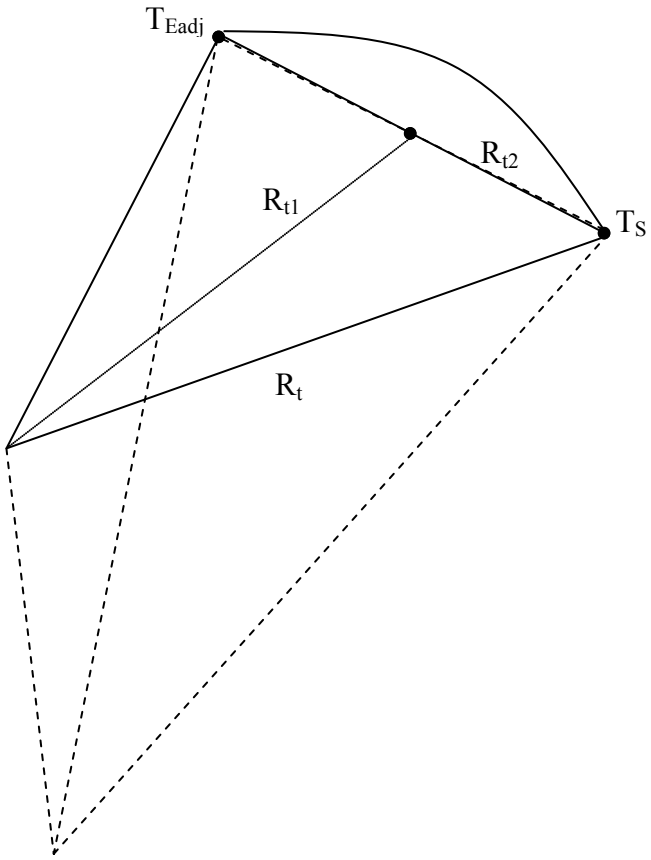

Figure 5.1-1 Turn Center plane in reference to ground

An adjusted End Turn vector is created from the End Turn unit vector and the magnitude of the Start Turn vector allowing any change in altitude between the End Turn and Start Turn vectors to be ignored when computing the turn center.

A local coordinate frame is defined with the turn plane aligned with the x-y plane. The X axis is the unit vector consisting of the difference between the adjusted End Turn and Start Turn vectors. The Y axis is defined as the cross product of the Start Turn and the adjusted End Turn vectors. The X axis is the cross product of the Y axis vector and the Z axis vector making this a right-handed coordinate system. A transformation matrix is computed to transform from this local frame to ECEF.

$$\vec{X} = \frac{\vec{T}_{Eadj} - \vec{T}_S}{\left|\vec{T}_{Eadj} - \vec{T}_S\right|}, \vec{Y} = \frac{\vec{T}_S x \vec{T}_{Eadj}}{\left|\vec{T}_S x \vec{T}_{Eadj}\right|}, \vec{Z} = \vec{X} x \vec{Y}$$

where

$\vec{T}_S$ - Start Turn location

$\vec{T}_{Eadj}$ - Adjusted End Turn location

Rt2 is then calculated as half the distance between Ts and $T_{Eadj.}$

$$R_{t2} = \frac{\left|\vec{T}_{Eadj} - \vec{T}_S\right|}{2}$$

where

$\vec{T}_S$ - Start Turn location

$\vec{T}_{Eadj}$ - Adjusted End Turn location

The Turn Center location is started at the halfway point between the adjusted End Turn and Start Turn vectors and 0,0, for the x and y location.

$$\vec{T}_C = [R_{t2}, 0, 0]$$

where

$\vec{T}_C$ - Turn Center

R_{t2} - halfway point between the adjusted End Turn and Start Turn vectors

R_{t1} can then be calculated by using the Pythagorean theorem

$$R_{t1} = \sqrt{R_t^2 - R_{t2}^{\ 2}}$$

where

R_{t2} - halfway point between the adjusted End Turn and Start Turn vectors

R_t - the specified turn radius

If the Turn Center will be applied to the left of the velocity vector

$$T_C = T_C + [0, -R_{t1}, 0]$$

If the Turn Center will be applied to the right of the velocity vector

$$T_C = T_C + [0, R_{t1}, 0]$$

T_C is then rotated back to ECEF and adjusted by the location of the Start Turn.

5.5.3.2.3 Turn Angle

The turn angle is determined by the following equation:

$$\theta_T = (V/R_T) \times dt$$

where

θ_T	-	turn angle (rads)
V	-	aircraft velocity (m/sec)
R_T	-	turn radius (m)
dt	-	internal FP time step (sec).

5.5.3.2.4 Straight-Line Flight

There are some cases where the angle between the platform's velocity vector and the destination vector is greater than or equal to 0.8 degrees that require straight flight. If the turn angle is greater than the angle between the destination vector and the velocity vector, then using curved flight would cause an overshoot. If the minimum of the commanded turn G, airframe max G, and the G-Available computed from the Hi-Fi model is zero then the radial acceleration is zero and therefore straight flight is required. In both of these cases, the aircraft is flown using straight-line flight as described in Subsection 5.5.3.1 and the Roll is set to zero.

5.5.3.2.5 Turn Plane Flattening

There are some cases where the turning plane is oriented at an exaggerated angle, causing straight up turns or turns into the ground. If not flying a maneuver and the angle between the destination vector and the velocity vector is less than the angle between the velocity vector and the up vector and less than the angle between the destination vector and the up vector then the turning plane tends to be oriented at an exaggerated angle, causing straight up turns or turns into the ground. In this case the destination vector is adjusted to be perpendicular to the velocity vector and in the local horizontal plane in order to make the turn flatter. The destination vector is only adjusted for the current integration time step.

5.5.3.2.6 Max Climb Angle

If the aircraft velocity is less than 2.5 times the minimum speed defined for the airframe and the current weight of the aircraft is greater than the empty weight plus the total weapon weight then the climb angle of the aircraft is limited by its thrust to weight ratio.

$$\text{Ratio} = \frac{T_{Max90}}{W}$$

where

Ratio - Thrust to weight Ratio (none)
T_{Max90} - 90% of max thrust defined for the airframe (N)
W - Current total weight of the aircraft including fuel and weapons (N)

The maximum climb angle can be computed from the thrust to weight ratio.

$$\theta_{max} = 90 - \cos^{-1}(\text{Ratio})$$

where

θ_{max} - Maximum climb angle relative to local horizontal plane (deg)
Ratio - Thrust to weight Ratio (none)

If the angle of climb required by the current destination vector is greater than the maximum climb angle, then the destination vector is adjusted so that the aircraft climbs at the maximum climb angle.

5.5.3.2.7 Aircraft Location

The turn will take place in a plane formed by the adjusted destination vector and the current velocity vector known as the turn plane. A local coordinate frame is defined with the turn plane aligned with the x-y plane.

For flights utilizing the Turn Center as described in section 5.105.10, the X axis is along the current position and the Turn Center position. The Y axis is defined as the cross product of the current location and the Turn Center vector. The Z axis is the cross product of the X axis vector and the Y axis vector making this a right-handed coordinate system.

For flights not utilizing the Turn Center, the Y axis is along the velocity vector. The Z axis is defined as the cross product of the velocity and the destination vector. The X axis is the cross product of the Y axis vector and the Z axis vector making this a right-handed coordinate system. The radius vector lies along the X axis. A transformation matrix is computed to transform from this local frame to ECEF. ***Figure 5.5-2*** shows the parameters used to compute the position and velocity in local coordinates.

If flying on a BPI route and at a Start Pattern waypoint, a right turn is always used for the aircraft as it turns through the turn angle. See MM Subsection 5.5.1.10 for an explanation on the use of BPI routes in EADSIM.

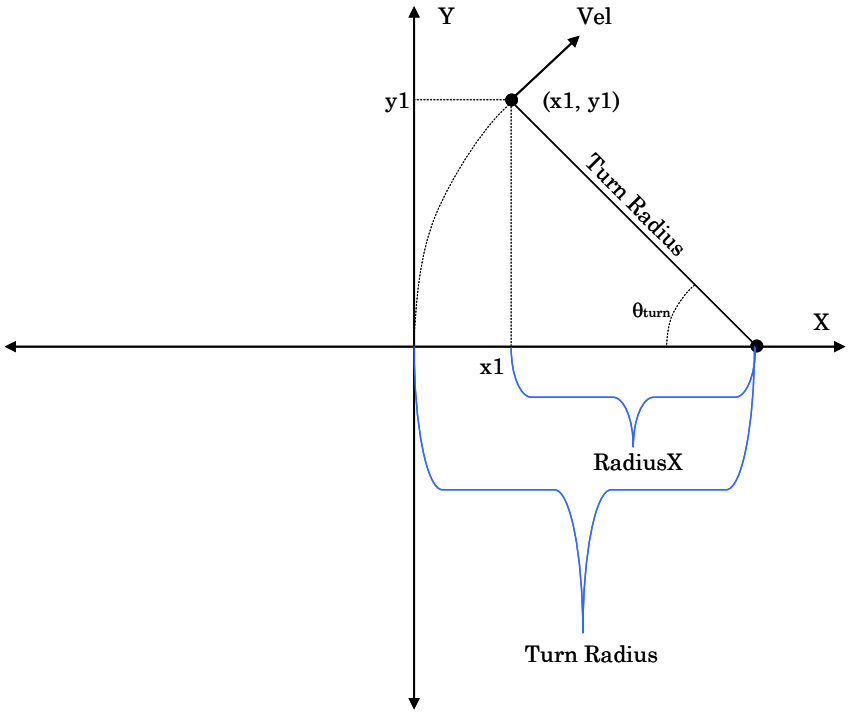

Figure 5.5-2 Right turn in local coordinate frame

The delta position vector is computed as follows.

$$R_X = R_{turn} \cdot \cos(\theta_{turn})$$

where

R_X - Distance from point (x1, 0) to the point (TurnRadius, 0) (m)
R_{turn} - Turn radius (m)
θ_{turn} - Turn angle (deg)

Then the local x coordinate of the new position is computed.

$$x1 = R_{turn} - R_X$$

where

x1 - Local X coordinate (m)
R_X - Distance from point (x1, 0) to the point (TurnRadius, 0) (m)
R_{turn} - Turn radius (m)

For flights not utilizing the Turn Center, there is zero change of position and velocity along the Z axis because the X-Y plane is defined as the turn plane, therefore Z is zero in local coordinates. The velocity vector is initialized with unit magnitude in the direction of the normal to the turn radius vector in the X-Y plane, relative to the new position.

For flights utilizing the Turn Center, there is a vertical velocity component which is applied to the Z axis of the position and velocity vectors. The vertical velocity is calculated by transforming the velocity vector into local coordinates with the transformation matrix, then transformed into a unit vector. The vector is then multiplied by the velocity magnitude. The z component is multiplied by Dt to get the z axis of the of the aircraft position in local coordinates. The roll is indicated by the bank angle specified on the Start Turn waypoint and is used for the duration of the turn.

The position delta vector and the velocity vector are then transformed into ECEF coordinates by calling the DblMatrixMultiply routine described in Subsection B10.4.18 with the transformation matrix. The new position is then computed using the sum of the previous position vector and the delta position vector. The magnitude of the new velocity is adjusted based on the acceleration, even though a constant velocity turn is assumed for the position calculation. The aircraft acceleration is computed as described in Subsection 5.5.2. If the acceleration cannot be computed, then the aircraft falls according to a ballistic trajectory with zero roll. The computed acceleration is used to compute the velocity.

$$\vec{V} = \vec{V}_{prev} + \vec{a} \cdot \Delta t$$

where

$\vec{V}$	-	Computed Velocity (m/s)
$\vec{V}_{prev}$	-	Previous velocity (m/s)
$\vec{a}$	-	Acceleration (m/s^2)
Δt	-	Integration time step (s)

5.5.3.2.8 Roll

Roll is defined as the angle about the lateral axis of the aircraft with zero roll defined as having the aircraft wings level in the lateral direction with the top of the aircraft pointing away from the ground. The aircraft may be pitched up or down, therefore the wings may or may not be level in the longitudinal direction. The roll is computed as the sum of the turn plane angle and the bank angle. First, a vector perpendicular to the velocity and position vectors is computed.

$$\vec{L} = \vec{V}_{ECEF} \times \vec{P}_{ECEF}$$

where

$\vec{L}$ - Vector perpendicular to the velocity and position vectors (m)
$\vec{V}_{ECEF}$ - ECEF Velocity vector (m)
$\vec{P}_{ECEF}$ - ECEF Position vector (m)

Then a vector perpendicular to the longitudinal axis of the aircraft and to the vector that is level in the lateral direction is computed for the left turn case.

$$\vec{T} = \vec{L} \times \vec{V}_{ECEF}$$

where

$\vec{T}$ - Vector that points out of the top of the aircraft normal to the wings when Roll is zero (m)
$\vec{L}$ - Vector perpendicular to the velocity and position vectors (m)
$\vec{V}_{ECEF}$ - ECEF Velocity vector (m)

The negative of the T vector is used for the right turn case. Next, the angle between the zero roll plane and the turn plane is computed.

$$\theta_{plane} = \cos^{-1}\left(\frac{\hat{z} \bullet \vec{T}}{|\hat{z}| \cdot |\vec{T}|}\right)$$

where

θ_{plane} - Angle between the zero roll plane and the turn plane (deg)
$\hat{z}$ - ECEF unit vector in direction of z axis of local coordinate frame
$\vec{T}$ - Vector which points out of the top of the aircraft normal to the wings when Roll is zero (m)

The bank angle is computed from the turn G.

$$\phi_{bank} = \tan^{-1}\left(G_{MaxTurn}\right)$$

where

ϕ_{bank} - Bank angle (deg)
$G_{MaxTurn}$ - Max Turn G

Finally, the roll is computed as the sum of the turn plane angle and the bank angle. For right turns the roll is set to a positive value, and for left turns the roll is set to a negative value.

$$\phi_{roll} = \theta_{plane} + \phi_{bank}$$

where

ϕ_{roll} - Magnitude of roll angle (deg)

θ_{plane} - Angle between the zero roll plane and the turn plane (deg)

ϕ_{bank} - Bank angle (deg)

If the angle between the zero roll plane and the turn plane is within 0.99 degrees of 90 degrees then the roll is set to zero.

5.5.3.3 Helicopter Straight-Line Flight

Like airplanes, the helicopter flight modes establish the direction that the helicopter needs to move. Straight-line flight will be used if the angle between the helicopter's current velocity vector and the destination vector is less than 0.8 degrees. For this case, the assumption is made that only slight modifications to the current flight path are required to maintain a course in the desired direction.

First, the acceleration required to achieve the commanded speed is computed as described in Subsection 5.5.2.2. Next, if the helicopter is out of fuel or flying slower than its minimum speed, the acceleration magnitude is set to acceleration due to gravity and the direction vector is adjusted so the platform will fall to the ground:

$$D_x = -UP_x$$

$$D_y = -UP_y$$

$$D_z = -UP_z$$

where

D_n - helicopter ECEF direction of flight in nth direction (m)
P_n - helicopter ECEF up vector in nth direction (m).

Next, the acceleration is applied along the direction vector to form the acceleration vector:

$$A_x = U_{D_x} \times a$$

$$A_y = U_{D_y} \times a$$

$$A_z = U_{D_z} \times a$$

where

A_n - helicopter ECEF acceleration in nth direction (m/sec^2)
U_{Dn} - unit helicopter ECEF direction of flight in nth direction (m)
a - desired acceleration magnitude (m/sec^2).

Finally, the acceleration vector is applied to the standard equations of motion for position and velocity to propagate the helicopter forward:

$$P_x = P_x + V_x \times D_t + \frac{1}{2} \times A_x \times D_t^2$$

$$P_y = P_y + V_y \times D_t + \frac{1}{2} \times A_y \times D_t^2$$

$$P_z = P_z + V_z \times D_t + \frac{1}{2} \times A_z \times D_t^2$$

and

$$V_x = V_x + A_x \times Dt$$

$$V_y = V_y + A_y \times Dt$$

$$V_z = V_z + A_z \times Dt$$

where

P_n - helicopter ECEF position in nth direction (m)
V_n - helicopter ECEF velocity in nth direction (m/sec)
D_t - integration time step (sec)
A_n - helicopter ECEF acceleration in nth direction (m/sec^2).

Roll is set to zero during straight-line flight.

5.5.3.4 Helicopter Curved Flight

Helicopters whose direction vectors are more than 0.8 degrees off the current velocity vector are flown through a coordinated turn to achieve the new heading. Once the helicopter is within 0.8 degrees, the straight-line flight algorithm explained in Subsection 5.5.3.3 takes over.

The turns are performed by computing the helicopter position through the turn, given the commanded G and the helicopter's current velocity. First, the acceleration required to achieve the commanded speed is computed as described in Subsection 5.5.2.2. Next, if the helicopter is out of fuel or flying slower than its

minimum speed, the acceleration magnitude is set to acceleration due to gravity and the direction vector is adjusted so the platform will fall to the ground:

$$D_x = -UP_x$$

$$D_y = -UP_y$$

$$D_z = -UP_z$$

where

D_n - helicopter ECEF direction of flight in nth direction (m)
P_n - helicopter ECEF up vector in nth direction (m).

Using this new direction vector and acceleration, the helicopter is propagated toward the ground using standard position and velocity equations of motion:

$$P_x = P_x + V_x \times D_t + \frac{1}{2} \times A_x \times D_t^2$$

$$P_y = P_y + V_y \times D_t + \frac{1}{2} \times A_y \times D_t^2$$

$$P_z = P_z + V_z \times D_t + \frac{1}{2} \times A_z \times D_t^2$$

and

$$V_x = V_x + A_x \times Dt$$

$$V_y = V_y + A_y \times Dt$$

$$V_z = V_z + A_z \times Dt$$

where

P_n - helicopter ECEF position in nth direction (m)
V_n - helicopter ECEF velocity in nth direction (m/sec)
D_t - integration time step (sec)
A_n - helicopter ECEF acceleration in nth direction (m/sec^2).

Roll is set to 0.0 for this case.

If the helicopter is not fuel limited and can fly to its new heading, turn radius is first computed:

$$R_t = \frac{V_{mag}^2}{G \times 9.80665}$$

where

R_t - helicopter turn radius (m)
V_{mag} - helicopter velocity magnitude (m/sec)
G - commanded turn G
9.80665 - acceleration due to gravity (m/sec^2).

Next, how much turn the helicopter can achieve in one time-step is computed:

$$\theta_{Obt} = \frac{V_{mag} \times D_t}{R_t}$$

where

θ_{Obt} - obtainable turn angle (rads)
V_{mag} - helicopter velocity magnitude (m/sec)
D_t - integration time step (sec)
R_t - helicopter turn radius (m).

The desired magnitude of the turn angle is then computed:

$$\theta_{Des} = U_{Vel} \bullet U_{Dir}$$

where

θ_{Des} - desired turn angle (rads)
U_{Vel} - unit ECEF velocity vector
U_{Dir} - unit ECEF direction of flight vector.

If the desired turn angle is greater than the obtainable turn angle, the helicopter will be limited to the obtainable turn angle for the single time step. Next, the local coordinate frame is computed within which the helicopter turn will be performed. The local x-axis is aligned along the direction of the ECEF velocity vector but in the local horizontal. The vector perpendicular to both the velocity and up vectors is first computed by crossing the velocity and position vectors:

$$UA = U_{Vel} \times U_{Pos}$$

where

UA - unit perpendicular vector
U_{Vel} - unit ECEF velocity vector
U_{Pos} - unit ECEF up vector at the platform position.

Then, the x-axis is computed as the cross product of the position vector and this new vector:

$$X_{Local} = U_{Pos} \times UA$$

where

X_{local} - unit local x-axis
U_{Pos} - unit ECEF up vector at the platform position
UA - unit perpendicular vector.

The local z-axis is the local up vector, so:

$$Z_{Local} = U_{Pos}$$

where

Z_{Local} - unit local z-axis
U_{Pos} - unit ECEF up vector at the platform position.

Finally, the local y-axis is orthogonal to the x- and z-axes. Thus:

$$Y_{Local} = Z_{Local} \times X_{Local}$$

where

Y_{Local} - unit local y-axis
Z_{Local} - unit local z-axis
X_{Local} - unit local z-axis.

Next, the distances moved along the local x- and y-axes are computed:

$$\Delta X_{Range} = R_t \times \sin(\theta_t)$$

$$\Delta Y_{Range} = R_t - R_t \times \cos(\theta_t)$$

where

ΔX_{Range} - local x-axis turn displacement (m)
R_t - turn radius (m)
θ_t - turn angle (rads)
ΔY_{Range} - local y-axis turn displacement (m).

Likewise, the velocity directional changes along the local x- and y-axes are computed within the turn:

$$\Delta X_{Vel} = \cos(\theta_t)$$

$$\Delta Y_{Vel} = \sin(\theta_t)$$

where

ΔX_{Vel} - local x-axis turn velocity change (m)
θ_t - turn angle (rads)
ΔY_{Vel} - local y-axis turn velocity change (m).

If the local y-axis opposes the desired direction vector or if flying on a BPI route and at a Start Pattern waypoint, a right turn is desired. Thus, the negatives of these displacements are used in the computations for the turn. See Subsection 5.5.1.10 for an explanation on the use of BPI routes in EADSIM.

Once computed, these displacement magnitudes in position and velocity are applied to the local x- and y-axes to form the position and velocity displacement vectors, which are then used to compute the new helicopter position and velocity at the end of the integration time step. The new position is computed by:

$$P_x = P_x + P_{X_x} + P_{Y_x}$$

$$P_y = P_y + P_{X_y} + P_{Y_y}$$

$$P_z = P_z + P_{X_z} + P_{Y_z}$$

where

P_n - helicopter position in ECEF nth direction (m)
ΔP_{X_n} - local x positional change in ECEF nth direction (m)
ΔP_{Y_n} - local y positional change in ECEF nth direction (m).

Likewise, the new velocity is computed by applying the current velocity magnitude in the directions established by the local x and y velocity directions:

$$V_x = V_{Mag} \times \Delta V_{X_x} + V_{Mag} \times \Delta V_{Y_x}$$

$$V_y = V_{Mag} \times \Delta V_{X_y} + V_{Mag} \times \Delta V_{Y_y}$$

$$V_z = V_{Mag} \times \Delta V_{X_z} + V_{Mag} \times \Delta V_{Y_z}$$

where

V_n - helicopter velocity in ECEF nth direction (m/sec)
V_{Mag} - velocity magnitude (m/sec)
ΔV_{X_n} - local x velocity in ECEF nth direction
ΔV_{Y_n} - local y velocity in ECEF nth direction (m).

The velocity magnitude used accounts for any acceleration or deceleration performed by the helicopter within this integration step.

Finally, the roll of the helicopter is computed. Roll is computed assuming the helicopter performs a coordinated turn, in which thrust balances the weight of the helicopter in the vertical direction and balances momentum forces in the horizontal direction. Thus, the force-balancing equations are:

$$W = T \times \cos(\Psi)$$

$$W \times \frac{V_{Mag}^{\ 2}}{R_t \times 9.81} = T \times \sin(\Psi)$$

where

W	-	helicopter weight (N)
T	-	thrust (N)
Ψ	-	roll angle (rads)
V_{Mag}	-	velocity magnitude (m/sec)
R_t	-	turn radius (m)
9.81	-	acceleration due to gravity (m/sec^2).

Combining these two equations and solving for roll angle yields:

$$\Psi = \tan^{-1}\left(\frac{V_{Mag}^{\ 2}}{R_t \times 9.81}\right)$$

where

Ψ	-	roll angle (rads)
V_{Mag}	-	velocity magnitude (m/sec)
R_t	-	turn radius (m)
9.81	-	acceleration due to gravity (m/sec^2).

This is the roll angle for the helicopter. If the helicopter is turning to the left, the negative of this value is used.

5.5.4 Aircraft Altitude Monitoring/Terrain Following

During all modes of aircraft flight, direction vector adjustments are made as required to avoid the aircraft's exceeding altitude limits or plowing into terrain. These adjustments are made using an over-damped feedback control loop on altitude rate.

Current aircraft altitude is first computed using DblECEFtoAlt. DblECEFtoUp is used to get the local vertical, i.e. the unit vector in the up direction.

The vector perpendicular to both the local up and direction vectors is next computed:

$$\vec{B} = \left| \vec{D} \times \vec{U} \right|$$

where

$\vec{B}$ - vector perpendicular to $\vec{D}$ and $\vec{U}$ (m)
$\vec{D}$ - aircraft current direction of flight vector (m)
$\vec{U}$ - aircraft local up vector (m).

Then, the vector planar to and in the direction of the current direction vector but in the aircraft's local horizontal plane is computed:

$$\vec{H} = \left| \vec{U} \times \vec{B} \right|$$

where

$\vec{H}$ - vector in a/c horizontal plane in direction of $\vec{D}$ (m)
$\vec{U}$ - aircraft local up vector (m).
$\vec{B}$ - vector perpendicular to $\vec{D}$ and $\vec{U}$ (m).

Next, the terrain altitude in the horizontal direction of the intended flight path is found. The position of the aircraft is found at varying times in the future along this horizontal:

$$\vec{P}_{Ahead} = \vec{P} + \vec{U}_H \times V_{Mag} \times D_t$$

where

$\vec{P}_{Ahead}$ - a/c position along the horizontal D_t sec in the future (m)
$\vec{P}$ - aircraft position vector (m)
$\vec{U}_H$ - unit horizontal vector in direction of current direction vector
V_{Mag} - magnitude of a/c velocity vector (m/sec)
D_t - time ahead to find a/c location (sec).

Using this equation, the a/c future position is found according to the user-specified, terrain-following control parameters input on the aircraft element window. The position of the aircraft after the initial DeltaT is found first. For subsequent positions, the look-ahead interval geometrically increases from the initial DeltaT value by multiplying each successive interval by the multiplicative factor. For each of these computed positions, the corresponding terrain altitude is found by accessing the terrain map database. For the special case where the desired altitude is measured in meters MSL, the terrain altitude is set to 0.0. This

terrain sampling is repeated until the specified number of terrain samples have been collected. The default settings for these control parameters allow for eight terrain samples to be taken, starting from 0.5 sec and increasing by the factor of 1.5. These default settings result in terrain samples for times ranging from 0.5 to 8.5 sec ahead of the current position of the aircraft. Once the terrain altitudes for the future positions are found, the highest terrain altitude is used to set the desired floor altitude used when the aircraft is descending and when it is terrain following.

Once the desired aircraft altitude has been found, the actual altitude rate of the aircraft is determined by finding the component of the aircraft velocity vector in the local vertical direction:

$$H'_{a/c} = \left| \vec{V} \bullet \vec{U} \right|$$

where

$H'_{a/c}$ - actual a/c altitude rate (m/sec)
$\vec{V}$ - a/c velocity vector (m/sec)
$\vec{U}$ - aircraft local up vector (m).

Next, the altitude difference between the current position and the desired altitude is found:

$$\Delta H = H_{Des} - H$$

where

ΔH - altitude diff between current and desired (m)
H_{Des} - desired altitude (m)
H - current a/c altitude (m).

The desired altitude rate is then computed and adjusted to overdamp the feedback control loop:

$$V_V = \frac{\Delta H}{Gain \times Dt_{Step}}$$

where

V_V - desired altitude rate (m/sec)
ΔH - altitude diff between current and desired (m)
Gain - adjustment for overdamping feedback loop
Dt_{Step} - integration step size = 0.5 (sec).

The gain in the previous equation is one of the terrain-following control parameters input on the aircraft element window. Its default setting is 15.

If the actual altitude rate is greater than the desired or if terrain following, the direction vector needs to be recalculated to avoid exceeding the altitude limit or flying into the ground. First, the a/c speed in the horizontal plane is computed assuming the speed in the vertical plane is the desired altitude rate:

$$V_H = \sqrt{V_{Mag}{}^2 - V_V{}^2}$$

where

V_H	-	aircraft velocity in horizontal direction (m/sec)
V_{Mag}	-	magnitude of a/c velocity vector (m/sec)
V_V	-	aircraft desired altitude rate (m/sec).

If the desired altitude rate is greater than the current a/c velocity, the V_H is set to 0.0 and the desired altitude rate is set to V_{Mag}. During terrain following, a limit is applied on climb angle to prevent the aircraft from changing altitude too quickly. This maximum climb angle limit is one of the terrain-following control parameters input on the aircraft element window. Its default setting is 20 deg. The climb angle from the computed direction velocities is determined by:

$$\phi = \tan^{-1}\left(\frac{V_V}{V_H}\right)$$

where

ϕ	-	climb angle (deg)
V_V	-	aircraft desired altitude rate (m/sec)
V_H	-	aircraft velocity in horizontal direction (m/sec).

If the climb angle is greater than the maximum climb angle, the horizontal velocity is recomputed for that maximum climb angle:

$$V_H = \sqrt{\frac{V_{Mag}{}^2}{1.0 + \tan(\theta_{Max})}}$$

where

V_H	-	aircraft velocity in horizontal direction (m/sec)
V_{Mag}	-	magnitude of a/c velocity vector (m/sec)
θ_{Max}	-	maximum climb angle (rads).

The corresponding altitude rate is then computed:

$$V_V = V_H \times \tan(\theta_{Max})$$

where

V_V - aircraft desired altitude rate (m/sec)
V_H - aircraft velocity in horizontal direction (m/sec)
θ_{Max} - maximum climb angle (rads).

Once the horizontal and vertical velocities have been computed, a new direction vector is computed by applying the velocities in the appropriate directions:

$$\vec{D} = V_H \times \vec{U}_H + V_V \times \vec{U}$$

where

$\vec{D}$ - new direction vector (m)
V_H - aircraft velocity in horizontal direction (m/sec)
$\vec{U}_H$ - unit horizontal vector in direction of current direction vector
V_V - aircraft desired altitude rate (m/sec)
$\vec{U}$ - aircraft local up vector (m).

5.5.4.1 Altitude Monitoring Settings

When an aircraft is not terrain following, direction vectors are adjusted if required to keep the aircraft from flying below the floor altitude or above the ceiling altitude. All flight modes use the airframe's service altitude limit as the ceiling altitude and a default floor altitude of 200 m except Waypoint and Wingman modes. For these modes, the ceiling altitude is still limited to the service altitude, but the floor altitude conditionally depends on the commanded altitude. For the Waypoint mode, the floor altitude is set as the minimum of the commanded altitude and 200 m. Similarly for the Wingman mode, the floor altitude is set according to this same minimum condition if the flight leader is flying waypoints. Floor altitudes for all modes are Above Ground Level (AGL). Ceiling altitudes for all modes are MSL.

To perform the altitude monitoring and direction vector adjustments described in Subsection 5.5.4, the desired altitude needs to be set. If the aircraft is not terrain following and its actual altitude rate is negative (aircraft is descending) or if its actual altitude is less than the floor altitude, the desired altitude is set to the floor altitude. For all other cases except terrain following, the desired altitude is set to the ceiling altitude.

5.5.4.2 Terrain-Following Settings

When an aircraft is scripted to fly terrain following, direction vector adjustments are made to keep the aircraft flying at the commanded altitude above terrain. In terrain following, only floor altitude is important. For this case, it is set

to the commanded altitude input by the user when deploying the platform. To perform the terrain following using the direction vector adjustments described in Subsection 5.5.4, the desired altitude is automatically set to the floor altitude.

5.5.5 Aircraft Constant Altitude Rate Flight

When an aircraft is being flown in the Waypoint, Engage, or RTB flight modes, the computed direction vector for each of these modes is modified to account for Earth curvature. Each of these modes computes the direction vector as the vector difference between the desired position and the current position vectors, resulting in a direction vector that potentially points through the Earth, directly at the desired position. The direction vector needs to point toward the desired position but along the Earth's surface. Additionally, when changing altitudes, aircraft do not actually fly in a straight line but in a more parabolic trajectory along the Earth's curvature. Both these phenomena are accounted for by the constant altitude rate flight.

To recompute the direction vector, the altitude difference between the current and desired positions is calculated by computing the altitude of the current position and the destination position utilizing DblECEFtoAlt. ΔH is then obtained by subtracting the current altitude from the destination altitude.

Next, the angle between the aircraft current position vector and the desired position vector is found via the Law of Cosines:

$$\varphi = \cos^{-1}\left(\frac{DP_{Mag}^{\ 2} + P_{Mag}^{\ 2} - D_{Mag}^{\ 2}}{2 \times DP_{Mag} \times P_{Mag}}\right)$$

where

φ	-	angle between a/c and destination vectors (rads)
DP_{Mag}	-	magnitude of destination position vector (m)
P_{Mag}	-	magnitude of current position vector (m)
D_{Mag}	-	magnitude of original direction vector (m).

The time to reach the destination point at the current commanded speed is computed by:

$$T_{TG} = \sqrt{\frac{\left(P_{Mag}^{\ 2} \times \varphi^{\ 2}\right) + \Delta H^{\ 2}\right)}{V_{Cmd}}}$$

where

T_{TG}	-	time to reach destination points (sec)
P_{Mag}	-	magnitude of current position vector (m)

φ - angle between a/c and destination position vectors (rads)
ΔH - positional altitude difference (m)
V_{Cmd} - commanded speed (m/sec).

Using this time, the altitude rate can now be computed by:

$$V_V = \left(\frac{\Delta H}{T_{TG}} \right)$$

where

V_V - altitude rate (m/sec)
ΔH - positional altitude difference (m)
T_{TG} - time to reach destination point (sec).

Given the commanded speed and the altitude rate result, speed along the surface of the Earth can be found by:

$$V_H = \sqrt{V_{Cmd}{}^2 - V_V{}^2}$$

where

V_H - speed along Earth's surface (m/sec)
V_{Cmd} - commanded speed (m/sec)
V_V - altitude rate (m/sec).

Next, the vector perpendicular to the surface of the Earth at the current position of the aircraft is found by first taking the cross product of the direction vector and the aircraft current position local up vector:

$$\vec{B} = \left| \vec{D} \times \vec{U} \right|$$

where

$\vec{B}$ - cross product of direction and position vectors (m)
$\vec{D}$ - original direction vector (m)
$\vec{U}$ - aircraft current position local up vector (m).

Then, the cross product of the aircraft current position local up vector and the resulting vector is found:

$$\vec{E}_H = \left| \vec{U} \times \vec{B} \right|$$

where

$\vec{E}_H$ - horizontal vector (m)

$\vec{U}$ - aircraft current position local up vector (m)

$\vec{B}$ - cross product of direction and position vectors (m).

The resulting vector is perpendicular to the ground plane at the aircraft current position. Thus, this is the vector in the horizontal direction. Next, the unit vectors of the horizontal direction vector is found:

$$U\vec{E}_H = \frac{\left(E_{H_x}\,\hat{i} + E_{H_y}\,\hat{j} + E_{H_z}\,\hat{k}\right)}{E_{H_{Mag}}}$$

where

$U\vec{E}_H$ - unit horizontal vector

E_{H_N} - component of horizontal vector in nth direction (m)

$E_{H_{Mag}}$ - magnitude of horizontal vector (m)

Since both the horizontal and vertical direction unit vectors are known, the speed in each direction can now be applied:

$$V_{ECI_N} = V_H \times UE_{H_N} + V_V \times U$$

where

V_{ECI_N} - aircraft velocity vector in ECEF (m/sec)

V_H - speed along the Earth's surface (m/sec)

UE_{H_N} - component of horizontal unit vector in nth direction

V_V - altitude rate (m/sec)

U - component of vertical unit vector in nth direction.

Finally, the new direction vector can be found by applying the integration step to the final speed vector:

$$D_N = V_{ECI_N} \times dt$$

where

D_N - new direction vector (m)

V_{ECI_N} - component of speed vector in nth direction (m/sec)

dt - integration step (sec).

If this new direction vector does not cause the aircraft to fly above the ceiling altitude or below the floor altitude, it is used to integrate the aircraft position over the time integration time step. If this vector does cause the aircraft to exceed the

altitude limits, the direction vector is adjusted, as described in Subsection 5.5.4, for altitude monitoring.

5.6 MODES OF PLATFORM MOVEMENT

5.6.1 Waypoint Mode

The waypoints of a given aircraft provide the nominal flight path for the platform. The aircraft will try to follow these user-defined points, based on its own perception, unless interrupted by the C3I battle management functions. A waypoint consists of the altitude, speed, on time, off time, terrain-following information, waypoint type, and the waypoint coordinate in either latitude and longitude or Military Grid Reference System (MGRS).

The Waypoint mode is defined by the user through Scenario Generation for both ground and airborne platforms. Airborne platforms transition to other modes based on decisions made by the C3I model. The aircraft is returned to the Waypoint mode when C3I commands it to resume normal operations. The FP model changes wingmen to Waypoint mode when their flight leader reaches its last waypoint. This allows the wingmen to fly to the last waypoint. The Waypoint mode can also be entered by a scripted flight that has finished its Scramble action.

In the Waypoint mode, platforms try to move from one user-defined waypoint to another, based on their own perception, in the order and at the velocity defined by the user. When the last waypoint is perceived to be reached, the platform is deactivated and removed from the scenario. The actual movement from waypoint to waypoint is different for ground and air platforms. Subsection 5.7 discusses this mode for ground platforms. This section is limited to aircraft movement.

Two modes are available for air platform movement between waypoints. The first mode utilizes representative aerodynamics to integrate the aircraft position and velocity over the scenario update interval. Thrust, gravity, lift, and drag are all accounted for in the calculation of accelerations required to achieve the desired speed. The second mode utilizes nonaerodynamic integration of aircraft position and velocity. Both of these modes are described below.

If the aircraft has a navigation element and the Accumulate Error Only option is not selected, the Waypoint flight mode will navigate using perception and will update perception according to the equations in Sections 5.4.1.3 and 5.4.1.4. Otherwise this flight mode navigates according to truth, and perception will be set and updated with the same values as the truth. Note that this means that in the following discussion, whenever it is stated that a perceived value is used, this will correspond with the truth value if the aircraft is navigating according to truth.

It is first important to understand what is meant in saying that the aircraft flies according to its own perception. The aircraft has a certain perception of its own position, velocity, and orientation. This perception can, in some cases, lead an aircraft to fail, by inaccurate navigation, to reach its true waypoint position, even though it knows that position. The simplest example would be an aircraft that has the correct perception of its own velocity and orientation, but the wrong perception of its position. Due to its mistaken perception of its own position, the aircraft will fly to a position offset from the waypoint by just as much as the aircraft's own perception of its position is offset from its true position.

5.6.1.1 Aerodynamic Waypoint Flight

If the nonaerodynamic waypoint flight has not been selected for the aircraft element that describes the flight leader or if that flight leader is returning from an engagement action and has not yet reached the next waypoint in the list, representative aerodynamics are used.

When flying an aircraft to its destination waypoint, total aircraft integration time D_T is initialized to the time of the aircraft's last scenario update. Then, the commanded speed is initialized to the minimum of the aircraft maximum speed and the user-defined waypoint speed for the destination waypoint. Certain waypoints result in automatic adjustments of speed to achieve arrival at the waypoints at the specified On Times for these waypoint types, i.e. Timed Routes, Start Turn, and End Turn. If the aircraft is a helicopter that has been commanded to hover by C3I, this commanded speed is set to 0.0 to force the helicopter to decelerate to a hover. The commanded speed will continue to be set to 0.0 until C3I sends an engagement action to proceed to the next waypoint.

The commanded turn g-force is initialized according to what type of flight path the aircraft is currently following. If flying in an orbit, the g-force is set to 3.0. Otherwise, it is set to the maximum allowable g-force for that aircraft structure. Finally, if the destination waypoint is set to terrain following, the aircraft will fly to that waypoint using the terrain-following methodology. Once the speed and turn g-force have been initialized, the direction in which the platform will fly is computed using the methodology described in section 5.5.1. The distance from the platform to the destination is then computed by taking the magnitude of the direction vector.

$$Dir_{Pmag} = \sqrt{Dir_{Px}^2 + Dir_{Py}^2 + Dir_{Pz}^2}$$

Where

Dir_{Pmag} - magnitude of perceived direction vector (m)

Next, the aircraft's turn radius is computed. If the aircraft is flying faster than its corner speed, its perceived turn radius is calculated by:

$$R_T = \frac{V^2}{\left(G_{cmd} \times 9.80665 m/sec^2\right)}$$

where

R_{PT} - perceived turn radius (m)
V_P - perceived aircraft current velocity (m/sec)
G_{cmd} - commanded turn g-force.

If the aircraft is flying slower than its corner speed, the turn radius is computed by:

$$R_{PT} = \frac{V_P}{\dot{\theta}}$$

where

R_{PT}	-	perceived turn radius (m)
V_P	-	perceived aircraft current velocity (m/sec)
$\dot{\theta}$	-	Angle rate curve-fit to aircraft velocity (rads/sec).

If the waypoint is set to Start Turn, the turn radius is not calculated but set to the value specified for the waypoint until the End Turn waypoint is reached. Between a Start Turn and End Turn pair of waypoints, other route waypoints may be defined along the turn to provide changes in altitude for the aircraft. These intermediate waypoints must be very close to the arc defined by the turn radius in the horizontal plane for proper operation.

Waypoint arrival is defined by satisfying two criteria. The aircraft's perceived turn radius must be greater than the perceived remaining distance to the waypoint, and the aircraft's perceived velocity vector must be abeam or greater than the line projected from the perceived location of the waypoint perpendicular to the aircraft's flight path. If these criteria are met and that waypoint is the last, the platform is deactivated. If not at its last waypoint, the aircraft is updated to the next waypoint and its destination altitude is adjusted to the next waypoint's altitude. The aircraft will also use the terrain-following status of this new destination waypoint. The commanded speed will be reset and the direction vector to the new destination waypoint will be recomputed as previously explained.

Next, the direction vector is adjusted to account for either terrain following or Earth curvature. If the destination waypoint requires terrain following, the terrain following methodology described in Subsection 5.4.3 is first used with truth data to modify the direction vector. This methodology, by definition, already accounts for Earth curvature. The floor altitude used by the terrain following is set by the commanded AGL altitude. Once terrain following has adjusted the truth direction vector appropriately, the new perceived direction vector is calculated, if necessary, using the methodology outlined in section Section 5.5.1.

If terrain following is not necessary, the perceived direction vector is modified to account for Earth curvature by flying the aircraft using perception data while maintaining a constant altitude rate. Subsection 5.4.4 explains the constant altitude rate methodology. If the perceived direction vector causes the aircraft to perceive that it is flying above the ceiling altitude or below the floor altitude, it is adjusted as described in Subsection 5.4.3 for altitude monitoring. The ceiling altitude for altitude monitoring is set to the airframe’s service altitude limit. The floor altitude is set at the minimum of 200 m or the commanded AGL altitude.

Once the perception direction vector has been found in this way, the method explained in Section 5.5.1 is used to derive the truth direction vector.

The aircraft is then flown over the integration time interval along the direction vector. For each integration, aircraft integration time T is updated by:

$$D_T = D_T + dt$$

where

D_T - total aircraft integration time (sec).
dt - integration time interval (sec)

During the integration, the aircraft is flown between the waypoints in a straight line or in a curve as described in Subsections 5.4.1 and 5.4.2, respectively. This entire procedure is repeated until D_T exceeds the current simulation time.

5.6.1.2 Nonaerodynamic Waypoint Flight

If the option to use nonaerodynamic flight has been selected for the aircraft element that defines a flight leader and the flight leader is flying waypoints, integration of aircraft position and velocity is performed without regard to aircraft aerodynamic and structural limitations. Just like aerodynamic waypoint flight, integration is carried out over integration steps until the end of the scenario update interval. Once the total update interval has been achieved, the new aircraft position and velocity are passed on to the other models.

Certain waypoints result in automatic adjustments of speed to achieve arrival at the waypoints at the specified On Times for these waypoint types, i.e. Timed Routes, Start Turn, and End Turn. For other waypoint types, the velocity between waypoints is adjusted by means of a linear interpolation between the previous and next waypoint speeds, as a function of range. First, the position of the destination waypoint relative to the previous waypoint is found:

$$\vec{W}_{P_{Diff}} = \vec{W}_{P_{Curr}} - \vec{W}_{P_{Prev}}$$

where

$\vec{W}_{P_{Diff}}$ - destination waypoint position relative to previous waypoint (m)
$\vec{W}_{P_{Curr}}$ - destination waypoint position vector (m)
$\vec{W}_{P_{Prev}}$ - previous waypoint position vector (m).

Next, the magnitude of the desired velocity difference between the current and previous waypoints is computed:

$$\Delta V = V_{Curr} - V_{Prev}$$

where

ΔV - velocity difference between current and previous waypoints (m/sec)
V_{Curr} - destination waypoint velocity magnitude (m/sec)
V_{Prev} previous waypoint velocity magnitude (m/sec).

Then, commanded speed is set by linearly interpolating between the waypoint desired velocities:

$$V_{Cmd} = V_{Curr} - \Delta V \times \left(\frac{D_{Mag}}{W_{Mag}} \right)$$

where

V_{Cmd} - commanded speed (m/sec)
V_{Curr} - destination waypoint velocity magnitude (m/sec)
ΔV - velocity difference between current and previous waypoint (m/sec)
D_{mag} - magnitude of aircraft direction vector (m)
W_{Mag} - magnitude of current to previous waypoint relative position vector (m).

If the aircraft is a helicopter that has been commanded to hover by C3I, this commanded speed is set to 0.0 to force the helicopter to decelerate to a hover. The commanded speed will continue to be set to 0.0 until C3I sends an engagement action to proceed to the next waypoint.

Using this commanded speed, the required acceleration is then computed according to:

$$A_{Mag} = \frac{V_{Cmd} - V_{Pmag}}{Dt}$$

where

A_{Mag} - required acceleration magnitude (m/sec^2)
V_{Cmd} - commanded speed (m/sec)
V_{Pmag} - perceived aircraft current velocity magnitude (m/sec)
Dt - integration step time (sec).

Once the direction vector have been adjusted for altitude limits and Earth curvature as described in Subsections 5.5.3 and 5.5.4, the current velocity and required acceleration are applied along the truth direction vector to get the truth velocity and truth acceleration vectors:

$$\vec{V} = \vec{U}_{Dir} \times V_{mag}$$

$$\vec{A} = \vec{U}_{Dir} \times A_{Mag}$$

where

$\vec{V}$ - velocity vector in desired direction of flight (m/sec)
$\vec{U}_{Dir}$ - unit desired direction of flight
V_{mag} - aircraft current velocity magnitude (m/sec)
$\vec{A}$ - acceleration vector in desired direction of flight (m/sec^2)
A_{Mag} - required acceleration magnitude (m/sec^2).

The velocity and acceleration vectors are then used to update the aircraft state. The following equations are used for updating truth:

$$\vec{P} = \vec{P} + \vec{V} \times Dt + \frac{1}{2} \times \vec{A} \times Dt^2$$

$$\vec{V} = \vec{V} + \vec{A} \times Dt$$

where

$\vec{P}$	-	true aircraft position vector (m)
$\vec{V}$	-	true velocity vector in desired direction of flight (m/sec)
Dt	-	integration step time (sec)
$\vec{A}$	-	true acceleration vector in desired direction of flight (m/sec^2).

Once the truth position and velocity vectors have been updated, the perception values are updated according to the methodology in Section 5.5.1.

This procedure is repeated until the scenario update interval has been achieved.

Turns using the nonaerodynamic flight are instantaneous. Nonaerodynamic waypoint flight only occurs for flight leaders flying Waypoint mode between two successive waypoints in their waypoint list. Flight leaders will also use nonaerodynamic flight if they are flying between waypoints in an orbit pattern. Flight leaders returning from an engagement do not use nonaerodynamic flight until they pass the next waypoint in the list. If the nonaerodynamic option is selected for wingmen, they use aerodynamic integration until they become the flight leader.

5.6.1.3 Route Waypoints Type

Route waypoints are used to fly an aircraft along a planned flight path. They are used by flight processing as the destination waypoint until the previously mentioned criteria for reaching a waypoint have been met. Normally, the aircraft will fly to these waypoints only once during the course of the flight. Only during orbits can an aircraft be flown to Route waypoints multiple times.

5.6.1.4 Timed Route Waypoints Type

Timed Route waypoints are used to fly an aircraft along a planned flight path and arrive at the waypoint location at the specified on time of the waypoint. Aircraft will fly to these waypoints only once during the course of the flight as the on time is specified from the start of the scenario.

As the aircraft is flying to the waypoint, the acceleration is computed to maintain a near constant acceleration to achieve the destination waypoint at the on time specified for that waypoint.

Constant acceleration is calculated by

$$A_{Mag} = \frac{2 \times (P - P_{Pred})}{t^2}$$

where

P	-	distance required to travel from current location to waypoint (m)
P_{Pred}	-	predicted distance to be travelled at current velocity of the aircraft (m)
t	-	time to fly until requested time of arrival (sec)
A_{Mag}	-	magnitude of constant acceleration of flight (m/sec^2).

The acceleration is limited by the maximum acceleration available to the airframe. The acceleration can be further limited by the Max Command Acc field on the Airframe Definition window. The speed is also constrained to be within the minimum and maximum speed on the aircraft and the minimum and maximum speeds set for the destination waypoint. During curved flight between timed route waypoints, the Max Climb Angle restriction, detailed in section 5.5.3.2.6, is ignored.

If the aircraft cannot arrive at the waypoint at the requested time due to speed limitations, the aircraft will continue to the waypoint at the maximum speed. If the aircraft arrives at a waypoint before the requested time due to minimum speed limitations, the aircraft will continue on to the next waypoint.

For the last waypoint in the waypoint list, aerodynamic flight is turned off and non-aerodynamic flight calculations are performed to allow the aircraft to arrive at the last waypoint on time. The on time for the last waypoint is still used for non-aerodynamic flight, but restrictions by the airframe are ignored e.g. turns are instantaneous.

5.6.1.5 Start/End Turn Waypoint Type

Start and End Turn waypoints are used to fly planned turns along an aircraft's flight path and arrive at the waypoint location at the specified on time of the waypoint. Aircraft will fly to these waypoints only once during the course of the flight as the on time is specified from the start of the scenario.

An aircraft will arrive at the Start Turn and End Turn waypoints by the specified On Time, adjusting speed and acceleration as described by the Timed Route waypoint type. The turn radius specified on the Start Turn is the turn radius that the aircraft will use until it reaches the End Turn waypoint. Turns are performed such that the radius of the resulting ground track is constant throughout the turn though altitude may be changed during the turn due to terrain following or to Route or Timed Route waypoints that are defined along the turn. If Timed Route waypoints are defined, the On Time on the Timed Route waypoint is ignored.

5.6.1.6 Start/End Pattern Waypoint Type

Start Pattern and End Pattern waypoints are used to define orbits in an aircraft's planned flight path. These orbits enable the aircraft to fly repeatedly around multiple defined waypoints until the orbit off time is exceeded. The Start Pattern waypoint designates where the orbit

begins. The End Pattern waypoint designates where it ends. In between the Start Pattern and End Pattern waypoints can exist any number of Route waypoints that simply define the legs of the orbit.

As the aircraft is flying its waypoint set, its destination waypoint becomes either the next Start Pattern, Route, or End Pattern waypoint. When the End Pattern waypoint is reached, the destination waypoint becomes the Start Pattern point. This process is repeated until the End Pattern off time is exceeded. At that time, the destination waypoint becomes the first waypoint after the End Pattern point. If the off time is reached and no more waypoints exist, the platform becomes a Return To Base platform and returns to its home air base.

When the aircraft flies an orbit, it flies at the speed and altitude set by the user for each of the waypoints that comprise the orbit. Aircraft turns are limited to 3.0 Gs. Terrain following will be performed by the aircraft if required. Otherwise, the aircraft will fly along the curvature of the Earth.

When the orbit(s) are set up by the user in Scenario Generation, the option to refill the waypoint set may be desired. A check box on the edit or deploy platform windows gives the user that capability. Selecting the Refill Waypoint Set check box allows nonscripted aircraft at an air base to be commanded by the Flexible Commander to refill the waypoint set.

5.6.1.7 PushOffPoint Waypoint Type

PushOffPoint waypoints are used to define a point after an orbit where the aircraft needs to be at a specified time to start the next phase of its mission. They can only be specified directly following End Pattern waypoints. The PushOffPoint forces the orbiting aircraft to leave the orbit prematurely, despite the off time for the orbit, so that the aircraft will arrive at the PushOffPoint at its on time. The PushOffPoint off time has no meaning.

When the aircraft leaves the orbit, it flies to the PushOffPoint at the speed and altitude set by the user for that waypoint. As when flying orbits, aircraft turns are limited to 3.0 Gs. Terrain following will be performed by the aircraft if required. Otherwise, the aircraft will fly along the curvature of the Earth.

5.6.1.8 Hover Waypoint Type

Hover waypoints are used to define a point where the aircraft is to stop and hover. These are only used by helicopters. Hover waypoints deployed for airplanes will be treated as Route waypoints. When the next waypoint in the list is a Hover waypoint, the range to the waypoint is continually monitored to trigger when the helicopter must begin decelerating from its current speed at the maximum deceleration Gs to be at zero velocity by the time the Hover waypoint is reached. Once the helicopter is at the Hover waypoint, it remains there until the hover duration time is reached. The off time for Hover waypoints defines this hover duration. For profiles, the hover duration is a direct input.

5.6.1.9 ARCP Waypoint Type

Airborne Refueling Control Point (ARCP) waypoints are used in mission-planned airborne refueling. They provide a rendezvous location for a receiver aircraft flight requiring refueling before continuing its mission and the refueling tanker flight from which the receiver flight is to take on fuel. When the next waypoint in the waypoint set is the ARCP waypoint, this triggers the Point-Parallel preplanned airborne refueling operations. The receiver flight leader begins assessing whether to go directly to the tanker or to proceed to the tanker's loiter to await the correct rendezvous time, the Airborne Refueling Control Time (ARCT), defined by setting the receiver flight leader's ARCP waypoint on time. The ARCP waypoint off time has no meaning. It is important to note that both the receiver flight and the tanker flight require this same ARCP waypoint in order for the refueling operations to work correctly. Only the on time for the receiver flight leader's ARCP waypoint serves as the ARCT. See Subsection 5.6.3 for a complete description of how the mission-planned and dynamic refueling operations work within flight processing. See Subsection 4.7.29 for a description of the Airborne Refueling (AR) Tanker ruleset.

5.6.1.10 Target Waypoint Type

Target waypoints are used exactly like Route waypoints to fly an aircraft along a planned flight path. However, Target waypoints cannot be skipped by ruleset commands. The Bomber, Fighter-Bomber, and AGAttacker rulesets issue a command to skip a waypoint after engaging a ground target or flying a profile for a target. Target waypoints ignore the ruleset commands and should be used when waypoints in the flight path should not be skipped by ruleset commands.

5.6.1.11 Report Waypoint Type

Report waypoints are used exactly like Route waypoints to fly an aircraft along a planned flight path. However, Report waypoints also have the additional capability of triggering the dissemination of track information at the report waypoint if scheduled reporting is selected as a Track reporting option.

5.6.1.12 Jammer Start Waypoint Type

Jammer Start waypoints are used exactly like Route waypoints to fly an aircraft along a planned flight path. However, rulesets which have Jamming Control phase inputs can optionally activate jammers or directed energy devices at the Jammer Start waypoint.

5.6.1.13 Jammer End Waypoint Type

Jammer End waypoints are used exactly like Route waypoints to fly an aircraft along a planned flight path. However, rulesets which have Jamming Control phase inputs can optionally deactivate jammers or directed energy devices at the Jammer End Waypoint.

5.6.1.14 Waypoint Deployment Options

Platforms can be deployed in a scenario through two waypoint deployment options: route deployment or direct waypoint input. The route deployment option allows multiple platforms

(aircraft and ground) to use identical waypoint sets. The direct waypoint input option allows each waypoint to have a unique waypoint set input by the user.

When using the route deployment option, a platform is deployed by selecting the route option and then selecting the desired route. The waypoints for this route are then used by the platform when the scenario is executed. All platforms that have this same route will use the same route waypoints. When deploying platforms using routes, randomness in platform activation time can be added by the setting of the platform Activation Time. If randomness is eliminated, the value entered for mean will be used as the activation time.

There are three types of routes that can be used for deploying platforms: Standard, Boost Phase Intercept (BPI) Route, and BPI Refill Route. All of the routes can be built with one or more of the available types of waypoints, dependent upon whether the platform is an aircraft or ground platform. Standard Routes are waypoint sets that can be defined for either aircraft or ground platforms. BPI Routes and BPI Refill Routes are only used for deploying aircraft and have special features described in Subsection 5.6.1.13. A fourth type of route exists that cannot be used for deploying platforms. This type is the Scramble Route, used in DCA operations by Flexible Commanders to scramble aircraft off of an air base to engage incoming threats. Further explanation on how Scramble Routes are used can be found in Subsection 4.7.2.2.3.

5.6.1.15 BPI CAP Operations

BPI CAP Operations are performed to allow TBMs to be engaged by aircraft while the TBMs are still in the boost phase. Aircraft fly on BPI CAPs, awaiting the launch of TBMs by the enemy forces. The aircraft fly the CAPs in two groups that counter-rotate about the CAP to keep one aircraft group (the hot aircraft) facing the perceived target area at all times. When a TBM is launched, the hot aircraft fly toward the target area and engage the threats using the Engage Flight mode. The cold aircraft turn towards the target area and can also engage the threats should the need arise. See Subsection 5.6.4 for an explanation of the Engage Flight Mode, and see Subsection 4.7.11 for an explanation on the Fighter Ruleset and how it pertains to BPI Operations.

Once the engagement is complete, the aircraft return to the BPI CAPs and continue to fly until needed again. If they should run low on fuel, BPI Refill CAPs provide replacement aircraft to assume the BPI role. The BPI aircraft then undergo dynamic refueling operations. After refueling, they assume the BPI Refill CAPs vacated by the replacement aircraft. If the BPI aircraft run out of weapons or if they have been on-station for an extended period of time, the BPI Refill CAPs again provide replacement aircraft for the BPI CAPs. Air bases then provide replacement aircraft for those departing the BPI Refill CAPs.

The Waypoint flight mode allows for the special capabilities required to capture the logistics of these BPI CAP Operations. These features include counter-rotating CAPs, two-stage CAP refilling, and special airborne refueling support for BPI Refill aircraft.

5.6.1.15.1 Counter-Rotating CAPs

Aircraft deployed using BPI Routes automatically perform counter-rotation as they orbit their pattern waypoints. Prior to reaching the pattern, two flights of aircraft deployed on a BPI

Route will fly the waypoints normally using the Waypoint flight mode. When the aircraft reach the waypoint just prior to the pattern, they will split up and head toward their respective Start Pattern waypoints.

When the BPI Route is associated with the flight leaders, the first flight leader receives the route's waypoint set exactly as specified. The second flight leader receives all the same waypoints prior to the pattern waypoints but then receives a slightly reordered version of the orbit pattern. The End Pattern waypoint for this second flight leader becomes the Start Pattern waypoint. For a two-waypoint orbit, the Start Pattern waypoint becomes the End Pattern waypoint. If more than two waypoints exist in the orbit, the Start Pattern waypoint becomes a Route waypoint and the rest of the waypoints are unchanged except the last, which then becomes the End Pattern waypoint. This juxtaposition of the orbit patterns allows the flight leaders to fly counter-rotation with respect to each other.

The timing for the counter-rotation is based on the arrival of each of the two flight leaders at their respective Start Pattern waypoints. The calculations for the timing account for the turn radius and the distance between each flight leader's current position, any subsequent waypoints, and their destination Start Pattern waypoint. As the two flights fly their respective orbits, the flight leader's throttle is adjusted up or down to ensure that it arrives at its Start Pattern waypoint at approximately the same time that the other flight leader arrives at its Start Pattern waypoint. The throttling is limited to the aircraft's maximum and minimum speeds input on the aircraft element.

BPI Routes require two flights of aircraft with no limit on the number of aircraft in each flight. The route itself is required to have at least an orbit pattern consisting of a Start Pattern waypoint, any number of other waypoints, and an End Pattern waypoint. Both requirements are enforced by the user interface and allow counter-rotation to be performed properly by the participant aircraft.

It is important to note that the optimum BPI CAP consists of a two-waypoint orbit pattern. Setting up a BPI CAP with more than two waypoints eliminates the capability for one of the two aircraft flights to be facing the target area at all times. Also, aircraft flying counter-rotation on BPI Routes always perform right turns when they reach the Start Pattern waypoint. This ensures that the two flights are truly counter-rotating and do not fly crisscrossing paths. Counter-rotation calculations account for the possibility that one of the two flights is not available since it has gone RTB or was destroyed. In this case, the remaining flight will fly the CAP waypoints using normal waypoint speeds.

5.6.1.15.2 BPI Two Stage CAP Refilling

Aircraft deployed using BPI Routes are automatically refilled by a two-stage refilling process. The first stage of the refilling comes from CAPs defined by aircraft deployed using BPI Refill Routes. When aircraft on a BPI Route need to go RTB or need to air refuel, the commander of the BPI Operations aircraft requests aircraft flying on the BPI Refill Route to replace the departing BPI aircraft.

The second stage of the refilling involves aircraft sitting at air bases assigned to the commander of the BPI Operations. When BPI Refill aircraft replace BPI aircraft going RTB, the commander scrambles aircraft off of one of the grunt air bases to replace the BPI Refill aircraft

on their refill CAP. This ensures that the level of aircraft assets dedicated to BPI Operations is maintained. If the BPI aircraft are just going to air refuel, no aircraft are scrambled off of the base since the BPI aircraft replace the BPI Refill aircraft on the BPI Refill Route once they have completed refueling. See Subsection 5.6.1.13.3 for an explanation on BPI air refueling. Also, see Subsection 4.7.2 for a description of the Flexible Commander's role in BPI operations.

5.6.1.15.3 BPI Air Refueling Support

Airborne refueling support is an integral component of the BPI Operations captured in EADSIM. When the refueling cell is deployed, the tankers in the cell should be added to the tanker lists of all the fighter aircraft, both in the air and on the ground, that are dedicated to the BPI operation. To keep the cell dedicated to BPI, no other aircraft deployed should have these tankers on its tanker list. The tankers should also be deployed as grunts of the BPI commander. This ensures that the commander can schedule defensive reactions for its BPI aircraft, should the need arise. See Subsection 4.7.2 for a description of the Flexible Commander's role in BPI operations.

Airborne refueling for BPI Operations functions exactly as any dynamic refueling operation described in Subsection 5.6.3.2. The only exception to this is for aircraft deployed using BPI Refill Routes. Any aircraft on the BPI Refill CAPs is subject to a Ready Alert Bingo Limit input for the BPI Refill Route. This fuel level defines when the BPI Refill aircraft need to air refuel. Typically, the Ready Alert Bingo Limit serves to keep the refill aircraft at a higher fuel state than the normal ARBingo fuel level so that it will always have plenty of fuel when replacing BPI aircraft. BPI aircraft are not subject to the Ready Alert Bingo Limit unless they replace BPI Refill aircraft on their CAP. BPI aircraft on the BPI CAPs are only subject to the ARBingo fuel level.

When BPI Refill aircraft reach the Ready Alert Bingo Limit, they vector to the tanker cell and perform dynamic refueling operations. While undergoing refueling, none of these aircraft can be commanded to replace BPI aircraft. Only when they have completed refueling can they replace aircraft on BPI CAPs.

When BPI aircraft needing refueling are replaced by BPI Refill aircraft, they leave the BPI CAP and vector to the refueling cell for dynamic refueling operations. Once refueling is completed, they become BPI Refill aircraft and are subject to the Ready Alert Bingo Limit. They too, cannot be commanded to replace BPI aircraft until the refueling operations are complete.

Likewise, aircraft scrambled off the air base to replace BPI Refill aircraft become BPI Refill aircraft themselves and again become subject to the Ready Alert Bingo Limit for dynamic refueling operations. See Subsection 5.6.1.13.2 for a description of the two stage CAP refilling procedure.

5.6.2 Wingman Flight Mode

The Wingman flight mode is specified by the user by designating a flight leader for an airborne platform. The Wingman mode can be changed to other flight modes by the C3I model. The C3I model returns wingmen to Wingman mode after an engagement or a drag maneuver has been completed. The Wingman mode is also used for airborne refueling operations. When a

receiver and its flight are flying in formation with a tanker during the actual fuel transfer, they are flying as wingmen to the tanker. Even if the tanker is flying as a wingman to another tanker flight leader, it becomes a refueling flight leader to those receivers that are refueling.

Each wingman's position in the flight is defined by a position vector relative to the flight leader. The relative position vector for each wingman is computed based on a wingman formation. The wingman formation can be either the default formation or a formation created by the user. Additionally, the refueling flight leader can have a special refueling formation defined for placement of those receiver aircraft that are refueling. If none exists, the refueling aircraft placement will be based on the default formation as well. See Subsection 5.6.3 for more information on the airborne refueling operations and Subsection 4.7.29 for more information on the AR Tanker ruleset.

To compute the relative position vector of a wingman, a body-centered coordinate system is established and centered on the flight leader. The x direction is aligned out the nose of the flight leader along its current velocity vector. The y-axis is out the left wing of the flight leader and perpendicular to its ECI position vector. The z-axis is perpendicular to both the x- and y-axes in the local up direction. Using the desired formation data, the relative position vector is found in the local coordinate frame. See Subsections 5.6.2.1 and 5.6.2.2 for the calculation of the relative position vector in the default and user-defined formations, respectively. Once computed, the vector is rotated into ECI coordinates. Then, the position to which the wingman must fly is computed as the vector sum of the flight leader's current position and the wingman's relative position vector.

When updating the wingman's position during flight, a smoothing function using a 20-sec look-ahead is applied to compute the wingman direction of flight vector:

$$\vec{D} = \vec{P}_{Des} - \vec{P}_{A/C} + 20.0 \times \vec{V}_{FL}$$

where

$\vec{D}$	-	wingman direction-of-flight vector (m)
$\vec{P}_{Des}$	-	position relative to flight leader to which wingman must fly (m)
$P_{A/C}$	-	current position of wingman (m)
20	-	look-ahead time (sec)
$\vec{V}_{FL}$	-	velocity of flight leader (m/sec).

To compute the velocity that the wingman will fly, the magnitude of the wingman direction-of-flight vector is computed:

$$D_{Mag} = \left|\vec{D}\right|$$

where

D_{Mag}	-	magnitude of wingman direction-of-flight vector (m)
$\vec{D}$	-	wingman direction-of-flight vector (m).

Then, the wingman velocity at which to fly along this direction-of-flight vector is calculated:

$$V_{Mag} = D_{Mag}/(20.0 + D_t)$$

where

V_{Mag}	-	velocity magnitude at which wingman is to fly (m/sec)
D_{Mag}	-	magnitude of wingman direction-of-flight vector (m)
20	-	look-ahead time (sec)
D_t	-	time difference between flight leader and wingman update times (sec).

If the direction-of-flight vector causes the aircraft to fly above the ceiling altitude or below the floor altitude, it is adjusted, as described in Subsection 5.4.3, for altitude monitoring. The ceiling altitude for altitude monitoring is set to the airframe's service altitude limit. If the flight leader is not flying in Waypoint mode, the wingman's floor altitude is defaulted to the minimum of 200 m or the flight leader's commanded altitude. If the flight leader is not flying Waypoint mode, the wingman's floor altitude is defaulted to 200 m.

If the difference between the wingman's direction-of-flight vector and velocity vector is less than 0.8 degrees, the wingman will fly straight to the destination point; otherwise, the wingman will fly in a curve to the destination point.

Any waypoint information entered by the user for a specific wingman will not be used in flying the wingman. When a wingman is engaging a ground target, it will use the terrain-following status and commanded altitude of the flight leader. The wingman will fly in the Engage flight mode, as described in Subsection 5.6.4. When an engagement has been completed or stopped, a wingman is given a destination position that will regroup the wingman with its flight.

5.6.2.1 Default Wingman Formation

If using the default formation, the wingman's relative position vector in the flight leader's local coordinate frame is computed in the order in which the wingman is deployed, as described below. Each flight member is identified by an ID number. The wingmen locations behind the flight leader are arranged in rows of four columns each. The flight ID divided by four results in a quotient used as the row and a remainder used as the column. The relative position vector consists of three component distances, X, Y, Z, from the flight leader's position. The X component distance is computed as:

$$X = -212.0 - (row) * 600.$$

The Y and Z component distances are based on the column assigned:

Column	Y value	Z value
1	1220	25

2	-1220	25
3	-1620	-15
4	1620	-15

5.6.2.2 User-Created Wingman Formation Element

If a user-created formation is used, the wingmen are deployed using wingman position data input by the user. The user creates a formation element in Scenario Generation. It consists of the location of all the wingmen in the flight relative to the flight leader's position. Each wingman location is defined by an angle measured off the flight leader's right wing, a lateral offset distance between the wingman and the flight leader, and a vertical offset distance. Once the formation element has been created, it is associated by the user to the ruleset of the flight leader.

The wingman relative position vector is computed by using this ruleset formation data:

$$X = \text{offset}_{\text{Lat}} \times \sin\left(\Sigma_{\text{Aft}} + 180\right)$$

$$Y = \text{offset}_{\text{Lat}} \times \cos\left(\Sigma_{\text{Aft}} + 180\right)$$

$$Z = \text{offset}_{\text{Vert}}$$

where

N	-	component of relative pos vector in nth direction (m)
$\text{offset}_{\text{Lat}}$	-	user-input lateral offset distance (m)
ε_{Aft}	-	user angle off flight leader right wing (sec deg)
180	-	coordinate alignment angle (deg)
$\text{offset}_{\text{Vert}}$	-	user-input vertical offset distance (m).

The coordinate alignment angle is used to orient the input angles measured relative to the right wing of the flight leader to the local coordinate frame that is 180 deg out of sync in the local x-y plane.

5.6.3 Airborne Refueling Mode

There are two types of airborne refueling operations available: Mission-Planned Refueling and Dynamic Refueling.

5.6.3.1 Mission-Planned Refueling

Mission-Planned Refueling allows preplanned refueling of platforms in a scenario prior to their completing other phases of their mission. The requirements for this type of refueling are:

1) Valid tankers have been deployed (see Subsection 4.7.29.5)
2) The aircraft element Max Fuel Receiving Rate is greater than zero

3) The platform has been selected as Refuel Capable
4) At least one tanker has been associated with this platform for refueling
5) The platform Refueling Amount is greater than zero
6) The platform has been deployed with an ARCP waypoint in its waypoint set.

When the receiver platform and its wingmen reach the waypoint prior to the ARCP waypoint, the Mission-Planned Refueling operations begin. At this time, the "refueling operations event" is logged for post-processing. Also at this time, the receiver flight leader ID is stored on the associated tanker's Timing List, causing timing checks to begin for determining when the receiver flight leader and the associated tanker's flight leader needs to go to the ARCP waypoint to arrive by the ARCP waypoint on time (the ARCT). In addition to this timing constraint, the receiver flight leader's associated tanker must also be available for refueling for the receiver flight to depart for the ARCP waypoint. If the associated tanker is not busy or if the associated tanker allows for refueling with another in the flight and that other tanker is not busy, the receiver flight and the tanker can depart for the ARCP.

If it is not time to go to the ARCP, the receiver flight leader adopts the tanker flight leader's loiter orbit. The "loiter orbit adoption" event is logged for post-processing. While in the loiter orbit, timing checks continue for reaching the ARCP by the ARCT. The receiver flight leader and its wingmen remain in this orbit until their time to depart.

When the time to depart has been reached, the receiver flight leader and the wingmen head directly to the ARCP. The "receivers' departure for the ARCP waypoint event" is logged for post-processing. Consequently, when the tanker time to depart has been reached, the tanker flight leader and its wingmen will head directly to the ARCP.

Once the receiver aircraft arrive at the ARCP waypoint, the receiver flight leader becomes a wingman to the associated tanker. This causes the receiver flight leader and its wingmen to go to rendezvous with the proper tanker, whether the tanker is the tanker flight leader or one of its wingmen. This "going to rendezvous" event is logged for post-processing.

If the associated tanker has a defined refueling formation, the receiver flight leader rendezvous with the associated tanker takes place when it gets to within 150% of the range from its formation location to the associated tanker. If using the default formation for the refueling wingmen, the receiver flight leader must fly to within 1300m. When the receiver flight leader has rendezvoused with the associated tanker, all the receiver flight leader's wingmen become wingmen to the associated tanker. The "successful rendezvous" event is logged for post-processing. At this time, the receiver flight leader and its wingmen are no longer available for scheduling for engagements by C3I. They fly with their refueling tanker flight leader until refueling operations are complete. If the tanker is attacked, the receivers drag along with the tanker. The receivers can only react to engagements themselves if attacked by a threat. See Subsection 4.7.29.2.1 for a description of the tanker defensive reactions.

At this time, fuel transfer begins. The receivers begin refueling up to the maximum number of simultaneous receivers that the tanker can service at one time. A "fuel transfer has begun" event is logged for post-processing for each receiver that begins taking on fuel. The fuel transfers from the tanker to the receiver at the minimum of the tanker's Max Off-Load Rate and

the receiver's Max Fuel Receiving Rate. The transfer takes place until there is no more fuel available on the tanker, the receiver's takeoff fuel weight is reached, or the receiver's Refuel Amount is reached. Fuel transfers can be temporarily halted due to a tanker defensive reaction. See Subsection 4.7.29.2.1 for a description of those reactions. When the fuel transfer is completed, the receiver stops taking on fuel and the next receiver begins refueling. As the receivers complete their refueling, a "fuel transfer has stopped" event is logged for post-processing.

This procedure is repeated until all aircraft in the refueling flight complete refueling. A "refueling operations have been completed" event is logged for post-processing. At this time, the receiver flight leader begins flying Waypoint mode again and resumes its mission waypoints. The receiver wingmen quit flying as wingmen to the associated tanker and resume flying as wingmen to the receiver flight leader. The receiver flight leader and its wingmen again become available for engagement scheduling by C3I. The tanker flight leader and its wingmen continue flying their orbit, awaiting the next receiver flight.

5.6.3.2 Dynamic Refueling

Dynamic Refueling allows for refueling of platforms in a scenario when they are getting low on fuel. The requirements for this type of refueling are:

1) Valid tankers have been deployed (see Subsection 4.7.29.5)
2) The aircraft element Air Refueling Bingo Limit is greater than zero
3) The aircraft element Max Fuel Receiving Rate is greater than zero
4) The platform has been selected as Refuel Capable
5) At least one tanker has been associated with this platform for refueling
6) The platform Refueling Amount is greater than zero
7) The Ready Alert Bingo Limit is greater than zero if the platform is a BPI Refill aircraft (see Subsection 5.6.1.13.3).

When the receiver platform or one of its wingmen reaches the AR Bingo fuel threshold or the Ready Alert Bingo Limit if on a BPI Refill Route (see Subsection 5.6.1.10), the Dynamic Refueling operations begin. At this time, C3I is notified that this aircraft and the rest of the flight must go to refuel. When notified, C3I then loops through the receiver's associated tanker list and finds the closest available tanker to send the flight to for refueling. Once FP receives the "Dynamic Refueling Operations" engagement action from C3I, the "dynamic refueling operations event" is logged for post-processing. If the assigned tanker is busy and the receiver cannot use another tanker in the flight, the receiver flight leader adopts the tanker's ARCP and flies toward the tanker flight. If, upon reaching the ARCP waypoint, the tankers are still not available, the loiter orbit is adopted by the receiver flight leader. The "loiter orbit adoption" event is logged for post-processing.

Once the tanker is no longer busy, the receiver flight leader drops the loiter orbit and becomes a wingmen to the assigned tanker. This causes the receiver flight leader and its wingmen to go to rendezvous with the assigned tanker, whether the tanker is the tanker flight leader or one of its wingmen. This "going to rendezvous" event is logged for post-processing.

The remainder of the dynamic refueling operation works exactly like the Mission Planned Refueling. Once the receiver flight leader rendezvous with the assigned tanker, fuel transfers begin. When all aircraft in the refueling flight competes refueling, a "dynamic refueling operations have been completed" event is logged for post-processing. At this time, the receiver flight leader begins flying Waypoint mode again and resumes its normal operations. The receiver wingmen stop flying as wingmen to the assigned tanker and resume flying as wingmen to the receiver flight leader. The tanker flight leader and its wingmen continue flying their orbit, awaiting the next receiver flight.

Dynamic refueling operations play a major part in BPI Operations as modeled in EADSIM. See Subsection 5.6.1.13 for an explanation of their role and the use of a special Ready Alert Bingo Fuel Limit for the scheduling of dynamic refueling operations for BPI aircraft.

5.6.4 Engage Flight Mode

The Engage flight mode is selected by the C3I model for an aircraft to fly a path toward a selected target. There are three aircraft propagation methods for the engage flight mode: standard engage mode propagation, bomber propagation, and fpole propagation for aircraft that have launched a semiactive weapon against a target with air-to-air capability.

An Engage Speed Table can also be used to specify how an aircraft will fly in the Engage flight mode. The table consists of ranges to the target, with desired speed, speed multiplier ratio, and max speed limiter for each range.

5.6.4.1 Standard Engage Propagation

The Standard Engage flight mode always uses the truth to navigate, because it is performed relative to targets that are in track. Perception errors will therefore wash out. However, if there is a navigation element, perception is updated according to the equations in Section 5.5.1. If the aircraft has no navigation element, then perception will be set and updated with the same values as the truth.

The standard engage propagation is used for flying all nonbomber aircraft that are vectored to engage ground or airborne targets, including TBMs. Only aircraft that are engaging air-to-air-capable targets and have launched a semiactive weapon are flown with this propagation method. This method computes the attacker's velocity to approach the target, performs a predictive intercept for ABT threats, computes a direction-of-flight vector, chooses whether to fly straight or in a curved path to the target, and updates the attacker's position and velocity after propagation is complete. It also adjusts the direction of flight vector for Earth curvature, azimuth and elevation limits, and altitude limits.

Once the updated position of the target has been obtained, a direction vector between the attacker and the target is computed:

$$\vec{D} = \vec{P}_{Tgt} - \vec{P}$$

where

$\vec{D}$ - direction-of-flight vector (m)
$\vec{P}_{Tgt}$ - target position vector (m)
$\vec{P}$ - attacker position vector (m).

Next, if the target is not a missile and the attacker is not currently locked on to the target, a predicted intercept point is computed. It is computed by first finding the maximum range of all the air-to-air weapons that the attacker is carrying. This range is the maximum launch range for successfully intercepting the target. If this range is greater than the current distance to the target, the missile can be launched immediately. The time to intercept is then computed using the methodology described in Appendix B-5. If an intercept is possible and the time to intercept is computed, the target aircraft is flown forward to this time using its current velocity direction:

$$\vec{P}_{Tgt} = \vec{P}_{Tgt} + T_{IP} \times \vec{V}_{Tgt}$$

where

$\vec{P}_{Tgt}$ - target position vector (m)
T_{IP} - time to intercept (sec)
$\vec{V}_{Tgt}$ - target velocity vector (m/sec).

Otherwise, if the target is a missile or the attacker is locked on or no intercept point could be found, the actual target position is used to guide the attacker. To provide altitude maintenance for the attacker, the target position is recomputed to the magnitude of the attacker position vector. This is accomplished by computing the attacker's altitude using the attacker's position and the DblECEFtoAlt function described in Appendix B10. Next, the latitude and longitude of the target is computed using the DblECEFtoLatLon as described in Appendix B10. Finally, the LLAToDblECEF function described in Appendix B10 is used to computed the new target position based on the target's current latitude and longitude and the attacker's current altitude.

After updating the direction-of-flight vector using the new target position vector, the direction vector is adjusted for terrain following if required. If the attacker's previous altitude mode was terrain following and the target is a ground platform, the Engage mode uses terrain following. Terrain-following adjustments are made using the methodology described in Subsection 5.4.3. The floor altitude used by the terrain following is set by the commanded AGL altitude. Commanded speed for terrain-following flight is set to the aircraft corner speed to maximize turn rate.

If terrain-following adjustments are not necessary, the commanded speed is set based on engagement geometry. If the difference between the attacker's and target's velocities is greater than the target's velocity, the attacker and target are flying toward each other. Commanded speed is set to maximum speed if the relative distance is greater than 6,500 m. If less, the commanded speed is set to corner speed. If the velocity difference is greater than the target's velocity, the aircraft are flying away from each other. Commanded speed is set to corner speed if the target is less than 2,500 m away. If the target is more than 2,500 m away, commanded speed is set to the lesser of its maximum velocity or 125% of the target's velocity.

Next, if not terrain following, the direction vector is adjusted for Earth curvature. The adjustments are made using the Constant Altitude Rate methodology described in Subsection 5.4.4.

If the attacker is not locked on to its target and the target is not a missile, the direction vector is adjusted to account for azimuth and elevation limits. Azimuth and elevation limits can be input for several ruleset phases. They can be used to keep the target aircraft within a desired FOV. To make the adjustments, the attacker's position and direction vector are passed into the DblECEF2BodyMatrix function described in Appendix B10. The target position relative to the attacker location is then rotated into the body frame coordinates using the DblDRotate1 function described in Appendix B10.

Next, the elevation and azimuth are both computed:

$$\varphi_{Tgt} = \frac{\pi}{2} - \cos^{-1}\left(Tgt_Z\right)$$

$$\theta_{Tgt} = \tan^{-1}\left(\frac{Tgt_X}{Tgt_Y}\right)$$

where

φ_{Tgt}	-	target elevation (rad)
Tgt_Z	-	local z-axis component of target position
θ_{Tgt}	-	target azimuth (rad)
Tgt_X	-	local x-axis component of target position
Tgt_Y	-	local y-axis component of target position.

If the target azimuth and elevation angles do not exceed the limits defined for the current ruleset phase, no adjustments are made. Otherwise, a direction vector in the local frame is set up that points along the local y-axis:

$$D_X = 0.0$$

$$D_Y = 1.0$$

$$D_Z = 0.0$$

where

D_n	-	direction vector component in local nth direction.

If the target azimuth exceeds the azimuth limit, the local direction vector is adjusted so the azimuth of the target is at the azimuth limit from the direction vector:

$$D_X = \pm \sin\left(\left|\theta_{Tgt}\right| - \left|\theta_{Lim}\right|\right) \text{for} \pm \theta_{Tgt}$$

$$D_Y = D_Y \times \cos\left(\left|\theta_{Tgt}\right| - \left|\theta_{Lim}\right|\right)$$

$$D_Z = 0.0$$

where

D_n	-	direction vector component in local nth direction
θ_{Tgt}	-	target azimuth angle (rad)
θ_{Lim}	-	azimuth limit angle (rad).

Similarly, if the target elevation exceeds the elevation limit, the local direction vector is adjusted so the elevation of the target is at the elevation limit from the direction vector:

$$D_X = D_X \times \cos\left(\left|\varphi_{Tgt}\right| - \left|\varphi_{Lim}\right|\right)$$

$$D_Y = D_Y \times \cos\left(\left|\varphi_{Tgt}\right| - \left|\varphi_{Lim}\right|\right)$$

$$D_Z = \pm \sin\left(\left|\varphi_{Tgt}\right| - \left|\varphi_{Lim}\right|\right) \text{for} \pm \varphi_{Tgt}$$

where

D_n	-	direction vector component in local nth direction
φ_{Tgt}	-	target elevation angle (rad)
φ_{Lim}	-	elevation limit angle (rad).

Once the direction vector in the local frame is computed, it is rotated into ECEF coordinates and becomes the direction-of-flight vector for attacker propagation.

When azimuth and elevation limit adjustments have been completed or are not necessary, the attacker direction vector is adjusted for the last time to account for altitude limits. These adjustments are made using the methodology described in Subsection 5.4.3. The ceiling altitude for altitude limits monitoring is set to the airframe's service altitude limit. The floor altitude is set to 200 m.

Once all direction vector adjustments are completed, the angle between the final direction vector and the aircraft current velocity vector is computed. If the angle is less than 0.8 degrees, the attacker will be flown straight and level toward the target. Otherwise, the attacker will be flown in a curve toward the target direction vector. This process is repeated in steps until the end of the update interval.

5.6.4.2 Bomber Propagation

The Bomber flight mode always uses the truth to navigate, because it is performed relative to targets that are in track. Perception errors will therefore wash out. However, if there is a navigation element, perception is updated according to the equations in Section 5.5.1. If the

aircraft has no navigation element, then perception will be set and updated with the same values as the truth.

The Engage flight mode for bombers is used only if the platform is a bomber and the target is a ground target. The altitude will be adjusted according to the flight method used, and the bomber will be flown for each internal time step within the scenario interval. The bomber's altitude is adjusted to the commanded altitude if terrain-following flight is selected. Otherwise, the altitude is the bomber's current altitude. At each time step, the current distance from bomber to target is updated.

The bomber's velocity is set to the current waypoint speed if the bomber is a flight leader. If the bomber is a wingman, it will use its flight leader's velocity. If the destination waypoint requires terrain following, the terrain following methodology described in Subsection 5.4.3 is used to modify the direction vector. The floor altitude used by the terrain following is set by the commanded AGL altitude.

If terrain following is not necessary, the direction vector is modified to account for Earth curvature by flying the aircraft while maintaining a constant altitude rate. Subsection 5.4.4 explains the constant altitude rate methodology. If the direction vector causes the aircraft to fly above the ceiling altitude or below the floor altitude, it is adjusted, as described in Subsection 5.4.3, for altitude monitoring. The ceiling altitude for altitude monitoring is set to the airframe's service altitude limit. The floor altitude is set to 200 m.

The angle between the bomber's velocity vector and the vector from bomber to target position will determine the flight method. An angle of less than 0.8 degrees causes the bomber to fly straight toward its target; otherwise, the bomber flies in a curve toward its target.

5.6.4.3 FPole Propagation

The FPole flight mode always uses the truth to navigate, because it is performed relative to targets that are in track. Perception errors will therefore wash out. However, if there is a navigation element, perception is updated according to the equations in Section 5.5.1. If the aircraft has no navigation element, then perception will be set and updated with the same values as the truth.

The FPole propagation is used under two circumstances. First, it is used to fly aircraft that have launched a semiactive missile at the target aircraft. Second, it is used to fly an aircraft carrying a laser in a turn-to-target maneuver against the threat. It flies the attacker such that the target is kept at a user-specified azimuth limit at all times to minimize closing rate while keeping the target in track. Elevation adjustments are applied when necessary to keep the target at the user-specified elevation limit as well. For the laser case, the aircraft is held at a constant altitude rather than maintaining the elevation limit.

After the target position has been updated to the current simulation time of the attacker, the command speed of the attacker is set. If its current speed is greater than the corner speed, command speed is set to the current speed. Otherwise, the command speed is set to the corner speed. For the laser case, the command speed is set to the speed of the current waypoint. Next, the azimuth and elevation limits are applied to the direction vector.

First, the body frame coordinate system is established using the attacker's position and the vector from the attacker to the target as the direction vector in the DblECEF2BodyMatrix function described in Appendix B10. This coordinate system is aligned with the local up direction. DblDRotate1 from Appendix B10, is then used to rotate the relative position to the target into this local coordinate system.

The target elevation angle is then computed:

$$\varphi_{Tgt} = \frac{\pi}{2} - \cos -1 \left(Tgt_Z \right)$$

where

φ_{Tgt} - target elevation (rad)
Tgt_Z - local z-axis component of target position.

If the target elevation does not exceed the elevation limit, a local direction vector is computed that points in the direction corresponding to the azimuth limit from the target relative position vector:

$$D_X = \sin (\theta_{Lim})$$

$$D_Y = \cos (\theta_{Lim})$$

$$D_Z = 0.0$$

where

D_n - direction vector component in local nth direction
θ_{Lim} - azimuth limit angle (rad).

This direction vector is then rotated to ECEF coordinates from the local coordinate system defined by the local x-axis and y-axis computed above and the attacker position vector which serves as the local z-axis.

If the target elevation does exceed the elevation limit, the local direction vector is computed so the azimuth and elevation of the target are at the azimuth and elevation limits from the target relative position vector:

$$D_X = \cos (\varphi_{Lim}) \times \sin (\theta_{Lim})$$

$$D_Y = \cos (\varphi_{Lim}) \times \cos (\theta_{Lim})$$

$$D_Z = \pm \sin (\varphi_{Lim}) \text{ for } \pm \varphi_{Tgt}$$

where

D_n - direction vector component in local nth direction
φ_{Lim} - elevation limit angle (rad)

θ_{Lim} - azimuth limit angle (rad)
φ_{Tgt} - target elevation angle (rad).

This direction vector is computed in a different local frame than the one computed above. The x-axis is the same as the local x-axis above, but the y-axis is the target relative position vector and the z-axis is perpendicular to the x-axis and the target relative position vector. The z-axis is computed by:

$$\vec{P}_Z = \left|\vec{P}_X \times \vec{R}el\right|$$

where

$\vec{P}_Z$ - local horizontal plane z-axis (m)
$\vec{P}_X$ - local horizontal plane x-axis (m)
$\vec{R}el$ target relative position vector (m).

The resultant direction vector from the azimuth and elevation adjustments is then rotated into ECEF coordinates.

Since the azimuth limit can be applied in both positive and negative directions around the target relative position vector, a second ECEF direction vector is computed using the local direction vector of:

$$D_{X_2} = -D_X$$

$$D_{Y_2} = D_Y$$

$$D_{Z_2} = D_Z$$

where

D_{n_2} - direction vector component in local nth second direction
D_n - direction vector component in local nth direction.

Commanded speed is then applied to a unit vector in both directions to compute the perspective attacker velocity. For other than lasers, closing rate is computed as the relative velocity difference between the attacker velocities in both directions and the target velocity. The direction vector that results in the smaller closing velocity is used to fly the aircraft. For laser platforms, the velocity vector which will result in the smaller angular change in direction is selected.

When azimuth and elevation limit adjustments have been completed, the attacker direction vector is adjusted for the last time to account for altitude limits. These adjustments are made using the methodology described in Subsection 5.4.3. The ceiling altitude for altitude monitoring is set to the airframe's service altitude limit. The floor altitude is set to 200 m.

Once all direction vector adjustments are completed, the angle between the final direction vector and the aircraft current velocity vector is computed. If the angle is less than 0.8 degrees, the attacker will be flown straight toward the target. Otherwise, the attacker will be flown in a curve toward the target direction vector. This process is repeated in steps until the end of the update interval.

5.6.5 Scramble Flight Mode

The Scramble flight mode is employed when the user assigns an aircraft a Home Base and sets the aircraft to be at Home Base in the Edit/Deploy Platform window. The C3I model can also initiate this action in response to a command center's request to refill a Combat Air Patrol that is leaving its waypoint set. This flight mode expects to handle flights of aircraft.

If the aircraft has a navigation element, the Scramble flight mode will update perception according to the equations in Section 5.5.1. In order for the aircraft to navigate using perception, it is also necessary that the Accumulate Error Only option is not selected, fuzzy vectoring is not being used, and direct scrambling is not being used. If any of these conditions fails, or if there is no navigation element, this flight mode navigates according to truth, and perception is set and updated with the same values as the truth. Note that this means that in the following discussion, whenever it is stated that a perceived value is used, this will correspond with the truth value if the aircraft is navigating according to truth.

For both scripted and nonscripted flights, the Scramble mode flies flight leaders and wingmen differently. Wingmen are flown in the Wingman mode during Scramble. If a flight leader's altitude is over 70% of its destination waypoint's altitude, the flight leader uses its default flight mode. The waypoint can use AGL altitude mode or MSL altitude mode. If the flight leader has reached its last waypoint, its next waypoint will be the first waypoint.

The flight leader climbs to its destination using a climb angle of at least 30 deg. The aircraft will fly at their maximum speed using one half of their maximum gravitational force. For each internal time step within the scenario interval, the aircraft is flown. The angle between the aircraft's perceived velocity vector and the vector to the perceived destination determines the flight method. An angle of less than 0.8 degrees causes the aircraft to fly straight to its destination; otherwise, the aircraft will fly in a curve to its destination.

5.6.6 Drag Flight Mode

The Drag flight mode always uses the truth to navigate, because it is performed relative to targets that are in track. Perception errors will therefore wash out.

However, if the aircraft has a navigation element, the Drag flight mode will update the aircraft’s perception of its own state, using the formulas in Section 5.5.1. If the aircraft has no navigation element, then perception will be set and updated with the same values as the truth.

The Drag flight mode is used to compute an escape direction vector for an aircraft that is being engaged by an attacking aircraft. It allows the user to choose whether the fleeing aircraft should perform a drag or a beam maneuver, and what kind of beam maneuver to perform. It also ensures that the fleeing aircraft flies within an altitude corridor specified by the user.

The C3I model uses the Drag flight mode if an aircraft is reacting to an engagement by a hostile aircraft. The C3I model breaks off the aircraft's current engagement and sends a drag command to FP. The FP model then determines whether the aircraft will fly a drag maneuver or a beam maneuver. Two conditions must be met for a drag maneuver to be executed: the angle threshold and the time-to-attacker threshold.

First, the line-of-sight vector from aircraft to attacker is computed according to:

$$\vec{LOS} = \vec{P}_{Atck} - \vec{P}_{AC}$$

where

$\vec{LOS}$ -line-of-sight to attacker vector (m)
$\vec{P}_{Atck}$ - attacker position vector (m)
$\vec{P}_{AC}$ - aircraft position vector (m).

Next, the cosine of the angle between the line-of-sight vector and the aircraft's velocity vector is computed by:

$$\cos_{\theta} = \left(\vec{U}_{LOS} \bullet \vec{U}_{Vel}\right)$$

where

$\cos_{\theta}$ - cosine of angle between line-of-sight and a/c velocity vectors
$\vec{U}_{LOS}$ - unit line-of-sight vector to attacker
$\vec{U}_{Vel}$ - attacker unit velocity vector.

If the angle between the aircraft velocity vector and the line-of-sight vector is greater than the drag phase angle threshold, the cosine of the angle is less than the cosine of the drag phase angle threshold and the first condition is met.

The range the aircraft can fly during the drag phase time threshold is computed by:

$$R_{Thresh} = V_{Mag} \times T_{Thresh}$$

where

R_{Thresh} - range aircraft flies in drag phase threshold time (m)
V_{Mag} - magnitude of aircraft velocity (m/sec)
T_{Thresh} - drag phase threshold time (sec).

If the time to reach the attacking aircraft is greater than the drag phase time-to-attacker threshold, the magnitude of the line-of-sight vector is greater than the range the aircraft can fly during the time threshold and the second condition is met.

If both of these conditions are met, the aircraft performs a drag maneuver. This is accomplished by the calculation of the drag maneuver direction vector according to:

$$\vec{D}_{Drag} = \vec{P}_{AC} - \vec{P}_{Atck}$$

where

$\vec{D}_{Drag}$ - drag maneuver direction vector (m)
$\vec{P}_{AC}$ - aircraft position vector (m)
$\vec{P}_{Atck}$ - attacker position vector (m).

If neither drag threshold condition is met, the fleeing aircraft executes a beam maneuver. First, a vector pointing in the direction to the right of the aircraft position and perpendicular to both the line-of-sight vector and the up vector at the aircraft is computed:

$$\vec{D}_{Beam} = \left(\vec{LOS} \times \vec{U}_{AC}\right)$$

where

$\vec{D}_{Beam}$ - beam maneuver direction vector if attacker on left (m)
$\vec{LOS}$ - line-of-sight vector (m)
$\vec{U}_{AC}$ - up vector at the aircraft position.

If the attacking aircraft is to the left of the fleeing aircraft, this vector becomes the beam direction vector. Otherwise, the beam direction vector is recomputed in the opposite direction according to:

$$\vec{D}_{Beam} = -\vec{D}_{Beam}$$

where

$\vec{D}_{Beam}$ - beam maneuver direction vector if attacker on right (m).

The user can adjust the beam direction vector by setting the drag phase beam angle to some angle other than 0.0. This angle is measured down from the horizontal plane defined by the beam direction vector. A unit vector in the direction 45 deg down from the horizontal is first computed by:

$$\vec{D}_{45} = \vec{U}_{D_{Beam}} - \vec{U}_{AC}$$

where

$\vec{D}_{45}$ - vector 45 deg down from horizontal plane
$\vec{U}_{D_{Beam}}$ - unit beam maneuver direction vector

$\bar{U}_{AC}$ - up vector at the aircraft position.

Next, this new vector is adjusted, based on the user-input beam angle, according to the following conditions:

If $315\,\text{deg} \le \theta_{Beam} \le 360\,\text{deg}$: $D_{Beam_Z} = D_{Beam_Z} \times [(\theta_{Beam} - 360)/45]$

or if $\theta_{Beam} \ge 225\,\text{deg}$: $D_{Beam_X} = D_{Beam_X} \times [(\theta_{Beam} - 270)/45]$

$$D_{Beam_Z} = -D_{Beam_Z}$$

or if $\theta_{Beam} \ge 135\,\text{deg}$: $D_{Beam_X} = -D_{Beam_X}$

$$D_{Beam_Z} = D_{Beam_Z} \times [(180 - \theta_{Beam})/45]$$

or if $\theta_{Beam} \ge 45\,\text{deg}$: $D_{Beam_X} = D_{Beam_X} \times [(90 - \ _{Beam})/45]$

otherwise: $D_{Beam_Z} = D_{Beam_Z} \times (\ _{Beam}/45)$

where

D_{Beam_n} - component of beam maneuver vector in nth direction

θ_{Beam} - drag phase beam angle (deg).

After the beam angle adjustment, this vector becomes the beam direction vector that is used to propagate the aircraft movement during integration.

Prior to integration, the direction vector is set to either the drag maneuver direction vector or the beam direction vector. Next, if the direction vector causes the aircraft to fly above the ceiling altitude or below the floor altitude, it is adjusted, as described in Subsection 5.4.3, for altitude monitoring. The ceiling and floor altitudes for altitude monitoring are set to the user inputs specified in the Drag phase of the aircraft ruleset. The ceiling altitude is set to the minimum of the airframe's service altitude limit and the ceiling altitude specified for the drag phase.

Once the direction vector is finalized, it is used to fly the aircraft for each internal time step within the scenario interval. If the angle between the aircraft's velocity vector and the direction vector is less than 0.8 degrees, the aircraft will fly straight at maximum velocity along the new direction vector. Otherwise, the aircraft will fly in a curved path at the aircraft's maximum cornering speed in a direction toward the new direction vector.

5.6.7 Return to Base (RTB) Flight Mode

The RTB flight mode is used for aircraft that have run out of weapons, targets, fuel, or time. The RTB Flight mode directs aircraft back to their home airbase, i.e., their first waypoint. The RTB mode expects aircraft to start from a normal flight mode, such as waypoints, rather than a reaction flight mode, such as Engage or Drag. When a platform is scheduled for the RTB mode, it is explicitly put into a normal flight mode.

The RTB mode sets the direction, velocity, force of gravity, and maximum altitude for the aircraft. A direction vector is obtained from the platform's current position to the platform's first waypoint. The velocity of the aircraft is adjusted according to its distance from its first waypoint. RTB aircraft are flown at their cornering speed if they are over 5,000 m from the base. Helicopter corner speed is 75% of maximum speed. Otherwise, the aircraft is flown at the greater rate of 50 m/sec or 30% above minimum velocity. The force of gravity used will be one half of the aircraft's maximum force of gravity. The maximum altitude will be the greater of the aircraft's commanded altitude at its current location or 5,000 m. Helicopters' maximum altitude during RTB is set to 20 m.

The aircraft is flown at each internal time step. The angle between the airbase position and the direction vector is obtained. If the aircraft is less than 15 deg to the airbase's position in the elevation plane and not out of fuel, terrain following will be used to adjust the climb angle of the aircraft. The terrain following methodology is described in Subsection 5.4.3. The floor altitude used by the terrain following is set by the commanded AGL altitude.

If terrain following is not necessary, the direction vector is modified to account for Earth curvature by flying the aircraft while maintaining a constant altitude rate. Subsection 5.4.4 explains the constant altitude rate methodology. If the direction vector causes the aircraft to fly above the ceiling altitude or below the floor altitude, it is adjusted, as described in Subsection 5.4.3, for altitude monitoring. The ceiling altitude for altitude monitoring is set to the airframe's service altitude limit. The floor altitude is set to 200 m.

The angle between the aircraft's velocity vector and the vector to the airbase determines the flight method to reach the climbing point. If the angle is less than 0.8 degrees, the aircraft will fly straight to the climbing point; otherwise, it will fly in a curve to the climbing point. If the distance to the airbase is less than 1,000 m, the aircraft's velocity vector is set to the commanded speed (the greater of 50 m/s or 30% above minimum velocity, as explained above in this section) times each component of the unit velocity vector. The floor altitude is also set to 0m, so that the aircraft can fly as low as necessary to land.

The wingman will follow the Wingman Flight mode procedures until 5 Km from homebase. Its airspeed will be commanded to the maximum of the aircraft approach speed (1.3 Min Speed) and 50 m/sec. If the distance to the airbase is less than 1 Km, the wingman will land at the airbase.

As the distance to the airbase is updated, FP determines if the aircraft has landed. The aircraft has landed at the airbase when the distance to the airbase is less than or equal to the aircraft's turning radius and the dot product of the unit direction and unit velocity vectors is negative.

5.6.8 Avoid Flight Mode

The Avoid flight mode always uses the truth to navigate, because it is performed relative to targets that are in track. Perception errors will therefore wash out.

However, if the aircraft has a navigation element, the Avoid flight mode will update the aircraft's perception of its own state, using the formulas in Section 5.5.1. If the aircraft has no navigation element, then perception will be set and updated with the same values as the truth.

The Avoid mode of flight for aircraft causes the aircraft to maintain a flight path normal to the vector for the aircraft being avoided. This flight mode is currently used by the Wild Weasel, AGAttacker, Fighter, and Flexible Commander rulesets. These rulesets makes use of this flight mode in a reaction to being locked by a SAM system.

The user controls the Avoid maneuver through the ruleset options.. The user can select the Wild Weasel or AGattacker to perform this maneuver during the lock phase. The use of this maneuver during the React-to-SAM-Lock phase is also controlled by the user. The user can specify the duration of the maneuver, as well as the altitude at which the maneuver will be performed. For more details on the ruleset control of this flight mode, see Subsection 4.7.24.

When operating in the Avoid flight mode, the aircraft attempts to maintain a course normal to the platform being avoided (i.e., the aircraft remains at a constant distance from the avoided platform). Within this flight mode, the aircraft may fly terrain following at the altitude commanded by the ruleset. The terrain following methodology is described in Subsection 5.4.3. The floor altitude used by the terrain following is set by the commanded AGL altitude.

If terrain following is not necessary and if the direction of flight causes the aircraft to fly above the ceiling altitude or below the floor altitude, it is adjusted as described in Subsection 5.4.3 for altitude monitoring. The ceiling altitude for altitude monitoring is set to the airframe's service altitude limit. The floor altitude is set to 200 m.

The target vector from the avoiding platform to the avoided platform is first computed. The avoidance vector that is normal to the target vector is next computed as the cross product of the target vector with the avoiding platform's position. This avoidance vector is the vector that defines the direction of flight for the platform. This avoidance vector will cause the platform to fly to the right while avoiding the platform. If the platform to be avoided is to the right of the avoiding platform's current velocity

vector, the avoidance vector undergoes a 180-deg phase shift to cause the platform to fly to the left to avoid the platform.

Once the avoidance vector, i.e., the direction vector, is established, the aircraft is flown using the aircraft flight modeling described in Subsection 5.4.

5.6.9 Vector Flight Mode

This flight mode flies a fighter in a direction rather than on a pursuit course after another platform. The C3I model computes an Intercept Point (IP) for an incoming airborne platform and vectors the fighter by sending to FP the heading, the ground range to the interception point, the altitude, and the speed the fighter should fly. The decision to use the Vector flight mode is made by C3I based on the fighter's ruleset. This flight mode is only available for aircraft using the fighter ruleset.

5.6.9.1 Vector Flight Mode Initialization

After making the decision to use the vector flight mode, the C3I model sends to FP the ground range and heading to the IP and the altitude and speed the fighter is to fly. This information is used to compute the intercept point including the ground range, heading and altitude. For the laser case, C3I sends to FP the position of the target centroid, so that the intercept point does not have to be computed from the heading and ground range. The calculation of the IP varies depending on the underlying Earth model.

5.6.9.2 Vector Flight Mode Platform Update

If the aircraft has a navigation element, the Vector flight mode will update perception according to the equations in Section 5.5.1. In order to navigate according to perception, it is also necessary that the Accumulate Error Only option is not selected, the laser target centroid maneuver is not being performed, and the target to which the aircraft is vectoring is not in track. If any of these conditions fails or if there is no navigation element, this flight mode navigates according to truth, and perception will be set and updated with the same values as the truth. Note that this means that in the following discussion, whenever it is stated that a perceived value is used, this will correspond with the truth value if the aircraft is flying according to the truth.

Platform movement for the vector flight mode is performed in steps up to the end of the update interval. At the beginning of the update interval, command speed V_{Cmd} is initialized to the desired speed sent from the C3I model. The perceived altitude H_P of the aircraft is computed using DblECEFtoAlt described in Appendix B10. The aircraft is flown at a constant altitude.

Next, the desired direction of flight vector is calculated. First the direction vector is computed using the methodology described in section 5.5.1 with the intercept point position as the destination. Then, the cross product of that direction vector to the intercept point position and the perceived up direction at the aircraft position is found:

$$\vec{C} = I\vec{P} \times \vec{U}_p$$

where

$\vec{C}$ - Cross product of $I\vec{P}$ and $\vec{U}_p$

$\vec{U}_p$ - Perceived up direction at the aircraft position

$I\vec{P}$ - Perceived vector from the aircraft to the intercept point position.

Then, the perceived up direction at the aircraft position is crossed with the resultant vector:

$$Di\vec{r} = \vec{U}_p \times \vec{C}$$

where

$Di\vec{r}$ - New direction vector

$\vec{C}$ - Cross product of $I\vec{P}$ and $\vec{U}_p$

$\vec{U}_p$ - Perceived up direction at the aircraft position.

The resultant vector is normal to the perceived up direction at the aircraft position and consequently is normal to the surface of the Earth. It also points in the direction of the intercept point. This is the desired perceived direction vector. If terrain following or an altitude adjustment is required, the terrain following methodology described in Subsection 5.5.3 is used to modify the direction vector. The floor altitude used by the terrain following is set by the commanded altitude. Due to the nature of the vector flight mode maneuver, the perceived altitude maintained will be measured MSL rather than AGL.

If terrain following or altitude adjustment is not necessary, the perceived direction vector is modified to account for Earth curvature by flying the aircraft while maintaining a constant altitude rate. Subsection 5.5.4 explains the constant altitude rate methodology. If the perceived direction vector causes the aircraft to fly above the ceiling altitude or below the floor altitude, it is adjusted as described in Subsection 5.5.3 for altitude monitoring. The ceiling altitude for altitude monitoring is set to the airframe's service altitude limit. The floor altitude is set to 200 m.

Once terrain following, altitude adjustment, and constant altitude rate have been checked and the perceived direction vector modified accordingly, the truth direction vector is calculated according to the translation methodology laid down toward the end of Section 5.5.1.

Next, the aircraft climbs or descends to the commanded intercept altitude while turning and proceeding toward the target. The climb/descent angle is limited to give a reasonable climb/descent. Once the fighter perceives that it has reached the commanded altitude, it flies to the intercept point at that altitude.

Another property of the vector flight mode is that the fighter will fly to its perceived intercept point and then fly beyond the point to a reattack range specified in the Vector Phase of the fighter ruleset. This allows the fighter to search for the attacker target for some distance after passing the perceived intercept point. Consequently, the fighter must still fly along the current direction of flight after passing the intercept point until the reattack range has been surpassed. Therefore, once the fighter passes the perceived IP and the computed direction vector is in the opposite direction of the fighter direction of flight, the negative of the direction vector is used to guide the fighter.

If an aircraft located within a Missile Engagement Zone (MEZ) is being vectored at a heading to intercept a target aircraft, it will follow the associated Low-Level Transit Route (LLTR), if any, to exit the MEZ before flying to the perceived IP point. Once out of the MEZ, it will transition to the Vector flight mode and fly to the perceived IP as described above.

5.6.10 Maneuver Flight Mode

The maneuver flight mode can be performed while an aircraft is executing a defensive maneuver to avoid a SAM Lock or SAM Launch or via a User Rules response. A maneuver is composed of multiple user-defined segments. The maneuver flight mode executes each of these segments and then returns to the default flight mode when the last segment is completed. Each segment lasts until one of the specified termination values has been reached, depending on the type of segment.

The aircraft is flown at internal time steps. At each time step, a check is performed to see if any of the segment's termination conditions are satisfied. If so, the aircraft transitions to the next segment and performs that action.

If the aircraft has a navigation element, the Maneuver flight mode will update perception as described in Subsection 5.5.1. If the Apply Error to Navigation option is selected for the navigation element, then the aircraft navigates according to perception; otherwise it navigates according to the truth and perception is set and updated with the same values as the truth. Note that this means that in the following discussion, whenever it is stated that a perceived value is used, this will correspond with the truth value if the aircraft is flying according to the truth.

If the Define Direction of Flight option is selected, then a vector in the desired direction of flight is computed as described in Subsection 5.5.1. The destination position is computed based on the selected maneuver segment execution values. If the Compute Direction of Flight option is selected then the direction vector is computed as described in Subsection 0. Once a perceived direction vector has been computed according to the

chosen maneuver segment definition, the angle between the aircraft's current perceived direction and this new perceived direction is then found. If the angle is less than 0.8 degrees, the aircraft will fly straight; otherwise, it will fly in a curve.

When the maneuver has been completed, a check is made to determine if any waypoints are perceived to have been passed and should therefore be skipped. Only route type waypoints will be skipped. If the current simulation time has exceeded the off time of a waypoint, it will be skipped. If the simulation time does not exceed the waypoint off time, but the off time will be exceeded before the aircraft can reach the waypoint, then that waypoint will be skipped.

Maneuver elements are defined by the user in Scenario Generation. Maneuvers are limited by the user defined floor and ceiling altitudes. The floor altitude is the minimum AGL altitude at which the aircraft is allowed to fly during the maneuver. The ceiling altitude is the minimum of either the airframe's service altitude limit or the maximum MSL altitude at which the aircraft is allowed to fly during the maneuver.

5.6.10.1 Maneuver Segments

Each maneuver segment type has different maneuver parameters available that can be used to define how the aircraft flies during the segment and when to transition to the next segment.

5.6.10.1.1 Direction of Flight

There are two options available for determining the desired direction of flight during each maneuver segment. If the Compute Direction of Flight option is selected then the direction of flight is computed based on the Roll, Roll Rate, Max Turn G, and Bank Angle execution values. This option allows the flight dynamics associated with a turn to be defined resulting in the direction of flight as the final outcome of the segment as opposed to directly defining the desired direction of flight. The Roll and Roll Rate are used in conjunction with the Max Turn G and Bank Angle to determine the turn plane and turn radius as described in section 5.6.10.1.2.1.2. The turn plane and turn radius determine the direction of flight that results from the turn. The bank angle and the angle between the local zero roll plane and turn plane about the longitudinal axis are computed as described in section 5.6.10.1.2.1.2. These parameters are then used to compute a destination vector in the turn plane orthogonal to the current velocity vector in body frame coordinates. If the Max Turn G is positive then the destination vector is computed as follows.

$$\vec{D} = \left\langle \cos\left(\phi_{tp}\right) \quad 0 \quad \sin\left(\phi_{tp}\right) \right\rangle$$

where

$\vec{D}$ - Destination vector in Body Frame coordinates (m)

ϕ_{tp} - Turn plane angle (deg)

If the Max Turn G is negative then the destination vector is computed as follows.

$$\vec{D} = \langle \sin(\phi_{tp}) \quad 0 \quad \cos(\phi_{tp}) \rangle$$

where

$\vec{D}$ - Destination vector in Body Frame coordinates (m)
ϕ_{tp} - Turn plane angle (deg)

The destination vector is transformed from Body Frame coordinates to ECEF coordinates using the ECEF2Body and Rotate2 functions described in Subsections B10.2.4 and B10.3.14 respectively. The destination vector, turn plane, and turn radius are then used to fly the aircraft in a curved flight path as described in Subsection 5.5.3.2. The resulting aircraft position is computed at the end of each integration interval until a termination value is satisfied. When the segment terminates, the aircraft continues to fly at the resulting flight path angle, heading, and roll. The Compute Direction of Flight option is only available for the Custom segment type. Carefully select the termination values used with the Compute Direction of Flight option because it is possible to setup segments which never meet the termination conditions.

The velocity at the start of the maneuver segment lies in the turn plane. Therefore, the turn is made with the turn plane oriented at the flight path angle of the aircraft at the start of the segment. Two maneuver segments can be used to define a maneuver segment using the Compute Direction of Flight option relative to a given FPA. Define the first maneuver segment using Define Direction of Flight that orients the aircraft at the desired FPA. If the maneuver segment that follows uses the Compute Direction of Flight option, then it will initiate from the FPA achieved in the preceding segment. A single custom maneuver segment can be used with the Compute Direction of Flight option to achieve a particular FPA using the FPA termination value as described in Subsection 5.6.10.1.2.2.2

If the Define Direction of Flight option is selected then the Flight Path Angle, Heading Angle, or Altitude Execution Values can be used to define the desired final direction of flight for this maneuver segment and Roll, Turn G, and Bank Angle execution values are used to compute and limit the G's used during the turn to point the aircraft in that direction.

5.6.10.1.2 Maneuver Parameters

For each maneuver segment, maneuver parameters describe how the maneuver segment is flown and when it is complete. A maneuver parameter can be defined as either an execution value or a termination value, or as both. Each maneuver parameter is defined by a random number distribution type. A random draw is made from each

distribution at the beginning of the maneuver segment to define the parameter used throughout the segment. This statistical representation can be used to create variability between aircraft maneuvers to simulate minor pilot adjustments. The mean value of each distribution is displayed in each maneuver parameter field. If randomness has been eliminated from the scenario, the mean value of each maneuver parameter is used.

5.6.10.1.2.1 Execution Values

The selected maneuver parameter execution values are applied to the aircraft while it is performing the current maneuver segment. These parameters can be used to define the desired final outcome of the maneuver segment or limit the aerodynamic flight of the aircraft during the segment. Only some execution values are applied as limits and they are only applied as limits for certain cases. The details of when and how these are applied as limits are discussed in each execution value subsection. If Define Direction of Flight is enabled then the Roll, Max Turn G, and Max Bank Angle are used to compute the G's pulled during any turns used to achieve the desired direction of flight. If Define Direction of Flight is enabled and the Roll, Max Turn G, and Max Bank Angle are not selected as execution parameters, then a max G turn is performed in order to achieve the desired direction of flight.

5.6.10.1.2.1.1 Roll Rate

Roll Rate is applied as the rate of change of roll which the aircraft can achieve during this maneuver segment. If the Define Direction of Flight option is selected and the maneuver requires the aircraft to roll, then the specified Roll Rate will be utilized to achieve the required roll. If the Compute Direction of Flight option is selected then the Roll Rate can be used with the Max Turn G or the Max Bank Angle to define the turn plane and the radius of the turn without defining the Roll. The turn plane and turn radius determine the direction of flight that results from the turn. In this case the Roll Rate is applied to compute the resulting roll at the end of each integration interval. A positive roll rate will cause a clockwise roll from the pilot's perspective in order to achieve the desired roll. A negative roll rate will cause a counterclockwise roll from the pilot's perspective in order to achieve the desired roll.

$$\phi_r = R_r \cdot \Delta t$$

where

ϕ_r	-	Roll at end of interval (deg)
R_r	-	Roll rate (deg/s)
Δt	-	Integration interval (s)

The resulting roll is then used in conjunction with either the defined Max Turn G or the Max Turn G computed from the Max Bank Angle to compute the turn plane and the turn radius as described in Subsection 5.6.10.1.2.1.2.

5.6.10.1.2.1.2 Roll

Roll is defined as the angle about the lateral axis of the aircraft with zero roll defined as having the aircraft wings level in the lateral direction with the top of the aircraft pointing away from the ground. The aircraft may be pitched up or down, therefore the wings may or may not be level in the longitudinal direction. The wings of the aircraft with zero roll define the zero roll plane discussed below. The zero-roll plane remains level in the lateral direction but remains aligned with the pitch and azimuth of the aircraft. Note that the longitudinal axis of the aircraft is aligned with the velocity vector and that positive roll is defined as a clockwise rotation from the pilot's perspective. If Compute Direction of Flight is selected and Roll is selected as an Execution Value then the Roll is used in conjunction with either the Max Turn G or the Bank Angle to determine the orientation of the turn plane and the turn radius, which define the direction of flight at the end of each integration interval until a termination value is satisfied. The Max Turn G is computed as described in Subsection 5.6.10.1.2.1.3. The Max Turn G is used to compute the bank angle relative to the turn plane. If the Max Turn G is positive then the bank angle is computed as follows.

$$\phi_{bank} = \tan^{-1}\left(G_{MaxTurn}\right)$$

where

ϕ_{bank}	-	Bank angle (deg)
$G_{MaxTurn}$	-	Max Turn G

If the Roll Rate execution value is also selected then the roll achieved at each integration interval is computed as described in Subsection 5.6.10.1.2.1.1. If the Max Turn G is negative then a nose down turn is being performed therefore a scale factor is applied to the bank angle calculation.

$$\phi_{bank} = \frac{R_{rp}}{90}\tan^{-1}\left(G_{MaxTurn}\right)$$

where

ϕ_{bank}	-	Bank angle (deg)
R_{rp}	-	Roll relative to roll plane (deg)
$G_{MaxTurn}$	-	Max Turn G

If the Bank Angle execution value is selected and if the magnitude of the Bank Angle is less than the magnitude of the computed bank angle then the Bank Angle execution value is used instead of the computed bank angle. Then, the bank angle and roll are used to compute the angle between the local zero roll plane and turn plane about the longitudinal axis.

$$\theta_t = R_{rp} - \phi_{bank}$$

where

θ_t	-	Angle between the zero roll plane and turn plane about the longitudinal axis (deg)
R_{rp}	-	Roll relative to roll plane (deg)
ϕ_{bank}	-	Bank angle – angle between aircraft wing plane and turn plane (deg)

The velocity and Max Turn G are then used to compute the turn radius.

$$R_{turn} = \frac{V^2}{G_{MaxTurn} \cdot g}$$

where

R_{turn}	-	Turn radius (m)
V	-	Instantaneous velocity at current integration interval (m/s)
$G_{MaxTurn}$	-	Max Turn G
g	-	The acceleration due to gravity (9.80665 m/s^2)

If Define Direction of Flight is enabled and Roll is selected as an Execution Value, then the Roll is used to limit the amount of roll that is achieved during a turn in this maneuver segment and hence limit the G's available for turns during this maneuver segment. The Rotation angle between zero roll plane and turn plane about the longitudinal axis of the aircraft is computed by first converting the direction vector to body frame coordinates using the ECEF2BodyMatrix function described in Subsection B10.2.6. Then the angle is computed from the Y versor vector.

$$\theta_t = \cos^{-1}\left(\frac{\vec{A} \bullet \vec{B}}{\left|\vec{A}\right|\left|\vec{B}\right|}\right)$$

where

θ_t	-	Rotation angle between zero roll plane and turn plane about the longitudinal axis of the aircraft with positive in direction of right wing down (deg)
$\vec{A}$	-	Body frame Y versor vector [0, 1, 0]
$\vec{B}$	-	Direction vector in body frame coordinates

The roll is used to compute the max bank angle limit. In this case the turn plane is defined by the current direction of flight and the desired direction of flight.

$$\phi_{bank} = R_{rp} - \theta_t$$

where

ϕ_{bank} - Bank angle relative to the turn plane (deg)
R_{rp} - Roll execution value (deg)
θ_t - Rotation angle between zero roll plane and turn plane about the longitudinal axis of the aircraft with positive in direction of right wing down (deg)

Then the max bank angle limit is used to compute the max G defined by the roll. The G defined by the roll is computed as follows.

$$G_r = \tan(R_{tp})$$

where

R_{tp} - Roll relative to turn plane or bank angle (deg)
G_r - Max G defined by the roll

The actual G used for the turn is set to the minimum of the G defined by the specified roll, the Max Turn G defined for the segment, and the max G available computed from the airframe.

5.6.10.1.2.1.3 Max Turn G

If Compute Direction of Flight is enabled and Max Turn G is selected as an Execution Value then the minimum of the Max Turn G, airframe available G, and the G computed from the Bank Angle is used in conjunction with the Roll and Roll Rate to determine the turn plane and radius which define the final direction of flight at the end of each integration interval until a termination value is satisfied.

If Define Direction of Flight is enabled and Max Turn G is selected as an Execution Value then the minimum of the Max Turn G, the turn G computed from the Bank Angle, the airframe available G, and the turn G computed from the Roll is used to define the amount of G's that can be achieved to perform a turn during this maneuver segment.

The Max Turn G can be defined as positive or negative. Positive Max Turn G causes the aircraft to use a nose up to execute a turn or change altitude while a negative Max Turn G causes the aircraft to use a nose down to execute a turn or change altitude.

Max Turn G Mode defines how the Max Turn G is specified for the segment. The available options are Absolute, %MIL, and %AB. The Absolute Max Turn G mode is defined as the ratio of the total acceleration experienced by the aircraft and the acceleration due to gravity at sea level. The %MIL mode allows specification of the Max

Turn G as a percentage of the available MIL G computed from the airframe MIL G available table for HIFI airframes or from the Max G airframe parameter for non-HIFI airframes. The %AB mode allows specification of the Max Turn G as a percentage of available MAX AB G computed from the airframe MAX AB G available table for HIFI or from the Max G airframe parameter for non-HIFI airframes.

5.6.10.1.2.1.4 Max Bank Angle

Bank angle is defined as the angle about the longitudinal axis of the aircraft relative to the turn plane of the aircraft. If Compute Direction of Flight is enabled and Max Bank Angle is selected as an Execution Value then the minimum of the Max Bank Angle and the bank angle computed from the Max Turn G is used in conjunction with the Roll and Roll Rate parameters to determine the turn plane and turn radius which define the direction of flight at the end of each integration interval until a termination value is satisfied.

If Define Direction of Flight is enabled and Max Bank Angle is selected as an Execution Value then the minimum of the Max Turn G, the turn G computed from the Bank Angle, the airframe available G, and the turn G computed from the Roll is used to define the amount of G's that can be achieved to perform a turn during this maneuver segment.

5.6.10.1.2.1.5 Flight Path Angle

If the Define Direction of Flight option is selected and the Execution Value option is selected for Flight Path Angle (FPA) but not for Altitude then the user defined FPA defines the desired direction of flight. A positive value represents a FPA in the up direction and a negative value represents a FPA in the down direction. The Heading Angle Execution Value can be used in conjunction with the FPA to define the desired direction of flight. If the Altitude Execution Value is enabled then the FPA input limits the flight path angle that can be flown to achieve the desired altitude to the range [-FPA, FPA]. The altitude is achieved using the largest magnitude FPA that the aircraft is capable of achieving.

There are three options for defining the FPA. In Absolute FPA mode, the FPA is defined as the angle between the velocity of the aircraft and the local horizontal plane. This is also the angle between the nose of the aircraft and the local horizontal since the orientation is aligned with the velocity vector for all maneuver segment types except for TSPI. This mode can be used to define a desired pitch when using a yaw, pitch, roll absolute coordinate frame. In the Relative FPA mode, the FPA is defined as an angle in the vertical direction relative to the FPA at the start of the segment. In the Relative LOS FPA mode, the FPA is defined as the desired angle between the line of sight (LOS) between the aircraft and the target to which the maneuver is relative and the aircraft's velocity vector. For example, this mode can be used to accomplish a drag away maneuver segment where the aircraft drags directly away from the target by setting the FPA to 180

degrees and setting the Heading Angle execution parameter to Relative LOS mode and 180 degrees. If the Altitude execution value is selected then the only mode available is the Absolute mode.

5.6.10.1.2.1.6 Heading Angle

Heading angle defines the heading to which the aircraft is to turn during the maneuver segment. The actual heading angle for the segment is determined by the specified value and the selected angle mode. If selected as an execution parameter, the aircraft will turn toward the specified heading until the angle is achieved or until the termination condition is reached. If the heading angle is achieved before a termination condition is met then the aircraft will continue to fly at that heading until a termination condition is met. If a termination condition is met before the heading angle is met then the current segment terminates.

There are four ways to define how the heading angle is specified for the segment. The available options are Absolute, Relative, Relative LOS, and Absolute LOS. An absolute heading is defined relative to North. A relative heading is defined relative to the aircraft's heading at the initiation of the maneuver segment. Both LOS heading modes are defined relative to the current line of sight vector between the aircraft and its target/attacker. When the Relative LOS mode is selected, the aircraft will turn in the direction that requires the smallest angle of turn to achieve the heading angle relative to the target. When the Absolute LOS mode is selected, the aircraft will fly directly to the angle specified relative to the current heading to the target. To specify an angle to the right of the target use a positive value, to the left of the target use a negative value.

5.6.10.1.2.1.7 Altitude

As an Execution Value, Altitude defines to what altitude the aircraft is to fly during the maneuver segment. The actual altitude desired for the segment is determined by the specified value and the selected altitude mode. If selected as an execution parameter, the aircraft will climb or dive to the altitude specified as quickly as possible limited by the FPA and maintain that altitude until the segment is terminated. Section 5.6.10.1.2.1.5 describes how the FPA is used to limit the aircraft's climb and descent.

Altitude mode defines how the altitude is specified for the maneuver segment. The available options for this field are 'Rel', 'AGL', and 'MSL'. Relative altitudes are specified as plus or minus the aircraft's altitude at the start of the maneuver segment. If the AGL option is selected, then the aircraft will fly terrain following. If the MSL option is selected, then the aircraft is flown to the specified MSL altitude without terrain following.

5.6.10.1.2.1.8 Speed

Defines what commanded speed to use while the aircraft is flying the current maneuver segment. The actual speed desired for the segment is determined by the specified value and the selected speed mode. If selected as an execution parameter, the aircraft will attempt to fly at the speed specified during the maneuver segment and maintain that speed until the segment is terminated.

Speed mode defines how the speed is specified for the maneuver segment. The available options are Absolute, Relative, % MIL Throttle Setting, % AB Throttle Setting, Target Multiplier, and Platform Multiplier. Relative speeds are specified as plus or minus the aircraft's speed at the start of the maneuver segment. Throttle Setting speeds are specified as a percentage of the thrust, AB thrust, or max thrust computed from the Hi-Fidelity thrust and AB thrust tables. For aircraft, the thrust is used to compute the acceleration which determines the speed. If the %AB option is used and the afterburners are not available for this airframe, then the maximum MIL thrust is used. For helicopters, if the %MIL option is selected, then the maximum thrust is computed as follows.

$$T_{Max} = \frac{1}{6} \cdot N_B \cdot C \cdot \rho \cdot V_T^{\,2} \cdot R_B \cdot C_L \cdot \left(\frac{P_{MIL}}{100} \right)$$

where

T_{max} - maximum producible rotor thrust (N)
N_B - number of rotor blades
C - blade chord (m)
ρ - air density (kg/m^3)
V_T - tip speed (m/sec)
R_B - blade radius (m)
C_L - blade coefficient of lift.
P_{MIL} - Percent MIL thrust (none)

If the %AB speed mode setting is selected for helicopters then the maximum thrust is computed using 100% or full Mil. If using Target Multiplier mode, the value specified is multiplied by the target's speed at the start of the maneuver segment to obtain the desired speed. This speed mode option should only be used for maneuvers adopted relative to a target through the User Rules. If using Platform Multiplier mode, the value specified is multiplied by the aircraft's speed at the start of the maneuver segment to obtain the desired speed.

5.6.10.1.2.2 Termination Values

The selected maneuver termination values determine the conditions for the aircraft to terminate the current maneuver segment when the specified values are reached. If multiple termination parameters are selected, the first termination criterion met will cause the transition to the next maneuver segment. The Duration termination value is monitored continuously to allow transition to the next segment at the exact time that the condition is met. All other termination values transition at the integration

interval. If this is the last segment for this maneuver then the aircraft will transition back to default flight mode, or, if the maneuver was initiated from the User Rules, then the aircraft will transition back to ruleset control at the end of the current scenario interval. In this case, the aircraft is flown using the maneuver flight mode until the end of the current scenario interval.

5.6.10.1.2.2.1 Roll

If Roll is selected as a Termination Value, the absolute value of the roll of the aircraft reaching or exceeding the user defined Roll threshold will terminate the maneuver segment. Roll is defined as the angle about the lateral axis of the aircraft with zero roll defined as having the aircraft wings level in the lateral direction with the top of the aircraft pointing away from the ground.

5.6.10.1.2.2.2 Flight Path Angle

If the Flight Path Angle Termination Value option is selected, then the maneuver segment terminates when the flight path angle of the aircraft reaches or crosses the specified parameter. If the FPA at the beginning of the maneuver segment is greater than the specified FPA, then the segment terminates when the FPA is less than or equal to the specified FPA. If the FPA at the beginning of the maneuver segment is less than the specified FPA, then the segment terminates when the FPA is greater than or equal to the specified FPA.

There are three options for defining the FPA. In Absolute FPA mode, the FPA is defined as the angle between the velocity of the aircraft and the local horizontal plane. This is also the angle between the nose of the aircraft and the local horizontal since the orientation is aligned with the velocity vector for all maneuver segment types except for TSPI. This mode can be used to define a desired pitch when using a yaw, pitch, and roll absolute coordinate frame. In the Relative FPA mode, the FPA is defined as an angle in the vertical direction relative to the FPA at the start of the segment. In the Relative LOS FPA mode, the FPA is defined as the desired angle between the line of sight (LOS) between the LOS from the aircraft to the target and the aircraft's velocity vector. If the Altitude execution value is selected, then the only mode available is the Absolute mode.

5.6.10.1.2.2.3 Heading Angle

If selected as a termination parameter, the segment will end when the aircraft reaches or crosses the specified heading angle. If the heading angle at the beginning of the maneuver segment is greater than the specified heading angle, then the segment terminates when the heading angle is less than or equal to the specified heading angle. If the heading angle at the beginning of the maneuver segment is less than the specified heading angle, then the segment terminates when the heading angle is greater than or equal to the specified heading angle.

There are four ways to define how the heading angle is specified for the segment. The available options are Absolute, Relative, Relative LOS, and Absolute LOS. An absolute heading is defined relative to North. A relative heading is defined relative to the aircraft's heading at the initiation of the maneuver segment. Both LOS heading modes are defined relative to the current line of sight vector between the aircraft and its target/attacker. When the Relative LOS mode is selected, the aircraft will turn in the direction that requires the smallest angle of turn to achieve the heading angle relative to the target. When the Absolute LOS mode is selected, the aircraft will fly directly to the angle specified. Positive values specify an angle to the right of the target, while negative values specify an angle to the left of the target.

5.6.10.1.2.2.4 Altitude

If selected as a termination parameter, the segment will end once the aircraft crosses the altitude specified. If the altitude at the beginning of the maneuver segment is greater than the specified altitude, then the segment terminates when the altitude is less than or equal to the specified altitude. If the altitude at the beginning of the maneuver segment is less than the specified altitude, then the segment terminates when the altitude is greater than or equal to the specified altitude.

Altitude mode defines how the altitude is specified for the maneuver segment. The available options for this field are 'Rel', 'AGL', and 'MSL'. Relative altitudes are specified as plus or minus the aircraft's altitude at the start of the maneuver segment. If the AGL option is selected, then the aircraft will fly terrain following. If the MSL option is selected, then the aircraft is flown to the specified MSL altitude without terrain following.

5.6.10.1.2.2.5 Speed

If selected as a termination parameter, the segment will end once the aircraft has reached or crossed the speed specified. If the speed at the beginning of the maneuver segment is greater than the specified speed, then the segment terminates when the speed is less than or equal to the specified speed. If the speed at the beginning of the maneuver segment is less than the specified speed, then the segment terminates when the altitude is greater than or equal to the specified speed.

Speed mode defines how the speed is specified for the maneuver segment. The available options are Absolute, Relative, % MIL Throttle Setting, % AB Throttle Setting, Target Multiplier, and Platform Multiplier. Relative speeds are specified as +/- the aircraft's speed at the start of the maneuver segment. Throttle Setting speeds are specified as a percentage of the MIL thrust, AB thrust, or max thrust computed from the Hi-Fidelity thrust and AB thrust tables. For aircraft, the thrust is used to compute the acceleration which determines the speed. If the %AB option is used and the afterburners are not available for this airframe then the maximum MIL thrust is used. For helicopters, if the %MIL option is selected then the maximum thrust is computed as follows.

$$T_{Max} = \frac{1}{6} \cdot N_B \cdot C \cdot \rho \cdot V_T^{\,2} \cdot R_B \cdot C_L \cdot \left(\frac{P_{MIL}}{100} \right)$$

where

T_{max} - maximum producible rotor thrust (N)
N_B - number of rotor blades
C - blade chord (m)
ρ - air density (kg/m^3)
V_T - tip speed (m/sec)
R_B - blade radius (m)
C_L - blade coefficient of lift.
P_{MIL} - Percent MIL thrust (none)

If the %AB speed mode setting is selected for helicopters then the maximum thrust is computed using 100% Mil. If using Target Multiplier mode, the value specified is multiplied by the target's speed at each integration interval to obtain the desired speed. This speed mode option should only be used for maneuvers adopted relative to a target through the User Rules. If using Platform Multiplier mode, the value specified is multiplied by the aircraft's speed at the start of the maneuver segment to obtain the desired speed.

5.6.10.1.2.2.6 Duration

Duration defines how long the current maneuver segment should be performed. If selected as a termination parameter, the aircraft will transition to the next maneuver segment in the maneuver element once the maneuver segment has been executed for the specified amount of time. The transition to another segment occurs at the exact time defined by duration regardless of the integration step size. If this is the last segment for this maneuver, then the aircraft will transition back to default flight mode, or, if the maneuver was initiated from the User Rules, then the aircraft will transition back to ruleset control at the end of the current scenario interval. In this case, the aircraft is flown using the maneuver flight mode until the end of the current scenario interval.

5.6.10.1.2.2.7 Min Vertical Velocity

When selected, the segment will end when the vertical component of the aircraft's velocity vector falls below the specified value. The vertical component of the aircraft's velocity is computed using the scalar projection of velocity onto a local up vector at the aircraft's position. The DblECEFtoUp routine described in Subsection B10.2.21 is used to compute the local up vector.

$$V_v = \frac{\vec{U} \bullet \vec{V}}{\left| \vec{U} \right|}$$

where

V_v - Vertical component of velocity
$\vec{U}$ - Local up vector in ECEF coordinates
$\vec{V}$ - Velocity vector in ECEF coordinates

5.6.10.1.2.2.8 Max Vertical Velocity

When selected, the segment will end when the vertical component of the aircraft's velocity vector exceeds the specified value. The vertical velocity is computed as described in section 5.6.10.1.2.2.7.

5.6.10.1.2.2.9 Min Available G

When selected, the segment will end when the aircraft's available G falls below the specified value.

5.6.10.1.2.2.10 Max Available G

When selected, the segment will end when the aircraft's available G meets or exceeds the specified value.

5.6.10.1.3 Maneuver Segment Types

A maneuver is created using one or more of the pre-defined maneuver segment types or a Custom segment type. More than one segment can be used to define a maneuver. An example maneuver with just one segment is to Drag Away from the target for 30 seconds. An example of a maneuver with more than one segment is to Turn to Relative Angle and Fly Straight and Level for 30 seconds. Only the valid combinations of maneuver parameters are available for each pre-defined maneuver segment type. The speed execution value is available for all segment types. If Speed is not selected as an execution value, then the aircraft's speed at the start of the segment will be maintained. Speed is used to set the commanded speed during each segment. The aircraft attempts to achieve the commanded speed using the aerodynamic parameters as described in Subsection 5.5.2.

5.6.10.1.3.1 Change Altitude Maneuver Segment

The Change Altitude maneuver segment can be used to specify a new altitude for the aircraft. The available execution values that apply are altitude, altitude mode, speed, speed mode, max turn G, Max Bank Angle, and FPA. The steepness of the climb or dive to the new altitude is limited to [-FPA, FPA]. The max turn G limits the turn used to achieve the FPA. The FPA allows the aircraft to achieve its new altitude as fast as possible given the specified elevation constraint. Altitude and duration are the termination values for the change altitude maneuver segment. The segment is flown until either the aircraft perceives that altitude has been reached or the segment Duration has been exceeded.

5.6.10.1.3.2 Drag Away Maneuver Segment

The Drag Away maneuver segment defines a type of maneuver which flies the aircraft directly away from the perceived location of the platform causing the reaction. If

a drag maneuver is to be performed, the direction vector is calculated by using the methodology described in Subsection 5.5.1 with the attacker's position as the destination. The negative of the direction vector is then computed. The calculations are similar to those in the discussion of the drag maneuver in Subsection 5.6.6. The execution values available for this maneuver are the Roll Rate, Roll, Max Turn G, Max Bank Angle, and the Speed at which the aircraft will attempt to fly during the drag maneuver. Roll Rate, Roll, Max turn G and Max Bank Angle are used as limits on the turn to the desired direction. The Duration of the drag is the only termination value that is used.

5.6.10.1.3.3 Drag Away Level Maneuver Segment

The Drag Away Level maneuver segment is similar to the Drag Away segment, except that the aircraft maintains its current perceived altitude. The segment is flown until the Duration time is reached.

5.6.10.1.3.4 Fly Abeam Maneuver Segment

The Fly Abeam maneuver segment causes the aircraft to fly with the LOS to the perceived location of the platform causing the reaction at a 90 degree heading relative to the nose of the aircraft. The direction vector is calculated as described for a beam maneuver in Methodology Manual Section 5.6. The execution values available for this maneuver are the Roll Rate, Roll, Max Turn G and the Speed of the aircraft. Roll Rate, Roll, and Max turn G are used to limit the turn to the desired direction as described in section 5.6.10.1.2.1.2 for Define Direction of Flight. The segment is flown until the Duration termination value is reached.

5.6.10.1.3.5 Fly Default Mode at New Altitude Maneuver Segment

The Fly Default Mode at New Altitude maneuver segment allows the user to specify a new altitude for the aircraft to fly during default flight mode. The default flight mode is the Waypoint flight mode for flight leaders or the Wingman flight mode for wingmen. Subsections 5.6.1 and 5.6.2 describe the Waypoint and Wingman flight modes respectively. The available execution values for this segment are the Max Turn G, Altitude, and Speed. The Altitude and Speed parameters are commanded values that the flight processing model attempts to achieve. The radius of the vertical turn used to achieve the altitude is limited by the Max Turn G. The segment is flown until the Duration termination value is reached.

5.6.10.1.3.6 Fly Straight and Level Maneuver Segment

The Fly Straight and Level maneuver segment allows the aircraft to fly at the current heading and MSL altitude. The available execution values for this maneuver are the Roll, Roll Rate, Max Turn G, Max Bank Angle, and Speed. The Speed is used to set the commanded speed and the minimum of the Max Turn G, the turn G computed from the Bank Angle, the airframe available G, and the turn G computed from the Roll is used to define the amount of G's that can be achieved during the vertical turn to level, if

required. The segment is flown until the Duration time is reached. If it is desired that a particular roll angle be maintained during the Fly Straight and Level segment, a Roll execution value can be specified, but the segment should be preceded by a Roll segment to get to aircraft to the desired angle.

5.6.10.1.3.7 Roll Maneuver Segment

The Roll maneuver segment allows the aircraft to perform a roll without resulting in a turn. The available execution values for this segment are the Roll Rate and speed. The aircraft rolls at the user-defined Roll Rate and attempts to accelerate to the Speed execution value for the Duration of the segment. The segment transitions to the next segment when the Duration termination value is reached.

5.6.10.1.3.8 Turn to Angle Off of LOS Maneuver Segment

The Turn to Angle Off of Line of Sight (LOS) maneuver segment allows the aircraft to turn to an angle relative to LOS to the perceived location of the target or platform initiating the maneuver response. The available execution values for this segment are Roll Rate, Roll, Max Turn G, Max Bank Angle, Heading Angle, Altitude, and Speed. The Heading Angle and Altitude parameters define the desired direction of flight during this segment. A perceived direction vector is computed from the target location and the perceived location of the aircraft.

The Roll Rate, Roll, Max Turn G, Max Bank Angle, Heading Angle, and Altitude parameters are used to limit the turn used to cause the aircraft to move in that direction. Once the aircraft has reached the Heading Angle termination value, the segment is terminated. To cause the aircraft to continue to fly at the new Heading Angle and Altitude, the Turn To Angle Off of LOS segment can be followed with the Fly Straight and Level segment. The Fly Straight and Level segment allows definition of the Duration of the aircraft flight at the current altitude and heading.

5.6.10.1.3.9 Turn to Beam Maneuver Segment

The Turn to Beam maneuver segment allows the aircraft to turn and fly such that the perceived target location is 90 degrees off nose in the horizontal direction. The available execution values for this segment are Roll Rate, Roll, Max Turn G, Max Bank Angle, and Speed. The Roll Rate, Roll, Max Turn G, and Max Bank Angle are used as limits on the G's used in the turn as described in section 5.6.10.1.2.1.2 for the Define Direction of Flight option. Once the aircraft has reached the new heading angle, the segment will terminate. To cause the aircraft to fly for a specified duration at this altitude or heading, this maneuver segment can be followed by a Flight Straight and Level maneuver segment. The direction vector is calculated as described for a beam maneuver in Subsection 5.6.

5.6.10.1.3.10 Turn to Relative Angle Maneuver Segment

The Turn to Relative Angle maneuver segment allows the aircraft to turn to an angle relative to its orientation at the start of the segment. The execution values for this segment are Roll Rate, Roll, Max Turn G, Max Bank Angle, Heading Angle, and Speed. The Heading Angle defines the relative angle that is desired. The Heading Angle defines the desired direction of flight. The Roll Rate, Roll, Max Turn G, Max Bank Angle, and Heading Angle parameters are used to limit the turn used to cause the aircraft to move in that direction as described in section 5.6.10.1.2.1.2 for the Define Direction of Flight option. Once the aircraft reaches the new heading angle termination value it terminates the segment. To cause the aircraft to fly for a specified duration at this altitude or heading, this maneuver segment can be followed by a Fly Straight and Level maneuver segment.

5.6.10.1.3.11 Custom

The Custom segment type allows any valid combination of maneuver parameters to be used to define a maneuver segment. The definition of each of the execution value maneuver parameters depends on the Direction of Flight setting described in section 0. The custom segment type can be used to define any of the pre-defined segment types except for TSPI and Fly Default Mode at New Altitude. But the additional execution and termination values available allow more flexibility to customize the segment. To define a custom segment type that behaves like another segment type, create a new segment with the desired type and then change the segment type to Custom. This will bring the default options for that segment over to the custom segment. The custom segment type allows the use of the Compute Direction of Flight option described in section 0.

5.6.10.1.3.12 Time Space Position Indicated

Time Space Position Indicated (TSPI) data includes the actual measured position and orientation of an aircraft performing a maneuver. TSPI data can be collected for an

aircraft and imported into EADSIM as a TSPI maneuver segment. The TSPI segment type allows TSPI data to be either imported from a text file or defined as inputs in Scenario Generation. The TSPI maneuver segment provides the specifics of a maneuver initiated relative to the current position and heading of the aircraft. When the segment is executed, the aircraft will snap to the defined flight path instead of using aerodynamic flight to fly to each point. This allows the exact flight path and orientation of the aircraft to be explicitly flown throughout the maneuver segment. The aircraft velocity is computed from the TSPI position and time data.

If the Apply Error to Navigation option is selected for a Navigation Device on the aircraft then the navigation error is accumulated throughout the maneuver segment and the aircraft is flown directly to the TSPI data points. When the segment completes the aircraft navigates according to perception.

5.6.10.1.3.12.1 Time

The time is defined in decimal seconds relative to the initiation of the maneuver. The data points do not have to be evenly spaced in time. In EADSIM, the position of each platform is updated at each integration interval in flight processing. There may not be a TSPI data point defined for each integration time step. Therefore the position and orientation at each integration time step are computed using the flight smoothing algorithm described in section 5.6.10.1.3.12.9.

5.6.10.1.3.12.2 Position and Orientation Coordinates

The TSPI position is defined in a local body coordinate system that remains fixed with the origin located where the aircraft initiated the maneuver segment, thus each position is defined as the offset from the location of the aircraft at initiation of the maneuver segment. This body frame coordinate system is defined as follows:

Z: Local up

Y: Perpendicular to plane formed by Z and Velocity vector

X: Orthogonal to YZ plane

The position is converted from body frame to ECEF coordinates using the DblECEF2BodyMatrix and Rotate2 routines described in Appendix sections B10.2.7 and B10.3.14 respectively.

The orientation is defined by the yaw, pitch, and roll rotation angles. Yaw is defined as the angle about the up vector measured positive clockwise relative to true north, pitch is defined as the angle between the nose of the aircraft and the local horizontal plane, and roll is defined as the angle about the lateral axis of the aircraft relative to level flight. Positive yaw is defined with the nose turning to right, positive

pitch is defined as nose towards up, and positive roll is defined as the right wing moving down. At runtime, the yaw of the first TSPI data point is used to translate the yaw of all other data points based on the aircraft's heading at the start of the segment.

$$\varphi_{TSPI\_trans\_n} = \left(\theta_{start} - \varphi_{TSPI\_1}\right) + \varphi_{TSPI\_n}$$

where

$\varphi_{TSPI\_trans\_n}$	-	Translated Yaw of n^{th} TSPI data point (deg)
θ_{start}	-	Heading of the aircraft at the start of the segment (deg)
φ_{TSPI_1}	-	Yaw of first TSPI data point (deg).
φ_{TSPI_n}	-	Yaw of TSPI n^{th} data point before translation (deg)

This allows the maneuver segment to be initiated from any angle in azimuth and still maintain its pitch and roll profile characteristics.

5.6.10.1.3.12.3 Segment Initialization

If the Smooth Transition option is selected then the transition to this maneuver segment from the default flight mode or previous segment is smoothed. Two segments of type Custom are automatically created to fly the aircraft to the initial orientation and velocity of the TSPI segment. The aircraft starts the TSPI segment when it reaches the orientation and velocity defined by the first two TSPI data points. This may cause a delay between when the segment starts and when the first TSPI point is reached. If the Smooth Transition option is not selected then the aircraft will snap directly to the first TSPI velocity and orientation at the start of the segment. This removes the possibility of a delay but adds the possibility for discontinuities in the aircraft orientation and velocity. If this option is not selected, a custom maneuver segment can be added by the user to precede the TSPI segment in order to allow the aircraft to reach a satisfactory state entry point.

5.6.10.1.3.12.4 Importing TSPI Data

TSPI data can be imported directly from a user-selected raw TSPI data file. User defined file format options allow flexibility in types of TSPI files that can be imported. There are several data format options which allow the raw TSPI data to be defined in different units and coordinate frames. Raw TSPI data is defined using absolute time and absolute position and orientation coordinates. Within the TSPI maneuver segment, TSPI data is defined using relative time and body frame coordinates for position of the aircraft. When the data is imported the time is translated into decimal seconds relative to the start of the maneuver segment and the position is transformed into local body frame coordinates.

EADSIM supports any ASCII text file format where the data is tab, comma, or space delimited. The user defines which column in the file contains each of the time, position, and orientation parameters, and what delimiter is used. The file can contain additional columns which are ignored during the import. If the first line of the file contains any characters other than the positive, negative, decimal point, and numeric characters (+ - . 1 2 3 4 5 6 7 8 9 0) then it is also ignored during import. This allows flexibility when importing raw TSPI data. The position and orientation must be in decimal notation.

There are several different data format options available for importing raw TSPI files. The time can be in either decimal seconds or decimal hours. The position can be defined in one of four available coordinate frames. There are two options available for the Latitude, Longitude, Altitude (LLA) coordinate frame and one option available for the Earth Centered Earth Fixed (ECEF) coordinate frame described in Methodology Manual section B10.1.1. Note that the ECEF coordinate frame must be defined relative to the WGS84 spheroid earth model. The ENU and NED coordinate frame options are also available. The ENU coordinate frame is described in section B10.1.3 and the NED coordinate frame is described in section 5.7.2.2. If the LLA option is selected, then the altitude can be defined in either feet or meters and the latitude and longitude can be defined as decimal degrees (dd.ddddd) or degrees minutes seconds (dd mm ss). The orientation is defined as a yaw, pitch, and roll which are described in section 5.6.10.1.3.12.2.

When the data is imported the time is translated from decimal hours to decimal seconds, if necessary, and then it is translated to be relative to the start of the segment. The time for the first point is set to zero. The time for all subsequent points is computed as follows.

$$t_{n\_rel} = t_n - t_1$$

where

t_{n_rel}	-	Time relative to start of maneuver for n^{th} point (s)
t_n	-	Absolute time from raw TSPI data for n^{th} point (s)
t_1	-	Absolute time from raw TSPI data for first point (s)

The orientation is imported directly without any translation. The position is transformed into relative local body frame coordinates based on the selected position format option as described in the following sections. The translated and transformed time, position, and orientation values are then displayed in the TSPI segment definition table.

5.6.10.1.3.12.5 Latitude Longitude Altitude to Body Frame

If the Latitude Longitude Altitude (LLA) position format is selected, then the raw TSPI position data is translated from geodetic LLA to the local body frame coordinates

defined in section 5.6.10.1.3.12.2 upon import. The Latitude and Longitude are translated from degrees/minutes/seconds (DMS) to decimal degrees (DD), if necessary, when the raw TSPI data file is read in. The location of the first point in body frame coordinates is set to (0, 0, 0). Each subsequent point is first transformed from LLA to ECEF coordinates using the LLAtoDblECEF routine described in section B10.2.10 and then to body frame using the DblECEF2BodyMatrix and Rotate1 routines described in sections B10.2.7 and B10.3.10 respectively. The ECEF position of the first point is used as the origin of the body frame coordinate frame and is sent into DblECEF2BodyMatrix as Position. The velocity sent into DblECEF2BodyMatrix is computed as described in section 5.6.10.1.3.12.10. Spherical earth is used for this transformation since the ECEF coordinate frame is an intermediate step in the transformation.

5.6.10.1.3.12.6 WGS84 Earth Centered Earth Fixed to Body Frame

If the WGS 84 Earth Centered Earth Fixed (ECEF) position format is selected then the raw TSPI position data is translated from ECEF to the local body frame coordinates defined in section 5.6.10.1.3.12.2 upon import. The location of the first point in body frame coordinates is set to (0, 0, 0). Each subsequent point is transformed from ECEF to body frame coordinates using the DblECEF2BodyMatrix and Rotate1 routines described in sections B10.2.7 and B10.3.10 respectively. The ECEF position of the first point is used as the origin of the body frame coordinate frame and is sent into DblECEF2BodyMatrix as Position. The velocity sent into DblECEF2BodyMatrix is computed as described in section 5.6.10.1.3.12.10. Oblate earth is used for this transformation since the data being imported is assumed to be relative to WGS84 spheroid.

5.6.10.1.3.12.7 North-East-Down to Body Frame

If the North-East-Down (NED) position format is selected then the raw TSPI position data is translated from NED to the local body frame coordinates defined in section 5.6.10.1.3.12.2 upon import. NED is an Earth-fixed system with the x-axis aligned with north, the y-axis aligned with east, and the z-axis opposing the local vertical as described in section 5.7.2.2. NED coordinates are first translated directly to ENU coordinates by swapping axes.

$$X_{NED} = Y_{ENU}$$

$$Y_{NED} = X_{ENU}$$

$$Z_{NED} = -Z_{ENU}$$

where

X_{NED} - NED X coordinate (m)
Y_{NED} - NED Y coordinate (m)
Z_{NED} - NED Z coordinate (m)

X_{ENU} - ENU X coordinate (m)
Y_{ENU} - ENU Y coordinate (m)
Z_{ENU} - ENU Z coordinate (m)

The resulting ENU coordinate positions are transformed to body frame as described in section 5.6.10.1.3.12.8.

5.6.10.1.3.12.8 East North Up to Body Frame

If the East North Up (ENU) position format is selected then the raw TSPI position data is translated from ENU to the local body frame coordinates defined in section 5.6.10.1.3.12.2 upon import. If the first ENU raw data position values are non-zero then the raw TSPI data is relative to a point other than the platform performing the maneuver, such as the location of the TSPI data collection sensor. The position points are translated so that they are relative to the aircraft location at the start of the maneuver with (0, 0, 0) as the position of the first ENU data point.

$$\vec{P}_{n\_ENU} = \vec{P}_{n\_ENU\_rel} - \vec{P}_{1\_ENU\_rel}$$

where

$\vec{P}_{n\_ENU}$ - ENU location of n^{th} TSPI point relative to aircraft performing the maneuver (m)
$\vec{P}_{n\_ENU\_rel}$ - ENU location of n^{th} TSPI point relative to data collection point (m)
$\vec{P}_{1\_ENU\_rel}$ - ENU location of first TSPI point relative to data collection point (m)

The location of the first point in body frame coordinates is set to (0, 0, 0). Each subsequent point is first transformed from ENU to BF coordinates using the DblENU2BodyMatrix and Rotate1 routines described in MM sections B10.2.28 and B10.3.10 respectively.

5.6.10.1.3.12.9 Flight Smoothing

A curve fit to a second order polynomial is used during runtime to determine the position and orientation at each integration interval. The curve fit is performed on each component of the position and orientation using the LeastSquares routine described in Subsection B10.4.22. LeastSquares computes the coefficients of the second order polynomial that most closely fits the data.

$$Y = b_0 + b_1 t + b_2 t^2$$

where

Y	-	Dependent value (m)
t	-	Difference in current time and time since segment began (s)
B_0	-	0^{th} degree coefficient
B_1	-	1^{st} degree coefficient
B_2	-	2^{nd} degree coefficient

The number of points used in the curve fit is defined by the user. The points used in the curve fit are selected so that the current point is as close to the middle of the curve as possible. No smoothing is performed for the first point since the aircraft will snap directly to it. For the first N/2 points in the segment, where N is the user defined number of points to use for smoothing and N/2 is rounded to the nearest integer, the smoothing is applied to the i^{th} point using points zero to 2*i where i is the point number starting with 1. For the last (N/2 – 1) points in the segment the smoothing is applied to the i^{th} point using points (i - N/2 + 2) to k, where k is the total number of TSPI data points. But the minimum number of points used for smoothing is three, so the last three points are always used to smooth each of the last two points. For example, if the total number of TSPI data points is 50 and the number of points to use for smoothing is 4, then the first point is not smoothed, the second point is smoothed over points one through three, the third point is smoothed over points one through four, the fifth point is smoothed over points two through five, and so on. Points 49 and 50 are smoothed over points 48 through 50. The EvaluateLeastSquares routine described in section B10.4.23 is used to compute each component of the position and orientation from the polynomial coefficients at each integration time step.

Three or more points are necessary to fit to a second order polynomial. If the user defined number of points is two, then a linear interpolation between the two closest points is used to determine the position and orientation at each integration interval. If the user defined number of points is one, then the closest point is used for the position and orientation at each integration interval.

Flight smoothing is applied in body frame coordinates. The smoothed data point is then converted to ECEF coordinates using the DblECEF2BodyMatrix and Rotate2 routines described in Subsections B10.2.7 and B10.3.14 respectively.

5.6.10.1.3.12.10 Velocity Calculation

The TSPI data may not include the velocity of the aircraft. Velocity is necessary for all platforms in EADSIM. Therefore, the velocity is computed from the smoothed TSPI position. First, the position polynomial coefficients are computed for each component of the position as described in section 5.6.10.1.3.12.9. Then the derivatives of each component of the position polynomials are computed to get the velocity.

$$V_i = 2 \cdot b2_i \cdot t + b1_i$$

where

V_i	-	X, Y and Z component of velocity (m/s)
$b2_i$	-	X, Y, and Z 2nd order position polynomial coefficient (m/s^2)
$b1_i$	-	X, Y, and Z 1st order position polynomial coefficient (m/s)
t	-	Difference in current time and time since segment began (s)

The current time is then used to compute each component of the velocity. The velocity is converted from body frame to ECEF coordinates using the ECEF2BodyMatrix and Rotate2 routines described in Appendix sections B10.2.6 and B10.3.14 respectively. Coordinated flight causes the aircraft's orientation vector to always be aligned with its velocity vector. When an aircraft is in the TSPI maneuver segment flight mode it is allowed to fly using uncoordinated flight if the defined orientation does not align with the computed velocity.

5.6.10.1.3.12.11 Fuel Consumption

There are two options available for calculation of fuel flow throughout the TSPI maneuver segment. The first option allows the fuel flow to be calculated from the TSPI position data. The second option allows an average fuel flow rate throughout the segment to be defined. In a tactical system, the ability to perform a maneuver segment and the fuel flow during the segment are both influenced by instantaneous weight throughout the segment. While performing a TSPI maneuver segment in EADSIM, the ability to perform a maneuver is only degraded if the aircraft runs out of fuel. The variable fuel flow throughout the segment can be captured using the Calculate Fuel Flow option.

If the Calculate option is selected, then the velocity computed as described in section 5.6.10.1.3.12.10 from the TSPI position data is used to compute the instantaneous fuel flow throughout the maneuver segment.

First, the required acceleration from the previous to the current integration time step is computed as the derivative of the velocity polynomial.

$$a_i = 2 \cdot b2_i$$

where

a_i	-	X, Y and Z component of required acceleration (m/sec2)
$b2_i$	-	X, Y, and Z 2nd order position polynomial coefficient (m/s^2)

If the number of curve fit points is less than or equal to two, then the acceleration is computed as the rate of change of velocity over the data time step.

$$a_i = \frac{V_{2i} - V_{1i}}{\Delta t}$$

where

a_i	-	X, Y and Z component of acceleration (m/s^2)
V_{2i}	-	X, Y, and Z component of current velocity (m/s)
V_{1i}	-	X, Y, and Z component of previous velocity (m/s)
Δt	-	Time step from previous TSPI data point to current TSPI data point (s)

Then, the drag is computed using parameters defined on the airframe as described in Subsection 5.5.2.1. Next, the required thrust is computed.

$$T = m \cdot a + D + W \cdot \cos(\phi)$$

where

T	-	Thrust required to achieve current speed (N)
m	-	Aircraft mass (kg)
a	-	Required acceleration (m/sec2)
D	-	Drag force (N)
W	-	Aircraft weight (N)
ϕ	-	Angle between velocity vector and local vertical (rads).

The required thrust is then used to compute the throttle setting and the throttle setting is used to compute the fuel consumed as described in Subsection 5.5.2.1. The thrust limits applied do not prevent the aircraft from being able to fly the TSPI data points. So if the thrust required is off the thrust tables or greater than the max thrust, the max fuel flow will be used but in reality the aircraft would consume more than this amount of fuel because it would be exceeding the max available thrust. To prevent this from happening, be sure the airframe is capable of producing the thrust and fuel flow required to fly the TSPI maneuver segment. Note that this is an approximate fuel consumption algorithm.

If fuel flow has been defined as an Average Fuel flow rate, then the specified value will be used to determine the fuel expended throughout the TSPI maneuver segment. If, at any time during the maneuver segment, the aircraft runs out of fuel, then the thrust is set to zero causing the aircraft to accelerate towards the center of the earth.

5.6.10.1.3.12.12 TSPI Report

The TSPI report within the Maneuver Definition can be used to view the smoothed position, velocity, average acceleration, orientation, and average orientation rates at a user specified time interval. The data will be smoothed using a curve fit to a second order polynomial over the user specified number of curve fit points as described in section

5.6.10.1.3.12.9. The segment data points can be generated relative to a starting location and heading of a platform. Selecting the LLA/ECEF Report option causes the report to be generated in absolute coordinates. The position will be generated in both LLA and ECEF (x, y, z) coordinates. The velocity will be generated in ECEF (x, y, z) coordinates. The orientation and orientation rates will be generated in absolute Yaw, Pitch, and Roll relative to the ENU coordinate axis. If the Body Frame Report option is selected, then the data will be generated in the relative body frame coordinate system defined in section B10.2.6.

Specify the same time interval that is used in runtime in order to generate data points that represent the smoothed data points that will be used in runtime. During runtime, if the scenario is running in an HLA federation and the ABT update rate is less than 1.0 second, the aircraft will be flown using the specified update time. Otherwise, a 0.5 second update time is used.

The velocity contained in the report is computed from the TSPI position data as described in section 5.6.10.1.3.12.10. The average acceleration is computed as described in section 0.

The average orientation rate is computed as the rate of change of orientation over the data time step.

$$\dot{\theta}_i = \frac{\theta_{2i} - \theta_{1i}}{\Delta t}$$

where

$\dot{\theta}_i$ - Yaw, Pitch, and Roll component of orientation (deg/s)
θ_{2i} - Yaw, Pitch, and Roll component of current orientation (deg)
θ_{1i} - Yaw, Pitch, and Roll component of previous orientation (deg)
Δt - Time step from previous TSPI data point to current TSPI data point (s)

```
!                              TSPI Report
!                              ------------
!
!     Report Option:  LLA/ECEF Report
!     Start Lat:  49.74709
!     Start Lon:  11.35717
!     Start Alt:  12787.88
!     Start Heading:  5.7
!     Time Interval:  0.5
!
!
!                                        Position                                  Velocity
! Time       Lat       Lon     Alt      X(ECEF)       Y(ECEF)       Z(ECEF)       X(ECEF)       Y(ECEF)       Z(ECEF)
!--------- -------------------------- ---------------------------------------- ----------------------------------------
   0.00000  49.74709   11.35717  12787 4044332.750000  812334.875000 4872272.500000    -323.988303    -22.380933    268.630515
   0.50000  49.74897   11.35746  12786 4044171.541315  812323.784976 4872406.565139    -318.592744    -21.258480    267.998983

                              Acceleration
X(ECEF)       Y(ECEF)       Z(ECEF)       Yaw     Pitch   Roll    Yaw Rate  Pitch Rate Roll Rate
---------------------------------------- ------- ------- ------- ---------- ---------- ----------
0.000000      0.000000      0.000000      5.70    2.10    0.00       0.00       0.67       0.67
16.995100     11.504361     22.661506     5.70    2.50    0.70       0.00       0.99       3.97
```

5.6.11 Processing of Captive Platforms

Captive platforms are created by placing a specific platform on the target list of a ground-capable Fighter, Bomber, AGAttacker, Fighter-Bomber, Red TEL, or SSFU host platform and specifying a waypoint at which the captive platform will be launched from the host platform. This state of being a captive is not treated as a true flight mode; instead, it causes the platform to be inactive in the scenario, while potentially contributing to the RCS of an airborne host platform.

The ruleset of the host platform makes the decision to transition a captive platform to a fully operational platform, i.e., to launch the platform. The ruleset sends the engagement commands to launch the weapon system and to activate the captive platform. These commands require the initialization of the captive platform's state. The initial state in terms of position and velocity of the captive platform is set to the state of the host platform at the time of launch. A captive launched by a stationary ground platform has its initial position set to that of the host platform at the time of launch, with its speed initialized to the airframe's minimum speed and its orientation set in the direction of its first waypoint. The captive platform will then proceed to fly in the user-selected Waypoint flight mode. Unlike other platforms, the captive platform will fly to its first waypoint rather than use it only to set the initial state of the platform.

5.6.12 Low-Level Transit Routes (LLTRs)

EADSIM models LLTRs across an MEZ. These routes allow for the safe passage of aircraft within an MEZ.

Aircraft will examine their flight routes at transitional times to see if the planned flight route will carry the aircraft within an MEZ that the aircraft knows about. Aircraft which determine that their planned flight route will carry them within an MEZ will search for and adopt an LLTR that allows them safe passage. If no LLTR is found which is suitable, the aircraft will proceed through the MEZ.

Once an LLTR has been adopted, it is simply followed as a set of waypoints. Normally, an aircraft will not depart the transit route once it is on the route. There are exception conditions which are described below. Aircraft may also adopt an LLTR and then abandon it prior to reaching the LLTR.

5.6.12.1 Aircraft Determination of MEZ Crossings

Aircraft will examine their flight routes at transitional times to see if the planned flight route will carry the aircraft within an MEZ of which the aircraft is aware.

The aircraft knowledge of the MEZ is based on the aircraft being explicitly associated with the MEZ during the setup of the scenario. This association is accomplished in Scenario Generation.

The transitional flight situations at which the aircraft examines the route are:

- when it is activated, either at the beginning of the scenario or later
- when it reaches one of its waypoints
- when it vectors toward a target in the early stages of an engagement, unless that target is in an MEZ
- when it executes either a scripted or dynamically determined takeoff
- when it enters the RTB flight mode
- when it has reached the end of an LLTR
- when it skips a waypoint (platforms with Bomber or Fighter-Bomber ruleset types only)
- when it returns to normal flight mode.

The planned aircraft route is not examined when the aircraft enters the latter stages of an engagement or when it is reacting to an attack.

In all cases, the aircraft will examine its flight route from its current position to its intended destination. When an aircraft is vectored to a target, it uses the current target position as the intended destination.

The aircraft will determine the distance from its current position to the point at which it would cross the boundary of the MEZ. In the case where the planned flight path will carry it within multiple MEZs, the closest MEZ (as determined by the distance to the boundary crossing) will be considered for selection of an LLTR.

5.6.12.2 Rules for Adoption of LLTRs

Aircraft which determine that their planned flight route will carry them within an MEZ will search for an LLTR associated with that MEZ which allows them safe passage. An LLTR may be used to cross an MEZ only if all of the following conditions are true:

- It is a two-way LLTR or a one-way LLTR running in the direction the aircraft wishes to cross.
- The aircraft does not need to cross the MEZ to reach the starting point of the LLTR.
- The aircraft does not need to cross the MEZ to fly from the end of the LLTR to its destination.

If several LLTRs are found to be valid routes across an MEZ, the one which gives the shortest flight path to the aircraft's objective will be used. If no valid LLTRs are found, the aircraft will fly across the MEZ.

5.6.12.3 Aircraft Flight on an LLTR

Once an LLTR has been adopted, it is simply followed as a set of waypoints. The aircraft will proceed from its current position to the entry point of the LLTR. During this time, it will be considered to be approaching the LLTR but not yet on it. It will then proceed from waypoint to waypoint along the LLTR.

A decision to stop or abort an engagement will interrupt the approach to the LLTR. The aircraft will drop the LLTR. A decision to stop or abort an engagement will not interrupt the transiting of the LLTR. The aircraft will proceed to the end of the LLTR. A decision to react to attack will cause the aircraft to be drawn off the LLTR. The aircraft will return to the LLTR later when the aircraft resumes normal flight.

A decision to return to base will cause the aircraft to drop the LLTR.

5.6.12.4 Effects of LLTR Adoption

The adoption of an LLTR has several impacts in the C3I model. These are briefly described here for clarity. A more in-depth description is contained in Section 4 of this document.

Aircraft on an LLTR are not considered for engagement by friendly SAMs operating in Truth mode. In this case, Friendly SAMs are those defined as being of the same alliance as the aircraft. If the aircraft is drawn off the LLTR by an enemy attack, it may be engaged by friendly SAMs until it returns to the LLTR. In perception mode, operating on an LLTR can feed point to the identification of the target through Procedural Identification.

Bombers and fighter-bombers will not select a target while on an LLTR. Fighters will select a target while on an LLTR but will not break off the LLTR to vector to the target.

5.6.13 Profile Deployment

Deployment of profile waypoint sets provides a dynamic capability for adopting a certain flight profile when a platform engages against a ground target. It allows the platform to leave its original waypoint set and follow the profile waypoints until within range of the target to engage. Once the engagement has been completed, the platform can then return to its original set of waypoints. Since the profile waypoints are set up at the ruleset level, the same flight profile can be used by multiple aircraft within the scenario, regardless of which target is being engaged and where that target is located within the scenario.

5.6.13.1 Profile Definition and Set-Up

Profiles are set up in the Target Select Phase of the Bomber, Fighter-Bomber, and AGAttacker rulesets. A profile is a series of waypoints that defines the path the platform will follow when reaching the target area. The path will be followed until the platform is within weapon's range of the target and the actual engagement can begin. The profile will be initiated once the flight leader flies to within a ruleset-defined initiation range of a target location. When this initiation range has been reached by the flight leader, each aircraft in the flight will adopt its individual profile and attempt to engage its target. Once the profiles have been initiated, the profile waypoints will be followed by each aircraft until it has found and engaged its targets or until the flight leader has orbited in its profile the number of orbits defined by the ruleset.

Each waypoint in the profile is defined relative to a generic target location. The waypoints are set by downrange and crossrange distances from the generic target location, speed at which to approach each waypoint, altitude of the waypoint in either MSL, AGL, or TGT units, and whether the platform should fly terrain following to the waypoint. TGT units are relative to the target. Terrain following is automatically set to 'No' when you select TGT or MSL altitude mode. Setting these waypoints using relative distances from a generic target rather than using absolute waypoint locations allows for the use of this ruleset and its defined profiles by multiple aircraft in the scenario engaging against a variety of ground targets located throughout the scenario.

Waypoint profiles can be set for each platform in a flight. The flight leader has its own unique profile. Each wingman can have the same default profile or a profile specific to the wingman position in the flight. For example, if a ruleset is set up with different profiles for the first two wingmen in the flight, the remainder of the wingmen will adopt the default wingman profile once the flight leader penetrates the initiation range to its target.

When defining the profiles for each aircraft in a flight, the user has the option to scale the profile for all platforms that use the ruleset, scale the profile for complex weapons only, or to not scale the profile. If profile scaling is enabled the profile will be scaled when the weapon is launched within the maximum profile waypoint downrange. Each waypoint is adjusted based on the horizontal and vertical scale factors. The horizontal scale factor is computed by dividing the launch range from the target by the greatest downrange of the profile waypoints. The vertical scale factor is computed by dividing the launch altitude by the first profile waypoint altitude. The horizontal scale factor is then multiplied by the downrange and the vertical scale factor is multiplied by the altitude of each profile waypoint to scale the profile. If profile scaling is not enabled the user has the option to designate which waypoint leg can be adjusted for range and altitude. This capability has been provided to allow for profile adjustments should initiation of the

profile be at a range or altitude relative to the target that is insufficient to allow the profile to be flown exactly as specified. Only one leg can be adjusted for range and only one for altitude. The same leg may be selected for both. Only legs where altitude changes occur may be selected for altitude adjustment. The minimum altitude for all waypoints can be designated using floor altitude. The default floor altitude is 200m.

When adoption of a profile occurs, absolute positions are computed for each profile waypoint. For flight leaders, the profile is simply inserted into its waypoint list prior to its next waypoint. The flight leader continues to fly in waypoint mode and flies these profile waypoints. For a wingman, the profile becomes its waypoint set since wingmen normally do not have waypoints. The wingman changes from Wingman flight mode to Waypoint flight mode and flies the waypoints. Once each aircraft is within weapon's range to its respective target, the aircraft engages the target using the Bomber flight mode and then drops the profile once the engagement is complete. If the flight leader never engages its target, it will orbit the specified number of times and then drop its profile and return to its original waypoint set. If a wingman never engages, it will orbit until the flight leader drops its profile and then it, too, will drop its profile. When the flight leader drops its profile, all the profile waypoints are simply removed from the waypoint list and it flies to the next remaining waypoint in the list. When the wingman drops its profile, it switches back to Wingman flight mode and vectors to its formation location relative to the flight leader's position.

To provide for a cruise missile capability, a profile option exists that allows the platform to drop all remaining original waypoints from its waypoint list when it adopts the profile. This allows for a one-way flight into the target. Once the last profile waypoint is reached, the platform will engage the target if it is within weapon's range. If not, the platform will simply disappear, since no more waypoints exist. This option allows a very specific maneuver to be performed by a cruise missile-like platform when it flies into its target.

5.6.13.2 Profile Waypoint Absolute Location Calculation

When computing the profile waypoint absolute locations, the local reference frame must be computed. The up directions at the target and platform are obtained using ECEFtoUp with the target and platform locations respectively. Then, the cross product of the up directions at the target and platform is computed:

$$\vec{C}_{tac} = \left| \vec{U}_T \times \vec{R}_{TP} \right|$$

where

$\vec{C}_{tac}$ - cross product of target and platform position vectors (m^2)

$\vec{U}_T$ - target position up vector

$\vec{R}_{TP}$ - Vector from target to platform.

Next, the x-axis of the local frame is computed. It points in the direction from the target toward the platform:

$$\vec{U}_x = \frac{\left|\vec{C}_{tac} \times \vec{U}_T\right|}{C_{mag}}$$

where

$\vec{U}_x$ - unit local x-axis pointing from target to platform
$\vec{C}_{tac}$ - cross product of target and platform position vectors (m^2)
$\vec{U}_T$ - target position up vector
C_{mag} - magnitude of target/platform cross product (m^2)

Once the local reference frame relative to the target's location is computed, the local z-axis is simply the unit vector of the target position vector. Taking this vector and the local x-axis, the local y-axis is found by:

$$\vec{U}_y = \left|\vec{U}_z \times \vec{U}_x\right|$$

where

$\vec{U}_y$ - unit local y-axis orthoganol to z- and x-axes
$\vec{U}_z$ - unit local z-axis pointing along target position vector
$\vec{U}_x$ - unit local x-axis pointing from target to platform.

Once the local reference frame has been computed, the profile for the platform must be found. If the platform is the flight leader, the first profile in the profile list off the ruleset structure is used. If the platform is a wingman and no specific profile was set up for it, the profile used is the default wingman profile which is the second profile in the list. If the platform is a wingman and a specific profile is set up for it, the profile list is looped over, starting from the third profile in the list until the correct profile is found.

Next, the necessary altitude and range adjustments are computed by:

$$\Delta R = R_{tac} - R_{prof}$$

$$\Delta H = \Delta H_{tac} - H_{prof}$$

where

ΔR - downrange difference (m)
R_{tac} - downrange from target to platform (m)
R_{prof} - downrange distance from target for first profile waypoint (m)
ΔH - altitude difference (m)
ΔH_{tac} - altitude difference between target and platform (m)
H_{prof} - altitude for first profile waypoint (m).

After initializing the range and altitude adjustment variables to 0.0, the profile waypoints are looped over from the last waypoint to the first and each waypoint absolute position is computed in ECEF coordinates. Adjustments, if necessary, are dependent upon where in the profile waypoint list the adjustment legs are located. If the current profile waypoint is used to adjust both range and altitude, the range and altitude differences are stored in the adjustment variables:

$$R_{adj} = \Delta R$$

$$H_{adj} = \Delta H$$

where

R_{adj} - downrange adjustment (m)
ΔR - downrange difference (m)
H_{adj} - altitude adjustment (m)
ΔH - altitude difference (m).

If the current waypoint is the altitude adjustment leg, the altitude difference is stored:

$$H_{adj} = \Delta H$$

where

H_{adj} - altitude adjustment (m)
ΔH - altitude difference (m),

Then the range adjustment value is computed based on certain conditions. If a previous leg represented a waypoint altitude change, if the platform's altitude above the target is greater than the previous waypoint's altitude, and if adjusting for range is allowed and a range adjustment has not yet occurred, similar triangles are used to compute the range adjustment value. First, the altitude difference between waypoints is computed:

$$H_{diff} = H_i - H_{i-1}$$

where

H_{diff} - waypoint altitude difference (m)
H_n - altitude for the nth waypoint (m)
i - waypoint number starting from the last waypoint.

Next, the range difference is computed:

$$R_{diff} = R_i - R_{i-1}$$

where

R_{diff} - waypoint downrange difference (m)
R_n - downrange for the nth waypoint (m)
i - waypoint number starting from the last waypoint.

Since the actual altitude difference based on the platforms current position and the previous waypoints altitude can be computed by

$$\Delta H_{act} = \Delta H_{tac} - H_{i-1}$$

where

ΔH_{act} - actual altitude difference (m)
ΔH_{tac} - altitude difference between target and platform (m)
H_{i-1} - altitude for the (i-1)th profile waypoint (m)
i - waypoint number starting from the last waypoint,

the actual downrange difference can be computed according to:

$$\Delta R_{act} = -(\Delta H_{act} R_{diff})/H_{diff}$$

where

ΔR_{act} - actual downrange difference (m)
ΔH_{act} - actual altitude difference (m)
R_{diff} - waypoint downrange difference (m)
H_{diff} - waypoint altitude difference (m).

The range adjustment is then computed as:

$$R_{adj} = \Delta R_{act} - R_{diff}$$

where

R_{adj} - downrange adjustment (m)
ΔR_{act} - actual downrange difference (m)
R_{diff} - waypoint downrange difference (m).

If a range adjustment has not yet occurred but is allowed, the altitude adjustment is left unchanged and the range adjustment is computed by:

$$R_{adj} = H_{adj}\left(\frac{R_i - R_{i-1}}{H_i - H_{i-1}}\right)$$

where

R_{adj} - downrange adjustment (m)
H_{adj} - altitude adjustment (m)
R_n - downrange for the nth waypoint (m)
H_n - altitude for the nth waypoint (m)
i - waypoint number starting from the last waypoint.

If no leg adjustment for range is necessary or if the range adjustment leg has already been passed, the altitude adjustment is left unchanged and the range adjustment is set to the actual downrange difference:

$$R_{adj} = \Delta R$$

where

R_{adj} - downrange adjustment (m)
ΔR - downrange difference (m).

If any adjustments were made in either altitude or downrange or both, the waypoint distances relative to the target location are computed as:

$$W_R = R_i + R_{adj}$$

$$W_H = H_i + H_{adj}$$

where

W_R - waypoint downrange from target (m)
R_i - downrange for ith waypoint (m)
R_{adj} - downrange adjustment (m)
W_H - waypoint altitude above target (m)
H_i - altitude for ith waypoint (m)
H_{adj} - altitude adjustment (m).

If no adjustments were allowed or necessary, the waypoint distances relative to the target location are set by:

$$W_R = R_i$$

$$W_H = H_i$$

where

W_R	-	waypoint downrange from target (m)
R_i	-	downrange for ith waypoint (m)
W_H	-	waypoint altitude above target (m)
H_i	-	altitude for ith waypoint (m).

Once the waypoint distances of downrange and altitude are set, the waypoint relative position direction vectors are computed relative to the target location in ECEF coordinates as:

$$\vec{X} = \vec{U}_x \times W_R$$

$$\vec{Y} = \vec{U}_y \times C_i$$

$$\vec{Z} = 0.0$$

where

$\vec{X}$	-	downrange vector in ECEF coordinates (m)
$\vec{U}_x$	-	unit local x-axis pointing from target to platform
W_R	-	waypoint downrange from target (m)
$\vec{Y}$	-	crossrange vector in ECEF coordinates (m)
$\vec{U}_y$	-	unit local y-axis orthoganol to z- and x-axes
C_i	-	crossrange distance from target (m)
$\vec{Z}$	-	local up vector in ECEF coordinates (m).

The ECEF relative position vector of the waypoint is then computed as the vector sum of the components of the direction vectors X, Y, and Z:

$$\vec{W}_{Rel} = \vec{X} + \vec{Y} + \vec{Z}$$

where

$\vec{W}_{Rel}$	-	waypoint relative position ECEF vector (m)
$\vec{X}$	-	downrange vector in ECEF coordinates (m)
$\vec{Y}$	-	crossrange vector in ECEF coordinates (m)
$\vec{Z}$	-	local up vector in ECEF coordinates (m).

The actual waypoint ECEF vector is then computed as the sum of the ECEF relative position vector of the waypoint and the target position:

$$\vec{W} = \vec{W}_{Rel} + \vec{T}$$

where

$\vec{W}$ - waypoint position vector in ECEF coordinates (m)
$\vec{W}_{Rel}$ - waypoint relative position ECEF vector (m)
$\vec{T}$ - target position vector in ECEF coordinates (m).

The latitude and longitude corresponding to this location are then computed. The position of this waypoint is then recomputed to account for the altitude of the waypoint in the user-desired AGL, TGT, or MSL units. If AGL, the waypoint altitude used is the value of W_H added to the terrain elevation at the waypoints' latitude and longitude. If MSL, the waypoint altitude used is the value of W_H. If TGT, the waypoint altitude used is the value of W_H added to the target elevation at the target's latitude and longitude. This new position, the latitude, longitude, and profile waypoint altitude are all stored in the waypoint structure. The bit designating this waypoint as a profile point is then set. If the profile point is a hover waypoint, the waypoint on time is set to 0.0 and the off time is set to the profile hover point duration. If the profile point is not a hover waypoint, the waypoint on and off times are set to the default values of 0.0 and 99,999.0, respectively. Once all of this has been completed, this waypoint is added to the platform's waypoint list, and the loop continues to the next profile waypoint, repeating the procedure until all profile waypoints have been inserted into the waypoint list.

5.6.14 State Vector Mode

State Vectors provide the precise position, velocity, and orientation of an aircraft platform at a specific time. A State Vector consists of waypoint number, time, latitude, longitude, altitude, speed, heading, pitch, bank, and yaw. In addition to state vectors, waypoints are specified in the state vector mode of operation for weapon delivery.

The State Vector mode is defined by the user through Scenario Generation for airborne platforms. Inputs controlling platform movement in this mode are read from a file, which contains both waypoints and state vectors.

In the state vector mode, the first state vector defines the platform OnTime. Platforms move from one user-defined state to another in the order defined by the user. For platform movement between time steps input by the user, all state vector parameters are adjusted by means of a linear interpolation. When the last state is reached, the platform is deactivated and removed from the scenario.

In state vector mode, the waypoints and state vectors specified in the input file maintain complete control over the platform position, velocity, orientation,

weapon delivery, etc. throughout the simulation. Any commands from the C3I model to vector to a new heading, begin drag maneuver, return to base, etc. for these platforms will be ignored in flight processing.

5.7 MISSILE FLIGHT

5.7.1 Introduction

EADSIM has the capability to represent a wide variety of missiles. The flight modeling for these missiles include options for 3 Degree of Freedom (DOF) theater and intercontinental ballistic missile flight, 3-DOF explicit interceptor missile flight, missile flight based on a trajectory curve fit model, constant velocity non-aerodynamic cruise missile flight, and constant velocity non-aerodynamic interceptor flight. Explicit Air-to-Surface missile flight utilizing the Aircraft airframe does not use the algorithms addressed in this section.

Ballistic missiles are introduced into EADSIM either through the detailed, internal 3-DOF missile model or by a trajectory curve fit model. The internal 3-DOF ballistic missile model and the curve fit model provide full compatibility with the dynamic launch decision processing found in the ruleset logic of the C3I model.

The internal 3-DOF ballistic missile model provides the capability to model multistage missiles with detailed pitch program guidance. Flight section options include the ability to set up multiple powered flight segments representing engine thrusting, unpowered segments representing ballistic flight, and missile staging events representing missile mass changes. All options can be timed or delayed to model actual missile weapon systems realistically. Guidance options such as minimum energy, depressed, lofted, non-thrust terminating reentry angle, thrust terminating reentry angle and disable launch iteration, gravity turns, or multiple guidance phases can be used to achieve the desired flyout. Guidance options can also be used to model post boost vehicle (PBV) maneuvers and deployment of debris, decoys, chaff, multiple independently targeted re-entry vehicles (MRVs), and maneuvering reentry vehicles (MaRVs).

The internal 3-DOF ballistic missile model is further supported by the launch iteration schemes, which determine the correct setting of specific parameters to allow the missile to fly the desired range to the target. The launch iteration schemes provide the capability to model the guidance options mentioned above. The algorithms were developed specifically for application in the EADSIM.

The internal 3-DOF model also provides a mechanism to take known missile deployment and detonation information for multiple RVs from a single missile launch and achieve trajectories for the RVs and associated objects that maintain the detonation timing for each RV. This option allows trains of objects including RVs and penetration aids to be fitted to user specified trajectory parameters without the need to accurately model the boost phase or post boost vehicle flight.

The trajectory curve fit model provides the capability to rapidly generate specific trajectories based on interpolation between known trajectories. These

known trajectories may include maneuver characteristics not supported by the internal ballistic model. Subsection 5.7.4 describes the methodology for using the curve fit model.

The 3-DOF missile model for interceptors is the same underlying flight dynamics model as that for the ballistic missile model. The major difference comes from the presentation to the user for configuring a specific missile, primarily in the area of available guidance options. This capability provides modeling of realistic trajectories associated with the boost, flyout, and homing of a single or multistage interceptor missile engaging a target. Guidance options during flyout include following a prescribed flight path angle curve, pitch profile, ballistic, ballistic with predictive proportional navigation (PPN), and guiding to a point in space for both powered and unpowered flight. Proportional navigation and predictive proportional navigation are available to steer the missile toward the target during endgame.

A number of options are included to provide flexibility so that the flight trajectory of virtually any homing interceptor can be realistically simulated. These options include control of both the azimuth and elevation pointing of the missile at time of launch. The missile is automatically pointed toward the target in azimuth or aligned with the Primary Target Line of the launcher. The latter modeling accounts for a launcher that is fixed along its Primary Target Line (PTL). In elevation, the user can specify a specific launch elevation above horizontal, a lead angle to allow the interceptor to pitch over due to gravity, or to align with a computed elevation angle if iterating on pitch for the launch solution. Iterating on pitch is provided as an option to obtain a trajectory that will ballistically fly the interceptor through the intercept point for long range intercepts.

Multistage interceptor missiles are modeled using stages and flight sections to represent the missile's properties. Each flight section consists of a separate specification for the aerodynamic, propulsion, and guidance parameters. Within a flight section, any combination or sequence of guidance modes can be selected including ballistic, prescribed flight path angle, pitch profile, or proportional navigation to a space point. A stage is one or more flight sections. Multiple flight sections within a stage will typically not exhibit a mass change upon transition from one flight section to the next; however, any parameter associated with the flight section can change if needed.

The flight trajectories during flyout and homing are realistically constrained by aerodynamic, structural, and propulsion capabilities. Lateral acceleration for steering is aerodynamically limited according to a user-specified maximum angle of attack. Also, a hard limit is placed on lateral steering acceleration representing maximum g-loading structural capabilities. A special exo-atmospheric steering option is included for modeling lateral (or divert motor) thrusters that act through the object's center of gravity.

5.7.2 Missile Coordinate Systems

The missile model uses three separate coordinate systems: the Earth-Centered Earth-Fixed (ECEF) coordinate frame, the North-East-Down (NED) frame, and the Missile Velocity Axis frame.

5.7.2.1 ECEF Coordinate Frame

All missile state calculations are performed in the ECEF coordinate frame as described in Appendix B10.

5.7.2.2 NED Coordinate Frame

The position of the missile's velocity vector relative to the Earth's surface is calculated in a right-handed NED frame. The origin of this frame is centered at the missile's launch point position defined by a Latitude and Longitude on the Earth's surface. Positive x points north, positive y points east, and positive z points down along the local vertical. The x and y axes define the NED ground plane.

This coordinate frame is primarily used to describe the orientation of the missile's velocity vector at launch in terms of azimuth and elevation angles. Azimuth is a positive rotation about the z axis. Elevation angle is the angle between the ground plane and the missile's velocity vector. During flight, the flight path angle is the angle between the ground plane and the missile's velocity vector.

5.7.2.3 Missile Velocity Axis Coordinate Frame

Calculations of forces acting on the missile are calculated in the velocity axis frame. This frame is centered at the missile's center of gravity. Positive x is defined as always being coincident with the missile's velocity vector.

After calculating the change in missile velocity due to thrust and drag, the velocity is resolved into vector components in the ECEF frame. The orientation of the missile's velocity axis is defined by rotating the ECEF frame through three consecutive Euler angle rotations. Initially aligned with the ECEF frame, the first rotation is a positive rotation about the ECEF z axis through the angle Ψ. This single rotation defines a new or second coordinate system. The second rotation is a positive rotation about the newly defined y axis through the angle θ, which again defines a new or third coordinate frame. In this third coordinate frame, the final rotation is about the newly defined x axis through the angle Φ. Thus a coordinate frame based on the missile's velocity vector is defined relative to the ECEF frame by the three consecutive angle rotations: Ψ, θ, and Φ. Note that the order of rotation is important, as the velocity vector's orientation will change with a different rotation order.

5.7.3 3-DOF Aerodynamic Missile Flight

Propagation of ballistic and explicit interceptor missiles is performed by an internal 3-DOF aerodynamic flight model. The model was formulated with particular attention to speed of execution. A unique point-mass model with wind axes is used to include the variation of drag with angle of attack as the missile maneuvers without invoking the computational burden associated with a 6-DOF rigid-body model for the missile. Trajectories generated with this modified point-mass model have been shown to match identically those generated by a very high fidelity 6-DOF simulation (within the resolution of time-history plots)[7].

5.7.3.1 Missile Equations of Motion

The missile's flight is modeled as a velocity vector acted upon by accelerations coincident and perpendicular to it as shown in ***Figure 5.7-1***. Acceleration due to drag and thrust act coincidentally to the velocity vector and acceleration due to steering acts perpendicular to the velocity vector. The effects of forces acting on the missile which slow it down or speed it up (drag, gravity, and axial thrust) are modeled by changing the magnitude of the velocity vector. The effects of forces acting on the missile which change its direction of flight (steering and gravity) are modeled by turning the velocity vector.

5.7.3.1.1 Change in Velocity Magnitude

The change in the magnitude of the velocity vector is calculated by:

$$\dot{V} = \frac{T\cos(\alpha) - qSC_d}{m} - g_x$$

where

T - Thrust (N)

α - Angle of attack (rad)

q - Dynamic pressure (N/m^2)

S - Aerodynamic reference area (m^2)

C_d - Drag coefficient

m - Mass (kg)

g_x - Gravity component acting along velocity vector axis (m/sec^2).

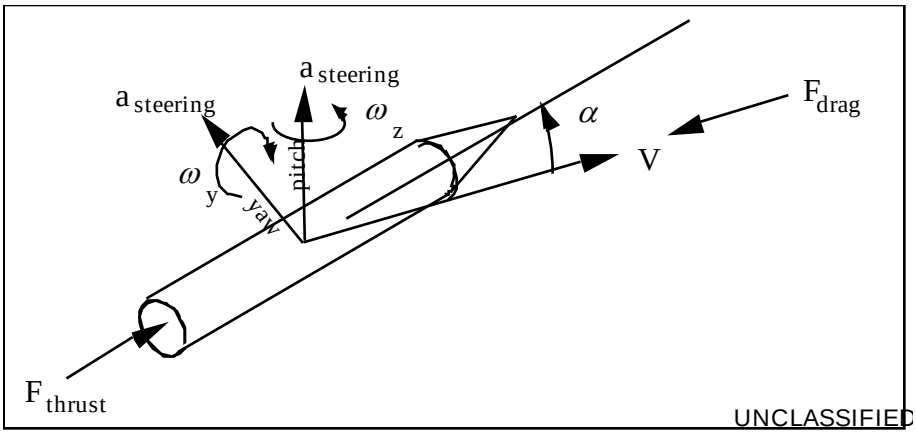

Figure 5.7-1 Missile Flight Model

The drag coefficient, C_d, is computed by table look-up as a function of Mach number and angle of attack. The velocity, V, is calculated:

$$V = \int \dot{V}\, dt$$

The missile's mass changes during boost as propellant is consumed. The change of mass rate is computed as:

$$\dot{m} = \frac{-T_{delivered}}{I_{sp} g_0}$$

where

$T_{delivered}$	-	delivered thrust (N)
I_{sp}	-	specific impulse (sec)
g_0	-	standard gravity
	=	9.80665 m/sec^2.

To account for nozzle exit losses due to the ambient atmospheric pressure, the delivered thrust is calculated as:

$$T_{delivered} = T_{vacuum} - A_{nozzle\ exit}\, P_{ambient}$$

where

T_{vacuum}	-	vacuum thrust (N)
$A_{nozzle\ exit}$	-	nozzle exit area (m^2)
$P_{ambient}$	-	local ambient pressure (N/m^2).

Mass (kg) is then calculated:

$$m = \int \dot{m}\, dt$$

5.7.3.1.2 Change in Velocity Vector Direction

The rate of turn of the velocity vector is given by

$$\omega = \frac{a_{steering}}{V}$$

where

ω	-	rate of turn of velocity vector
$a_{steering}$	-	applied acceleration for steering
V	-	magnitude of velocity.

Note that if the angle of attack is nonzero, the axial thrust will also tend to turn the velocity vector. Given the accelerations a_y and a_z in the missile velocity axis frame y and z channels, respectively, the rotation rate of the velocity vector is computed by:

$$\omega_y = \frac{-a_z}{V} \text{ and } \omega_z = \frac{a_y}{V}.$$

The derivatives of the Euler angles that describe the missile velocity vector's orientation with respect to the ECEF coordinate system are defined using the computed velocity vector turning rates:

$$\dot{\Psi} = \frac{\omega_2 \cos\Phi + \omega_y \sin\Phi}{\cos\Theta}$$
$$\dot{\Phi} = \dot{\Psi}\sin\Theta$$
$$\dot{\Theta} = \omega_y \cos\Phi - \omega_z \sin\Phi \quad .$$

The Euler angles are then calculated:

$$\Phi = \int \dot{\Phi}\, dt$$
$$\Theta = \int \dot{\Theta}\, dt$$
$$\Psi = \int \dot{\Psi}\, dt\,.$$

5.7.3.1.3 Steering Constraints

Before they are utilized in the preceding equations, the accelerations for steering, a_y and a_z, are constrained by aerodynamic, structural, and divert motor capability limitations, as well as a dynamic lag. This limited acceleration is used to compute an equivalent angle of attack in the pitch and yaw channels. The complete logic for limiting the commanded acceleration from guidance is shown in ***Figure 5.7-2.*** Following is a mathematical description of each constraint.

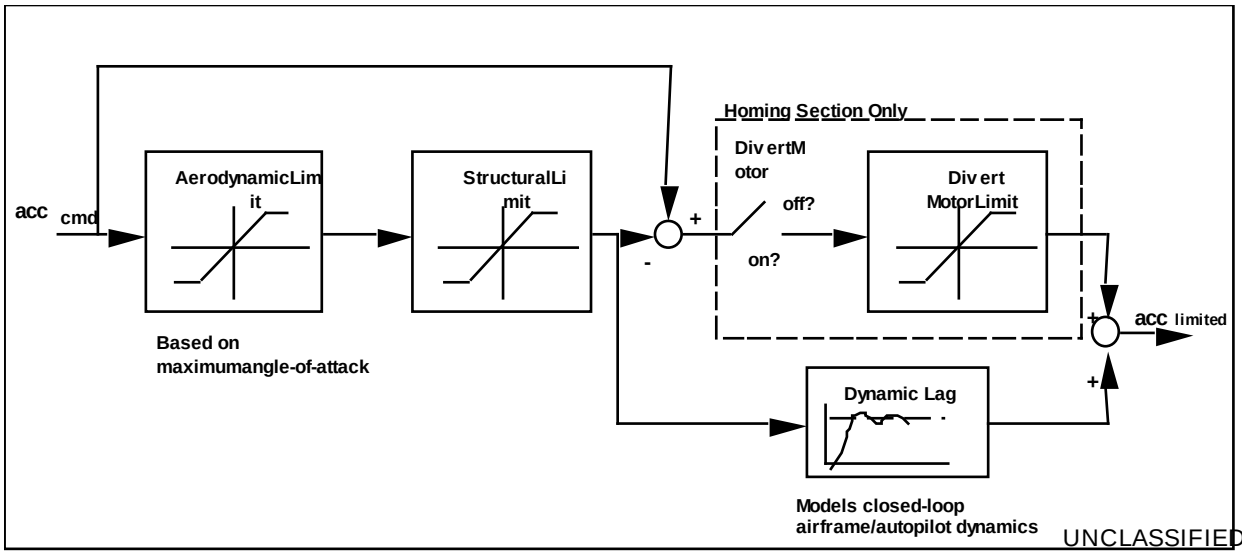

Figure 5.7-2 Missile Model Maneuver Constraints

An aerodynamic constraint on acceleration is computed based on a user-specified maximum angle of attack:

$$a_{max} = \left(F_{thrust}\sin\alpha_{max} + qSC_{1\alpha}\alpha_{max}\right)/\text{Mass}$$
$$\approx \left((F_{thrust} + qSC_{1\alpha})\alpha_{max}\right)/\text{Mass}$$

where

$\sin \alpha_{max} \approx \alpha_{max}$

$C_{1\alpha}$ = linearized coefficient of lift (/rad).

The linearized coefficient of lift (or lift slope) approximates the variation of this coefficient with angle of attack, as shown in ***Figure 5.7-3***. The linearized lift coefficient is computed by table look-up as a function of Mach number.

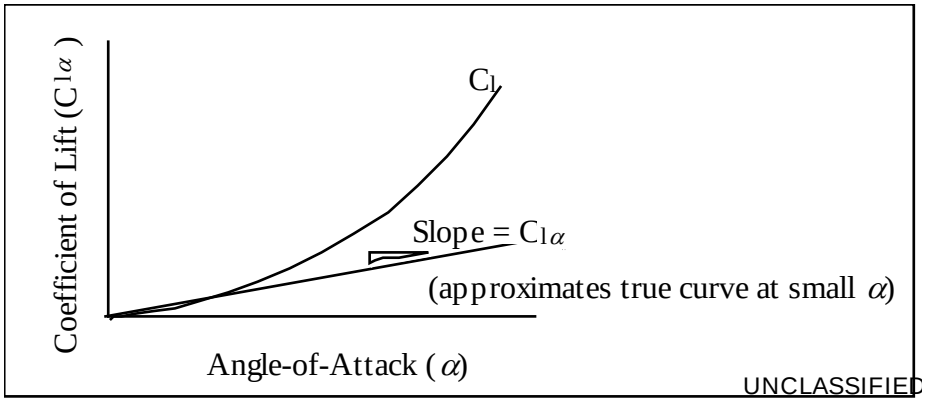

Figure 5.7-3 Linearized Approximation of Lift Coefficient Variation

At low altitude and hypersonic velocities, some missiles are able to generate very large steering accelerations at relatively small angles of attack. To prevent the

missile from maneuvering at levels where it would fail structurally, the aerodynamic acceleration is further constrained by a hard limit, as shown by the saturation function in ***Figure 5.7-4.***

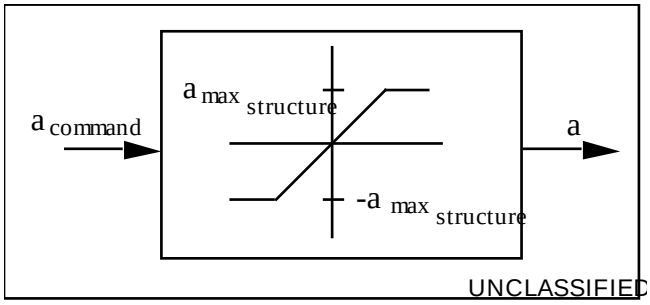

Figure 5.7-4 Structural Constraint on Steering Acceleration

The missile's airframe does not respond immediately to lateral acceleration commands; it takes some time for the airframe to rotate and produce lift with the angle of attack under control of the autopilot. ***Figure 5.7-5*** illustrates the second-order transfer function that is used to model the resulting response lag between the commanded and achieved acceleration. The speed of response is specified by the damping coefficient, z, and the natural frequency, w_n. To provide a "higher-level" parameter for the user who may not be familiar with how to specify z and w_n, z is fixed at 0.7 and w_n is computed within the model by a user-specified 90% response time:

$$\omega_n = \frac{7.04\zeta^2 + 0.25}{2\zeta t_r}$$

where t_r is the time for response to reach 90% of a step command input (sec).

Fixing z = 0.7 provides a good speed of response with minimum overshoot and is frequently the specification for design of closed-loop control systems, including missile autopilots.

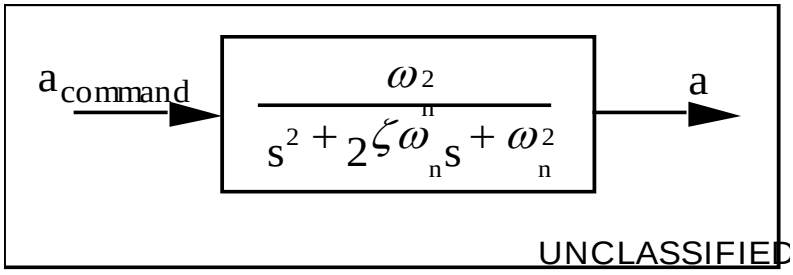

Figure 5.7-5 Second-Order Transfer Function Used to Model Response Lag

A special exo-atmospheric steering option is included for modeling lateral (or divert motor) thrusters that act through the object's center of gravity. Here, the air density is insufficient to provide significant aerodynamic steering so thrusters that provide steering forces approximately through the missile's center of gravity

augment steering. If this option is utilized, any difference between the commanded acceleration from guidance and the aerodynamic acceleration (from angle of attack) is made up by the divert motors up to the divert motor's ability to accelerate the missile. This is illustrated in ***Figure 5.7-2.*** The magnitude and duration of thrust from the divert motors are constrained by a user-specified acceleration limit and divert velocity. A minimum dynamic pressure for utilizing the aerodynamic as a function of whether the interceptor is in ballistic PPN or homing is provided for the interceptor. Also, a maximum dynamic pressure for utilizing the divert motors for each of these guidance types is also used for the interceptor.

The actual angle of attack is computed given the constrained acceleration. The main purpose for calculating this angle of attack is to establish this as a parameter in computing axial force in the previous equation for $\dot{V}$. Angle of attack is needed to resolve the boost thrust vector to the velocity vector and in the table look-up of the drag coefficient, C_d. Angle of attack is computed as:

$$\alpha = \frac{ma}{F_{thrust} + qSC_{1\alpha}}$$

where

$\sin\alpha$ - α

a = $\sqrt{a_{pitch}^2 + a_{yaw}^2}$

a_{pitch} - pitch acceleration after limiting (m/sec^2)

a_{yaw} - yaw acceleration after limiting (m/sec^2)

m - mass (kg).

5.7.3.1.4 Acceleration - Gravity

When operating with the spherical Earth gravity option, potential gravity is calculated by:

$$G_P = - G_C / X_{mag}^3$$

where

G_P - potential gravity (1/s^2)

G_C - universal gravity constant of 3.986 x 10^{14} (m^3/s^2)

X_{mag} - missile ECEF position vector magnitude (m).

Acceleration due to gravity is calculated in ECEF coordinates according to:

$$A_{Gx} = G_P \times X_x$$

$$A_{Gy} = G_P \times X_y$$

$$A_{Gz} = G_P \times X_z$$

where

A_{Gn} = gravity acceleration in ECEF n-direction (m/sec^2)
G_P = potential gravity (1/sec^2)
X_n = missile position in ECEF n-direction (m).

The Earth Gravity Model, EGM-96, option is a more detailed model of the Earth's gravity which can be used in conjunction with the WGS-84 modeling of the Earth. The following excerpt from NIMA TR8350.2, Third Edition, Amendment 1, 3 January 2000, provides an overview of the EGM96 model. This document also provides the coefficients for a degree 18, order 18 implementation. References to other sections and documents within the excerpt have been removed.

"The form of the WGS 84 EGM96 Earth Gravitational Model is a spherical harmonic expansion of the gravitational potential (V). The WGS 84 EGM96, complete through degree (n) and order (m) 360, is comprised of 130,317 coefficients. EGM96 was a joint effort that required NIMA gravity data, NASA/GSFC satellite tracking data and DoD tracking data in its development. The NIMA effort consisted of developing worldwide 30′ and 1° mean gravity anomaly databases from its Point Gravity Anomaly file and 5′x 5′mean GEOSAT Geodetic Mission geoid height file using least-squares collocation with the Forsberg Covariance Model to estimate the final 30′x 30′ mean gravity anomaly directly with an associated accuracy. The GSFC effort consisted of satellite orbit modeling by tracking over 30 satellites including new satellites tracked by Satellite Laser Ranging (SLR), Tracking and Data Relay Satellite System (TDRSS) and GPS techniques in the development of EGM96S (the satellite only model of EGM96 to degree and order 70). The development of the combination model to 70 x 70 incorporated direct satellite altimetry (TOPEX/POSEIDON, ERS-1 and GEOSAT) with EGM96S and surface gravity normal equations. Major additions to the satellite tracking data used by GSFC included new observations of Lageos, Lageos-2, Ajisai, Starlette, Stella, TOPEX and GPSMET along with GEOS-1 and GEOSAT. Finally, GSFC developed the high degree EGM96 solution by blending the combination solution to degree and order 70 with a block diagonal solution from degree and order 71 to 359 and a quadrature solution at degree and order 360."

If selected, the Earth Gravity Model 1996 (EGM96) is applied to the flight of ballistic missiles and SAM interceptors. Usage of EGM96 is activated by the EGM 96 gravity option in addition to selection of the WGS-84 oblate model.

EADSIM implements the EGM96 model using a selectable degree/order up to the maximum degree 360 and maximum order 360. The coefficients for this model were obtained from Goddard Space Flight Center (GSFC) / National Imagery and Mapping Association (NIMA) publications for EGM96. These publications were obtained from multiple website locations, to include:

http://cddisa.gsfc.nasa.gov/926/egm96/egm96.html,

http://www.pha.jhu.edu/~hanish/wgs84fin.pdf, and
http://earth-info.nga.mil/GandG/wgsegm/egm96.html.

Within EADSIM, these coefficients are stored in binary files in the executable directory. When a scenario is loaded into EADSIM utilizing the EGM96 model, the coefficients are read into memory based on the user selected degree and order. The greater the degree and order selected, the more memory that is used and the greater the computation times. The speed of operation should be evaluated against desired accuracy to determine the degree and order to utilize.

The gravitational potential function (V) is defined as:

$$V = \frac{GM}{r}\left[1 + \sum_{n=2}^{n_{max}} \sum_{m=0}^{n} \left(\frac{a}{r}\right)^n \overline{P}_{nm}(\sin\phi')\left(\overline{C}_{nm}\cos(m\lambda)\right) + \left(\overline{S}_{nm}\sin(m\lambda)\right)\right]$$

where,

V	=	Gravitational potential function (m^2/s^2)
GM	=	Earth's gravitational constant = 398601620553728.0
r	=	Distance from the Earth's center of mass
a	=	Semi-major axis of the WGS 84 Ellipsoid
n,m	=	Degree and order, respectively (user-specified)
ϕ'	=	Geocentric latitude
λ	=	Geocentric longitude = geodetic longitude
$\overline{C}_{nm}, \overline{S}_{nm}$	=	Normalized gravitational coefficients (read from file)

In addition to the descriptions of the algorithms, FORTRAN source code for performing the computation of the gravitational potential, V, was also obtained from the aforementioned websites. These algorithms were converted to C and used to compute the gradient of V, which is equal to the gravitational acceleration.

Therefore, gravitational acceleration is calculated by:

$$\overline{A}_G = \nabla V$$

where

$\overline{A}_G$	=	gravitational acceleration (m/s^2)
V	=	gravitational potential function (m^2/s^2)

5.7.3.1.5 Acceleration - Rotating Earth

The acceleration vector computed in Subsection 5.7.3.1.4 must be further adjusted if Earth rotation effects are modeled. These effects impact the gravitational forces in the x and y directions.

$$Ax = Ax + \omega E\ (\ 2\ Vy + \omega E\ (Xx))$$

$$Ay = Ay + \omega E\ (-2\ Vx + \omega E\ (Xy))$$

where

A_n	-	acceleration in ECEF n-direction (m/sec^2)
X_n	-	missile position in ECEF n-direction (m).
V_n	-	missile velocity in ECEF n-direction (m).
ωE	-	Earth rotation rate (rads/sec).

If operating with the rotating Earth, these acceleration terms are computed and summed into the acceleration due to gravity prior to rotating the acceleration into the missile velocity axis frame.

5.7.3.1.6 Integration Algorithm

All derivatives in the model are integrated using a second-order Runge Kutta method. This method is particularly effective for the solution of differential equations containing non-smooth functions. The missile model contains many non-smooth functions, including linear interpolation with table-ups and discontinuities in the missile state definition during the transitions between the different flight sections.

The numerical integration for expressions defined by

$$y = \int f(y,t)\,dt$$

is performed using the formula

$$y_{j+1} = y_j + \Delta t f(y^*_{j+1/2}, t_{j+1/2})$$

where

$$y^*_{j+1/2} = y_j + \frac{\Delta t}{2} f(y_j, t_j)$$

$$t_{j+1/2} = t_j + \frac{\Delta t}{2} \quad .$$

5.7.3.2 Missile Guidance Options

The guidance options described in this section are available for interceptors and ballistic missiles.

5.7.3.2.1 Pitch Profile Guidance

Pitch profile guidance is used to control the angle at which thrust is applied during the missile flyout. Both pitch angle and rate are measured clockwise down

from the local vertical. The acceleration steering command, a, in the pitch channel is generated by a closed-loop controller, based on angle of attack, i.e., the difference between the commanded thrust vector pitch angle and the current flight path angle, as follows:

$$a = \left(T\sin\alpha + qSC_{1\alpha}\alpha\right)/\text{Mass}$$

where

T - axial thrust (N)
α - angle of attack (rad)
q - dynamic pressure (N/m^2)
S - aerodynamic reference area (m^2)
$C_{1\alpha}$ - linearized coefficient of lift (/rad).

For ballistic missile, the user-specified pitch angle may be altered due to pitch iterations for Lofted, Depressed, Non-Thrust Terminating ReEntry Angle, Thrust Terminating ReEntry Angle, and Minimum Energy guidance types. In this case, the pitch value that will be used from the beginning iteration stage, flight section and flight mode until the ending iteration stage, flight section and flight mode will be the iterated pitch value.

When operating in pitch profile mode, azimuth control is also conducted as described in Subsection 5.7.3.4.7.8.

5.7.3.2.2 Flight Path Angle Guidance

The flight path angle guidance mode attempts to adjust the missile flight to maintain the specified flight path angle. The flight path angle is the angle between the velocity vector and the local horizontal. The acceleration command in the pitch channel is generated by a closed-loop controller, which multiplies the difference between the commanded flight path angle and the current flight path angle by a gain to generate the acceleration steering command. The controller is shown in ***Figure 5.7-6.*** The dynamics of the closed-loop system can be approximated as a first-order transfer function with a time constant, *t*. The time constant is the time it takes the system to reach 63.2% of its steady state value in response to a step input. The time constant is related to the gain, k, by:

$$k = \frac{V}{\tau}$$

where

V = missile velocity (m/sec).

The time constant, t, is converted to an equivalent 90% response time by:

$t_r = 2.30259\tau$

This conversion is performed to provide a uniform response-time specification that is consistent with the 90% response time used to specify the speed of response for the autopilot/airframe dynamics second-order transfer function described earlier.

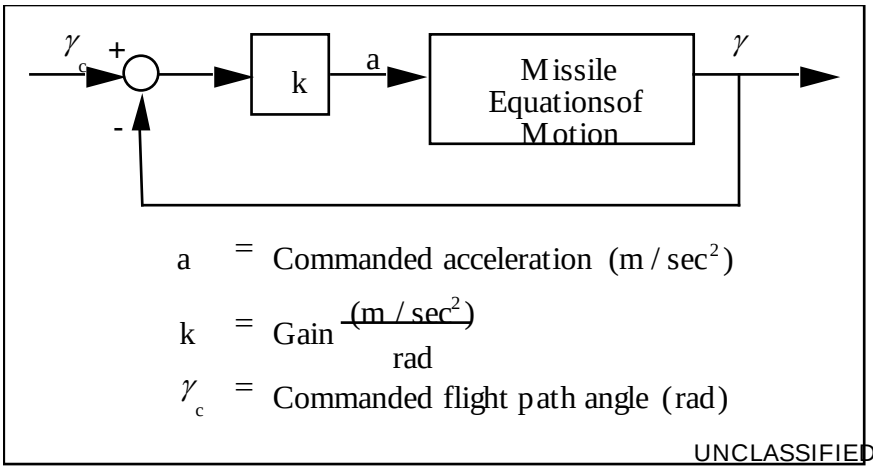

Figure 5.7-6 Flight Path Angle Closed-Loop Controller

Since a response lag is already inherent in the implementation of the flight path controller, the second-order transfer function is bypassed in computing acceleration constraints on the resulting commanded acceleration, a.

5.7.3.2.3 Ballistic

Ballistic flight is modeled simply as:

$a_y = 0$

$a_z = 0$

The only forces acting on the missile are gravity and drag; no steering acceleration is applied.

5.7.3.2.4 Ballistic PPN Guidance

The Ballistic PPN flight mode guides the missile to the predicted target location using predictive proportional navigation. Ballistic PPN guidance differs from the pure ballistic flight in that a predicted intercept is periodically computed based on the current state of the missile continuing with pure ballistic flight. The ballistic PPN will only be pure ballistic flight if the remaining flight sections, ignoring endgame homing, are ballistic; otherwise, the PPN will be applied to a predictive flight of the remaining flight sections. If the computed miss distance is outside of the user input tolerance for the ballistic PPN, then proportional navigation is applied based on the miss distance. The prediction is performed at each integration interval, until the miss distance is predicted to be within the

tolerance. Once the miss distance is within tolerance, the missile will fly according to ballistic guidance until a subsequent target state update, at which time miss distance will be evaluated and PPN initiated to bring the missile back within tolerance. Ballistic flight is used for the Ballistic PPN flight mode during the launch iteration.

When performing ballistic PPN guidance, the missile is flown with ballistic guidance and the target is flown ballistically if a missile or in a straight line for other threat types until the missile passes the target, i.e. achieves the point of closest approach. The vector from the missile's predicted position at intercept to the target at intercept is next computed, i.e. the miss vector. This miss vector is rotated into the missile velocity axes coordinate frame at the missile's current position. The acceleration is then computed in both the y and z channels based on the proportional navigation gain computed by a table look-up as a function of time-to-go.

$$a_y = K\frac{M_y}{T_{go}^2}$$

$$a_z = K\frac{M_z}{T_{go}^2}$$

where,

a_y, a_z = acceleration in y and z axes.

K = proportional navigation gain.

M_y, M_z = Miss vector.

T_{go} = Time until intercept.

The proportional navigation gain is input as a function of time-to-go. Time-to-go is calculated every integration time step using the algorithm in Subsection 5.7.3.5.7. For PN gain table look-ups, the absolute value of time-to-go is used.

PN gain values are not interpolated from the PN table. Instead, the PN gain value of the nearest time-to-go less than the observed time-go-value is used. If the time-to-go value is less than the first value on the table, the first gain input is used as the PN gain. If the time-to-go is greater than the last value on the table, the last input is used as the PN gain.

The PN gain values for interceptor homing are input in a table as a function of slant range. Slant range is measured from the launcher location to the predicted intercept point. PN gain values are not interpolated from the PN table. Instead, the PN gain value of the nearest slant range less than the calculated slant range value will be used. If the slant range value is less than the first value on the table, the first gain input is used as the PN gain. If the slant range value is greater than the last value on the table, the last input is used as the PN gain. When reading in

previous airframe element versions, the homing guidance gain table will have a single entry. For the single entry, the slant range value will be zero and the guidance gain value will be the homing guidance gain value specified in the older version airframe element.

The desired acceleration based on the PPN gain is calculated once per scenario interval. It is then applied across the entire scenario interval, saving significant computation time over computing the PPN gain on each integration step. Application of the acceleration over an entire scenario interval could result in overshooting the goal of zero miss distance. An acceleration limit is computed based on the acceleration needed to zero the miss distance over a scenario interval. The gain is set to 2.0 and the scenario interval is used as the time parameter to compute the acceleration that will result in zero miss distance at the end of the scenario interval.

$$a_y = 2.0 \frac{M_y}{I^2}$$

$$a_z = 2.0 \frac{M_z}{I^2}$$

where,

a_y, a_z = acceleration in y and z axes.

M_y, M_z = Miss vector.

I = Scenario Interval.

By comparing between the acceleration found using PN gain values and the acceleration determined for a scenario interval, the acceleration chosen is then limited to the minimum of these two values.

The speed control option causes the terminal object to perform angle of attack adjustments in order to achieve a desired speed at a specified altitude. The speed control option can only be used with the ballistic PPN flight mode. Speed control is implemented by first determining the acceleration needed to achieve the desired speed. This acceleration is calculated based on the user specified speed and values determined from the ballistic PPN flyout. There are two points of interest during the PPN flyout in which values of missile speed, altitude and time since launch are collected and stored for interpolation purposes. The first point of interest is the integration time step right before the missile crosses the desired altitude and the second point is the time step right after the desired altitude is crossed. The speed and time since launch can be linearly interpolated to the user specified altitude using these stored values. The difference between the specified speed and the interpolated speed at the desired altitude can be used in conjunction with the time-

to-go to the desired altitude to determine the acceleration needed to achieve the specified speed.

$$\dot{v} = \frac{\Delta v}{Tgo}$$

where

$\dot{v}$ = Acceleration needed to achieve desired speed (m/s^2).

Δv = Difference between desired speed and interpolated speed (m/s)..

Tgo = Time until desired altitude (s).

The change in velocity magnitude of a non thrusting vehicle is due to aerodynamic and gravitational forces. Setting the above acceleration equal to the change in velocity magnitude allows for the determination of a drag coefficient needed to acquire the desired speed.

$$C_d = \frac{m \cdot (G_x - \dot{v})}{q \cdot S}$$

where

C_d = Drag coefficient

m = Mass of the missile (kg)

G_x = Gravitational acceleration component coincident with the velocity vector (m/s^2)

$\dot{v}$ = Acceleration needed to achieve desired speed (m/s^2)

q = Dynamic pressure (N/m^2)

S = Aerodynamic reference area (m^2)

Drag coefficients are stored in two dimensional tables with independent values of Mach number and angle of attack. The angle of attack which produces the calculated drag coefficient can be interpolated from the two dimensional table given the current Mach number of the missile. A steering acceleration can be determined from the angle of attack and is restricted to the pitch channel of the missile and restricted further to only pitch up.

$$a_z = \sqrt{\frac{\sin(\alpha)^2 \cdot (q \cdot S \cdot Cl_\alpha)^2}{m^2 - a_y{}^2}}$$

where

a_z = Steering acceleration in the pitch channel (m/s^2)

a_y = Steering acceleration in the yaw channel (m/s²)
α = Angle of attack (rad)
Cl_α = Linear lift coefficient
m = Mass of the missile (kg)

The angle of attack required for speed reduction is balanced with the angle of attack required for reducing the miss distance. If a greater angle of attack is required from the PPN steering commands, then that angle of attack is used; otherwise, the angle of attack generated for speed control is applied.

5.7.3.3 Post Boost/Terminal Maneuvers Guidance

Post Boost/Terminal maneuvers can be used to achieve a desired flight profile relative to a ground target. Post Boost maneuvers can also be used to search and guide to a target using an onboard seeker. Terminal maneuvers can be defined for RV, debris, decoy, and chaff object types and for an interceptor. Maneuver execution is controlled using the ballistic, pull up, pull down, ballistic PPN, and FPA flight modes. The pull up and pull down maneuvers are described in section 5.7.3.3.1. The ballistic, ballistic PPN, and FPA flight modes are described in section 5.7.3.2. The ballistic aim point option determines whether or not a maneuver is flown during the launch iteration process. If the ballistic aim point option is selected, then the launch iteration uses ballistic flight during the final stage of flight, in place of any defined maneuvers, to select a trajectory which impacts the user defined ballistic aim point as described in section 5.7.3.10. At least one associated terminal object must use the ballistic PPN guidance mode in order to use the ballistic aim point option. A ballistic aim point maneuver is useful against targets which require detection via an onboard sensor during the terminal guidance phase in order to guide to the target. The sensor detection of the target allows engagement of moving ground targets or stationary targets with tracks containing a high level of error which require seeker guidance for improved accuracy. The seeker can be modeled either explicitly or implicitly as described in section 5.7.3.4.10. As the missile's seeker activates and detects the target, the perceived target location is updated using the track on the target, including any track error. If the seeker does not detect the target, then the terminal object guides to the perceived target location that was provided at the time of launch. The weapon CEP is applied relative to the ballistic aim point if the ballistic aim point option is selected.

If the ballistic aim point option is not selected and the terminal object has any flight modes which are not ballistic, the terminal maneuver is flown during the launch iteration process used to find a launch solution as described in section 5.7.3.10. The ballistic PPN flight mode will operate the same as the ballistic flight mode for terminal maneuvers that do not use the ballistic aim point option. The

weapon CEP is applied relative to the true target location to compute the offset from the perceived target location as described in MM section 4.4.5.

5.7.3.3.1 Pull Up/Pull Down

The pull up flight mode allows in-plane lateral accelerations to be applied in the up or down direction. The Angle of Attack option orients the object at a user specified angle of attack in the up direction relative to the local horizontal. The pull down flight mode orients the object at a user specified angle of attack in the down direction relative to the local horizontal. The angle of attack is defined as the angle within the trajectory plane between the missile orientation and velocity vectors.

The acceleration applied to the object is computed from the drag and lift that is caused by the angle of attack. The acceleration consists of an axial component and a lateral component.

The lateral or steering acceleration is related to the angle of attack by

$$a_{steering} = \frac{q \cdot S \cdot C_{1\alpha} \cdot \sin(\alpha)}{m}$$

Where

α	-	Angle of attack (rad)
$a_{steering}$	-	Steering acceleration in the pitch direction (m/sec^2)
q	-	Dynamic pressure (N/m^2)
S	-	Aerodynamic reference area (m^2)
m	-	Mass (kg).
$C_{1\alpha}$	-	Linearized coefficient of lift (/rad).
α	-	Angle of attack (rad)

For the small angles of alpha that are valid in EADSIM the following approximation can be used

$$\sin(\alpha) \approx \alpha$$

The steering acceleration is therefore computed using

$$a_{steering} = \frac{q \cdot S \cdot C_{1\alpha} \cdot \alpha}{m}$$

If the commanded lateral acceleration option is selected, then an attempt is made to apply the commanded acceleration using aerodynamics and the available divert capability. The divert thrust is only applied if the maximum angle of attack is reached and still more acceleration is necessary. The divert capability is described in section 5.7.3.1.3. The pull up flight mode applies the acceleration in

the up direction while the pull down flight mode applies the acceleration in the down direction.

The RV angle of attack during the maneuver is realistically constrained by aerodynamic and structural limitations. Lateral acceleration for steering is aerodynamically limited according to a user-specified maximum angle of attack. A hard limit is placed on lateral steering acceleration representing maximum G-loading structural capabilities.

The steering acceleration is then used as described in section 5.7.3.1. A tactical missile airframe does not respond immediately to lateral acceleration commands; it takes some time for the airframe to rotate and produce lift with the angle of attack under control of the autopilot. A second order transfer function is used to model this response lag as described in section 5.7.3.1.3. The acceleration in the yaw direction is zero for the pull up and pull down flight modes.

The axial acceleration is then computed as described in section 5.7.3.1. The axial acceleration depends on the coefficient of drag which is a function of the angle of attack and Mach number.

5.7.3.4 Ballistic Missile Guidance Options

All ballistic missiles are comprised of one or more missile stages that allow them to fly to their desired range. Typically, the stages are booster, sustainer, post boost vehicle (PBV) and Reentry Vehicle (RV). Staging events define both mass changes and thrust changes. Booster thrust is usually greater than sustainer thrust. PBVs apply various thrust levels while maneuvering for RV deployment, and the RV thrust is usually zero. The range of guidance options available at various stages of missile flight include ballistic, pitch profile, PBV ignition, PBV turn to RSA, PBV turn to RIA, targeting, PBV disposal, pull up, pull down, flight path angle (FPA), and ballistic PPN. The pitch profile, FPA, ballistic, ballistic PPN, pull up, and pull down flight modes are available for interceptors and ballistic missiles and are therefore described in section 5.7.3.2. PBV ignition, PBV turn to RSA, PBV turn to RIA, targeting, and PBV disposal are unique to ballistic missiles and are described in the following sections. A special rail option is used immediately after launch at which point the missile is commanded to follow a straight line for a hard coded distance of 5 meters from the launch point.

5.7.3.4.1 Ballistic Missile Multiple Object Type Identification

A number of different unique types of objects may be created during the flight of a ballistic missile. At the most basic level, the missile may be composed of a single object that is propagated all the way from boost to impact. Other missiles may launch as a single entire missile object through boost, but then separate into one or more objects at different stages in the flight, such as a spent booster tank and an RV. These missiles may create other debris objects and penetration aids as

they progress through their flight stages. Each of these objects may be explicitly flown with the internal 3DOF ballistic missile flight model, allowing the missile pieces to be detected, classified, tracked and engaged. Alternatively, these multiple object missiles may be represented simply with a single critical path object reflecting the trajectory from launch to impact.

For creation of unique detectability and engageability parameters for each object type, as well as for ease of identification of different missile objects in Scenario Playback, each object type is identified by a unique compound object type name. The word "compound" is used to refer to the object naming convention that combines the overall missile element name and the name of the particular separating object defined in the missile's stage summary table. These compound object names can be used when defining classes, indicating objects/classes of interest for various ruleset options, etc. These names can also be specified in the construction of the DIS element cross reference to allow better mapping to the characteristics of each object type.

As objects are activated by the missile model, they take on a user-specified object type. For example, the "at launch" object type indicates that the entire missile is intact, i.e., no separation events have occurred. At first stage separation, the object type transitions to a type that indicates the first stage has fallen away. At the same time, the spent first stage tank may optionally be generated. These object type transitions are logged for post processing and scenario playback support. Similar object type transitions may occur throughout missile flight until the missile transitions to a terminal object, i.e., debris, RV, Decoy, or Chaff.

Compound object names are constructed as shown below. A "~" is used as the separator character between the overall weapon element name and the missile object name. In the example below, "TBM" is the overall weapon element name. Additionally, the TBM weapon element contains a stage summary table specifying object names, Booster1, Booster1Tank, Booster2, Booster2Tank, PBV1, DECOY1, and RV1. Over the course of the missile's trajectory, the following missile objects will be created:

TBM~Booster1
TBM~Booster1Tank
TBM~Booster2
TBM~Booster2Tank
TBM~PBV1
TBM~DECOY1
TBM~RV1

where:

TBM~Booster1	– indicates entire missile
TBM~Booster1Tank	– indicates spent Booster1
TBM~Booster2	– indicates remainder of entire missile minus Booster1
TBM~Booster2Tank	– indicates spent Booster2

TBM~PBV1	– indicates remainder of entire missile minus Booster1 and Booster2
TBM~DECOY1	– indicates DECOY1 object type (penaid)
TBM~RV1	– indicates RV1 object type (re-entry vehicle)

The various data elements contain numerous lists that include ballistic missile element names as a potential target type, such as Pk tables, fire doctrines, and NCTR probability tables. Both the overall missile element name and the compound object type names are available in the selection lists. When a lookup is performed on target type, the compound object type name is considered a more detailed definition of the object than the overall missile element name. If the specific compound object name is not found in a list but the missile element name is, then the data for the missile element is used.

5.7.3.4.2 Ballistic Missile Objects, Stages, and Flight Sections

The ballistic missile flyout is described by the use of multiple flight sections combined into missile objects, which are deployed in stages. Multiple missile objects can be deployed during a single stage of flight. If only a single object is deployed for a stage, the stage corresponds directly with the object. Each flight section consists of a separate specification of aerodynamic, propulsion, and mass properties, as well as guidance commands. An object can consist of a single flight section or multiple flight sections. Both dry and fuel mass are specified for each object. If a single object has multiple flight sections, the transition from one flight section to another for the object utilizes the remaining fuel mass from the previous flight section, whereas, a transition between objects resets the dry and fuel mass specified for the next object representing a mass transition event, i.e. staging event. The mathematical formulation for each flight section is identical.

Each stage/object definition contains specifications for dry mass, fuel mass, and flight section data. The dry mass and fuel mass specified for any given object reflects only the dry mass and fuel mass of that object. The dry mass and fuel mass of all objects defined in subsequent flight stages are added into the dry mass and fuel mass of the current object to determine the current total mass. The lone exception is the mass of debris objects. The debris objects are assumed to come from the mass of the prior stage and are therefore excluded from the current total mass. Some data sources report and some models utilize a specification of the total mass at the beginning of each flight stage; however, the EADSIM algorithm expects the mass to only be the dry mass and fuel mass of the individual object. If total mass is entered for each object, the missile will be less capable as a result of the increased missile mass.

Each flight section consists of specifications for:

- Transition trigger
- Maximum dynamic pressure

- Specific impulse
- Nozzle exit area
- Aerodynamic reference area
- Aerodynamic tables for lift and drag
- Maximum Angle of Attack
- Response Time
- Numerical integration step size
- Guidance instructions

Within a flight section, guidance modes consist of a table specifying periods of operation, by means of sequentially listed guidance mode start triggers, for combinations of the available guidance modes:

- Ballistic
- Pitch Profile
- Post Boost Vehicle (PBV) Maneuver
- Reentry Vehicle (RV) Maneuver

This table of sequentially listed guidance modes also contains a specification of the vacuum axial thrust level to be applied during the guidance mode.

Flight section and flight mode transitions can be triggered using time, altitude relative to target while ascending, altitude relative to target while descending, altitude MSL while ascending, altitude MSL while descending, altitude rate, or completion of previous mode. Flight sections also have the additional transition options of time in section, thrust terminate, and maximum dynamic pressure (max Q). The flight section and mode transition triggers are start triggers, and therefore cause a transition from the preceding flight section/mode in the list to the associated flight section/mode. The max Q is an end trigger, and therefore if the dynamic pressure exceeds the max Q then this flight section transitions to the next flight section. If the time in section option is selected, the section will start at the specified time relative to the start of the previous flight section. If the transition on thrust terminate option is selected for a flight section then the section will start that section upon thrust termination of the previous flight section. For flight sections, time is defined as the time since launch at which the flight section starts. For flight modes, time is defined as the time since the current flight section start at which the flight mode begins.

If one of the altitude options is selected, the flight section/mode will start upon crossing the specified altitude. The altitude can be defined relative to the target or MSL. If one of the ascending options is selected then the flight section/mode will start when the missile crosses the specified altitude while ascending. If one of the descending options is selected then the flight section/mode will start when the missile crosses the specified altitude while descending. If the altitude rate trigger is selected, the flight section/mode will start when the altitude rate of the missile crosses the specified altitude rate. The altitude rate can be

specified as positive or negative. If completion is selected then the flight mode will start when the previous flight mode is complete. Completion is only available for the PBV ignition, PBV turn to RSA, PBV turn to RIA, targeting, go to, and PBV disposal flight modes. Completion allows a flight mode to start once the specific goal defined by the maneuver type is achieved for the previous flight mode.

If the transition on detect option is selected for a flight mode, the mode will transition from the previous mode in the list to the associated mode if the target is detected or if the transition trigger condition is met. This option is available in addition to the other trigger options in order to allow flight mode transitions to occur even if the target is not detected.

Describing the missile as a series of flight stages, objects, and sections provides the capability to model many types of ballistic missiles. For many cases, an object will correspond to a single flight section, but this does not necessarily have to be the case. Within a single object, multiple flight sections could represent aerodynamic or thrust profile changes during a single stage of missile flight.

Following is a description of how the acceleration commands are formulated for the different ballistic missile guidance options.

5.7.3.4.3 Ballistic PPN

The Ballistic PPN flight mode guides the missile to the predicted target location using predictive proportional navigation. The prediction is performed at each integration interval, until the miss distance is predicted to be within the tolerance. Once the miss distance is within tolerance, the missile will fly according to ballistic guidance until a subsequent target state update, at which time miss distance will be evaluated and PPN initiated to bring the missile back within tolerance.

Missiles can receive an update on the perceived target state from a track produced by an onboard sensor. See section 5.7.3.2.4 for details on the acceleration calculations for the Ballistic PPN guidance mode.

5.7.3.4.4 Flight Path Angle Guidance

The flight path angle guidance mode attempts to adjust the missile flight to maintain the specified flight path angle. The ballistic missile allows the flight path angle during the commanded flight path angle guidance mode to be defined as a table of flight path angle commands versus time in section. If the apply PPN option is selected for the flight path angle guidance mode, the specified PPN gain will be applied as described for ballistic PPN based on the remaining time to go in the flight section in an attempt to reduce the miss distance as the terminal object closes on the target location. The PPN gain is not applied during the launch iteration.

See section 5.7.3.2.2 for details on the acceleration calculations for the Flight Path Angle mode.

When operating in flight path angle mode, azimuth control is also conducted as described in section 5.7.3.4.7.8.

5.7.3.4.5 Pitch Profile Guidance

Each pitch profile guidance phase consists of a start time (s), pitch angle (deg), pitch rate (deg/s), axial thrust value (N) and a flag for performing a gravity turn. Both pitch angle and rate are measured clockwise down from the local vertical. The start time tells the model when to transition to the guidance phase. The pitch angle defines how the missile should be aligned at the start of the phase. If it is the first guidance phase or if the gravity turn option is selected, the pitch angle tells the model to rotate the missile body-axis to this absolute angle. Otherwise, the pitch angle is the relative number of degrees to rotate the missile from its current orientation. The pitch rate defines the speed at which the missile should pitch until the end of the current guidance phase. The gravity turn flag tells the model to limit the angle of attack, the angle between the velocity vector and the missile body axis, to zero. Thrust and drag are then aligned with the velocity vector so that gravity is the only force that causes the missile to pitch. This multiphase capability allows for vertical flight segments, constant attitude flight segments, thrusted and unthrusted gravity turns, and pitch programs whose inputs can be found in weapon description documents. See section 5.7.3.2.1 for details on the acceleration calculations for the Pitch Profile flight mode.

5.7.3.4.6 Next Object

The Next Object flight mode allows the deployment of RVs as well as multiple penetration aids (Penaids) and other objects, such as debris, decoys and chaff clouds. Mass, guidance, signature, aerodynamics and demise altitude are specified for each object as well as a deployment timing delay and separation velocity. Objects may be deployed in a variety of configurations. An RV may be deployed with a group of objects that includes decoys and chaff. Or, a bare RV may be deployed with no decoy or chaff. Additionally, a group of objects may be deployed that include only chaff, only decoys, or chaff and decoys together. When operating with a single RV, the last stage spent booster may also be part of the train. With the Create New Train option selected, all of the objects specified in a Next Object flight mode window, except for boosters and PBVs, will be part of the same train; i.e., a group of objects from the same missile launch proceeding along similar ballistic trajectories. With the Create New Train option deselected, all of the objects specified in the Next Object flight mode window, including the boosters and PBVs, will be added to the previous train of objects. Also with the Create New Train option deselected, the booster or PBV that is deploying the objects is included in the previous train of objects. The true knowledge of which objects are part of the same train is maintained by making all members of the train except for the critical path

object itself children of the critical path object and by assigning a non-zero Train ID and Object ID to each object, see Section 5.7.3.4.6.1 below for details. If the Next Object Flight Mode contains a booster or PBV deployment and the Create New Train option is selected, all objects but the booster or PBV will be assigned a common train ID. In this case, it is assumed that the booster or PBV will generate sufficient separation to the other ballistic objects to not be part of the same train. If the Next Object Flight Mode contains a booster or PBV deployment and the Create New Train option is deselected, all objects including the booster or PBV will be assigned the common train ID that was used for the last train of objects. In the special case of deployment from a PBV and no RV is specified on the Next Object flight mode list of objects, the first RV object in the stage summary table will be used for internal prediction of RV flight to impact. During PBV maneuvers, the prediction of RV flight to impact is used to determine when to terminate maneuvers and be able to deploy the specified critical path object on a trajectory that passes through the target. The only difference between the paths of the objects in the train will be slight perturbations due to small user-specified separation velocities imparted at deployment.

Only one Next Object flight mode may be specified for a booster. This flight mode is specified at the end of the booster stage. There can be multiple deployments within this single Next Object flight mode; however, they will all occur at the same time. There are three possible combinations of object type deployments from a booster stage. These combinations are Booster/Debris, PBV/Debris, and RV/Decoy/Chaff/Debris. Multiple booster stages may be modeled by deploying a booster from a booster stage, but only a single Booster may be deployed and once a Booster has been selected for deployment as a next object, it is no longer possible to deploy any other objects but Debris from that stage. Multiple deployments are allowed for all other object types.

A PBV stage can deploy any combination of the following object types: RV, Decoy, Chaff, or Debris. Multiple, varied deployments are allowed for all of these object types from a single PBV. Although at least one object to be deployed must be specified for each Next Object mode, there is never a requirement that it be any specific object type, nor that there be more than one object deployed. The only requirement is that the objects deployed by a stage fall within one of the legitimate combinations as described above.

For a missile with no PBV, the final stage booster deploys the objects specified in its Next Object flight mode against the next available target on the launching platform's target list. While boosters may only specify a single Next Object flight mode at the end of the stage, PBVs may specify multiple Next Object flight modes. PBVs may be used to deploy trains of objects against multiple targets. In this case, each execution of a PBV Next Object flight mode increments forward to the next available target on the launching platform's target list and deploys all

objects specified in the Next Object flight mode against that target. Targets are processed sequentially as specified in the target list.

During execution of Next Object flight modes, object depletion takes place. At some point, the number of available objects of a given type may go to zero. For a Next Object flight mode where no objects of any of the specified types are available for deployment, the flight mode is skipped. Otherwise, if at least one object of any of the specified types is available for deployment, the available objects are deployed against the associated target.

An additional parameter, cloud growth rate, is specified for Chaff type objects. The point mass associated with the propagated chaff cloud establishes the center of the cloud. The cloud growth rate is then used at any time since deployment to determine the cloud size. The truth location of the chaff cloud center along with the truth based time since deployment information determines the current size of the cloud for such operations as determining if other objects are masked by the chaff cloud.

5.7.3.4.6.1Multiple Object Traceability

As explained above, a train is a group of objects from the same missile launch proceeding along similar trajectories, accomplished by deployment from the same Next Object flight mode. Knowledge of objects generated from a single launch event is maintained by the setting of a complex ID. Knowledge of objects in the same train is maintained by making them all children of the same critical path object and by assigning a non-zero Train ID. Trains are numbered sequentially from 1, in the order they are deployed by the missile. Thus, if the Next Object window defines the sequence of objects "Chaff-RV-Decoy," with the RV as a critical path object, the Chaff and Decoy objects become children of the RV object and are assigned a Train ID equal to the current number of trains deployed by this launch complex plus one, i.e. if there were two trains in this launch complex before this deployment event the new train is given a train ID of three. Additionally, the Chaff and Decoy objects are given an Object ID indicating the order in which they deployed within each train,i.e. in the example above, if the Chaff deploys first then it is ObjectID of 1 and Decoy is ObjectID of 2. If none of the objects in the Next Object flight mode is a critical path object, the first RV in the window is treated as the critical path object. If there is no RV included in the Next Object flight mode, then the first object to be deployed is treated as the critical path object. Only the first critical path object to be deployed from each Next Object flight mode window will have meaning as a critical path object, all others just add to the current train.

When a booster object executes a Next Object flight mode, it does not create a new missile entity for the critical path object. Rather, its missile and Truth ID transition to the critical path object.

5.7.3.4.6.2 Object Traceability Examples

The following are examples of the traceability.

Table 5.7-1 Example 1: Single Booster/Separating RV with explicit spent tank, Critical Path goes to RV

Stage Number	Object Name	Stage Type	#Objects	Critical Path Flag	Deploy From
1	Booster1	Booster		YES	
2	Booster1Tank	Debris		NO	Booster1
2	RV1	RV	1	YES	Booster1

Time of launch:

The full missile (Missile 1) is created, establishing the launch complex for all future object deployments.

Missile 1 Type = TBM~Booster1
ID = 28766
ComplexID = 28766
TrainID = 0
ObjectID = 0

After Booster1 Burnout:

The missile ID follows the critical path. Missile 1 maintains the ID 28766, but its object type is now RV1. The spent booster tank is deployed as a new missile object (Missile 2) in the same train as the RV object.

Missile 1 Type = TBM~RV1
ID = 28766
ComplexID = 28766
TrainID = 1
ObjectID = 1

Missile 2 Type = TBM-Booster1Tank
ID = 28767
ComplexID = 28766
TrainID = 1
ObjectID = 2

Table 5.7-2 Example 2: Single Booster/Single RV with explicit spent tank, Critical Path goes to Tank

Stage Number	Object Name	Stage Type	#Objects	Critical Path Flag	Deploy From
1	Booster1	Booster		YES	
2	Booster1Tank	Debris		YES	Booster1
2	RV1	RV	1	NO	Booster1

Time of launch:

The full missile (Missile 1) is created, establishing the launch complex for all future object deployments.

Missile 1 Type = TBM~Booster1
ID = 28766
ComplexID = 28766
TrainID = 0
ObjectID = 0

After Booster1 Burnout:

The missile ID follows the critical path. Missile 1 maintains the ID 28766, but its object type is now Booster1Tank. The RV is deployed as a new missile object (Missile 2) in the same train as the spent tank.

Missile 1 Type = TBM~Booster1Tank
ID = 28766
ComplexID = 28766
TrainID = 1
ObjectID = 1

Missile 2 Type = TBM~RV1
ID = 28767
ComplexID = 28766
TrainID = 1
ObjectID = 2

Table 5.7-3 Example 3: MIRV Example of Object Transition – Critical Path goes to RV#1, Create New Train selected on all Next Object flight mode windows

Stage Number	Object Name	Stage Type	#Objects	Critical Path Flag	Deploy From
1	Booster1	Booster		YES	
2	Booster1Tank	Debris		NO	Booster1
2	Booster2	Booster		YES	Booster1
3	Booster2Tank	Debris		NO	Booster2
3	PBV1	PBV		YES	Booster2
4	DECOY1	DECOY	50	NO	PBV1
4	RV1	RV	10	YES	PBV1

Next Object Window for First RV1 Launch

Object	#	Timing
DECOY1	2	5
RV1	1	10

Next Object Window for Second RV1 Launch

Object	#	Timing
RV1	1	3
DECOY1	1	5

Time of launch:

The full missile (Missile 1) is created, establishing the launch complex for all future object deployments.

Missile 1 Type = TBM~Booster1
ID = 28766
ComplexID = 28766
TrainID = 0
ObjectID = 0

After Booster1 Burnout:

The missile ID follows the critical path. Missile 1 maintains the ID 28766, but its object type is now the second stage booster, Booster2. Boosters are not assigned train IDs, so the first stage spent booster tank is deployed as a new missile object (Missile 2) and is the sole member of the first train.

Missile 1 Type = TBM~Booster2
ID = 28766
ComplexID = 28766
TrainID = 0
ObjectID = 0

Missile 2 Type = TBM~Booster1Tank
ID = 28767
ComplexID = 28766
TrainID = 1
ObjectID = 1

After Booster2 Burnout:

The original missile ID continues to follow the critical path. Missile 1 maintains the ID 28766, but its object type transitions to PBV1. The first stage tank missile object continues to fly. PBV objects are not assigned train IDs, so the second stage spent booster tank is deployed as a new missile object (Missile 3) in a new train.

Missile 1 Type = TBM~PBV1
ID = 28766
ComplexID = 28766
TrainID = 0
ObjectID = 0

Missile 2 Type = TBM~Booster1Tank
ID = 28767
ComplexID = 28766

TrainID = 1
ObjectID = 1

Missile 3 Type = TBM~Booster2Tank
ID = 28768
ComplexID = 28766
TrainID = 2
ObjectID = 1

After Decoy#1 and Decoy#2 Deployment:

Two decoys are launched by the PBV before it launches the first RV. As the decoys are not critical path objects, the PBV retains the Missile 1 object. The decoys are created as new missile objects (Missile 4, 5) in a new train.

Missile 1 Type = TBM~PBV1
ID = 28766
ComplexID = 28766
TrainID = 0
ObjectID = 0

Missile 2 Type = TBM~Booster1Tank
ID = 28767
ComplexID = 28766
TrainID = 1
ObjectID = 1

Missile 3 Type = TBM~Booster2Tank
ID = 28768
ComplexID = 28766
TrainID = 2
ObjectID = 1

Missile 4 Type = TBM~Decoy1 (first decoy)
ID = 28769
ComplexID = 28766
TrainID = 3
ObjectID = 1

Missile 5 Type = TBM~Decoy1 (second decoy)
ID = 28770
ComplexID = 28766
TrainID = 3
ObjectID = 2

After RV#1 Deployment:

The RV takes over Missile 1 as the new critical path object, but it is assigned as the third object in the same train as the first two decoys. The PBV is created as a new missile object (Missile 6).

Missile 1 Type = TBM~RV1
ID = 28766
ComplexID = 28766
TrainID = 3
ObjectID = 3

Missile 2 Type = TBM~Booster1Tank
ID = 28767
ComplexID = 28766
TrainID = 1
ObjectID = 1

Missile 3 Type = TBM~Booster2Tank
ID = 28768
ComplexID = 28766
TrainID = 2
ObjectID = 1

Missile 4 Type = TBM~Decoy1 (first decoy)
ID = 28769
ComplexID = 28766
TrainID = 3
ObjectID = 1

Missile 5 Type = TBM~Decoy1 (second decoy)
ID = 28770
ComplexID = 28766
TrainID = 3
ObjectID = 2

Missile 6 Type = TBM~PBV1
ID = 28771
ComplexID = 28766
TrainID = 0
ObjectID = 0

After RV#2 Deployment:

The first RV retains the Missile 1 object. However, as a critical path object, the second RV takes over Missile 6 from the PBV and a new missile (Missile 7) is created to allow the PBV to continue to fly. The second RV becomes the first object in a new train.

Missile 1 Type = TBM~RV1
ID = 28766

ComplexID = 28766
TrainID = 3
ObjectID = 3

Missile 2 Type = TBM~Booster1Tank
ID = 28767
ComplexID = 28766
TrainID = 1
ObjectID = 1

Missile 3 Type = TBM~Booster2Tank
ID = 28768
ComplexID = 28766
TrainID = 2
ObjectID = 1

Missile 4 Type = TBM~Decoy1 (first decoy)
ID = 28769
ComplexID = 28766
TrainID = 3
ObjectID = 1

Missile 5 Type = TBM~Decoy1 (second decoy)
ID = 28770
ComplexID = 28766
TrainID = 3
ObjectID = 2

Missile 6 Type = TBM-RV1 (Second RV)
ID = 28771
ComplexID = 28766
TrainID = 4
ObjectID = 1

Missile 7 Type = TBM~PBV1
ID = 28772
ComplexID = 28766
TrainID = 0
ObjectID = 0

After Decoy#3 Deployment:

The final decoy is deployed as a new missile object (Missile 8) in the same train as the second RV.

Missile 1 Type = TBM~RV1
ID = 28766
ComplexID = 28766

TrainID = 3
ObjectID = 3

Missile 2 Type = TBM~Booster1Tank
ID = 28767
ComplexID = 28766
TrainID = 1
ObjectID = 1

Missile 3 Type = TBM~Booster2Tank
ID = 28768
ComplexID = 28766
TrainID = 2
ObjectID = 1

Missile 4 Type = TBM~Decoy1 (first decoy)
ID = 28769
ComplexID = 28766
TrainID = 3
ObjectID = 1

Missile 5 Type = TBM~Decoy1 (second decoy)
ID = 28770
ComplexID = 28766
TrainID = 3
ObjectID = 2

Missile 6 Type = TBM-RV1 (Second RV)
ID = 28771
ComplexID = 28766
TrainID = 4
ObjectID = 1

Missile 7 Type = TBM~PBV1
ID = 28772
ComplexID = 28766
TrainID = 0
ObjectID = 0

Missile 8 Type = TBM~Decoy1 (Third Decoy)
ID = 28773
ComplexID = 28766
TrainID = 4
ObjectID = 2

Table 5.7-4 Example 4: Multiple Booster/Single RV with explicit spent tank, Critical Path goes to Tank, Create New Train selected on Booster1 and deselected on Booster2

Stage Number	Object Name	Stage Type	#Objects	Critical Path Flag	Deploy From
1	Booster1	Booster		YES	
2	Booster1Tank	Debris		NO	Booster1
2	Booster2	Booster		YES	Booster1
3	Booster2Tank	Debris		YES	Booster2
3	RV1	RV	1	NO	Booster2

Time of launch:

The full missile (Missile 1) is created, establishing the launch complex for all future object deployments.

Missile 1 Type = TBM~Booster1
ID = 28766
ComplexID = 28766
TrainID = 0
ObjectID = 0

After Booster1 Burnout:

The first stage spent booster tank is deployed as a new missile object (Missile 2) and is a member of the first train. The missile ID follows the critical path. Missile 1 maintains the ID 28766, but its object type is now the second stage booster, Booster2. Since the Create New Train option is deselected on Booster2, Booster2 is also a member of the first train.

Missile 1 Type = TBM~Booster2
ID = 28766
ComplexID = 28766
TrainID = 1
ObjectID = 2

Missile 2 Type = TBM~Booster1Tank
ID = 28767
ComplexID = 28766
TrainID = 1
ObjectID = 1

After Booster2 Burnout:

The original missile ID continues to follow the critical path. Missile 1 maintains the ID 28766, but its object type transitions to Booster2Tank. Since the Create New Train option is deselected on Booster2, Booster2Tank is deployed as a member of the first train. The RV is

deployed as a new missile object (Missile 3) and is also placed in the first train due to the Create New Train option being deselected. The first stage tank missile object continues to fly.

Missile 1 Type = TBM~ Booster2Tank
ID = 28766
ComplexID = 28766
TrainID = 1
ObjectID = 3

Missile 2 Type = TBM~Booster1Tank
ID = 28767
ComplexID = 28766
TrainID = 1
ObjectID = 1

Missile 3 Type = TBM~RV1
ID = 28768
ComplexID = 28766
TrainID = 1
ObjectID = 4

5.7.3.4.6.3Object Deployment Timing

Timing for Next Object deployments is dealt with in one of two ways, depending on whether the stage from which deployment occurs is a booster or PBV stage.

For a booster stage, the only timing specified is the time for the Next Object to occur. At this time, all the objects specified in the Next Object flight mode are deployed simultaneously from the booster. Neither the timing distribution field nor the Post Drop Settling Time field of the Next Object window apply when objects are deployed from boosters.

For a PBV stage, a timing distribution is specified for each object type in the Next Object flight mode. When the PBV executes the Next Object flight mode, a random draw is performed for each user-specified deployment. Several deployments may use the same timing; in this case several random draws are performed using the same distribution. Deployments are then performed in the proper chronological order.

A single post-object drop timing is also specified in the Next Object flight mode for PBVs. A random draw from this distribution is made to establish the post-object drop settling time. The time drawn is then added to the last chronological object deployment time in the mode. Post-drop settling time is used to account for any necessary settling prior to subsequent PBV maneuvers.

For old scenarios, pre-object drop time and post-object drop time will be used to populate a uniform time distribution value. The pre-object drop time distribution is treated as the timing distribution for the first object in the Next Object window. This will always be an RV and there will never be more than one object in the Next Object window. Pre-object drop time and post-object drop time are not used when deploying objects from a booster.

5.7.3.4.7 PBV Maneuver Guidance

PBVs are utilized to release multiple RVs on one or more targets from a single launch vehicle. PBV guidance maneuvers are based primarily on computations of the range sensitive axis (RSA) and the range insensitive axis (RIA). Thrusting along the RSA maximizes the change in velocity needed for each subsequent object deployment. Thrusting along the RIA provides a stable deployment axis upon which multiple objects may be targeted to the same impact point. PBV maneuver guidance options include; PBV Ignition, PBV Turn to RSA, PBV Turn to RIA, PBV Targeting, Next Object, PBV Disposal and the GoTo flight mode. The following is a typical guidance loop for multiple RV deployments from a PBV:

Mode #	Flight Mode
1	PBV Ignition
2	PBV Turn to RSA
3	PBV Targeting
4	PBV Turn to RIA
5	Next Object
6	GoTo (Mode #2)
7	PBV Disposal

During pitch/cut-off iterations for the boost phase, PBV Turn to RSA and PBV Turn to RIA are executed as full turns and PBV Targeting is executed for three seconds to facilitate an RV #1 target impact solution. For all subsequent RV deployments, these same maneuvers may execute for a variable amount of time to accommodate impact solutions associated with RV targets 2 thru N.

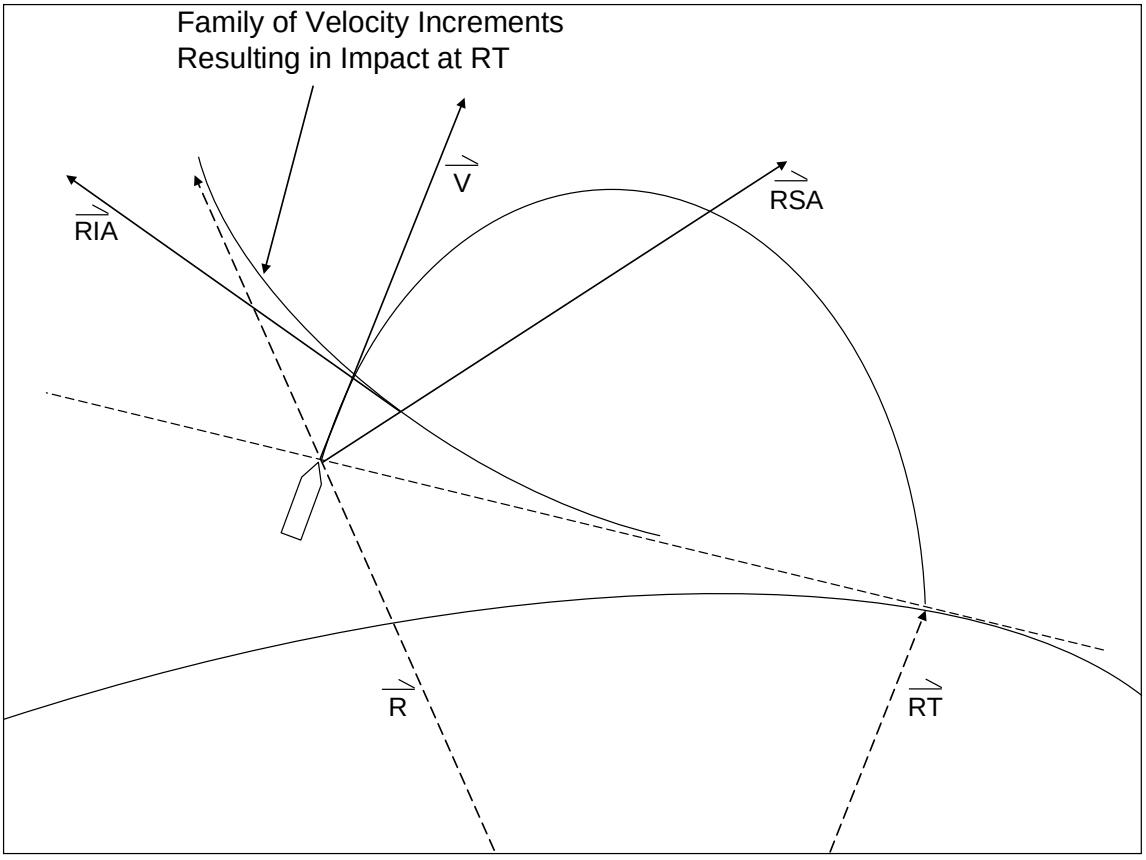

Figure 5.7-7 Range Sensitive/Insensitive Axes

Figure 5.7-7 illustrates the relationship between the RSA, RIA and missile trajectory. During PBV flight, the RSA and RIA are recomputed each integration time step. The RSA and RIA are axes in-plane with the target and are used to execute the user-specified PBV guidance loop. Turns to these axes are controlled by user-specified turn rates. In some cases, only partial turns may be necessary to accomplish a solution for the next target. The up or lofted RIA direction is defined by an angle measured counter clockwise from local horizontal. The lofted RIA angle is computed by applying a velocity perturbation to the PBV state vector and iterating on the altered direction of the applied perturbation until the impact point of the propagated state is within 1 meter of the unperturbed propagated state vector. The RSA angle for maximizing range extension is a negative 90 degrees from the lofted RIA angle. The RSA angle for reducing ground range during maneuvers is a positive 90 degrees from the lofted RIA angle.

Steering accelerations during PBV Turn to RSA, PBV Turn to RIA and PBV Targeting are based on angle of attack and are computed in the same manner as Pitch Profile Guidance.

5.7.3.4.7.1 PBV Ignition

The PBV Ignition guidance option simulates the delay before start of engine burn based on a user-specified ignition delay. During the PBV Ignition delay, missile flight is modeled simply as ballistic flight. The only forces acting on the missile are gravity and drag; no steering acceleration is applied.

5.7.3.4.7.2 PBV Turn to RSA

The PBV Turn to RSA guidance option simulates the turn to the next targeting direction, i.e., the next user-specified RV impact point. The PBV Turn to RSA maneuver is used to gain or reduce velocity, based on whether the next target is up range or down range. For a next target that is down range, PBV Turn to RSA would turn to the positive RSA direction so that thrust is applied with velocity. For a next target that is up range, PBV Turn to RSA would turn to the negative RSA direction so that thrust is applied against velocity.

The PBV must be in-plane with the target prior to a turn to the RSA. Azimuth control is conducted as described in Subsection 5.7.3.12 below. The turn to RSA initiates when the azimuth is within a 0.3 degree tolerance of being in plane

If the user-specified deployment axis is the RIA, then at each integration time step during the turn to RSA, a turn back to user-specified deployment axis is made. When the thrust vector is fully rotated to the deployment axis, the RV is flown ballistically to impact. Once the target ground range has been crossed, a delta time within the current integration time step is computed to accomplish target impact within a 100 meter tolerance.

Otherwise, if the user-specified deployment axis is the RSA, the RV is flown ballistically to impact during the turn at each integration time step. Once the target ground range has been crossed, a delta time within the current integration time step is computed to accomplish target impact within a 100 meter tolerance.

If a full turn is made to the RSA and additional velocity is needed to accomplish an impact solution, the PBV Targeting maneuver would then be executed if specified by the user's guidance loop.

5.7.3.4.7.3 PBV Turn to RIA

The PBV Turn to RIA guidance maneuver simulates rotation to the RIA. The RIA is also referred to as the Null Range Direction. The null range direction is a direction along which velocity perturbations do not cause a target miss. Consider a RV separated from a ballistic missile with a given velocity increment deltaV. Then there exists a family of velocity increments (see ***Figure 5.7-7***), which result in impact at the same point, only varying the loft of the trajectory (i.e., time of flight). Although RV deployments can occur from the RSA, normally RVs are deployed from the RIA.

When operating in the PBV Turn to RIA mode, azimuth control is conducted as described in section 5.7.3.12.

5.7.3.4.7.4PBV Targeting

The PBV Targeting guidance option simulates thrust along the RSA in order to gain or reduce velocity to accomplish a target impact solution.

If the user-specified deployment axis is the RIA, then at each integration time step during PBV Targeting, a turn back to user-specified deployment axis is made. When the thrust vector is fully rotated to the deployment axis, the RV is flown ballistically to impact. Once the target ground range has been crossed, a delta time within the current integration time step is computed to accomplish target impact within a 100 meter tolerance.

Otherwise, if the user-specified deployment axis is the RSA, the RV is flown ballistically to impact during the turn at each integration time step. Once the target ground range has been crossed, a delta time within the current integration time step is computed to accomplish target impact within a 100 meter tolerance.

When operating in the PBV Targeting mode, azimuth control is conducted as described in section 5.7.3.12.

5.7.3.4.7.5PBV Disposal

The PBV Disposal guidance option simulates the PBV maneuver following the last RV deployment. There are three options for the PBV disposal: (1) continue along the current thrust direction, (2) PBV goes inactive, and (3) fly the PBV the maximum distance possible out of plane with the last RV.

5.7.3.4.7.6Next Object

The Next Object guidance option simulates the deployment of an object from the PBV. The Next Object flight mode is detailed in Section 5.7.3.4.6,above.

5.7.3.4.7.7 GoTo FlightMode

The GoTo FlightMode guidance option is simply a loop construct. This option allows the user to specify guidance loops for deployment of multiple objects.

5.7.3.4.7.8Azimuth Guidance

The azimuth guidance mode attempts to zero the azimuth of the target relative to the velocity vector and bring the target into the trajectory plane. When operating in azimuth mode, azimuth control is conducted as described in section 5.7.3.12. Ballistic flight is used once the azimuth is within tolerance.

5.7.3.4.8 GEMS Guidance

GEMS guidance consists of a start time (s), maneuver direction, maneuver rate (degrees/second), maneuver maximum (degrees) and axial thrust value (N). The maneuver direction may be either up or down. The maneuver rate defines the rate at which the missile will pitch in the maneuver direction. The start time tells the model when to transition to the guidance phase. The axial thrust value defines the amount of thrust applied during GEMS guidance. Only one GEMS flight mode is allowed per missile. The GEMS flight mode must occur after the iterated pitch guidance phase.

The acceleration steering command, a, in the pitch channel is generated by a closed-loop controller, based on angle of attack, i.e., the difference between the commanded thrust vector pitch angle and the current flight path angle, as follows:

$$a = \left(T\sin\alpha + qSC_{1\alpha}\alpha\right)/\text{Mass}$$

where

T - axial thrust (N)

α - angle of attack (rad)

q - dynamic pressure (N/m^2)

S - aerodynamic reference area (m^2)

$C_{1\alpha}$ - linearized coefficient of lift (/rad).

At completion of a GEMS maneuver, the angle of attack is held constant until the end of the GEMS flight mode. While holding the angle of attack constant, the missile thrust vector pitch angle is computed as

$$Pitch = Gamma + \alpha$$

if the maneuver is up. Otherwise, the pitch is computed as

$$Pitch = Gamma - \alpha$$

where *Gamma* is the current missile flight path angle. The acceleration steering command is then computed as described above.

5.7.3.4.9 When operating in GEMS mode, azimuth control is also conducted. The azimuth control utilizes the same closed loop system but is attempting to zero the azimuth of the missile relative to the current plane from the missile to the target impact point. This azimuth control works to zero the azimuth of the missile flight relative to the target impact point.Disable Launch Iteration

Disable Launch Iteration is a missile guidance mode that disables all pitch and burntime iterations. This guidance mode flies the missile using the user-specified guidance options without making any adjustments. Burntime cut-off is controlled completely by user-specified stage/section information, including fuel mass, specific impulse and cut-off time. If the missile definition contains GEMS flight modes, no GEMS maneuver will occur during the GEMS flight mode. The angle of attack at start of GEMS will be held constant until the end of the GEMS flight mode.

5.7.3.4.10 Ballistic Missile Seekers

The ballistic missile supports implicit and explicit seeker modeling. Both options allow the target location to be updated during fly out and therefore allow a dynamic terminal maneuver to be performed. A seeker can be added for each missile object. This allows different seekers to be used during the different stages of missile flight. For example, separate sensors could be associated with a PBV and RV on a missile to model a threat with mid-course and terminal seekers.

If a ballistic aim point is selected but there are no sensors associated with the missile then the sensor is modeled implicitly. If the guide to truth option is selected then the terminal maneuver guides to the truth target location. This models an implicit seeker that always detects the target. Otherwise the terminal maneuver guides to the perceived target location at the time of launch.

Seekers can be explicitly modeled on ballistic missiles by associating sensor elements with the various objects. An AGAttacker ruleset must be associated with each missile object that has sensors to provide the mechanisms for sensor data management and targeting decisions. MM Section 4.7.27 describes the AGAttacker ruleset. The track state information is handed over when a parent object with a seeker deploys an object that has a sensor or an object that will eventually deploy an object that has a sensor. The sensor data is used to refine and update the target location during the terminal phase of flight. If the target is not detected by the sensor, the maneuver guides to the perceived target location at the time of launch.

Sensor activation can be controlled using the flight mode seeker active option or through the user rules. The flight mode seeker active option allows all of an object's on-board sensors to be set to active or inactive during a user specified flight mode. MM section 4.12 describes the user rules operations.

5.7.3.4.11 Flight on Rail

Flight on Rail is not a user interface option for ballistic missiles. However, a rail is modeled internally for ballistic missile to fly the missile along a straight line for a prescribed distance just after launch. This capability is useful to keep the missile from pitching over to the ground after nonvertical launch or to simulate launch from a rail or tube. Most missiles do not have enough velocity just after launch to keep them flying on a straight path. The rail supports the missile until it has sufficient time to accelerate to sufficient flight-sustaining velocity. This condition is modeled by setting the velocity vector turning rates to zero:

$$w_y = 0$$

$$w_z = 0 \quad .$$

The rail length used internally for ballistic missiles is 5 meters.

5.7.3.4.12 Impact Overshoot Correction Backup

If the missile should overshoot the target altitude during integration, it is necessary to back up the missile to the target altitude or impact with the ground. When this occurs, the model uses a linear interpolation to create a new Δt to reach the desired target altitude.

Using this new integration step, the model recalculates the missile position and velocity vectors and decrements the total flyout time by the value of Δt. This process is repeated until the missile is within 1 m of the target altitude.

5.7.3.4.13 Missile Orientation

By default, the body frame of a TBM is oriented along its velocity. In reality, a missile that leaves the atmosphere, however, is not forced by drag nor gravitational forces to conform its attitude to its trajectory. Options are available that allow the user to define how the characteristics of missile body axis orientation will change throughout the exo-atmospheric portion of the missile flight. The attitude options will be employed upon missile burnout or when the missile exits the atmosphere.

A stabilization phase has also been incorporated that is initiated upon missile reentry into the atmosphere. This phase models a damping effect and subsequent missile orientation alignment along the velocity vector due to atmospheric drag and gravitational forces. The stabilization computations are performed in a body frame coordinate system, in which the body-axis is rotated from ECEF into a coordinate system aligned with the body of the missile. The y-axis lies along the orientation vector, the x-axis is horizontal to the right, and the z-axis is upwards so as to form a right-handed system.

If any of the options are selected, and the missile being modeled never exits the atmosphere, from launch to burnout, missile orientation is set to the thrust vector by the guidance pitch parameters. After burnout, the body frame is oriented along the velocity vector.

The missile attitude enhancements do not affect missile dynamics or propagation. The changes in ballistic missile orientation only affect aspect dependent detection and pk calculations.

5.7.3.4.13.1 Velocity Vector Aligned

This option allows the missile orientation to align with the velocity vector once thrust terminates. The orientation aligns with the thrust vector from launch to burnout. This is the default option.

5.7.3.4.13.2 Maintain Attitude

This option allows a post-burnout missile to maintain a constant attitude when above the Earth's atmosphere. The cases described below demonstrate how EADSIM sets missile body frame orientation when the attitude maintenance feature is selected.

The missile will begin using attitude maintenance at burnout. While post-burnout, the body frame is oriented along the last orientation vector attained by the missile before burnout. This static attitude is maintained until the missile reenters the atmosphere. After the missile reentry, the missile orientation linearly converges from its orientation at the reentry altitude to its velocity vector at the stable attitude altitude. When the stable attitude altitude is set equal to the reentry altitude, the model immediately forces missile orientation to adopt the velocity vector. The stable attitude altitude is always less than or equal to the reentry altitude.

The stabilization computations are performed from information gained by pre-flying the missile to the reentry altitude and then to the stable attitude altitude:

$$\dot{\theta} = \frac{\cos^{-1}(\vec{O}_R \bullet \vec{V}_S)}{t_s - t_r}$$

$$\theta = -\dot{\theta} \cdot I$$

$$O_x = 0.0$$

$$O_y = \cos(\theta)$$

$$O_z = \sin(\theta)$$

where

θ = Alignment rate, the total change in orientation angle over the time interval between reentry and stabilization (rad/sec)

$\vec{O}_R$ = Missile orientation unit vector at the reentry altitude (body frame coordinates)

$\vec{V}_S$ = Missile velocity unit vector at the stable attitude altitude (body frame)

t_r = Time when the missile reenters the atmosphere (sec)

t_s = Time when the missile orientation stabilizes (sec)

θ = Angle between previous orientation and new orientation (rad)

I = Scenario interval (sec)

$\vec{O}_{x,y,z}$ = New orientation vector (body frame).

The new orientation is then rotated from local body frame to the earth-centered inertial (ECEF) coordinate system.

5.7.3.4.13.3 Missile Precession / Coning

This option allows a post-burnout exo-atmospheric missile to precess about its body axis. The cases described below demonstrate how EADSIM sets missile body frame orientation when the precession feature is selected.

The missile will begin using the precession feature at burnout. When post-burnout, the missile precession is established. A random draw is performed to determine the cone angle. The cone angle is the angular displacement between the attitude vector and the missile body axis. A second random draw is made to determine the rate at which the missile body precesses about the body axis. The precession angle is calculated by multiplying the randomized precession rate by the user-specified scenario interval. The new orientation vector is then computed as a function of the precession angle and the coning angle. This process is repeated every scenario interval until the missile reaches a user-defined reentry altitude. If randomness is eliminated, the values entered for mean are used as the cone angle and precession rate. The calculations are as follows:

$$O_x = \sin(\theta) \cdot \cos(\phi)$$
$$O_y = \cos(\theta)$$
$$O_z = \sin(\theta) \cdot \sin(\phi)$$

where

$\bar{O}$ = Precessed orientation vector (body frame)

θ = Cone angle off body axis (rad)

ϕ = Precession angle, initially generated by a random draw (rad).

The new orientation is then rotated from local body frame to the ECEF coordinate system.

From the time the missile reenters the atmosphere until it descends to the stable attitude altitude, a stabilization occurs. The cone angle between the missile body and the body axis linearly decreases to zero as the precession rate linearly increases to the rate defined by the sum of the precession rate and precession rate increase. Precession rate increase is intended to model the increase in precession rate observed by cone angle decrease and spin rate increase due to the missile atmospheric reentry. When the stable attitude altitude is set equal to the reentry altitude, the model immediately forces missile orientation to adopt the velocity vector, since the cone angle instantaneously decreases to zero. The stable attitude altitude is always less than or equal to the reentry altitude.

The stabilization computations are performed from information gained by pre-flying the missile to the reentry altitude and then to the stable attitude altitude:

$$R = \frac{(t - t_r)}{(t_s - t_r)}$$

$$\theta = \theta \cdot (1.0 - R)$$

$$\dot{\phi} = \dot{\phi} + \dot{\phi}_{inc} \cdot R$$

where

R = Linear ratio of time between reentry and stabilization

t = Current time (sec)

t_r = Time at which the missile reenters the atmosphere (sec)

t_s = Time at which the missile attitude stabilizes (sec)

θ = Current cone angle (rad)

$\dot{\phi}$ = Current precession rate (rad/sec)

$\dot{\phi}_{inc}$ = Amount of precession rate increase achieved by stabilization time (rad/sec).

5.7.3.4.13.4 Attitude Maintenance with Precession/Coning

The previous section discusses missile precession and coning under the assumption that the precession axis and the missile velocity vector are co-linear. An additional option is available for a post-burnout exo-atmospheric missile that allows the user to specify that the precession axis (i.e. the missile body attitude without precession) is fixed upon leaving the atmosphere. Under this option, the missile behaves in the same manner as with precession alone, except that the cone angle is viewed in reference to the constant body axis rather than the time-dependent velocity vector. When the missile reaches the stabilization stage defined between the reentry and stable attitude altitudes, three stabilization changes occur as a synthesis of the maintain attitude and precession stabilization processes. The body-axis linearly aligns itself from its static orientation to the velocity vector; the cone angle linearly decreases to zero; and the precession rate linearly increases to the rate defined by the sum of the precession rate and precession rate increase.

5.7.3.4.13.5 Missile In-Plane Tumble

This option allows a post-burnout exo-atmospheric missile to tumble in the trajectory plane of the missile. The case described below demonstrates how EADSIM sets missile body frame orientation when the in-plane tumble feature is selected.

The missile will begin using the in-plane tumbling feature at burnout. When post-burnout, the missile tumble is established. A random draw is performed using the defined in-plane angular tumble rate. The change in in-plane orientation angle is calculated by multiplying the randomized tumble rate by the length of the user-specified scenario interval. Using the previous attitude as the point of reference,

the new orientation vector is then computed as the previous attitude, offset by the change in in-plane missile orientation angle. This process is repeated until the missile reaches a user- defined reentry altitude. If randomness is eliminated, the value entered for mean is used as the tumble rate. The computations are as follows:

$$\Delta\theta = \dot{\theta}_T \cdot I$$

$$\theta = \tan^{-1}\left(\frac{O_z}{O_y}\right)$$

$$r = \sqrt{(O_y)^2 + (O_z)^2}$$

$$O_y = r \cdot \cos(\theta - \Delta\theta)$$

$$O_z = r \cdot \sin(\theta - \Delta\theta)$$

where

$\Delta\theta$ = Orientation angle displacement off the current attitude (rad)

$\dot{\theta}_T$ = Tumble rate generated by a uniform random draw (rad/sec)

I = Scenario interval (sec)

θ = Angle between the y and z components of the current attitude (rad)

$\vec{O}_{x,y,z}$ = Orientation vector (body frame)

r = Resultant of the vector between the y and z components of the current attitude.

The new orientation is then rotated from local body frame to the ECEF coordinate system.

The tumbling process continues until the missile reaches the reentry altitude. The missile will undergo the stabilization process between the specified reentry altitude and the specified stable attitude altitude. The stabilization computations are performed from information gained by pre-flying the missile to the reentry altitude and then to the stable attitude altitude. Pre-flying the missile gives the reentry time, the stable attitude altitude time, and the flight path angle at the stable attitude altitude. See Subsection 5.7.3.5.4 for the derivation of the flight path angle. The missile is then actually flown in the model. Stabilization will begin once the missile reaches the reentry altitude. In order to derive the orientation angle at reentry, it is necessary to rotate the orientation into a satellite body coordinate system. See Subsection 6.4.2.3.2 for further information on the satellite body coordinate system. The computations of the stabilization process are as follows:

$$\theta = \tan^{-1}\left(\frac{O_z}{O_y}\right)$$

$$\alpha = \cos^{-1}\left(\vec{O} \bullet \vec{V}\right)$$

where

θ = Angle between the y and z components of the current attitude (rad)

$\vec{O}_{x,y,z}$ = Orientation unit vector (body frame)

α = Angle between the current orientation and velocity unit vectors (rad)

$\vec{V}$ = Velocity unit vector (body frame).

The new orientation is then rotated from local body frame to the ECEF coordinate system. The stabilization process continues until the missile is at the stable attitude altitude. At this point the missile orientation is aligned with the velocity vector.

5.7.3.4.13.6 Missile Random Tumble

This option allows a post-burnout exo-atmospheric missile to randomly tumble in its flight. The cases described below demonstrate how EADSIM sets missile body frame orientation when the random tumble feature is selected.

For missiles that burn out before leaving the atmosphere, orientation is set to the missile thrust vector from launch to burnout. From the final burnout time until the missile leaves the atmosphere, the missile body frame is oriented along the velocity vector. When exo-atmospheric, the missile tumble is established. A random draw is performed in three dimensions, where the maximum angle to which the missile may rotate in any plane from its current position is specified by the user-defined maximum rotation angle. Using the previous attitude as the point of reference, the new orientation vector is then computed as the previous attitude, offset by the computed change in missile orientation angle. This process is repeated every scenario interval until the missile impacts. If randomness is eliminated, the value entered for mean is used to orient the missile. The calculations are as follows:

$$\Delta\theta = \theta_{RT} \cdot R$$

$$\theta = \tan^{-1}\left(\frac{O_p}{O_q}\right)$$

$$r = \sqrt{(O_p)^2 + (O_q)^2}$$

$$O_p = r \cdot \cos(\theta - \Delta\theta)$$

$$O_q = r \cdot \sin(\theta - \Delta\theta)$$

where

$\Delta\theta$ = Orientation angle displacement to be subtracted from the current attitude (rad)

θ_{RT} = Max rotation angle in three dimensions (rad)

R = Uniform random number in the interval [0,1)

θ = Angle between the p and q components of the current attitude (rad)

$\bar{O}_{x,y,z}$ = Orientation vector (body frame)

(p,q) = (y,z) in the first rotation, (x,y) in the second rotation

r = Resultant of the vector between the p and q components of the current attitude.

The new orientation is then rotated from local body frame to the ECEF coordinate system.

For missiles that burn out after leaving the atmosphere, the missile body axis is oriented to the thrust vector from launch to burnout. From the final burnout time until impact, the missile attitude changes as described by the random tumble process.

5.7.3.5 Interceptor Guidance Options

The range of guidance options available provides the flexibility to reproduce many different flight trajectories. During flyout, ballistic, ballistic PPN, flight path angle curve, pitch profile, and proportional navigation to a space point can be used in any sequence. A special rail option is provided for just after launch at which point the missile can be commanded to follow a straight line for a prescribed distance from the launch point. Proportional navigation or predictive proportional navigation may be used in the homing section to steer toward a predicted intercept point. The guidance options are specified uniquely within the flight sections defining an interceptor. These guidance options ultimately feed the process of computing the steering accelerations for the interceptor.

5.7.3.5.1 Interceptor Missile Stages and Flight Sections

The interceptor missile flyout is described by the use of multiple flight sections combined into stages. Each flight section consists of a separate specification of aerodynamic, propulsion, and mass properties, as well as guidance

commands. A stage can be a single flight section or multiple flight sections. The mass value specified for the first flight section of a stage is the initial mass of the missile at the start of the stage. The mass from subsequent stages is summed with the current stage mass to determine the missile's total mass. This includes both dry/inert mass and onboard propellant. Any mass value specified for a flight section other than the first flight section of a stage is the amount of inert mass to drop at the beginning of the flight section. If a single stage has multiple flight sections, the transition from one flight section to another within the stage utilizes the remaining fuel mass from the previous flight section minus the dropped inert mass, whereas, a transition between stages resets the total remaining mass (i.e., both dry/inert and fuel mass) for the missile.

The interceptor thrust profile specification contains an option to enter thrust as a function of either Time in Section or Time in Stage. If Time in Section is selected, thrust is linearly interpolated from the profile table during the current flight section. If Time in Stage is selected, thrust is linearly interpolated from the profile table during the current flight stage. A minimum and maximum dynamic pressure threshold is specified for each flight section of the interceptor. If the dynamic pressure falls below or exceeds the specified thresholds, the interceptor will transition to the next flight section within a stage. For the last flight section of a stage, if the thrust value goes to zero before the end time of the last flight section of the stage and the dynamic pressure falls below the minimum threshold, the interceptor will transition to the next stage. For the last flight section of a stage, flight section transitions due to dynamic pressure exceeding the specified maximum dynamic pressure threshold will not occur. Stage transition will also occur when the trigger of the first flight section of the next stage is met.

Transitions between flight sections can be triggered using time since launch,, altitude relative to target while ascending, altitude relative to target while descending, altitude MSL while ascending, altitude MSL while descending, altitude rate, or time since start of flight section. The transition triggers are start triggers, and therefore cause a transition from the preceding flight section in the list to the next flight section.

When reading in airframe element versions prior to EADSIM Version 14.00, the minimum and maximum dynamic pressure thresholds will be set to defaulted values. For the last flight section within a stage, the minimum dynamic pressure threshold will be defaulted to the dynamic pressure threshold value specified in the older version, while the maximum dynamic pressure threshold will be defaulted to 99999.9. For all other flight sections, the maximum dynamic pressure threshold will be defaulted to the dynamic pressure threshold value specified in the older version, while the minimum dynamic pressure will be defaulted to zero.

The mathematical formulation for each section is identical with an additional capability for the final section, designated as the homing section. This section has

the special characteristic that it provides a user selectable option to specify the use of proportional navigation or predictive proportional navigation to home on the target when the decision is made in the C3I model, although it operates identically to the other sections prior to the initiation of endgame homing. An additional user selectable option allows a stage transition to occur upon initiation of homing. This transition models mass change prior to homing and only occurs during the next to last flight section of the missile.

Each flight section consists of specifications for:

- Transition Trigger
- Transition Value
- Minimum dynamic pressure threshold
- Maximum dynamic pressure threshold
- Vacuum thrust profile
- Specific impulse
- Nozzle exit area
- Aerodynamic reference area
- Aerodynamic tables for lift and drag
- Maximum Angle of Attack
- Numerical integration step size
- Guidance instructions
- Mass
- Stage Number.

Within a flight section, guidance instructions consist of a table specifying periods of operation, by means of transition triggers and transition values, for combinations of the available guidance modes:

- Ballistic
- Ballistic PPN
- Flight Path Angle Curve
- Pitch Profile
- Fly to a space point
- Pull Up
- Pull Down

For 3 DOF complex weapons utilizing the AGAttacker ruleset, the transition on detect and seeker active options are implemented for each flight mode entry. The flight mode will transition from the previous flight mode in the list to the associated flight mode if the target is detected or if the transitions trigger condition is met. The seeker will become active once a flight mode with the seeker active option selected is reached. The seeker activation time specified on the target select phase window will not be used.

Describing the missile as a series of flight stages and sections provides the capability to model many types of missiles. For many cases, the flight stage will correspond to a single flight section, but this does not necessarily have to be the case. Within a single stage, multiple flight sections could represent aerodynamic, thrust profile, or mass changes during a single stage of missile flight.

Following is a description of how the acceleration commands (a_y and a_z in the equations for w_z and w_y, respectively) are formulated for the different flight guidance options.

5.7.3.5.2 Ballistic Flight

The ballistic guidance mode may optionally be used to model a failed interceptor's flight once the failure has occurred. If the Continue Flight Upon Failure option is selected, any failure or intentional abort of the interceptor, once the explicit interceptor is in the air, will result in the interceptor operating with ballistic guidance until ground impact. The interceptor will continue to transition flight sections from the point of failure through its final flight section, but the ballistic guidance mode will be used throughout.

5.7.3.5.3 Ballistic PPN Guidance

Ballistic PPN guidance differs from the pure ballistic flight in that a predicted intercept is periodically computed based on the current state of the interceptor continuing with pure ballistic flight. The prediction is performed at each integration interval, until the interceptor is predicted to be within the tolerance. Once within tolerance, the interceptor will fly according to ballistic guidance. After each inflight target update (IFTU) controlled by the IFTU/Guidance Update rates specified on the weapon element definition, the PPN is applied once again. Typically, against a ballistic missile target, none or only minor adjustments will be required once within the tolerance for the first time. The initial entrance into ballistic PPN will require less time to guide to the solution if the planned intercept solution from the flyout table is accurate for the interceptor definition. See section 5.7.3.2.4 for details on the acceleration calculations for the Ballistic PPN guidance mode.

When reading in airframe element versions prior to EADSIM Version 14.00, the PN gain specified at the flight section level in older versions will be moved to PN gain tables for those flight modes that use PN gain. Each Ballistic PPN and Fly to a Space Point flight mode entry will have a PN gain table associated with it. The table will have a single entry. For the single entry, the time-to-go will be zero and the gain value will be the PN gain value from the corresponding flight section.

Predictive proportional navigation, as described above, may optionally be selected for use during endgame homing. Accelerations for PPN during homing are computed as described above once per integration interval. However, the guidance

gain defined for endgame homing is used rather than the flight section proportional navigation gain. An additional characteristic of PPN during homing is that its application is not based on a miss distance tolerance. Once PPN during homing is initiated, it is continuously applied until the interceptor passes the target, i.e. achieves the point of closest approach.

The PN gain values for homing are input in a table as a function of slant range. Slant range is measured from the launcher location to the predicted intercept point. PN gain values are not interpolated from the PN table. Instead, the PN gain value of the nearest slant range less than the calculated slant range value will be used. If the slant range value is less than the first value on the table, the first gain input is used as the PN gain. If the slant range value is greater than the last value on the table, the last input is used as the PN gain. When reading in previous airframe element versions, the homing guidance gain table will have a single entry. For the single entry, the slant range value will be zero and the guidance gain value will be the homing guidance gain value specified in the older version airframe element.

5.7.3.5.4 Flight Path Angle Guidance

During the commanded flight path angle curve guidance phase, a table of flight path angle commands versus a combination of time in section, elevation angle and ground range defines the desired flight path angle for the missile. The elevation angle is the angle between the local horizontal plane at the launcher and the predicted intercept point measured positive up. Ground range is the distance along the Earth's surface from the point of launch to the predicted intercept position. A decision to home while operating in a phase with flight path angle guidance is valid. Upon the decision to home, guidance will transition to a user specified PN or PPN guidance with a potentially altered integration time step to steer to the target. See section 5.7.3.2.2 for details on the acceleration calculations for the Flight Path Angle mode.

The commanded flight path angle may be altered from that specified in the table if the iteration on flight path angle for launch solution option is selected. In this case, the flight path angle that will be used from the beginning iteration flight section and flight mode until the ending iteration flight section and mode will be the iterated flight path angle. This iterated flight path angle value may be the value for the iteration during computing the launch solution or it may be the final value computed from the iteration when actually flying the interceptor.

When operating in flight path angle mode, azimuth control is also conducted once the interceptor has flown for the input begin azimuth control time. The azimuth control utilizes the same closed loop system but is attempting to 0 the azimuth of the interceptor relative to the current plane from the interceptor to the intercept point. In this case, the gain must be applied relative to the horizontal speed rather than the total velocity. This azimuth control works to keep the

azimuth component zero when the interceptor is aligned initially with the intercept point or to zero the azimuth when the interceptor is fixed along the PTL.

5.7.3.5.5 Pitch Profile

During the pitch profile flight mode, pitch rate commands control the direction of the missile's thrust vector. Pitch rate is the rate at which the missile body pitches until the end of the current guidance phase. Pitch rate is defined as a function of either slant range or altitude. Slant range is calculated from the point of launch to the predicted intercept point. Altitude is measured from the earth's surface to the predicted intercept point. At the beginning of the flight mode, if pitch profile was not the previous flight mode, the missile's pitch angle will be set to the flight path angle plus the angle of attack. The pitch value will then be updated every integration time step based on the pitch rate. If slant range is selected as the look up value, the slant range is evaluated upon entering a pitch profile flight mode. If the slant range value is below the first slant range entry, the interceptor will skip this flight mode and immediately transition to the next flight mode. If the first slant range value is non-zero, a flight mode other than Pitch Rate must follow the Pitch Profile flight mode. When operating in pitch profile mode, azimuth control is also conducted once the interceptor has flown for the input Begin Azimuth Control time. Pitch rate is measured clockwise down from the local vertical. See section 5.7.3.2.1 for details on the acceleration calculations for the Pitch Profile flight mode.

5.7.3.5.6 Pull Up/Pull Down

The pull up and pull down flight modes allow in-plane accelerations to be applied in either the up or down direction. Either Angle of Attack or Commanded Lateral Acceleration is used to specify the command for the Pull Up/ Pull Down Guidance. See section 5.7.3.3.1 for more information about the Pull Up and Pull Down Guidance modes.

5.7.3.5.7 Proportional Navigation

Proportional navigation may be optionally specified for steering to a space point and/or for steering to a target during endgame homing. During flyout, the missile can optionally steer to a space point supplied by C3I in the scenario update message. This space point is currently always the computed intercept point. During homing, the missile steers to a target whose coordinates are also supplied by C3I. The guidance algorithm is identical in both cases.

A simple proportional navigation steering law is used to generate the acceleration commands based upon the line-of-sight rate of the target (or space point):

$$a_y = \Lambda V_r \dot{\theta}_{az}$$

$$a_z = -\Lambda V_r \dot{\theta}_{el}$$

where

a_y	-	commanded acceleration, yaw (m/sec^2)
a_z	-	commanded acceleration, pitch (m/sec^2)
V_r	-	relative velocity between missile and target (m/sec)
$\dot{\theta}_{yaw}$	-	line-of-sight rate, yaw (rad/sec)
$\dot{\theta}_{pitch}$	-	line-of-sight rate, pitch (rad/sec)
Λ	-	proportional navigation gain
	=	1 (pure pursuit)
	=	3-4 (normal against moving target).

A proportional navigation gain of one always steers the missile toward the current target position (a "tail chase"). Proportional navigation gains greater than one establish a lead angle which aims the missile in front of the target's current flight direction.

The line-of-sight rates are computed based upon the range and range rate. The range and range rates are first computed in the ECEF coordinate frame:

$$
\begin{aligned}
r_x &= x_{target} - x_{missile} \\
r_y &= y_{target} - y_{missile} \\
r_z &= z_{target} - z_{missile} \\
\dot{r}_x &= \dot{x}_{target} - \dot{x}_{missile} \\
\dot{r}_y &= \dot{y}_{target} - \dot{y}_{missile} \\
\dot{r}_z &= \dot{z}_{target} - \dot{z}_{missile}
\end{aligned}
$$

where

r	-	missile-to-target range (m)
x,y,z	-	ECEF position coordinates of missile/target (m)
$\dot{r}$	-	missile to target range rate (m/sec)
$\dot{x},\dot{y},\dot{z}$	-	ECEF velocity coordinates of missile/target (m/sec).

The ECEF range and range rate vectors are then transformed to the ECEF2LaunchAz coordinate frame.

The ECEF2LaunchAz range and range rate vectors are then transformed to the Missile Velocity Axis frame with the Euler rotations, Ψ, Θ, and Φ.:

$$
\begin{bmatrix} x_{MVA} \\ y_{MVA} \\ z_{MVA} \end{bmatrix} = \begin{bmatrix} \cos\Theta\cos\Psi & \cos\Theta\sin\Psi & -\sin\Theta \\ \sin\Phi\sin\Theta\cos\Psi - \cos\Phi\sin\Psi & \sin\Phi\sin\Theta\sin\Psi + \cos\Phi\cos\Psi & \cos\Theta\sin\Phi \\ \cos\Phi\sin\Theta\cos\Psi + \sin\Phi\sin\Psi & \cos\Phi\sin\Theta\sin\Psi - \sin\Phi\cos\Psi & \cos\Theta\cos\Phi \end{bmatrix} \begin{bmatrix} x_{ECI} \\ y_{ECI} \\ z_{ECI} \end{bmatrix}
$$

Finally, the line-of-sight rates are computed:

$$\dot{\theta}_{yaw} = \frac{r_x \dot{r}_y - r_y \dot{r}_x}{\sqrt{r_x^2 + r_y^2} + r_z^2}$$

$$\dot{\theta}_{pitch} = \frac{r_z \dot{r}_x - r_x \dot{r}_z}{\sqrt{r_x^2 + r_y^2} + r_z^2}$$

where the range and range rates, r and $\dot{r}$, are in the Missile Velocity Axis frame.

Miss distance and time-to-go until closest approach are computed as quantities to monitor the effectiveness of the proportional navigation steering. Time-to-go, t_{go}, is computed based upon the range, r, and range rate, $\dot{r}$:

$$t_{go} = \frac{\dot{r}_x r_x + \dot{r}_y r_y + \dot{r}_z r_z}{\dot{r}_x^2 + \dot{r}_y^2 + \dot{r}_z^2} \quad .$$

Time-to-go is negative up until the time of closest approach, at which point its sign turns positive. Thus, a positive time-to-go is used as a signaling condition that the target has been passed.

Miss distance at any time is defined as the distance between the missile and target at time of closest approach, if the missile's and target's positions were propagated forward in time at constant velocity. Miss distance is computed based upon the time-to-go. First, the miss distance along each axis (in the Missile Velocity Axis frame) is computed:

$$miss_x = r_x - t_{go} \dot{r}_x$$

$$miss_y = r_y - t_{go} \dot{r}_y$$

$$miss_z = r_2 - t_{go} \dot{r}_z \quad .$$

The total miss distance is the root-sum-square of the three components of miss:

$$miss = \sqrt{miss_x^2 + miss_y^2 + miss_z^2} \quad .$$

5.7.3.5.8 Flight on Rail

This option is provided to fly the missile along a straight line for a prescribed distance just after launch. This capability is useful to keep the missile from pitching over to the ground after nonvertical launch or to simulate launch from a rail or tube. Most missiles do not have enough velocity just after launch to keep them flying on a straight path. In cases where target engagement planning necessitates a nonvertical launch, a rail or launch tube points and supports the missile until it has sufficient time to accelerate to sufficient flight-sustaining velocity. This condition is modeled by setting the velocity vector turning rates to zero:

$$w_y = 0$$

$$w_z = 0 \quad .$$

The specified rail length does not have to correspond physically to an actual rail. This option is also useful for keeping the missile's flight path angle constant after launch.

5.7.3.5.9 Internal Target Model

During Homing and Ballistic PPN, an internal target model is used to propagate the target position state between scenario updates. The target's position must be propagated internally since several hundred integration cycles normally occur within the missile model between scenario updates. The target state is initialized based on the message from C3I and is propagated forward in time at constant velocity:

$$x_{target} = \int \dot{x}_{target} dt$$

$$y_{target} = \int \dot{y}_{target} dt$$

$$z_{target} = \int \dot{z}_{target} dt$$

where

$x_{target}, y_{target}, z_{target}$ - ECEF target position (m)

$\dot{x}_{target}, \dot{y}_{target}, \dot{z}_{target}$ - ECEF target velocity (m).

The target velocity state and initial position are in the scenario update message from C3I.

If the target is a ballistic missile, the target state will be propagated using the ballistic flight model accounting for gravity, the atmosphere, and other propagation effects.

This target state propagation is only an approximation of the true target position maintained outside of the missile model, since the true target's velocity may change due to accelerations (drag, maneuvers, or gravity). The target propagation model maintained in the missile model is reinitialized to the true position with each scenario update message.

5.7.3.6 Interceptor Missile Launch Schemes

At launch, the direction of the missile's velocity must first be established to initialize the equations of motion. There are three options for the setting of the elevation angle if the interceptor is launched from a Flexible SAM or SAM Launcher. The initial elevation angle can be either degrees above the local horizontal, degrees above the line of sight to the intercept point, or set equal to the iterated elevation angle. The degrees above horizontal models a launcher that launches from a fixed elevation angle. The degrees above line of sight models a rough angle method to get the interceptor above the line of sight to the threat. The

iterated elevation angle can only be used when iterating on flight path angle to determine a launch solution and also if the iteration begins at the first flight section and flight mode. This case models a launcher that is oriented and adjusted in elevation as it launches the interceptor. For interceptors launched from a Fighter or AGAttacker, the elevation angle is set to the pitch of the launching platform. The pitch is calculated by using function ComputeHeadingPitch as described in Appendix B10.2.

For flights containing terminal maneuvers against ground targets, the launch solution will be computed using ballistic flight for the final stage if the ballistic aim point option is selected. Otherwise, the launch iteration scheme flies any terminal maneuvers to the perceived target location. See section 5.7.3.10 for more information about the ballistic aim point.

For interceptors launched from a Flexible SAM or SAM Launcher, the azimuth angle will either be the azimuth to the intercept point from the launch location or it will be the azimuth defined for the PTL. For interceptors launched from a Fighter or AGAttacker, the azimuth angle is set to the direction of travel of the launching platform relative to North in degrees based on the velocity vector. The heading is calculated by using the function ComputeHeadingPitch as described in Appendix B10.2.

Using the initial elevation and initial latitude and longitude, the next step is to create the ECEF2LaunchAz rotation matrix by calling GenXYZRotMatrix as described in Appendix B10.3. With the rotation matrix, the initial Euler angles for the interceptor are calculated using the launch elevation and an azimuth of 90 degrees. The initial elevation and azimuth of the missile's velocity vector are calculated in the NED frame, so the ECEF space point, i.e. the target intercept point, is converted to the NED frame.

First, the launch location is determined in ECEF based on the launch latitude, longitude, and elevation. The launch location is determined by calling LLAToDblECEF, described in Appendix B10.

$$x_{translated} = x_{ECEF} - L_X$$
$$y_{translated} = y_{ECEF} - L_Y$$
$$z_{translated} = z_{ECEF} - L_Z$$

where

x_{ECEF}, y_{ECEF}, z_{ECEF} - space point ECEF coordinates
L – Position vector of the launch location in ECEF coordinates

The translated vector is then rotated through the angles' longitude and latitude:

$$\begin{bmatrix} x_{NED} \\ y_{NED} \\ z_{NED} \end{bmatrix} = \begin{bmatrix} \cos\left(-\theta_{lat}-\frac{\pi}{2}\right)\cos\theta_{lon} & \cos\left(-\theta_{lat}-\frac{\pi}{2}\right)\sin\theta_{lon} & -\sin\left(-\theta_{lat}-\frac{\pi}{2}\right) \\ -\sin_{lon} & \cos_{ion} & 0 \\ \sin\left(-\theta_{lat}-\frac{\pi}{2}\right)\cos\theta_{lon} & \sin\left(-\theta_{lat}-\frac{\pi}{2}\right)\sin\theta_{lon} & \cos\left(-\theta_{lat}-\frac{\pi}{2}\right) \end{bmatrix} \begin{bmatrix} x_{translated} \\ y_{translated} \\ z_{translated} \end{bmatrix} .$$

If the interceptor is to be aligned to the intercept point in azimuth, the azimuth is computed in the NED coordinate frame.

$\theta_{azimuth}$ - $\tan^{-1}(y_{NED}, x_{NED})$
$\theta_{FlyToPoint}$ - $\tan^{-1}(y_{NED}, x_{NED})$

If the interceptor is launched from a Fighter or AGAttacker, the azimuth is initialized to the heading of the launching platform. Otherwise, if the interceptor is to be aligned to the PTL, then azimuth is simply set to the azimuth specified for the PTL.

If the interceptor is launched from a Fighter or AGAttacker, the elevation angle is initialized to the pitch of the launching platform. Otherwise, if the elevation angle is degrees above local horizontal, then the elevation angle is set to the input value. If it is to be the iterated flight path angle value, it will be set according to the launch iteration scheme described below. If it is to be set to some angle above the line of sight to the intercept point, then the elevation angle is computed from the NED values.

$\theta_{elevation}$ - $\tan^{-1}(-z_{NED}/x_{NED})$

where care should be taken to ensure that $\theta_{azimuth}$ is in the correct quadrant. An optional elevation lead angle can be added to $\theta_{elevation}$. The angular rotation's longitude, latitude, azimuth, and elevation can now be used to define the rotation matrix to set the Euler angles in a coordinate frame where the X-Y plane is aligned with the azimuth to target.

The Euler angles describe the initial orientation of the missile's velocity vector in the Launch azimuth frame:

$$\Phi = \tan^{-1}\left(\frac{-\sin\theta_{az}\cos\theta_{lat}}{\sin\theta_{el}\cos\theta_{az}\cos\theta_{lat}-\cos\theta_{el}\sin\theta_{lat}}\right) -\pi \le \Phi \le \pi$$

$$\Theta = \sin^{-1}(\cos\theta_{el}\cos\theta_{az}\cos\theta_{lat}-\sin\theta_{el}\sin\theta_{lat}) -\frac{\pi}{2} \le \Theta \le \frac{\pi}{2}$$

$$\Psi = \tan^{-1}\left(\frac{\sin\theta_{el}\cos\theta_{lat}\sin\theta_{lon}+\cos\theta_{el}(\sin\theta_{az}\cos_{lon}-\cos\theta_{az}\sin\theta_{lat}\sin\theta_{lon})}{\sin\theta_{el}\cos\theta_{lat}\sin\theta_{lon}-\cos\theta_{el}(\sin\theta_{az}\sin_{lon}-\cos\theta_{az}\sin\theta_{lat}\sin\theta_{lon})}\right) -\pi \le \Psi \le \pi$$

where

$\Theta_{lat} = 0.0$
$\Theta_{lon} = 0.0$

$$\Theta_{az} = \Theta_{FlyToPo\,\mathrm{int}} - \left(\Theta_{azimuth} - \Pi/2\right)$$

$$\Theta_{el} = LaunchElevation$$

The iteration on flight path angle option is designed for computing an angle for a portion of a flight path angle curve specification to allow the interceptor to fly through the target without additional guidance. The target position for this case is the position computed from the intercept planning algorithm using the flyout table; thus, the more accurate the flyout table, the less requirement for flight adjustment post the flight path angle curve flight.

The initial guess, FPA0, for the iteration scheme is to use the elevation angle above the local horizontal from the launch location to the intercept point. This initial guess will tend to produce an initial result below the intercept point. With the initial guess, the interceptor will be flown utilizing pure ballistic for any ballistic PPN modes and with no consideration of proportional navigation, i.e. attempting to hit the point ballistically.

The elevation angle, Elev0, above the local horizontal from the launch location to the final interceptor position at which it passed the target point is next computed. The desire is to zero the difference between this elevation angle and the desired elevation angle, i.e. the angle to the computed target point.

The second guess, FPA1, is computed as an adjustment to first guess using the outcome of that guess. The division by ten will slow the iteration, but help insure that a shot near maximum range will select the correct solution.

FPA1 = FPA0 + (FPA0 – Elev1)/10;

The interceptor is now flown to the resultant intercept point. On each subsequent iteration, the secant method is applied on the two prior flight path angle values and results to determine the next value of fpa. FPA1 and FPA0 are adjusted by subtracting the desired elevation to the target point out of these values. TargetMiss and TargetMiss0 are the miss distances from the prior iterations.

FPA = FPA1 – TargetMiss* (FPA1 – FPA0)/(TargetMiss – TargetMiss0)

The FPA value is bounded so that once a value is found below the target, no FPA value lower will be considered and once a value is found above the target, no FPA value greater will be considered. The MinFPA and MaxFPA are initialized to -90 and +90 degrees respectively. If an FPA value is computed below the minimum by the above equation, the FPA is recomputed as

FPA = 0.5*(FPA1 + MinFPA).

If an FPA value is computed above the maximum, then the FPA value is recomputed as

FPA = 0.5*(FPA1 + MaxFPA).

If the interceptor final altitude is less than zero then the FPA value will be recomputed and the interceptor flown again until the impact altitude is greater than zero. Whenever the interceptor is impacting the ground prior to passing the target, the resultant trajectory does not provide information that can aid the iteration. If on an iteration past the first iteration, then the FPA value is recomputed as

FPA = 0.5*(FPA + FPA0).

If | ((FPA1-FPA0)/FPA1) | < 0.001 then the interceptor is unable to reach the target without impacting the ground. In this case, the missed distance will be set to the iteration tolerance to accept the solution. If on the first iteration and the interceptor intercepts the ground, then the FPA value is recomputed as

FPA = FPA + 0.1. and the FPA0 is set to the current FPA value which resulted in a ground impact without reaching the target.

Once the interceptor missed distance value is less than the iteration tolerance, the iteration has been successful. If after fifteen iterations no solution has been found, the pitch value which resulted in the least miss distance is used to fly the interceptor one more time utilizing the Ballistic PPN mode. If no solution is found after this iteration, the interceptor cannot reach the target and the latest solution will be used.

5.7.3.7 Interceptor Explicit Flight From Burnout

When analysis of boost phase flyout is not required, interceptors may optionally be flown explicitly using a ballistic propagation of the interceptor's final burnout state. This capability requires the interceptor's weapon definition to include a flyout table that specifies burnout state. Flyout tables specifying burnout state may be constructed in an automated fashion using the flyout table generator as described in Appendix B5. This capability is generally applied to interceptors that achieve exo-atmospheric final burnout and intercept.

At interceptor launch time, the burnout state associated with the planned intercept position is computed by C3I. Burnout state actions are used to send the computed burnout time, position and velocity vectors to flight processing. Flight processing then creates the interceptor platform and uses the burnout state to initialize the interceptor's position and velocity. Also, the interceptor's time in flight is initialized using the burnout time provided by C3I. The interceptor is then activated and flown ballistically from burnout time to intercept time. There is no possibility for any type of maneuver during post burnout flight. This should be taken into consideration when defining interceptor airframes to automatically generate flyout tables. This burnout state capability is also based on a further

assumption that the interceptor is flown in-plane from launch to the interceptor position. Therefore, the burnout position vector is computed in-plane from launcher position to intercept position using the burnout ground range and burnout altitude from the flyout table. Furthermore, the velocity vector direction is also computed in the same plane using the burnout speed and burnout flight path angle from the flyout table. The details of the burnout state calculations are provided below. During actual post burnout flight, the last flight section of the interceptor's airframe is used to determine aerodynamics and signature.

The burnout position, BOPos, is calculated using the standard function to calculate an end position from a starting position, altitude, heading and ground range. The ground range and altitude at the time of burnout are used to calculate the end position. Refer to section 5.2.8.4.2.4 for a detailed description of these calculations.

For calculating burnout velocity, the rotation from local velocity frame to ECEF must be computed.

$$\text{RotMat} = \begin{vmatrix} BOX_X & BOX_Y & BOX_Z \\ BOY_X & BOY_Y & BOY_Z \\ BOZ_X & BOZ_Y & BOZ_Z \end{vmatrix}$$

where BOX, BOY and BOZ are computed as follows:

The local up vector BOZ is computed at the burnout position using the function ECEFtoUp. Refer to section B.10.20 for a detailed description of this calculation.

Next , the vector from burnout position to intercept position is computed.

$$\text{BOToIntPos} = \text{TgtPos} - \text{BOPos}$$

Compute Y and X direction vectors at burnout position

$$\text{BOY} = \text{BoToIntPos} \times \text{BOZ}$$
$$\text{BOX} = \text{BOY} \times \text{BOZ}$$

The BOX, BOZ plane is the plane in which the burnout velocity is established in local coordinates as follows:

$$Vel_X = \text{Cos}(\text{FpAngle}) * \text{BOSpeed}$$
$$Vel_Y = 0$$
$$Vel_Z = \text{Sin}(\text{FpAngle}) * \text{BOSpeed}$$

The burnout velocity in ECEF coordinates is then calculated using the Rotate2 function in which the transpose of the transformation matrix is multiplied

by the weapon velocity in local coordinates. For a detailed description of the Rotate2 function refer to appendix B10.3.13.

where,

FpAngle	- Flight Path Angle at the time of BurnOut
RotMat	- Transformation Matrix
BOPos	- Position calculated at the time of burnout
TgtPos	- Target intercept position
BOX	- X direction vector at burnout position
BOY	- Y direction vector at burnout position
BOZ	- Z direction vector at burnout position
BOSpeed	- Speed at the time of burnout.

5.7.3.8 Milestone Event Data

At major milestones throughout the flyout, the model stores data to be analyzed at the completion of the flyout. These milestones are launch, flight section transitions, apogee, reentry, and impact. The data stored are the times at which events occur (sec), altitudes (m), ground ranges (m), missile velocities (m/sec), and flight-path angles (deg). All data are used to shape missile trajectories to match documented classified or unclassified trajectories for particular missile systems and are provided to the user through the graphical interface. They are not available during the run-time processes.

Ground range is calculated using the DblGroundRange function described in Appendix B10.

Flight-path angle is calculated by:

$$w = \text{atan}\left[V_z / \text{sqrt}(V_x^2 + V_y^2)\right]$$

where

w - angle between the missile velocity vector and the local horizontal measured positive up (deg)

V_n - Missile velocity in ECEF n-direction (m).

5.7.3.9 Ballistic Missile Launch Schemes

The Launch Iteration Scheme supports seven modes of ballistic missile guidance: Lofted, Depressed, Minimum Energy, Burn Time Iteration, Non-Thrust Terminating ReEntry Angle, Thrust Terminating ReEntry Angle and Disable Launch Iteration. To model these modes, three primary iteration schemes are used. The Lofted/Depressed Iteration Scheme determines a pitch rate or angle that allows

the missile to fly the range to the target. The Minimum Energy Iteration Scheme determines the combination of optimum pitch angle or rate and minimum burn time that allows the missile to fly the range to the target. The Burn Time Iteration Scheme uses the user-specified pitch angles and rates to determine a burn time that allows the missile to fly the range to the target. The Non-Thrust Terminating Reentry Angle Iteration Scheme utilizes Guidance Energy Management System (GEMS) maneuvers to generate ballistic missile flyout solutions. The Thrust Terminating Reentry Angle Iteration Scheme determines the combination of pitch angle/rate and burn time for achieving the desired reentry angle and range to the target. Disable Launch Iteration is a missile guidance mode that disables all pitch and burntime iterations to allow for easier missile characterization checkout. All the schemes measure pitch angle from the launch reference frame local vertical. In discussing the iteration schemes, pitch refers to both the iterated pitch angle and pitch rate unless otherwise specified.

Predictive proportional navigation (PPN) is disabled for the Ballistic PPN and FPA PPN flight modes during the launch iteration. With PPN disabled, Ballistic PPN uses pure ballistic flight and FPA flies as if the PPN option is disabled.

In addition to supporting the basic guidance modes, the launch iteration schemes support the rotating Earth option. If Earth rotation is being modeled, the launch azimuth must be adjusted for the missile to land on the desired target location. The algorithm to accomplish this adjustment is described in Subsection 5.7.3.11.

5.7.3.9.1 Data Initialization

All iteration schemes require the iteration pitch parameter and the iteration burn time parameter to be initialized prior to the start of the iterations. The pitch parameter is initialized to the user-specified value of the pitch angle or rate for the guidance phase being iterated. The burn time parameter is initialized to the minimum of the cut-off time of the flight section being iterated and the actual burn time allowable due to the amount of fuel present on the iterated flight section. Initialization for the iteration schemes also includes the determination of two initial guesses of pitch angle, pitch rate, or burn time, along with their corresponding ranges. Additionally, tests are performed prior to the iteration search to ensure that the target range is bounded by the user-specified minimum and maximum ranges of the missile.

5.7.3.9.2 Lofted/Depressed Iteration Scheme

The Lofted/Depressed Iteration Scheme determines a pitch angle or rate that allows the missile to fly the range to target using the secant algorithm discussed in Subsection 5.7.3.13.1. The lofted mode yields a pitch angle or rate that results in the most vertical trajectory for achieving the desired range. Conversely, the

depressed mode yields a pitch angle or rate that results in the most horizontal trajectory desired.

5.7.3.9.3 Launch Time Iteration Scheme

The Launch Time Iteration Scheme determines the missile launch time needed to reach the desired target impact time, using the current missile launch iteration scheme. This routine is used when the user selects Update Launch on either the Target Window or the Target Spreadsheet.

When performing the iterations, the missile's launch time is set to zero and the ground range solution of the missile is then found by using the launch iteration scheme specified for the weapon. If the resultant flight time is greater than the desired impact time to the target, the missile cannot converge on a launch time to reach the target at the desired impact time. Otherwise, the first initial launch time and impact time guesses are stored. Next, the second guess is found by setting the launch time to the desired impact time minus the time of flight at zero launch time and flying the missile. The secant equation is then used to converge on a launch time:

$$X_n = X_{n-1} - \frac{Y_{n-1} \times (X_{n-1} - X_{n-2})}{Y_{n-1} - Y_{n-2}}$$

where

X_n - new launch time
X_{n-1} - second launch time guess
X_{n-2} - first launch time guess
Y_{n-1} - impact time resulting from the second launch time
Y_{n-2} - impact time resulting from the first launch time.

Next, the impact time given the new launch time is found. The secant equation variables are then reset:

$X_{n-2} = X_{n-1}$
$Y_{n-2} = Y_{n-1}$
$X_{n-1} = X_n$
$Y_{n-1} = Y_n$

where

X_n - current launch time
X_{n-2} - first launch time guess
X_{n-1} - second launch time guess
Y_n - current impact time
Y_{n-2} - impact time resulting from first launch time guess
Y_{n-1} - impact time resulting from second launch time guess

This procedure is repeated until either the impact time is within 0.1 sec of the desired impact time or 20 iterations have been executed.

5.7.3.9.4 Application of Secant Method

Two initial pitch guesses and their corresponding ranges are determined. The first guess is a lofted or depressed pitch value, depending on the missile guidance mode, whose range is greater than the desired target range. The second guess is a pitch value whose range is less than the desired target range. These guesses, as well as all subsequent pitch guesses, are bounded by the pitch program and the physical limits of the missile. The missile is flown through all guidance phases previous to the iterated phase. Given that q is the missile pitch orientation after these phases and Dt is the time spent in the iterated guidance phase, the lower bounds are:

$\theta_L = \frac{-\theta}{\Delta t}$ for pitch rate

$\theta_L = -\theta$ for pitch angle.

The upper bounds are:

$\theta_U = \frac{(180-\theta)}{\Delta t}$ for pitch rate

$\theta_U = 90$ for the first phase pitch angle

$\theta_U = (180 - \theta)$ for all other phase pitch angles.

After the two pitch guesses and corresponding ranges have been determined, the Secant Method is used to calculate subsequent pitch guesses. For first-time calculations, a modified secant equation is used for stability purposes:

$$X_n = X_{n-1} - .5 \times \left[\frac{Y_{n-1} \times (X_{n-1} - X_{n-2})}{Y_{n-1} - Y_{n-2}} \right] \quad .$$

For all subsequent pitch guess calculations:

$$X_n = X_{n-1} - \frac{Y_{n-1} \times (X_{n-1} - X_{n-2})}{Y_{n-1} - Y_{n-2}}$$

where

X_n - new pitch
X_{n-2} - first pitch guess
X_{n-1} - second pitch guess
Y_{n-2} - first range resulting from first pitch

Y_{n-1} - second range resulting from second pitch.

If the new pitch X_n is greater than θ_U or less than θ_L or if more than 50 iterations have occurred, the Interpolation Iteration Scheme described in Subsection 5.7.3.13.2 is used to determine the correct pitch.

The missile is flown using the new pitch guess. If the resultant range Y_n is greater than the maximum achieved range, the maximum achieved range is reset to this value. The secant equation variables are then updated:

$$X_{n-2} = X_{n-1}$$
$$Y_{n-2} = Y_{n-1}$$
$$X_{n-1} = X_n$$
$$Y_{n-1} = Y_n$$

where

X_n - current pitch
X_{n-2} - first pitch guess
X_{n-1} - second pitch guess
Y_n - range resulting from new pitch
Y_{n-2} - first range resulting from first pitch
Y_{n-1} - second range resulting from second pitch.

The secant equation is again used to determine a new pitch guess. This process is repeated until the resulting range is within 100m of the range to target or until 50 iterations have been executed.

5.7.3.9.5 Application of Interpolation Scheme

The Interpolation Iteration Scheme is not a primary scheme. Instead it is used as a backup when the Lofted/Depressed Iteration Scheme does not converge on a pitch angle or rate. The Lofted/Depressed Iteration Scheme keeps track of the initial range and pitch guesses, the maximum achieved range and its corresponding pitch, and the last range achieved and its corresponding pitch. These values serve as the bounds for the initial interpolation/extrapolation calculation.

The missile pitch is calculated using the interpolation/extrapolation equation:

$$X_n = X_1 + (X_2 - X_1) \times \frac{Y_n - Y_1}{Y_2 - Y_1}$$

where

X_n - new pitch
X_1 - initial pitch guess
X_2 - last lofted/depressed pitch guess

Y_n - target range
Y_1 - range resulting from initial pitch guess
Y_2 - range resulting from last pitch guess.

If the new pitch X_n is greater than θ_U or less than θ_L described in Subsection 5.7.3.9.3, it is reset to the average of the pitch values which bound the desired pitch:

$$X_n = \frac{\theta_{BH} + \theta_{BL}}{2.0}$$

where

θ_{BH} - last lofted/depressed pitch value
θ_{BL} - initial pitch value.

θ_{BH} represents the pitch that gives a range closest to but greater than the target range. θ_{BL} represents the pitch which gives a range closest to but less than the target range. The missile is flown using the computed pitch X_n to get the output range Y_n. The interpolation/extrapolation variables are then updated:

$X_1 = X_2$
$Y_1 = Y_2$
$X_2 = X_n$
$Y_2 = Y_n$

where

X_n - current pitch
X_1 - first pitch guess
X_2 - second pitch guess
Y_n - current output range
Y_1 - range resulting from first pitch guess
Y_2 - range resulting from second pitch guess.

If the output range is less than the desired target range, but greater than the closest range to the target range thus far, θ_{BL} is reset to the pitch corresponding to this output range. Similarly, if the output range is greater than the desired target range but less than the best range greater than the target range, θ_{BH} is reset. Finally, if the output range is greater than the maximum achieved range, the maximum achieved range is reset to the output range.

Once all of the interpolation and boundary variables are reset, a new pitch value X_n is found using the interpolation/extrapolation equation and is then, if necessary, limited by the boundary variables. This procedure is repeated until the output range is within 100 m of the range to target or until 100 iterations have been

executed. If 100 iterations have been executed, an error message is returned stating that the search was unsuccessful. The missile failed flag is also set.

5.7.3.9.6 Minimum Energy Iteration Scheme

The Minimum Energy Iteration Scheme determines the combination of the optimum pitch with minimum burn time for achieving the range to the target using the secant and the least squares algorithms described in Subsections 5.7.3.13.1 and 5.7.3.13.3. This scheme is computationally more intensive than any of the other schemes.

When performing the iterations, the minimum burn time BT_{MIN} for the iterated flight section is the cut-off time of the previous flight section. The maximum burn time BT_{MAX} is the minimum of the iterated flight section cut-off time and the burn time allowable due to fuel constraints. The burn time iteration parameter is initially set to BT_{MAX}. The maximum range of the missile is then found for this maximum burn time setting. This maximum range search varies the pitch while maintaining a constant burn time to achieve the maximum range possible. See Subsection 5.7.3.9.7 for more information on the maximum range search. If the resultant maximum range is not greater than or equal to 100 m of the range to target, the search iterations are aborted since the missile cannot fly farther than this maximum range. Otherwise, the maximum range and corresponding burn time serve as the first of two initial guesses.

The second guess is found by resetting the burn time to half of the maximum value and flying the missile using the same pitch as the maximum range case. Then, using these two guesses, the secant equation is used to converge on a new burn time:

$$X_n = X_{n-1} - \frac{Y_{n-1} \times (X_{n-1} - X_{n-2})}{Y_{n-1} - Y_{n-2}}$$

where

X_n	-	new burn time
X_{n-1}	-	second burn time guess
X_{n-2}	-	first burn time guess
Y_{n-1}	-	range resulting from second burn time guess
Y_{n-2}	-	range resulting from first burn time guess.

If the new burn time X_n is greater than the maximum burn time BT_{MAX}, X_n is reset to BT_{MAX}. Similarly, if X_n is less than the minimum burn time BT_{MIN}, X_n is reset to BT_{MIN}.

Next, the maximum range for the missile using the new burn time is found. The secant equation variables are reset using the maximum range Y_n and the current burn time X_n as the new second guess:

$X_{n-2} = X_{n-1}$
$Y_{n-2} = Y_{n-1}$
$X_{n-1} = X_n$
$Y_{n-1} = Y_n$

where

X_n	-	current burn time
X_{n-2}	-	first burn time guess
X_{n-1}	-	second burn time guess
Y_n	-	maximum range resulting from the current burn time
Y_{n-2}	-	range resulting from first burn time guess
Y_{n-1}	-	range resulting from second burn time guess.

If the resultant maximum range is greater than the maximum achieved range, the maximum achieved range is reset to the maximum range.

This procedure is repeated until the maximum range is within 100 m of the desired range to the target or 40 iterations have been executed. This scheme generally converges within 10 iterations, so if the number of iterations does exceed 40, the algorithm switches to a secondary search using the Burn Time Iteration Scheme described in Subsection 5.7.3.9.8. If any ranges during the 40 iterations were within $\pm$300 m of the range to target, the average of these pitch values is used when iterating on burn time in the Burn Time Iteration Scheme. Otherwise, the average of the pitch values over the entire number of iterations is used.

5.7.3.9.7 Maximum Range Scheme

The Maximum Range Scheme determines the maximum range of a missile, using the least squares algorithm described in Subsection 5.7.3.13.3 and the Maximum-Minimum Theorem of Calculus.

Keeping the input burn time constant, a step search is performed by varying the missile pitch and flying the missile until the current range is less than the previous range. ***Figure 5.7-8*** shows a curve of pitch versus range.

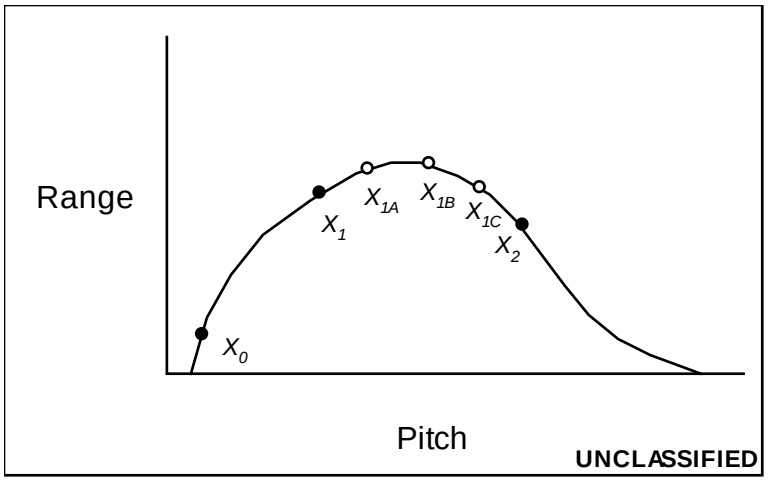

Figure 5.7-8 Missile Pitch Versus Achieved Ground Range

The search starts at pitch X_0 and flies the missile, resulting in the range associated with point X_0. The pitch is then varied, resulting in the range associated with point X_1, which is greater than the range at X_0. This is repeated until the pitch is varied enough to pass the maximum range and yield a decrease in range, as at point X_2. The step increment is then decreased and the routine collects five pitch-versus-range points: X_1, X_{1A}, X_{1B}, X_{1C}, and X_2. This finer search is performed to get a better curve profile.

Once these points are found, the least squares algorithm is used to approximate a second-degree polynomial that expresses range as a function of pitch. Range values of less than 60% of the maximum range value are first filtered out to ensure that the least squares polynomial will have a maximum value close to the actual maximum value. Then, the least squares algorithm computes the coefficients of a polynomial of the form:

$$Y = b_0 + b_1X + b_2X^2$$

where

Y	-	range
b_n	-	second order equation coefficients
X	-	pitch.

The Maximum-Minimum Theorem of Calculus states that the maximum value of Y occurs when the derivative of Y with respect to X equals 0; therefore:

$$0 = b_1 + 2b_2X$$

and solving for X yields:

$$X = -\frac{b_1}{2b_2}.$$

This value is the pitch at which the missile range is at a maximum. The missile pitch is then set to this value and the missile is flown. The resultant range is the maximum range.

5.7.3.9.8 Burn Time Iteration Scheme

The Burn Time Iteration Scheme determines missile burn time needed to reach the desired target range, using the current missile pitch. This routine may serve as either a primary or secondary iteration scheme.

The missile is flown using the maximum burn time BT_{MAX}. See Subsection 5.7.3.9.5 for more details on BT_{MAX} and BT_{MIN}. If the resultant range is less than the desired range to the target, the missile cannot converge on a burn time to reach the target. Otherwise, the first initial range and time guesses are stored. Next, the second guess is found by setting the burn time to half the maximum value and flying the missile. The secant equation is then used to converge on a burn time:

$$X_n = X_{n-1} - \frac{Y_{n-1} \times (X_{n-1} - X_{n-2})}{Y_{n-1} - Y_{n-2}}$$

where

X_n - new burn time
X_{n-1} - second burn time guess
X_{n-2} - first burn time guess
Y_{n-1} - range resulting from the second burn time
Y_{n-2} - range resulting from the first burn time.

If the new burn time X_n is greater than the maximum burn time BT_{MAX}, X_n is reset to BT_{MAX}. Similarly, if X_n is less than the minimum burn time BT_{MIN}, X_n is reset to BT_{MIN}.

Next, the maximum range given the new burn time is found. The secant equation variables are then reset:

$X_{n-2} = X_{n-1}$
$Y_{n-2} = Y_{n-1}$
$X_{n-1} = X_n$
$Y_{n-1} = Y_n$

where

X_n - current burn time
X_{n-2} - first burn time guess

X_{n-1} - second burn time guess
Y_n - current output range
Y_{n-2} - range resulting from first burn time guess
Y_{n-1} - range resulting from second burn time guess

and if the resultant maximum range is greater than the maximum achieved range, the maximum achieved range is reset to the maximum range.

This procedure is repeated until either the maximum range is within 100 m of the desired range to target or 100 iterations have been executed.

5.7.3.9.9 Thrust Terminating ReEntry Angle Iteration Scheme

The Thrust Terminating Reentry Angle Iteration Scheme determines the combination of pitch angle/rate and burn time for achieving the desired reentry angle and range to the target using the secant algorithm described below. When performing the iterations, pitch angle is bounded between 0.1 degrees and 90.0 degrees measured clockwise from missile pitch angle at the start of the first iterated flight mode. Pitch Rate is bounded between 0.001 degrees per second and 90 degrees per second. Burn time for the iterated cut-off section is bounded by the flight section cut-off time. User-specified reentry flight path angle limits are measured clockwise from 0.0 degrees local horizontal.

The maximum ground range is generated to see if the desired ground range is achievable by using the Maximum Range Scheme described in Section 5.7.3.9.7. If the maximum ground range is less than the desired ground range, a solution will not be found for the desired range at the specified reentry angle limits.

If the starting iterated flight mode is the first flight mode, “beginning pitch angle” is set to 0.0 degrees for an iterated parameter of Pitch Angle or to the absolute value specified by the flight mode for an iterated parameter of Pitch Rate. If the starting iterated flight mode is not the first flight mode, the missile is flown to obtain the “beginning pitch angle” at the start of the first iterated flight mode.

If the iterated parameter is “Pitch Angle”, the reentry angle iteration scheme begins by setting the iterated pitch angle to 0.1 degrees minus the “beginning pitch angle” and calling the Burn Time Iteration Scheme. The Burn Time Iteration Scheme determines the missile burn time needed to reach the desired target range, using the current missile pitch. See Subsection 5.7.3.9.8 for more information on the Burn Time Iteration Scheme. The “Pitch Angle” increment is one degree.

If the iterated parameter is “Pitch Rate”, the reentry angle iteration scheme begins by setting the pitch rate and calling the Burn Time Iteration Scheme. If the starting iterated flight mode is the first flight mode, pitch rate is set to 0.001 degrees/second. If the starting iterated flight mode is not the first flight mode, the

pitch rate is then set to the negative of "beginning pitch angle"/(Duration of iterated flight modes). The "Pitch Rate" increment is 0.4 degrees/second.

As long as no solution is found that satisfies target range, pitch angle/rate is incremented by the pitch increment and the missile is flown again calling the Burn Time Iteration Scheme. The first solution found that satisfies target range establishes the first guess. If multiple solutions are found that satisfy target range with resultant reentry angles above the maximum reentry angle, the first guess continues to be updated. If the resultant reentry angle for the first guess is less than the minimum reentry angle, the search iterations are aborted since the missile cannot satisfy the user-specified reentry flight path angle limits. Otherwise, the pitch angle/rate is incremented by the pitch increment and flown using the Burn Time Iteration Scheme until a) a solution is found that satisfies both target range and reentry angle limits, or b) the resultant reentry flight path angle is less than the user-specified minimum reentry angle. The later case establishes the second guess and begins the secant method described below.

The secant method begins using a first guess pitch angle/rate value that achieves a reentry angle greater than the maximum reentry angle and a second guess pitch angle/rate value that achieves a reentry angle less than the minimum desired reentry angle.

Using the two guesses, the secant equation is used to converge on a new pitch value:

$$X_n = X_{n-1} - \frac{Y_{n-1} \times (X_{n-1} - X_{n-2})}{Y_{n-1} - Y_{n-2}}$$

where

X_n	-	new pitch value
X_{n-1}	-	second pitch value guess
X_{n-2}	-	first pitch value guess
Y_{n-1}	-	reentry angle resulting from second pitch value guess
Y_{n-2}	-	reentry angle resulting from first pitch value guess

Next, the reentry angle for the missile using the new X_n is found. If a solution is not found that meets the target ground range, the secant equation is used to converge on a new pitch value:

$$X_n = X_{n-1} - \frac{Z_{n-1} \times (X_{n-1} - X_{n-2})}{Z_{n-1} - Z_{n-2}}$$

where

X_n	-	new pitch value
X_{n-1}	-	second pitch value guess
X_{n-2}	-	first pitch value guess

Z_{n-1} - ground range resulting from second pitch value guess
Z_{n-2} - ground range resulting from first pitch value guess

Otherwise, the secant equation variables are reset using the new reentry angle Y_n and the current pitch value X_n as the new second guess:

$$X_{n-2} = X_{n-1}$$
$$Y_{n-2} = Y_{n-1}$$
$$X_{n-1} = X_n$$
$$Y_{n-1} = Y_n$$

where

X_n - current pitch value
X_{n-2} - first pitch value guess
X_{n-1} - second pitch value guess
Y_n - reentry angle resulting from the current pitch value
Y_{n-2} - reentry angle resulting from first pitch value guess
Y_{n-1} - reentry angle resulting from second pitch value guess.

The secant equation is repeated to find a new X_n until the reentry angle is within the user-specified reentry flight path angle limits or 40 iterations have been executed.

5.7.3.9.10 Thrust Terminating ReEntry Angle Maximum Range Scheme

The Thrust Terminating Reentry Angle Maximum Range Scheme determines the maximum range of a missile that meets the user-specified reentry angle limits, using the least squares algorithm described in Subsection 5.6.3.5.6.3 and the Maximum-Minimum Theorem of Calculus.

Using maximum burn time, a step search is performed by varying the missile pitch and flying the missile until the resultant reentry angle is less than the maximum reentry angle. The figure below shows a curve of range versus reentry angle.

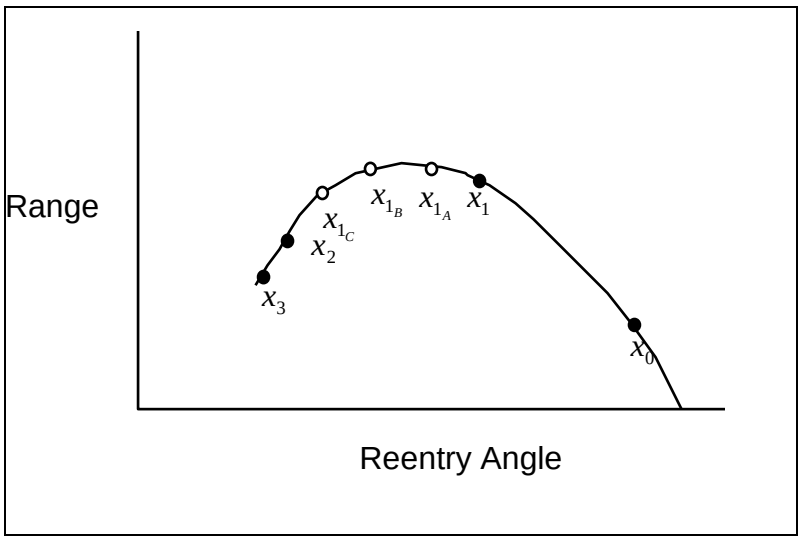

Figure 5.7-9 Ground Range Versus Reentry Angle

If the starting iterated flight mode is the first flight mode, "beginning pitch angle" is set to 0.0 degrees for an iterated parameter of "Pitch Angle" or to the absolute value specified by the flight mode for an iterated parameter of "Pitch Rate". If the starting iterated flight mode is not the first flight mode, the missile is flown to obtain the "beginning pitch angle" at the start of the first iterated flight mode.

If the iterated parameter is "Pitch Angle", the search begins by setting missile pitch angle to 0.1 degrees minus the "beginning pitch angle" and flying the missile to produce the associated ground range and reentry angle. The "Pitch Angle" increment is one degree.

If the iterated parameter is "Pitch Rate", the search begins by setting missile pitch rate and flying the missile to produce the associated ground range and reentry angle. If the starting iterated flight mode is the first flight mode, pitch rate is set to 0.001 degrees/second. If the starting iterated flight mode is not the first flight mode, pitch rate is set to the negative of "beginning pitch angle"/(Duration of iterated flight modes). The "Pitch Rate" increment is 0.4 degrees/second.

The pitch angle/rate is then incremented until the reentry angle is less than the maximum reentry angle. This establishes the first solution point for range versus reentry angle. The pitch angle/rate is then incremented until the resultant reentry angle is less than the minimum desired reentry angle. This establishes the second solution point for range versus reentry angle.

If both solutions lie in a region of increasing ground range, such as point X_0 and X_1 in ***Figure 5.7-9***, then a pitch angle/rate that produces a reentry angle closest to the minimum reentry angle is calculated yielding the maximum ground range solution. The pitch angle/rate is calculated by using the secant equation described

in Subsection 5.7.3.13.1 with the minimum reentry angle being the desired reentry angle.

If both solutions lie in a region of decreasing ground range, such as X_2 and X_3 in ***Figure 5.7-9***, the pitch angle/rate that produces a reentry angle closest to the maximum reentry angle is calculated yielding the maximum ground range solution. The pitch angle/rate is calculated by using the secant equation with the maximum reentry angle being the desired reentry angle.

If one solution lies in a region of increasing ground range and the next solution lies in a region of decreasing ground range, such as X_1 and X_2 in ***Figure 5.7-9***, the pitch angle/rate step increment is then adjusted and the scheme collects five range-versus-reentry angle points: X_1, X_{1A}, X_{1B}, X_{1C}, and X_2. This finer search is performed to get a better curve profile.

Once these points are found, the least squares algorithm is used to approximate a second-degree polynomial that expresses range as a function of pitch angle/rate. Range values of less than 60% of the maximum range value are first filtered out to ensure that the least squares polynomial will have a maximum value close to the actual maximum value. Then, the least squares algorithm computes the coefficients of a polynomial of the form:

$$Y = b_0 + b_1X + b_2X^2$$

where

Y	-	range
b_n	-	second order equation coefficients
X	-	pitch.

The Maximum-Minimum Theorem of Calculus states that the maximum value of Y occurs when the derivative of Y with respect to X equals 0; therefore:

$$0 = b_1 + 2b_2X$$

and solving for X yields:

$$X = -\frac{b_1}{2b_2} \quad .$$

This value is the pitch angle/rate at which the missile range is at a maximum. The missile pitch angle/rate is then set to this value and the missile is flown. The resultant range is the maximum range.

5.7.3.9.11 Non-Thrust Terminating ReEntry Angle Iteration Scheme

The Non-Thrust Terminating Reentry Angle Iteration Scheme utilizes Guidance Energy Management System (GEMS) maneuvers to generate ballistic missile flyout solutions. GEMS is a boost phase energy wasting technique whereby in-plane pitch up or pitch down maneuvers may be used in combination with iterated pitch angle/pitch rate flight modes for achieving desired reentry angle and range to the target.

When performing the iterations, iterated pitch angle is bounded between 0.1 degrees and 90.0 degrees measured clockwise from the missile pitch angle at the start of the first iterated flight mode. Iterated Pitch Rate is bounded between 0.001 degrees per second and 90 degrees per second. GEMS maneuvers are bounded by user-specified maximums. Reentry flight path angle limits are user-specified and are measured clockwise from 0.0 degrees local horizontal.

If the starting iterated flight mode is the first flight mode, "beginning pitch angle" is set to 0.0 degrees for an iterated parameter of Pitch Angle or to the absolute value specified by the flight mode for an iterated parameter of Pitch Rate. If the starting iterated flight mode is not the first flight mode, the missile is flown to obtain the "beginning pitch angle" at the start of the first iterated flight mode.

If the iterated parameter is "Pitch Angle", the iteration scheme begins by setting the iterated pitch angle to 0.1 degrees minus the "beginning pitch angle". If the GEMS Maneuver direction is up, the "Pitch Angle" increment is one degree. Otherwise, the GEMS maneuver is down and the pitch increment is negative one degree.

If the iterated parameter is "Pitch Rate", the iteration scheme begins by setting the pitch rate. If the starting iterated flight mode is the first flight mode, pitch rate is set to 0.001 degrees/second. If the starting iterated flight mode is not the first flight mode, the pitch rate is then set to the negative of "beginning pitch angle"/(Duration of iterated flight modes). If the GEMS Maneuver direction is up, the pitch increment is 0.4 degrees/second. Otherwise, the GEMS maneuver direction is down and the pitch increment is negative 0.4 degrees/second.

The search begins by finding an initial guess pitch as described in Subsection 5.7.3.9.10. With the initial guess being the start pitch, the iteration scheme begins by holding the start pitch constant and performing a GEMS Search. The GEMS Search finds a maneuver which meets the desired ground range. See Subsection 5.7.3.9.11.2 for more detail on the GEMS Search. After completing the GEMS Search, the resultant reentry angle is stored. If the reentry angle is within the user-specified limits, the search is complete. Otherwise, the desired reentry angle is set to the mid-point of the user-specified reentry angle limits. If the reentry angle is greater than the desired reentry angle, the reentry angle and associated pitch angle/rate are stored as ReentryAngle_1 and Pitch_1, otherwise, they are stored as

$ReentryAngle_0$ and $Pitch_0$. The iterated pitch angle/rate is incremented and the GEMS search continues until both $Pitch_0$ and $Pitch_1$ are stored. Once $Pitch_0$ and $Pitch_1$ are found, the Re-Entry Angle (REA) Search described in Subsection 5.7.3.9.12 is performed to find the pitch that meets the desired reentry angle limits. If no solution is found that satisfies the reentry angle limits, all iteration values are reset, the iterated pitch angle/rate is incremented and the GEMS Search starts over. This is repeated until the reentry angle is within the user-specified reentry angle limits or the iterated pitch angle/rate has been incremented 60 times. Through experimentation and testing, solutions were never found beyond sixty increments of iterated pitch angle/rate beyond the initial guess pitch.

5.7.3.9.11.1 Initial Guess Pitch

The initial guess pitch search begins by flying the missile to impact with no GEMS maneuver and storing the achieved ground range. If the ground range is less than the desired ground range, the pitch angle/rate is incremented by the absolute value of the "pitch increment". The missile is flown to impact again with no GEMS maneuver and the ground range is stored. The pitch angle/rate continues to be incremented, by the absolute value of the "pitch increment", until the ground range is greater than the desired ground range or the ground range begins to decrease. The initial guess pitch is set to one pitch increment less than the pitch angle/rate value when the initial guess iteration stopped.

5.7.3.9.11.2 GEMS Search

The purpose of the GEMS Search is to find a maneuver that achieves the desired ground range. The GEMS maneuver direction and maneuver rate are user-specified. If the GEMS Maneuver direction is up, the maneuver increment is 10 degrees. Otherwise, the maneuver increment is negative 10 degrees. The GEMS Search begins by flying the missile to impact using a combination of the iterated pitch angle/rate value and an initial GEMS maneuver. If the GEMS maneuver direction is up, the initial GEMS maneuver is no GEMS maneuver. Otherwise, the initial GEMS maneuver is a maximum down maneuver. A maximum down initial maneuver was chosen because most solutions are found closer to the maximum down point than the current pitch at start of the GEMS flight mode. This reduces runtime for determining solutions. During missile flight to impact, the candidate GEMS maneuver is executed and the angle of attack achieved at the end of the maneuver is held constant until the end of the GEMS flight mode. If the achieved ground range is greater than the desired ground range, the GEMS maneuver and ground range values are stored as M_1 and GR_1, otherwise, they are stored as M_0 and GR_0. The GEMS maneuver then continues to be incremented by the maneuver increment until both M_0 and M_1 have been stored. The following interpolation equation is used to compute the GEMS maneuver value required to satisfy the desired ground range.

$$DesiredManeuver = \frac{GR_0 \times M_1 + GR_1 \times M_0}{GR_1 + GR_0}$$

The missile is then flown using the current iterated pitch angle/rate value and the DesiredManeuver. If the resulting ground range is greater than the desired ground range,

M_1 = DesiredManeuver
GR_1 = ResultingGroundRange

otherwise,

M_0 = DesiredManeuver
GR_0 = ResultingGroundRange

The GEMS Search is repeated to find a new DesiredManeuver until the ground range is within the 100 meter tolerance or 20 iterations have been executed.

5.7.3.9.12 REA Search

The purpose of the REA Search is to find the pitch angle/rate that achieves a reentry flight path angle that is within the user-specified limits. The following interpolation equation is used to compute the iterated pitch angle/rate value that satisfies the user-specified reentry angle limits.

$$DesiredPitch = \frac{REA_0 \times Pitch_1 + REA_1 \times Pitch_0}{REA_1 + REA_0}$$

After DesiredPitch has been calculated, the GEMS Search is initiated for the DesiredPitch to find the GEMS maneuver which will reach the desired ground range. The reentry angle is stored from the solution found in the GEMS Search. If the resulting reentry angle is greater than the desired reentry angle,

$Pitch_1$ = DesiredPitch
REA_1 = ResultingReentryAngle

otherwise,

$Pitch_0$ = DesiredPitch
REA_0 = ResultingReentryAngle

The REA Search is repeated to find a new DesiredPitch until the reentry angle is within the user-specified reentry flight path angle limits or 20 iterations have been executed.

5.7.3.9.12.1 Non-Thrust Terminating ReEntry Angle Maximum Range Scheme

The Non-Thrust Terminating Reentry Angle Maximum Range Scheme determines the maximum range of a missile that meets the user-specified reentry angle limits, using GEMS maneuvers and maximum burn time. A step search is performed by varying the missile's iterated pitch angle/rate value and GEMS maneuver. The missile is then flown to impact until the resultant reentry angle is within the user-specified reentry angle limits and the ground range is at a maximum.

When performing the iterations, iterated pitch angle is bounded between 0.1 degrees and 90.0 degrees measure clockwise from missile pitch angle at the start of the first iterated flight mode. Iterated Pitch Rate is bounded between 0.001 degrees per second and 90 degrees per second. GEMS maneuvers are bounded by user-specified maximums. Reentry flight path angle limits are user-specified and are measured clockwise from 0.0 degrees local horizontal.

If the starting iterated flight mode is the first flight mode, "beginning pitch angle" is set to 0.0 degrees for an iterated parameter of Pitch Angle or to the absolute value specified by the flight mode for an iterated parameter of Pitch Rate. If the starting iterated flight mode is not the first flight mode, the missile is flown to obtain the "beginning pitch angle" at the start of the first iterated flight mode.

If the iterated parameter is "Pitch Angle", the iteration scheme begins by setting the iterated pitch angle to 0.1 degrees minus the "beginning pitch angle". The "Pitch Angle" increment is one degree.

If the iterated parameter is "Pitch Rate", the iteration scheme begins by setting the pitch rate. If the starting iterated flight mode is the first flight mode, pitch rate is set to 0.001 degrees/second. If the starting iterated flight mode is not the first flight mode, the pitch rate is then set to the negative of "beginning pitch angle"/(Duration of iterated flight modes). The pitch increment is 0.4 degrees/second.

The iteration begins by finding an initial guess pitch. The missile is flown to impact with no GEMS maneuver. If the resultant reentry angle is outside the reentry angle limits, the solution is discarded. Otherwise, the achieved ground range is stored as a candidate maximum ground range solution. The iterated pitch angle/rate is incremented by the pitch increment and the missile is again flown to impact and the ground range is stored. This continues until the ground range begins to decrease or the resultant reentry angle goes outside the user-specified reentry angle limits. The initial guess pitch is set to one pitch increment less than the pitch angle/rate value when the initial guess iteration stopped.

The maximum ground range solution exists between ± one pitch increment from the initial guess pitch. The upper and lower bound pitch angle/rate values are

within one pitch increment of the initial guess pitch. The upper bound pitch angle/rate is the value that produced the most ground range with no GEMS maneuver. The lower bound pitch angle/rate is the value that produced the least ground range. The desired reentry angle is set to the limit closest to the reentry angle of the initial guess pitch.

If the initial guess pitch search ended due to the resultant reentry angle of the last candidate maximum range solution being outside the user specified limits, iterated pitch angle/rate is set to the mid-point of the upper and lower bound. The GEMS Maneuver Maximum Range Iteration Search then begins as described in Subsection 5.7.3.9.12.2. Otherwise, the initial guess pitch ended due to the decrease in achieved ground range of the last candidate maximum range solution and the iterated pitch angle/rate is set to the mid-point of the upper and lower bound. The GEMS Maneuver Maximum Range Least Squares Search then begins as described in Subsection 5.7.3.9.12.3.

If the GEMS Maneuver Maximum Range Iteration Search or the GEMS Maneuver Maximum Range Least Squares Search returns no solution, the upper bound is set to the iterated pitch angle/rate. Otherwise, if the resultant ground range from the GEMS search is greater than the ground range associated with the upper bound pitch angle/rate, the iterated pitch angle/rate is set to the upper bound pitch. Otherwise, the iterated pitch angle/rate is set to the lower bound pitch. Next, a binary search is performed between the upper bound and the lower bound to reset the iterated pitch angle/rate. For each step in the binary search, either the GEMS Maneuver Maximum Range Search or the GEMS Maneuver Maximum Range Iteration Search is repeated. The binary search continues until the difference between upper bound and lower bound is less than 0.001 degrees or 10 iterations have been executed.

5.7.3.9.12.2 GEMS Maneuver Maximum Range Iteration Search

The GEMS Maneuver Maximum Range Iteration Search holds the current iterated pitch angle/rate constant and finds the GEMS maneuver that produces the max ground range solution. The search begins by flying the missile to impact using the current iterated pitch angle/rate with no GEMS maneuver. If the resultant reentry angle is greater than the desired reentry angle, the GEMS maneuver and reentry angle values are stored as M_1 and REA_1. Otherwise, they are stored as M_0 and REA_0. The GEMS maneuver then continues to be incremented by 10 degrees until both M_0 and M_1 have been stored or the maximum allowable maneuver has been reached.

If both M_0 and M_1 are stored, the following interpolation equation is used to compute the GEMS maneuver value required to satisfy the desired reentry angle.

$$DesiredManeuver = \frac{REA_0 \times M_1 + REA_1 \times M_0}{REA_1 + REA_0}$$

The missile is then flown using the current iterated pitch angle/rate value and the DesiredManeuver. If the resulting ground range is greater than the desired ground range,

M_1 = DesiredManeuver
REA_1 = ResultingReentryAngle

otherwise,

M_0 = DesiredManeuver
REA_0 = ResultingReentryAngle

The GEMS Maneuver Maximum Range Iteration Search is repeated to find a new DesiredManeuver until the reentry angle is within 0.1 degrees of the desired reentry angle or 10 iterations have been executed.

If M_0 and M_1 cannot be found and the maximum maneuver is reached, the GEMS Maneuver Maximum Range Iteration Search ends with no solution.

5.7.3.9.12.3 GEMS Maneuver Maximum Range Least Squares Search

The GEMS Maneuver Maximum Range Least Squares Search holds the current iterated pitch angle/rate constant and finds the GEMS maneuver that produces the maximum ground range solution. The search begins by finding an initial maneuver guess which results in a maximum ground range. To determine the initial guess maneuver, the missile is flown to impact using the current iterated pitch angle/rate with no GEMS maneuver. The ground range is stored and the maneuver is incremented by 10 degrees and the missile is flown to impact again. This is repeated until the ground range begins to decrease.

The initial guess maneuver is set as X_2 and its resulting ground range is set as Y_2.

X_0 = initial guess maneuver – 10 degrees
X_1 = initial guess maneuver – 5 degrees
X_3 = initial guess maneuver + 5 degrees
X_4 = initial guess maneuver + 10 degrees

For each X, the missile is flown to store the respective Y ground range values.

Once these maneuver and ground range points are found, the least squares algorithm is used to approximate a second-degree polynomial that expresses range

as a function of pitch angle/rate. The least squares algorithm computes the coefficients of a polynomial of the form:

$$Y = b_0 + b_1X + b_2X^2$$

where

Y	-	ground range
b_n	-	second order equation coefficients
X	-	maneuver.

The Maximum-Minimum Theorem of Calculus states that the maximum value of Y occurs when the derivative of Y with respect to X equals 0; therefore:

$$0 = b_1 + 2b_2X$$

and solving for X yields:

$$X = -\frac{b_1}{2b_2}.$$

This value is the GEMS maneuver at which the missile range is at a maximum. The missile's GEMS maneuver is then set to this value and the missile is flown to impact. The resultant ground range is the maximum range.

5.7.3.10 Ballistic Aim Point

Interceptors and the ballistic missiles both allow use of the ballistic aimpoint option. If the ballistic aimpoint option is selected, then the ballistic aimpoint is targeted during the launch iteration utilizing ballistic flight throughout final stage of flight. At the time of launch, the ballistic aimpoint is computed using random draws from the user defined down range and cross range offset distributions. The down range and cross range distributions define the ballistic aimpoint as an offset relative to the perceived target location. The randomized ballistic aim point is then used to compute a launch solution. If randomness is eliminated, the mean values of the down range and cross range distributions are used to specify the ballistic aimpoint. Any terminal stages will use ballistic flight throughout the final stage of flight regardless of the maneuvers or flight modes that are defined. The specified flight modes will be used during the actual fly out of the missile. The ballistic aimpoint option is only available for missiles that have terminal objects that use the Ballistic PPN flight mode. If the ballistic aimpoint option is not selected, then the launch iteration scheme flies any terminal maneuvers to the perceived target location during the launch iteration with PPN disabled. See section 5.7.3.3 for details on post boost maneuver guidance.

5.7.3.11 Iteration for Rotating Earth

If Earth rotation effects are activated, an additional iteration to determine the launch azimuth is required. First, the true ground range and azimuth to the target location are determined relative to the launch location. These computations are done in the ECEF coordinate frame. An imaginary or virtual target is created at this same true ground range and azimuth from the launch location. For the first iteration, this position will be the actual target location. Once this position is established, the iteration loop is begun.

With each iteration the missile is first flown to the current virtual target using the launch algorithms previously described in this Section. As the missile is flown to the computed true range, the Earth rotation effects will be applied. The miss distance between the final impact position and the desired target location is evaluated. If within the tolerance of 100 meters, the iteration is complete and a solution has been found. If not within the tolerance, the next iteration is performed. The actual azimuth from the launch location to the impact location produced by the previously flown launch azimuth is computed. The difference between this azimuth and the desired real target azimuth is then computed. This difference is now added to the previously flown virtual azimuth. A new virtual target is created at the true ground range and the new virtual azimuth. The iteration process is then repeated until the miss distance is less than the 100 meter tolerance. Typically, one to two iterations are required for convergence as a result of the predictability and stability of the iteration on azimuth.

5.7.3.12 Ballistic Missile Launch Azimuth

The Launch Azimuth capability allows ballistic missiles to be launched in an azimuth direction other than the azimuth associated with the shortest flight path. Along with launch direction, a user specified time to begin azimuth control allows the user to control boost phase over-flight of geographical locations and impact of boost phase debris.

Ballistic missile launch azimuth may be specified as “To Target” or “To Azimuth”. The “To Target” launch azimuth option causes the flight model to compute a launch azimuth associated with the shortest flight path to the target. The “To Azimuth” option causes the flight model to align the missile at time of launch based on the azimuth value specified in the Targets window.

In addition to options that align the missile at time of launch, a “Begin Time Az Control” value is also specified. For missile time in flight prior to the user specified begin azimuth control time, azimuth control utilizes a closed loop system that attempts to zero the azimuth between the current flight plane (i.e., velocity vector) and the azimuth at time of launch. After the user specified begin azimuth control time, azimuth control utilizes a closed loop system that attempts to zero the

azimuth between the current flight plane (i.e., velocity vector) and the plane defined by the missile's current position and the current target's position.

If earth rotation effects are activated, an iteration is used to vary the real target location creating virtual target locations. This algorithm will continue to operate as it currently operates. When not operating with this option, the missile is aligned in-plane with the virtual target each iteration to eventually zero the miss distance resulting from earth rotation. For this case, the missile is aligned at time of launch each iteration as specified in the "Launch Azimuth Specification" section above and then the azimuth control seeks to guide to the virtual target position after the "Begin Time Az Control" is met.

5.7.3.13 Missile Launch Scheme Support Algorithms

The following algorithms are used by the iteration schemes when attempting to determine a pitch and/or burn time that allows the missile to fly the range to the target. The algorithms are the Secant Algorithm, Interpolation Algorithm, and the Least Squares Algorithm.

5.7.3.13.1 Secant Algorithm

The Secant Algorithm is a method for finding roots of an equation. It is derived from the Newton-Raphson Method; the derivation is as follows:

Begin with the Newton formula:

$$X_n = X_{n-1} - \frac{f(X_{n-1})}{f(X_{n-1})^1} \quad \text{for } n > 1 \quad .$$

When evaluated functions are not continuous or do not have easily computed derivatives, it is necessary to use an approximation for the derivative term in the Newton formula.

By definition,

$$f'(x_{n-1}) = \lim_{X \to X_{n-1}} f \frac{(X) - f(X_{n-1})}{X - X_{n-1}} \quad .$$

Now, letting $X = X_{n-2}$:

$$f'(X_{n-1}) \cong \frac{f(X_{n-2}) - f(X_{n-1})}{X_{n-2} - X_{n-1}}$$

$$f'(X_{n-1}) \cong \frac{f(X_{n-1}) - f(X_{n-2})}{X_{n-1} - X_{n-2}}$$

This equation is simply the equation for the slope between two points; the derivative term in Newton's formula is also another expression for slope.

Consequently, the derivative term in Newton's formula can be replaced by the equivalent two-point slope term:

$$X_n = X_{n-1} - \frac{f(X_{n-1})}{f(X_{n-1})^1}, \text{ Newton Formula}$$

$$X_n = X_{n-1} - \frac{f(X_{n-1}) \times [X_{n-1} - X_{n-2}]}{[f(X_{n-1}) - f(X_{n-2})]}, \quad \text{Secant Formula}$$

The technique, using this formula, is called the Secant Algorithm.

5.7.3.13.2 Interpolation Algorithm

The Interpolation algorithm is also a method for finding the roots of an equation. It is derived as follows:

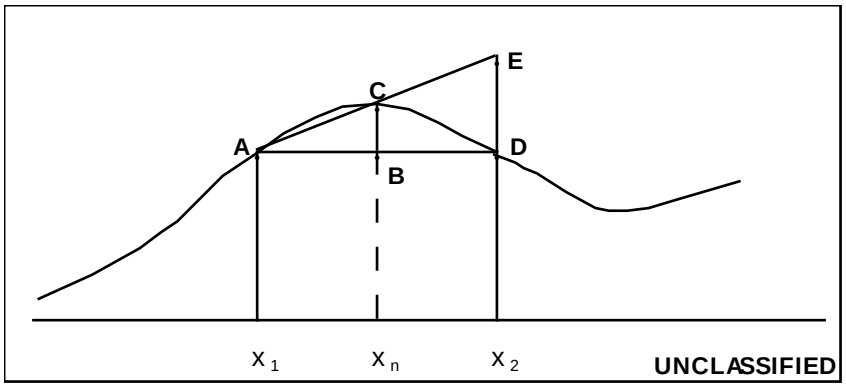

Figure 5.7-10 Illustration of Interpolation Methodology

Referring to ***Figure 5.7-10*** and using the law of similar triangles, we have the proportion

$$\frac{BC}{BA} = \frac{DE}{DA}.$$

Solving for BC:

$$BC = BA \times \frac{DE}{DA}$$

$$BC = [X_n - X_1] \times \left[\frac{f(X_2) - f(X_1)}{X_2 - X_1}\right] \quad .$$

The resulting interpolated value for $f(X_n)$

$$f(X_n) = f(X_1) + \frac{X_n - X_1}{X_2 - X_1} \times [f(X_2) - f(X_1)]$$

$$f(X_n) - f(X_1) = \frac{X_n - X_1}{X_2 - X_1} \times [f(X_2) - f(X_1)] \quad .$$

Solving for X_n:

$$\frac{f(X_n) - f(X_1)}{[f(X_2) - f(X_1)]} = \frac{X_n - X_1}{X_2 - X_1}$$

$$\frac{X_n - X_1}{X_2 - X_1} = \frac{f(X_n) - f(X_1)}{f(X_2) - f(X_1)}$$

$$X_n - X_1 = [X_2 - X_1] \times \frac{f(X_n) - f(X_1)}{f(X_2) - f(X_1)}$$

$$X_n = X_1 + [X_2 - X_1] \times \frac{f(X_n) - f(X_1)}{f(X_2) - f(X_1)} \quad .$$

The technique using this equation is called the Interpolation Algorithm.

5.7.3.13.3 Least Squares Algorithm

The Least Squares Algorithm is a technique for approximating a polynomial through a given set of data points. For our purposes this method was used to approximate a second-order polynomial with missile pitch as the independent variable and the resulting range as the response variable.

Figure 5.7-11 shows an example least squares curve that approximates a series of data points. The curve fitted by a least squares approximation is the one that makes the sum of squares of all these vertical discrepancies as small as possible.

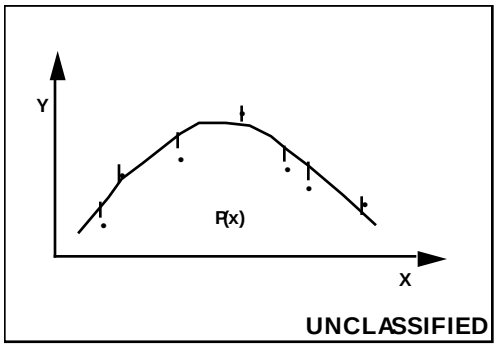

Figure 5.7-11 Example Curve for Least Squares Approximation

The general form of a second-order polynomial is:

$$Y = b_0 + b_1 X_1 + b_2 X_2^2$$

For the discrete case:

$$Y_i = b_0 + b_1 X_{1_i} + b_2 X_{2_i}^2 \quad .$$

To minimize the error between the polynomial $P_{(x)}$ and Y_i it is necessary that:

Error = $(Y_i - P(X))^2 = 0$.

To find the coefficients which minimize the actual error:

$$\frac{\partial Y}{\partial b_0}, \frac{\partial Y}{\partial b_1} \text{ and } \frac{\partial Y}{\partial b_2} = 0$$

$$\begin{aligned} Y_i &= b_0 + b_1 X_i + b_2 X_i^2 \\ X_i Y_i &= b_0 X_i + b_1 X_i^2 + b_2 X_i^3 \\ X_i^2 Y_i &= b_0 X_i^2 + b_1 X_i^3 + b_2 X_i^4 \end{aligned} \quad .$$

Expanding these equations:

$$\begin{aligned} \sum_{i=0}^{n} Y_i X_i^0 &= b_0 \sum_{i=0}^{n} X_i^0 + b_1 \sum_{i=0}^{n} X_i + b_2 \sum_{i=0}^{n} X_i^2 \\ \sum_{i=0}^{n} Y_i X_i &= b_0 \sum_{i=0}^{n} X_i + b_1 \sum_{i=0}^{n} X_i^2 + b_2 \sum_{i=0}^{n} X_i^3 \\ \sum_{i=0}^{n} Y_i^2 X_i &= b_0 \sum_{i=0}^{n} X_i^2 + b_1 \sum_{i=0}^{n} X_i^3 + b_2 \sum_{i=0}^{n} X_i^4 \end{aligned} \quad .$$

The coefficients b_0, b_1 and b_2 can be determined using linear algebra:

$$\begin{bmatrix} \sum_{i=0}^{n} Y_i X_i^0 \\ \sum_{i=0}^{n} Y_i X_i \\ \sum_{i=0}^{n} Y_i X_i^2 \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{n} X_i^0 & \sum_{i=0}^{n} X_i & \sum_{i=0}^{n} X_i^2 \\ \sum_{i=0}^{n} X_i & \sum_{i=0}^{n} X_i^2 & \sum_{i=0}^{n} X_i^3 \\ \sum_{i=0}^{n} X_i^2 & \sum_{i=0}^{n} X_i^3 & \sum_{i=0}^{n} X_i^4 \end{bmatrix} \times \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} \quad .$$

Simplifying this equation yields

$$\overline{Y} = \overline{X} \times \overline{b}$$

and multiplying both sides by $\overline{X}^{-1}$

$$\begin{aligned} \overline{Y} \times \overline{X}^{-1} &= \overline{X} \times \overline{X}^{-1} \times \overline{b} \\ \overline{Y} \times \overline{X}^{-1} &= 1 \times \overline{b} \end{aligned} \quad .$$

Once $\overline{X}^{-1}$ is determined, it is multiplied by the Y matrix, resulting in coefficients b_0, b_1, and b_2.

These coefficients represent the 0th degree, first-degree, and second-degree coefficients of a second-degree polynomial. The resulting equation is:

$Y = b_0 + b_1 X + b_2 X^2$.

In order to find a relative maximum of this equation, the first derivative is set to zero.

$$0 = b_1 + 2b_2X$$

Solving for X

$$X = \frac{-b_1}{2b_2} \quad .$$

The X_1 value represents the nominal input value for the least squares polynomial.

5.7.3.14 Missile Launch Scheduling

Launches can be scheduled for a scripted time or by command within the C3I model. The C3I model schedules missiles for launch against the targets specified by the user in the Targets list on the Edit Platform window of Scenario Generation. The missiles are launched at the user-specified launch time.

If a launch event is planned to occur between scenario intervals, the missile launch command is sent from the C3I model and is processed by the FP model at the beginning of the interval in which the missile is planned for launch. The missile is launched and is flown for the time delta between launch time and the end of the scenario interval. This produces the correct missile state at the end of the interval in which it is launched. For subsequent scenario interval updates, the missile is flown for an entire scenario interval or until the missile impacts.

The C3I model also schedules commanded launches for the surface-to-surface fire unit (SS FU) ruleset. In this case, the missiles are launched in response to a command from the artillery chain.

Ballistic missiles with MRV capability may be launched from platforms using the Red Transporter Erector Launcher (Red TEL) or SS FU ruleset. Ballistic missiles with MRV capability may be launched from these platforms at a predefined target, if that target is within weapon range. The time of the launch is specified by the user when deploying the Red TEL or SS FU platform. The platform targets contained in the Targets list are evaluated starting at the first target. If a weapon is available that satisfies the target range, it is assigned to the target. Then the next target is evaluated and assigned a weapon. This continues until all targets have been assigned a weapon or until there are no longer weapons available for assignment. In the event that a ballistic missile with MRV capability is assigned to a target, a number of succeeding targets in the target list are also associated with the MRV weapon based on the number of re-entry vehicles contained on the weapon. For example, if a MRV weapon capable of deploying ten RVs is assigned to a target, the first RV is deployed against that target and the nine remaining RVs are deployed against the nine succeeding targets in the Targets list.

5.7.3.15 Separate Ballistic Missile Launch Process

The launch solution iteration described in Subsection 5.7.3.9 is performed for every TBM launched. Since this iteration is performed while FP is updating platform state within a scenario interval, FP waits for the iteration to be complete for each missile launched within that interval before updating the other models. This causes all four models to pause while the search iterations are performed.

On the External Connections window located off the Edit Scenario window, an option exists to create a separate TBM launch process. Flight processing farms out the job of performing the launch solution search to this separate process, allowing the run-time models to continue updating without pausing to wait. One result of this procedure is that the missile does not get launched at exactly the time that C3I requested. It may in fact be one or more scenario intervals later before the missile launch is actually recognized by all four of the run-time models. How long it actually takes depends on the flyout range of the shot being processed and how much CPU time can be dedicated to finding the launch solution. The slower the machine, the longer the search for the launch solution. As a consequence, using the separate TBM launch process introduces an element of unrepeatability into the run of a scenario. This process should only be used when running with the DIS interface, which is inherently unrepeatable by itself, or when repeatability is not a factor.

5.7.3.16 Fitted Trajectory Option

The fitted trajectory option provides a mechanism to take known missile deployment and detonation information for multiple RVs from a single missile launch and achieve trajectories for the RVs and associated objects that maintain the detonation timing for each RV. This option allows trains of objects including RVs and penetration aids to be fitted to user specified trajectory parameters without the need to accurately model the boost phase or post boost phase flight.

5.7.3.16.1 Launcher Platform Restrictions

When using the fitted trajectory option, only one ballistic missile launch event is allowed per launching platform. Furthermore, only the first ballistic weapon type listed on the platform's system definition having range capability for the first target will be used. All targets on the platform's target list are considered to be the group of targets associated with the single missile launch event. For example, if a three reentry vehicle (RV) multiple independently targeted reentry vehicle (MIRV) ballistic missile weapon is the first ballistic weapon type listed on the system, the first three targets listed on the target list are targeted by the launch event. Three separate object trains are fitted to deploy against the three listed targets.

5.7.3.16.2 Optional Propagation of Spent Stages

When using the fitted trajectory option, boost, final spent upper stage and spent post boost vehicle (PBV) objects may be optionally propagated. The boost option backfills the position and velocity from the beginning of the missile launch using a linear calculation. The linear calculation takes the launch state and deployment state to calculate a straight line of flight for the booster. If the user wants to ignore this phase in terms of run-time visibility, the signature values can be set low. The final spent upper stage and PBV objects may also be optionally masked by chaff. This means that a chaff object will be co-located with the spent object during exo-atmospheric flight. A delta velocity is specified for both spent upper stage and spent PBV. The specified delta velocity for the upper stage is applied along the velocity vector of the first train reference object. The specified delta velocity for the PBV is applied orthogonal to the plane of flight for the reference object of the last train. This delta velocity is imparted orthogonal to the plane of flight in a direction obtained by taking the cross product of the position and velocity vectors of the last train reference object.

5.7.3.16.3 Fitted Trajectory Algorithm Options

There are two options for user input, Position Only and Full State, each of which operates a unique algorithm to fit the trajectory to the input information. The Position Only option requires a computationally intensive iteration to fit a trajectory to a few known data items about the trajectory. For the Position Only option, the position of the deployment state is provided as an input. In addition to the deployment position, the detonation location and time and the reentry flight path angle provide the extent of known information for the iteration. For the Full State option, the user must also provide deployment velocity and time, in addition to the position, in order for the algorithm to function. The algorithm also requires the detonation position and time. The user can additionally include state information at reentry and detonation for the Full State option. Further details on the functionality of each algorithm are provided in section 5.7.3.16.4

When using either fitted trajectory option, all targets specified on the launching platform's target list are considered to be targeted by a single launch event. The launch time of the single launch event corresponds to the specified launch time of the first listed target. Detonation time, however, is specified for each target. Detonation time is the arrival time of a train's reference object at the height of burst above the target location. After fitting the reference object trajectory using the Position Only option, deployment time for the reference object is computed by subtracting the reference object's time-of-flight from detonation time. For both Position Only and Full State options, all objects within a train are deployed at the same location and at the same time. Therefore, the deployment time of the fitted reference object will be the deployment time for all objects within the train. Deployment time for each object will be stored in the object's runtime missile

structure. For the Full State option, deployment time specified by the user is used as an initialization parameter for the Full State iteration algorithm. After fitting the reference object trajectory, deployment time for the reference object is held fixed as the user input value and the difference in time-of-flight shows up in the detonation time.

5.7.3.16.4 Deploying a RV/Chaff Train against a Target

For a given target, the first step is to fit a trajectory for the reference object in the RV/Chaff train. If the train type is specified to contain an RV, the RV is the reference object to be fitted. Otherwise, the train type specifies an "empty" chaff only train. In this case, the front most chaff cloud, i.e., the chaff cloud that re-enters first, is the reference object to be fitted. The remaining objects in the RV/Chaff train are fitted to satisfy the user specified chaff spacing distance. The chaff spacing distance is measured as the slant range distance between chaff cloud centroids. RV/Chaff train objects are fitted first and then decoys are fitted and clustered around a user specified object within the RV/Chaff train. RV, chaff and decoy objects will be fitted and explicitly flown from deployment to impact/demise.

Figure 5.7-12 illustrates the deployment of the RV/Chaff train. An iterative technique is used to find the best trajectory solution based on the launch position, the target position, and the deployment data provided. Earth rotation effects are compensated for during the iteration process. Input data may be defined as deployment position only or as full state information for deployment, re-entry, and detonation points. When the Position Only option is selected, the deployment latitude, longitude, and altitude, and the reentry flight path angle are specified and used along with the detonation position, as defined by the target platform's position and the height of burst, to define the trajectory to be fitted for the reference object. The iteration varies the flight path angle and speed at deployment and propagates the reference object to impact. Both the Position Only and Full State options assume a single plane of flight for the reference object. The plane of flight is defined by three points; earth center, deployment location and impact location. The iteration process continues until a deployment velocity vector is found that produces a trajectory passing through the deployment and impact locations and achieves a reentry flight path angle accurate to within one degree of the user specified reentry flight path angle.

An alternative option is to input Full State data, i.e. velocity vector, position vector and time, of the reference object at deployment, reentry and detonation to specify the trajectory to be fitted. The Full State option, starting from the given deployment state, iterates to reduce differences in detonation time and position of the fitted trajectory from the input values. The deployment position and time are the only inputs that are not subject to change as the trajectory is fitted. First the iteration varies speed at deployment to reduce the timing difference between the resultant detonation time of the fitted trajectory and the input detonation time.

Once the timing difference is within tolerance the iteration continues by varying the deployment velocity vector to reduce positional differences between the fitted trajectory's detonation point and the specified target location. Varying the deployment velocity vector changes the timing of the fitted trajectory, requiring further iterations on speed to reduce timing differences. The iteration process continues until a deployment velocity vector is found that produces timing and positional differences at detonation that are within tolerance or when the user defined total number of iterations is reached. The entire iteration process takes place during pre-processing. For further discussion on this pre-processing step refer to section 5.7.3.16.12. During runtime initialization, checks are made to ensure the fitted trajectory solution produces a detonation time and position within tolerance of the user specified values.

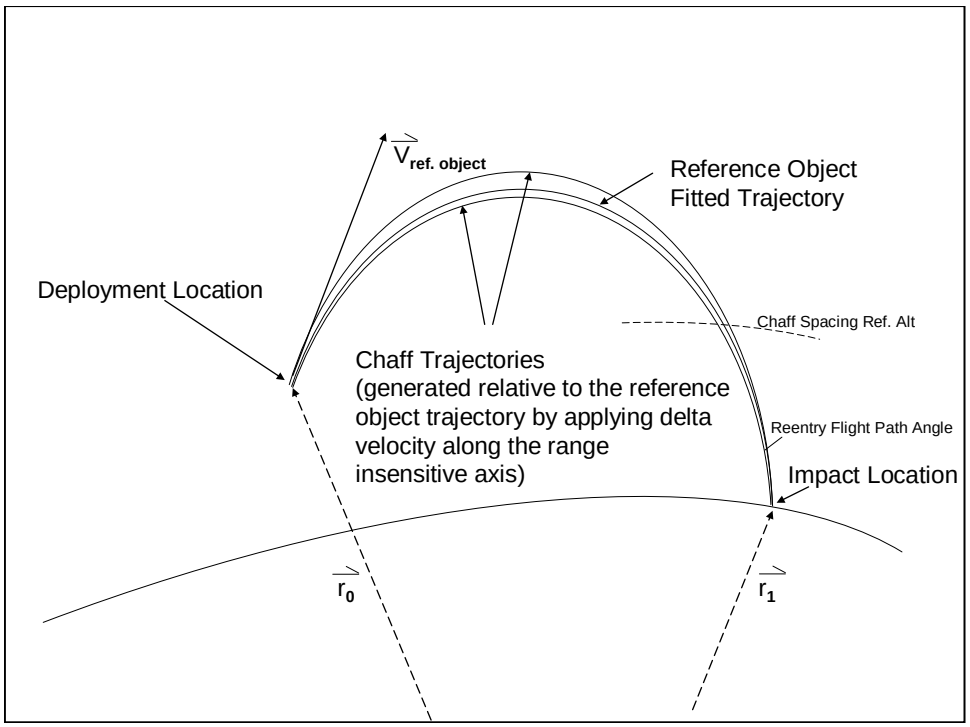

Figure 5.7-12 RV/Chaff Train Deployment

Once the reference object in the RV/Chaff train is fitted, the other chaff objects are then fitted to satisfy the specified chaff spacing. If an RV is present in the train, the RV is simply co-located with a chaff cloud. So, when a chaff cloud reaches the post apogee chaff spacing reference altitude, the chaff cloud must be separated from the next trailing chaff cloud by the specified chaff spacing. Since all chaff cloud trajectories pass through both the deployment point and impact point, they are said to be deployed along the range insensitive axis (RIA).

The iterative technique used to fit the chaff cloud trajectories first begins by computing the range insensitive axis at the deployment location of the reference object. Since all objects in the RV/Chaff train occupy the same trajectory plane,

delta velocities along the RIA may be applied to the reference object velocity vector to space the chaff clouds. An iterative technique is used to vary the delta velocity along the RIA. This processing continues until the user specified spacing between successive chaff object centroids is achieved as chaff objects pass through the chaff spacing reference altitude. If an RV is present in the RV/Chaff train, a random number draw based on a uniform distribution will determine how many of the total number of chaff clouds trajectories will be above and below the trajectory of the RV and its co-located chaff cloud. When scenario options indicate randomness is to be eliminated, one half of the chaff objects will be above and one half will be below the RV.

5.7.3.16.5 Deploying Decoy Clusters within a RV/Chaff Train

Several options are available for decoy deployments. The first option deploys no decoys. In this case, the train consists of the objects specified for the RV/Chaff train only.

A second option allows a single decoy to be deployed and co-located with each object in the RV/Chaff train. Also, for this second option, the decoys are deployed with the same deployment velocity vector as the co-located object, i.e., the decoy and the co-located object occupy the same exo-atmospheric trajectory.

A third option allows multiple decoy clusters to be deployed within a train. 0 illustrates decoy clusters within the RV/Chaff train. When deploying a decoy cluster in a train, a reference object is specified for the cluster. For a given cluster, the user specified number of decoys will be clustered around the reference object based on a user specified deployment method. Deployment methods include; 1) relative distance separation offsets, 2) relative distance and time separation offsets, 3) fixed spacing along the RIA, and 4) variable spacing along the RIA.

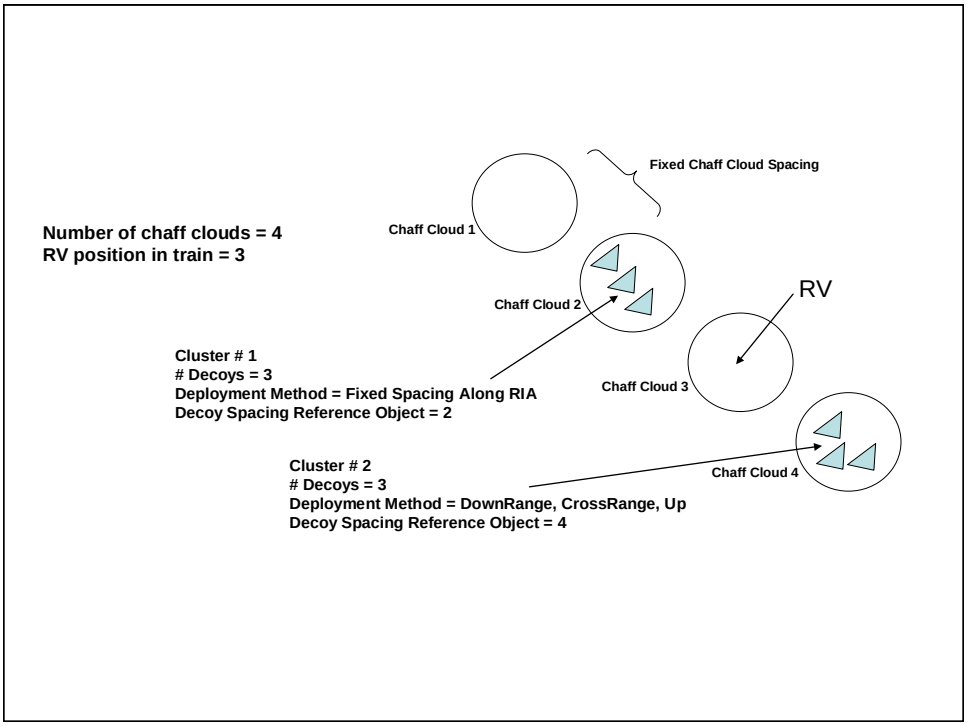

Figure 5.7-13 Decoy Cluster Deployments

The first decoy cluster deployment method requires each decoy in the cluster to pass through a point in space defined by a down range, cross range and local up offset. Furthermore, the decoy object time of flight from deployment to the decoy spacing reference altitude will equal the time of flight of the reference object time of flight to the decoy spacing reference altitude. A down range, cross range and local up offset is specified for each decoy in the cluster. These offsets are relative to the point in space where the reference object passes through the decoy spacing reference altitude. The down range separation offset is defined as the in-plane distance measured in the local horizontal plane of the reference object at the decoy spacing reference altitude. Positive values indicate down range/toward the target. Negative values indicate up range/away from target. Cross range is defined as the distance orthogonal to the plane of flight of the reference object at the decoy spacing reference altitude. Positive values indicate cross range to the left of the plane of flight when looking from the reference object toward the target. Negative values indicate cross range to the right of the plane of flight.

The second decoy cluster deployment method requires each decoy in the cluster to pass through a point in space defined by a down range and cross range offset. For this method, a time of flight separation is also applied. The time of flight separation is the time delta between the reference object achieving the decoy spacing reference altitude and the decoy achieving the decoy spacing reference altitude. Down range and cross range offsets are specified for each decoy in the cluster. These offsets are relative to the point in space where the reference object

passes through the decoy spacing reference altitude. The down range separation offset is defined as the in-plane distance measured in the local horizontal plane of the reference object at the decoy spacing reference altitude. Positive values indicate down range/toward the target. Negative values indicate up range/away from target. Cross range is defined as the distance orthogonal to the plane of flight of the reference object at the decoy spacing reference altitude. Positive values indicate cross range to the left of the plane of flight when looking from the reference object toward the target. Negative values indicate cross range to the right of the plane of flight.

The third decoy cluster deployment method deploys decoys along the RIA with fixed spacing. This deployment method uses a fitting technique identical to that used to deploy chaff clouds along the RIA with fixed spacing. A random number draw based on a uniform distribution will determine how many of the total number of decoy trajectories in the cluster will be above and below the trajectory of the reference object. When scenario options indicate randomness is to be eliminated, one half of the decoy trajectories will be above and one half will be below the reference object trajectory.

The fourth decoy cluster deployment method deploys decoys along the RIA with user specified variable spacing. This deployment method uses a fitting technique identical to that used to deploy chaff clouds along the RIA. In this case, variable spacing is applied rather than fixed spacing. A random number draw based on a uniform distribution will determine how many of the total number of decoy trajectories in the cluster will be above and below the trajectory of the reference object. When scenario options indicate randomness is to be eliminated, one half of the decoy trajectories will be above and one half will be below the reference object trajectory.

5.7.3.16.6 Object Demise

Object demise altitudes are specified for chaff and decoy objects in the weapon's stage summary table. When chaff or decoy objects reach the specified post apogee demise altitude, the objects will deactivate and disappear.

5.7.3.16.7 Fitting Algorithm Considerations

When using the fitted trajectory option, the fitting algorithm will support any combination of earth rotation, earth model, and gravity model.

The fitting algorithm implements tolerances on key iteration variables. Detonation miss distance tolerance is 100 meters. Reentry flight path angle tolerance is one degree. For the Full State option, the tolerance on detonation timing difference is 1/10 of a second. Chaff Spacing tolerance is 20 meters. Decoy offset space point tolerance is 20 meters. Decoy fixed/variable spacing tolerance is 20 meters.

5.7.3.16.8 Object Grouping

Object trains deployed using the fitted trajectory option are grouped in a manner consistent with object groupings produced by the internal flight model when performing normal boost and post boost vehicle flight deployments.

A separate train of objects is deployed against each specified target. All objects in a train share a common train ID. The train ID is assigned consistent with the internal model. The train IDs are numbered sequentially from 1 for all trains from a single launch event.

In addition to train IDs, object parent/child links must also be established. The parent ID of the first target train reference object will be the launching platform truth ID. The parent ID of all other objects in the first target train will be the reference object's truth ID. All objects in the first target train other than the reference object will be linked as children of the reference object.

The second, third, etc. target trains will also be grouped and linked as described above. However, the parent ID of the second train reference object will be the truth ID of the first train reference object. The parent ID of the third train reference object will be the truth ID of the second train reference object, etc.

5.7.3.16.9 Compound Object Name Mappings

Fitted trajectory trains may contain RVs, chaff and decoy objects. The user interface provides a field to specify the compound object name to be used when propagating an RV object, a chaff object and/or a decoy object. The compound object name mapping provides the mass, aerodynamic and signature characteristics throughout an object's trajectory.

5.7.3.16.10 Event Logging

A launch event will be logged at the launch time for the first target specified on a launching platform's target list. Since the boost phase is not being flown, the burnout event will be the next event logged after the launch event. The burnout event will be logged at the time of the first target train deployment. Object deployment events for objects within a train will be logged consistent with existing internal flight model deployments.

5.7.3.16.11 Predictive Propagation

EADSIM performs forward and backward object propagation to support launch point predictions and engagement planning. The forward/backward propagation of an object that is truly chaff or decoy type will use RV mass and aerodynamic characteristics. The RV characteristics used will be that of the RV associated with the object train. In the case that no RV is associated with the train, the first RV listed in the missile's stage summary table will be used.

5.7.3.16.12 Fitted Trajectory Pre-Processing

With state information known at deployment, reentry and detonation, feedback is provided regarding the accuracy of the fitted trajectory. Using this feedback adjustments can be made to the aerodynamic and Earth models used during pre-processing to ensure a "best fit" for the fitted trajectory to the input missile states. A tool is provided within Scenario Generation that performs the iterative process for fitted trajectory missiles. The fitted trajectories may be flown from either the Fitted Trajectory State Definition window or the Target Spread Sheet. The feedback provided consists of final differences in the state variables at deployment, reentry, and detonation, along with the total number of iterations needed to find a solution. The Fitted Trajectory State Definition window allows the user to fly a single fitted trajectory. When using the Full State option, the flight is limited to a user-defined total number of allowed iterations. When using the Position Only option, the number of allowed iterations is set internally. Feedback is provided directly on the Fitted Trajectory State Definition window in the State Info Error Table. The Target Spread Sheet allows the user to fly multiple fitted trajectories at once, restricting the trajectories to a single user-defined total number of iterations. The user has the option of either flying all or a selection of fitted trajectory objects from the Target Spread Sheet. By default, when the user selects to fly all fitted trajectories, the only reference objects that are flown and iterated are ones that do not already have a velocity solution in memory from a previous iteration. The user has the additional option of flying all fitted trajectory objects regardless of whether there is already a velocity solution in memory. Feedback is provided to the user in an output file whose name and location is specified by the user. For each trajectory, the number of iterations completed is reported on the spread sheet.

The feedback provided can be used to adjust the characteristics underlying the missile modeling to improve the trajectory fit to the input data. When initially evaluating a missile type, limiting the number of iterations to 1 will show the differences of directly using the initial velocity coupled with the aerodynamic characterization of the missile and the parameters used to model the Earth. The iteration results can be used to evaluate how closely the fitted trajectory matches the specified trajectory. For example, the user can run the iteration 1 time then examine the difference in velocity at detonation. Feedback might show that the fitted trajectory has a slower velocity at detonation than the input detonation velocity, indicating a higher drag force on the reference object. Based on these results, the user can lower the drag coefficient for the reference object thus reducing the effects of the high drag force. Drag force effects can interact with integration interval. For larger integration intervals, the non-linear errors accumulate during the flight and essentially reduce the effect of drag on the object and so the object will detonate with a greater speed; the opposite is true for reducing the integration interval. When trying to fit the detonation velocity it is recommended that the drag profile of the object be adjusted first until the desired results are reached then if

further adjustments are needed adjust the integration interval. The user should note that a threshold will be reached where further reductions in integration interval will have minimal effects on the flight. Reducing the integration interval will also increase the amount of processing time for the trajectory. Reducing the integration interval once the object reenters the atmosphere improves the modeling of the non-linear atmospheric interactions without imposing the processing penalty for the full trajectory.

Once overall speed differences at detonation are acceptable, the user can examine the state at reentry. Differences can occur from deployment to reentry, which is mostly exo-atomospheric flight, due to a mismatch in the model of Earth's gravitational potential. There is no tell-tale signs of which model to use based on the feedback from the iteration only that there is most likely a mismatch in gravitational models if there are unreasonable differences from deployment to reentry. Gravitational potential is modeled by a spherical representation of the potential and EGM96 in EADSIM. The user should note that changing the integration interval will have an effect on this part of the flight as well. Having adjusted these characteristics for one RV of a specific type should allow all trajectories for that RV type to align correctly.

The iterated deployment velocity generated for each fitted trajectory will be stored in the associated laydown file after the iteration is complete, removing the need to perform the fitted trajectory iteration at run time. Each fitted trajectory will be flown once during initialization of the first Monte Carlo run to verify that the stored deployment state produces a trajectory with a detonation state within tolerance of the input values. If the fitted trajectory was not within tolerance of the input values, the runtime will abort operation and output an error statement to the C3ILOG. In this case, the pre-processing step will need to be executed within Scenario Generation to produce or update the requisite iterated deployment velocity vector.

5.7.4 Curve-Fit Missile Model

The curve-fit missile model provides the capability to incorporate characteristics of flight which are not incorporated into the standard flight model in EADSIM, while maintaining the capability to quickly laydown a number of launch and target points for a set of TBMs. This is achieved by entering a number of trajectories consisting of key data items along those trajectories. A curve-fit is used to determine the trajectories between the input data points. For trajectories at ground ranges other than those explicitly supplied in the data file, the trajectory is interpolated between two of the input trajectories.

Input trajectories consist of several key data points along the trajectory, as opposed to data points at constant one or two second spacing. Each trajectory has an equal number of data points describing the trajectory. The data points that are selected for the input trajectories are critical to the performance of the algorithm.

When establishing the input trajectory files, trajectories should be flown at several ranges and evaluated against the known trajectories to verify accurate operation of the curve fit with respect to the input trajectories.

The computed trajectory assumes no earth rotation. The trajectory is flown from 0MSL at launch to 0MSL at impact, which means introduction of some inaccuracy during early boost and just prior to impact. The mechanism has the added advantage of being computationally inexpensive. This can provide a good alternative to the EADSIM internal flight model when run speed is highly desired, e.g. a real time experiment.

5.7.4.1 Launching a Curve-Fit Missile

The launch solution for a missile being flown according to the curve-fit model consists of establishing the coefficients of the curve fit. The input trajectories are assumed to be in ascending order from shortest range to longest range trajectory. If the range to the target is greater than the longest range trajectory or less than the shortest range trajectory, then the missile will not be launched. The code also checks for the minimum and maximum ground ranges of the missile defined by the user and will not launch if this check fails.

The bounding trajectories are next identified from the input data. These two curves are the trajectories with the closest range shorter than the desired range and with the closest range longer than the desired range. The data points for the desired trajectory are computed by linear interpolation between the two bounding curves. The interpolation is performed based on scaling between the range of the trajectories and the desired range of the specific trajectory.

$$S = \frac{R_D - R_I}{R_{I+1} - R_I}$$

where,

S - Interpolation scale factor,
R_D - Desired ground range for trajectory,
R_I - Ground range of shorter bounding trajectory, and
R_{I+1} - Ground range of longer bounding trajectory.

During the initial reading of the trajectory data, the input data is converted into ground range in meters, altitude in meters, velocity in meters/second. The input is expected to be in kilometers and kilometers/second. The input flight path angle is then used to compute ground range rate and altitude rate.

$$R_{Im} = R_{Ikm} \times 1000.$$
$$A_{Im} = A_{Ikm} \times 1000.$$
$$V_{Im/s} = V_{Ikm/s} \times 1000.$$

where,

R_{Im} - Ground Range in meters,
R_{Ikm} - Ground Range in kilometers,
A_{Im} - Altitude in meters,
A_{Ikm} - Altitude in kilometers,
$V_{Im/s}$ - Ground Range in meters/second, and
$V_{Ikm/s}$ - Ground Range in kilometers/second.

The input flight path angle and adjusted velocity vector are then used to compute the ground range rate and altitude rate for each of the data points.

$$\dot{R}_I = \frac{V_I \cos(\theta_I)}{1 + A_I/RE}$$

$$\dot{A}_I = V_I \sin(\theta_I)$$

where,

$\dot{R}_I$ - Ground range rate for point I,
$\dot{A}_I$ - Altitude rate for point I,
V_I - Velocity for point I,
θ_I - Flight path angle for point I,
A_I - Altitude for point I, and
RE - Earth radius.

The array of data points for the specific desired trajectory are computed using the above scaling based from the ground range of the trajectory. The interpolated data for each trajectory point consists of ground range, altitude, ground range rate, and altitude rate. Each of these is computed using the following standard interpolation equation.

$$x = x_I + S(x_{I+1} - x_I)$$

where,

x - Interpolated value of x,
x_I - Value of x in shorter trajectory,
x_{I+1} - Value of x in longer trajectory, and
S - Computed scale factor between trajectories.

The curve fit coefficients are next computed for each segment along the trajectory. A segment is each pair of data points along the trajectory. A loop is performed over each segment. For each segment, the matrix P is established.

$$P = \begin{bmatrix} 0 & 0 & \Delta t^3 & 3\Delta t^2 \\ 0 & 0 & \Delta t^2 & 2\Delta t \\ 0 & 1 & \Delta t & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

where,

Δt - the time difference between the data entries.

The matrix [R] is next loaded with the range and range rate values for the segment. The matrix [A] is loaded with the altitude and altitude rate values for the segment.

$$[R] = \begin{bmatrix} R_I \\ \dot{R}_I \\ R_{I+1} \\ \dot{R}_{I+1} \end{bmatrix}$$

$$[A] = \begin{bmatrix} A_I \\ \dot{A}_I \\ A_{I+1} \\ \dot{A}_{I+1} \end{bmatrix}$$

The coefficients for the equations are then obtained by multiplying the inverse of the matrix P by the R and A matrix.

$$[R_C] = [P]^{-1}[R]$$
$$[A_C] = [P]^{-1}[A]$$

5.7.4.2 Computing Missile State at a Time

Once the coefficients are established, the location of the missile at any point in time is computed from the curve fit. Time is the independent variable in the array of trajectory data points. The specific segment of the curve fit is established based on the desired time.

$$R = R_{c[0]}\Delta t^3 + R_{c[1]}\Delta t^2 + R_{c[2]}\Delta t + R_{c[3]}.$$
$$A = A_{c[0]}\Delta t^3 + A_{c[1]}\Delta t^2 + A_{c[2]}\Delta t + A_{c[3]}.$$
$$\dot{R} = 3R_{c[0]}\Delta t^2 + 2R_{c[1]}\Delta t + R_{c[2]}.$$

The range rate must be adjusted for the altitude of the trajectory point.

$$\dot{R} = \dot{R}(1 + A/RE).$$

Altitude rate is computed from the curve fit.

$$\dot{A} = 3A_{c[0]}\Delta t^2 + 2A_{c[1]}\Delta t + A_{c[2]}.$$

Finally, speed and flight path angle are computed.

$$\text{Speed} = \sqrt{\dot{A}^2 + \dot{R}^2}$$

$$\theta = \tan^{-1}\left(\dot{A} \diagup \dot{R}\right)$$

where,

θ - Missile flight path angle.

The position and velocity of the interceptor in ECEF must finally be computed. This requires translating the ground range, altitude, speed, and flight path angle into the ECEF coordinate system.

First two unit vectors are obtained.

$$\vec{U}_I = \frac{\vec{P}_I}{\left|\vec{P}_I\right|}.$$

where,

$\vec{P}_I$ - Missile impact position.

$$\vec{U}_{IL} = \frac{\vec{P}_{IL}}{\left|\vec{P}_{IL}\right|}.$$

where,

$\vec{P}_{IL}$ - Vector from missile impact to missile launch.

The angle B is then computed.

$$B = \cos^{-1}\left(\vec{U}_{IL} \bullet \vec{U}_I\right).$$

The angle A is then computed. Angle A is the angle at the center of earth between the impact position and the current position.

$A = (R_D - R)/RE.$

Angle C is computed from angles A and B.

$C = B - A.$

The distance L from the impact point to the point directly beneath the desired position can be computed from angles A and C.

L = RE(sin (A)/sin (C)).

The unit vector in the direction of the current position can then be computed.

$$\vec{V}_M = \vec{P}_I + L\vec{U}_{IM}.$$

$$\vec{U}_M = \frac{\vec{V}_M}{\left|\vec{V}_M\right|}.$$

The position can finally be computed using the computed altitude A, the radius of the earth, and $\vec{U}_M$.

$$\vec{P}_M = \vec{U}_M(RE + A).$$

The velocity vector is then computed by setting a vector in the trajectory plane at the computed flight path angle to the computed speed.

At this point, the position and velocity of the missile is established. The orientation vector will be governed by the attitude options specifiable through the Missile Definition Window.

5.7.5 Cruise Missile Flight

This subsection discusses the modeling of the simple cruise missile weapon type. More accurate modeling of cruise missiles is obtained through the captive platform capabilities. Cruise missiles are modeled with straight-line, constant velocity flight between initial location and target location. The missile is flown at a constant velocity at 200 m above ground level. A cruise missile is flown based on the missile's current position and its target's current position.

The distance vector between the missile and the target is used to determine the impact and the need for terrain following for the missile. If the distance is less than the distance the missile will fly at its current velocity during the scenario interval, the missile will impact its target during this interval. If the distance is less than the distance the missile will fly at its current velocity during three scenario intervals, terrain following will not be continued. The missile is flown for each internal time step of 0.5 sec during the scenario interval. If the missile is terrain following, a commanded altitude of 200 m is set.

5.7.6 Constant Velocity Nonaerodynamic Interceptor Flight

This subsection describes the method used for Constant Velocity Nonaerodynamic Interceptor Flight, which entails the missile's being propagated through space without any type of aerodynamic limitations. This capability may severely misrepresent a system and is not recommended for most usage. The kinematic capabilities are defined by the Constant Velocity or the Flyout Table associated with the weapon. State vectors are computed at the end of each time

interval. Positional changes of the missile are computed in straight line segments at a constant velocity. If using the Constant Velocity option, the constant velocity defined for the weapon is used to propagate the missile forward. If a flyout table is defined, the table is accessed at time of missile launch based on the target's downrange and altitude with respect to the Flexible SAM launching the weapon. The resultant time of flight from this table is converted into a velocity based on the range to the target. This velocity is then used to propagate the missile forward. Turns are instantaneous.

This model also accounts for command guided, non-homing flight versus semi-active/active homing flight. If command guided, the missile flies to the space point provided by C3I. If semi-active or active and thus homing on the target position, the missile guides according to the track position and velocity of the target platform. As the missile is being propagated forward, the target is also being propagated forward along its velocity vector so that the missile does not wind up trailing the target platform.

This model continuously checks for closest approach, which is reached when the missile does not close on the target position (i.e., the distance between the target and the missile is increasing instead of decreasing). At that time, the closest approach is flagged, allowing C3I to be notified that the intercept is complete so the kill determination and assessment can be performed. At this time, the missile itself is "backed up" to its closest approach position, since this position could have been reached in the time interval between updates. This position is used for the missile position for the kill determination and kill assessment.

5.7.7 Atmosphere Parameter Calculations

Pressure, temperature, air density, and speed of sound are calculated using pressure curve fits and temperature gradients derived from the 1962 Standard Atmosphere data. The input altitude and the calculated atmospheric conditions are all in metric units.

Four data tables of 1962 Standard Atmosphere data are used to calculate the atmosphere parameters: temperature, temperature gradient, pressure, and corresponding reference altitudes. ***Table 5.7-5*** shows these four data tables combined.

Table 5.7-5 1962 Standard Atmosphere Data Tables

Altitude H_t (m)	Temperature T_t (°K)	Temp Gradient ^2T_t (°K)	Pressure P_t (N/m^2)
0.0	288.15	-0.0065	101325.0
11000.0	216.65	0.0	22632.0
20000.0	216.65	0.0010	5474.87
32000.0	228.65	0.0028	868.014
47000.0	270.65	0.0	110.905
52000.0	270.65	-0.002	59.0005
61000.0	252.65	-0.004	18.2099
79000.0	180.65	0.0	1.0377
90000.0	180.65	0.003	0.16438
100000.0	210.65	0.005	3.0075e-2
110000.0	260.65	0.010	7.3544e-3
120000.0	360.65	0.020	2.5217e-3
150000.0	960.65	0.015	5.0617e-4
160000.0	1110.65	0.010	3.6943e-4
170000.0	1210.65	0.070	2.7926e-4
190000.0	1350.65	0.005	1.6852e-4
230000.0	1550.65	0.004	6.9604e-5
300000.0	1830.65	0.0033	1.8838e-5
400000.0	2160.65	0.0026	4.0304e-6
500000.0	2420.65	0.0017	1.0957e-6
600000.0	2590.65	0.0011	3.4502e-7
700000.0	2700.65	0.0	1.1918e-7

UNCLASSIFIED

These tables are referenced using altitudes expressed in geopotential meters. One geopotential meter is defined as the vertical distance through which a one kilogram mass must be moved to increase its potential energy by 9.80665 joules[1]. Thus, the input altitude H_o in geometric meters is converted to altitude H in geopotential meters using:

$$H = (H_o \times R_e / (H_o + R_e)) \times (g_o / G)$$

where altitudes H and H_o are in meters, Earth radius R_e and acceleration due to gravity go are dependent on latitude, and geopotential G = 9.80665 m^2/sec^2. Since the Earth is assumed to be perfectly spherical, then g_o is a constant 9.80665 m/sec^2 and R_e = 6,356,766 m, corresponding to 45-deg latitude on a non-perfect spherical Earth model using a gravity relation by Lambert[2]. Rearranging the geopotential altitude equation, it reduces to:

$H = [H_o / (1 + H_o / R_e)]$.

The next step is to determine in which reference altitude bin the geopotential altitude resides. For altitudes less than 130,000 m, dividing the altitude by 10,000 yields the starting altitude bin number for a binary search due to the structure of the reference altitude table. For altitudes greater than or equal to 130,000 m, a binary search is performed limited to the altitude bins above 130,000 m, and below the maximum table value of 700,000 m.

Once the altitude index I is found, the delta between the geopotential altitude and the table reference altitude is calculated:

$$\Delta H = H - H_t[I]$$

where ΔH, H, and altitude table H_t values are in meters.

Next, temperature T at altitude is calculated according to:

$$T = T_t[I] + \Delta T_t[I] \times \Delta H$$

where temperature T, temperature table T_t values, and temperature gradient table ΔT_t values are in °K, and ΔH is in m.

When calculating pressure P, two different curve fits to 1962 Standard Atmosphere Pressure data are used. The first curve fit is for altitudes less than 90,000 m. If $\Delta T_t[I]$ for the current table index is equal to zero:

$$P = P_t[I] \text{ x e } (A_{ag} \times \Delta H / T_t[I])$$

If $\Delta T_t[I]$ is not equal to zero:

$$P = P_t[I] \times (T_t[I] / T) (A_{ag} / \Delta T_t[I])$$

where P and pressure table P_t values are in N/m^2; T_t values, ΔT_t values, and T are in °K; ΔH is in meters; and

$$A_{ag} = A_w / R \times 9.80665 \text{ m/sec}^2,$$

where air molecular weight A_w = 28.9644 kg/kmol and gas constant R = 8314.32 joules/°Kkmol.[3,4]

For altitudes above 90,000 m, five pressure coefficients are calculated:

$C_0 = R_e + H$

$C_1 = R_e + H_t[I]$

$C_2 = T_t[I] / \Delta T_t[I]$

$C_3 = 1.0 / (C_1 - C_2)$

$C_4 = - A_{ag} \times R_e^{\ 2} \times C_3 / \Delta T_t[I]$

where R_e, H, and H_t values are in meters, and T_t and ΔT_t values are in °K.

Then

$$P = P_t[I] \times e (C_4 \times C_3 \times \log (C_1 \times (\Delta H / C_2 + 1.0) / C_0) - \Delta H / C_1 / C_0)$$

where P and P_t values are in N/m^2, and ΔH is in meters.

Once P is calculated, air density r is calculated according to:

$$r = (A_w / R) \times P / T$$

where r is in kg/m^3, A_w = 28.9644 kg/kmol, R = 8314.32 joules/°Kkmol, P is in N/m^2, and T is in °K.

Finally, speed of sound C is calculated according to:

C = sqrt [1.4 x T / (A_w / R)]

where C is in m/s, T is in °K, A_w = 28.9644 kg/kmol, and R = 8314.32 joules/°Kkmol.

Table 5.7-6 through ***Table 5.7-9*** provide verification of this methodology. ***Table 5.7-6*** shows documented atmospheric data for a 1962 Standard Atmosphere in English units. ***Table 5.7-7*** shows the corresponding data generated by the EADSIM atmosphere model. Only minor discrepancies occur.

Table 5.7-6 1962 Standard Atmosphere Data in English Units[5]

Geopotential Altitude (ft)	Pressure (psi)	Density (slug/ft^3)	Speed of Sound (ft/s)	Temperature (°R)
0.0	14.70	2.377e-3	1116.4	518.7
10000.0	10.11	1.7553e-3	1077.4	483.0
20000.0	6.754	1.2664e-3	1036.8	447.3
30000.0	4.364	0.8893e-3	994.7	411.7
40000.0	2.720	0.5851e-3	968.1	390.0
50000.0	1.682	0.3618e-3	968.1	390.0
60000.0	1.040	0.2238e-3	968.1	390.0

UNCLASSIFIED

Table 5.7-7 Atmosphere Model Data in English Units

Geopotential Altitude (ft)	Pressure (psi)	Density (slug/ft^3)	Speed of Sound (ft/s)	Temperature (°R)
0.0	14.692	2.3765e-3	1116.5	518.7
10000.0	10.104	1.7550e-3	1077.4	483.0
20000.0	6.752	1.2662e-3	1036.9	447.3
30000.0	4.363	0.8891e-3	994.7	411.7
40000.0	2.719	0.5850e-3	968.1	390.0
50000.0	1.682	0.3618e-3	968.1	390.0
60000.0	1.040	0.2237e-3	968.1	390.0

UNCLASSIFIED

Table 5.7-8 shows documented atmospheric data for a 1976 Standard Atmosphere in metric units. ***Table 5.7-9*** shows the corresponding data generated by the EADSIM atmosphere model. Comparisons of the data again reveal only minor differences.

Table 5.7-8 1976 Standard Atmosphere Data in Metric Units[6]

Geopotential Altitude (m)	Pressure (N/m^2)	Density (kg/m^3)	Speed of Sound (m/s)	Temperature (°K)
0.0	101325.0	1.2250	340.3	288.2
5000.0	54019.0	0.73612	320.5	255.7
10000.0	26436.0	0.41271	299.5	223.2
15000.0	12044.0	0.19367	295.1	216.7
20000.0	5475.0	0.088035	295.1	216.7

UNCLASSIFIED

Table 5.7-9 Atmosphere Model Data in Metric Units

Geopotential Altitude (m)	Pressure (N/m^2)	Density (kg/m^3)	Speed of Sound (m/s)	Temperature (°K)
0.0	101325.0	1.2250	340.3	288.1
5000.0	54019.9	0.73612	320.5	255.7
10000.0	26436.3	0.41271	299.5	223.1
15000.0	12044.5	0.19367	295.1	216.6
20000.0	5474.9	0.088035	295.1	216.6

UNCLASSIFIED

References

1. *Handbook of Chemistry and Physics*, 55th ed., Chemical Rubber Company, 1974, p. F-191.

2. *Handbook of Geophysics and the Space Environment*, 1985, pp. 14-7 to 14-8.

3. *Handbook of Geophysics and the Space Environment*, 1985, p. 14-7.

4. *Handbook of Chemistry and Physics*, 55th ed., Chemical Rubber Company, 1974, p. F-221.

5. *Airplane Aerodynamics and Performance*, Roskam Aviation and Engineering Corporation, 1981, pp. 11-12.

6. *Airplane Aerodynamics and Performance*, Roskam Aviation and Engineering Corporation, 1981, pp. 13.

7. "Comparison of KDEC Aries Test Article Trajectories to Off-line Model," H.R. Sells, Teledyne Brown Engineering, Technical Letter SSDCSY-93-61AB01-1533100801-0573 to Ms. Jackie Steele, USASSDC, Systems Analysis Division, 10 August 1993.

5.8 GROUND PLATFORM MOVEMENT

Ground platforms move using one of two methods. When using the constant velocity method, ground platforms move between waypoints in a straight line at a constant velocity. When using the detailed mobility model method, the speed, slope, acceleration, and deceleration are limited by maxima defined by the user for each platform. There are also two different ways to control the movement of ground platforms. Scripted ground platforms travel between waypoints based on the waypoint on and off times, speed, and vehicle characteristics defined by the user. Commanded ground platforms move between user-defined waypoints based on these parameters but additionally allow commanded assignments to control movement. Waypoints for ground platforms can be defined individually on the Edit Platform window or by using the Add Road option.

5.8.1 Ground Platform Waypoint Definition

Ground platforms move by following waypoints. Ground platforms are moved at each scenario interval. Ground platform waypoints can be defined individually in Scenario Generation by using the mouse to select locations on the map or by entering the exact latitudes and longitudes of each waypoint. Multiple waypoints along a route can be added using the Add Road option, which is currently limited to retrieving road data from the VMAP database. In both cases, the path starts at the last waypoint in the list and ends with a waypoint selected by mouse. The waypoint parameters speed, on time, off time, etc. for the added waypoints are copied from the previous waypoint if one exists. If there is no previous waypoint then the speed and on time for the newly created waypoints are set to zero. If there is no previous waypoint then the off time is set to zero for waypoints created using the Add Road option and END_SCEN when the waypoints are generated using other methods.

The Add Road option allows the user to select a series of roads as an extension of the current path. The Add Road option requires a road database, the terrain and mobility libraries, and the definition of the detailed mobility at the system level. The Add Road routing algorithm displays the current path of the vehicle to which it appends a section of roads nearest the destination which the user selects by mouse. The end of the current path is the beginning of the new road section. The route selection algorithm is based on the A* search algorithm as described in 5.8.5.2.1 and is implemented via the mobility library as described in 5.8.5.2. This feature simplifies the weighting algorithm to speed up the path selection. This method does not apply a weighting factor to the road physical characteristics, road soil types, elevation changes, or cross-country paths that may provide a more optimal route but instead selects the optimum route based only on minimum distance.

The road specific data, i.e., Road ID, Latitude, and Longitude, is retrieved from the road database into the waypoint which is labeled as a “Road” waypoint. A

single road route can consist of multiple Road waypoints in the list with the same Road ID and a straight line between any two successive Road waypoints. Typically, the end Road waypoint of one road will coincide with the starting waypoint of the next Road waypoint. In this case the duplicate Road waypoint for the end of the first road is deleted from the list and the Road waypoint for the start of the second road is kept. Thus, the start of a new road is identified with a change in the Road ID in the waypoint list.

5.8.2 Scripted Ground Platforms

A scripted ground platform moves based on user-defined on and off times and speeds for each of the platform's waypoints. The on time determines when a platform is active at a waypoint and may contribute to the platform's constant speed calculation for reaching the next waypoint. A scripted ground platform does not start to move from its first or current waypoint until the simulation time is greater than or equal to the off time for that waypoint. The off time determines when the platform is no longer active at that waypoint.

5.8.3 Commanded Ground Platforms

Commanded ground platforms require the use of the SS FU ruleset. A commanded ground platform requires more than one waypoint. The platform moves at the speed computed by the selected movement algorithm. It will not move if velocity is not specified for the next waypoint. Movement between waypoints is governed by commanded assignments, waypoint off times, and phase timing. The SS FU ruleset is described in more detail in Methodology Manual section 4.7.9.

5.8.4 Constant Velocity Movement Algorithm

When using the constant velocity movement method, ground platforms move between waypoints at a constant velocity in the direction of the next waypoint. The constant velocity is defined by the user for each waypoint as the waypoint speed. If the speed defined for the next waypoint is greater than zero, then the platform is moved at that speed. If the speed is less than or equal to zero for a destination waypoint and the on time of the next waypoint is equal to the off time of this waypoint, then the user wants the platform to travel to the next waypoint in zero time, which is an error. This is handled by moving the platform at a speed that allows it to arrive at the next waypoint in a single scenario interval. If the speed is less than or equal to zero for a destination waypoint and the on time of the next waypoint is not equal to the off time of this waypoint then the speed is calculated for the platform using the distance remaining to the next waypoint and the on time of the next waypoint. If the speed calculated is less than or equal to zero the platform is deactivated.

$$S_p = \frac{D}{t_{on} - t_{sim}}$$

where

S_p	-	Platform's speed (m/s)
D	-	Distance remaining from platform to waypoint (m)
t_{on}	-	Waypoint's On Time (s)
t_{sim}	-	Current simulation time (s)

Moving a platform includes calculating the platform position and setting the platform's velocity vector toward the next waypoint. The magnitude of the difference vector between the platform's position vector and the destination waypoint's position vector is the distance the platform must be moved. The platform's velocity vector is oriented in the direction of the difference vector with the desired speed as its magnitude. The orientation vector is set to the unit vector velocity. The new platform position is calculated using the velocity.

$$\vec{P} = \vec{P} + \vec{V} \cdot \Delta t$$

where

$\vec{P}$	-	ECEF Platform position vector (m)
$\vec{V}$	-	ECEF Platform velocity vector (m/s)
Δt	-	Scenario interval (s)

A platform is moved until it reaches the new waypoint or the waypoint's off time is exceeded and is a non-zero value. The platform is considered to be at the new waypoint if the distance calculated from the speed at which the platform travels during the scenario time step is greater than or equal to the distance that the platform must travel to reach the new waypoint. The platform's position is set to the new waypoint's position once the platform arrives at the waypoint.

If the platform has not reached the new waypoint, then its position is updated to the location of the distance covered in the direction of the new waypoint. The standard algorithms described in Appendix B10 are applied to the updated position. The function ECEFtoLatLon determines the latitude and longitude of the updated position. If the AGL option is selected or terrain following is on then the elevation of this position is obtained based on the terrain data. Otherwise the elevation is set to zero. The altitude of the platform is then computed as the terrain elevation added to the specified waypoint altitude. The LLAToECEF function is then used to compute the ECEF position of the ground platform. The AGL or terrain following options in addition to non-zero waypoint altitudes are recommended for ground platforms in order to prevent terrain masking of sensors and communications devices. If a waypoint's off time is exceeded, the platform's next waypoint is set as its destination.

5.8.5 Detailed Mobility Movement Algorithm

When using the detailed mobility model, ground systems move to their next waypoint at a velocity which depends on the slope of the terrain, soil type, and the

vehicle characteristics. The mobility and terrain libraries must be installed in order to use the detailed mobility model. If these are not installed then platforms with the detailed mobility model option selected use the constant velocity model. The detailed mobility model must be enabled in order to use the "Add Road" option for creating waypoints on a platform.

The main difference between the detailed mobility algorithm and the constant velocity algorithm is the method used to compute the speed at which to move the platform. Another difference between the two movement algorithms is the use of the on and off times. The detailed mobility model does not use the on and off times to compute the speed at which to move a platform. However, the off time can be used to make a platform stop and wait at a waypoint. If the off time is zero then it is ignored for scripted ground platforms using the detailed mobility model.

To compute the speed at which to move the platform the soil type is obtained using the road database. Currently Unknown and Asphalt are the only two soil types available through the use of VMAP. The soil type is used to lookup the vehicle characteristic parameters in the table defined on the Detailed Mobility Definition window. If a soil type is not found, then the vehicle characteristic parameters defined for the Unknown soil type are used. The speed at which the platform moves is computed by calling the CalcSpeedLL function with the vehicle characteristics as inputs. The CalcSpeedLL function is located in the mobility library and is described in section 5.8.5.2.2.1. The user-specified speed for a waypoint is the desired speed of movement between waypoints. The actual speed at which the platform travels is limited by the calculated speed of the vehicle. If the desired speed is set to zero, or if it is greater than the calculated speed, then the calculated speed of the vehicle is utilized. If the computed speed is greater than zero, then the platform is moved at the computed speed. If the computed speed is less than or equal to zero, then the platform remains stationary. The speed is used with the scenario interval to move the platform in the direction of the next waypoint at the calculated velocity.

Moving a platform includes calculating the new platform position and setting the platform's velocity vector in the direction toward the waypoint with a magnitude of the calculated speed. The magnitude of the difference vector between the platform's current position vector and the destination waypoint position vector is the distance the platform must be moved in order to reach the destination waypoint. A vector along the surface of the terrain is computed as a vector from the current platform location to a point at zero AGL located 10 meters in front of the platform in the direction of the next waypoint. The platform's velocity vector is oriented along a vector along the surface of the terrain in the direction of the destination waypoint. The magnitude of the velocity is then set to the magnitude of the speed. If the platform will not reach the next waypoint in the current scenario interval, its position is updated to the location of the distance covered in that scenario interval. The standard eadmath library algorithms described in Appendix

B10 are applied to the updated position. The function ECEFtoLatLon determines the latitude and longitude of the updated position. If the terrain following option or AGL is selected for the next waypoint then the platform position is adjusted so that it is at the next waypoint's altitude relative to the terrain. If the MSL option is selected without terrain following then the platform moves to the next waypoint at the altitude defined for that waypoint relative to zero MSL. The AGL or terrain following options in addition to non-zero waypoint altitudes are recommended for ground platforms in order to prevent terrain masking of sensors and communications devices. The LLAToECEF function is then used to compute the ECEF position of the ground platform.

A platform is moved until it reaches the next waypoint. If the next waypoint's off time is exceeded and is a non-zero value then the platform is deactivated. If the next waypoint's off time is zero then the platform will continue to move to that waypoint and start moving to the next waypoint immediately upon reaching it. The platform is considered to be at the next waypoint if the distance calculated from the speed and the scenario time interval is greater than or equal to the distance that the platform must travel to reach the new waypoint. The platform's position is set to the new waypoint's position once the platform arrives at the waypoint.

EADSIM can extract bridge data from VMAP databases. A bridge consists of a point or a series of points with connecting line segments. These points can be retrieved into the waypoint list of a ground platform using the Edit Platform Window of Scenario Generation. These points are labeled as a "Bridge" waypoint. The basic bridge data includes the identification number of the bridge, a series of waypoints, and additional details concerning the physical characteristics of the bridge.

Bridges are implemented as additional segments that overlay road segments with the same identification number as the road. The physical width of the bridge is set to 10 meters, and the length is determined from the distance between the endpoints of the bridge segments. The GUI displays the bridge segment as a different color and width, currently hardcoded to Red with a 2x line width, from the underlying road segment.

The elevation at coordinates of the bridge may not accurately reflect the true elevation of the bridge structure over the surrounding terrain. EADSIM implements two options to overcome this. The exact elevation of the bridge can be entered in the bridge waypoint, similar to entering an elevation for any other waypoint. A global option exists to allow the user to designate a constant offset elevation for all bridges. To use this option, the elevation of the bridge waypoint should be left at zero. The bridge offset is not applied in any terrain masking calculations or in the ALARM radar calculations.

5.8.5.1 Wet Soil Conditions

A global option determines the calculation of the vehicle speed under wet soil conditions. If this option is selected, then all platforms operating with the detailed mobility model will use the parameters defined in the "wet soil" portion of their detailed mobility definition table.

5.8.5.2 EADSIM Mobility Library

The EADSIM mobility library is used to determine the speed at which to move ground platforms and to compute ground platform routes using the data available from terrain and soil databases. The library is implemented as a DLL on Windows machines and as a shared object on Linux and SUN.

5.8.5.2.1 A* Search Algorithm

The A* search algorithm is used for the Add Road option to choose the least expensive road route based on distance. A description of the A* search algorithm can be found on the web at the Wikipedia – The Free Encyclopedia website, http://en.wikipedia.org/wiki/A%2A.

5.8.5.2.2 Mobility Library Functions

The interface with the detailed mobility library consists of two functions. The detailed mobility library can be replaced with a custom mobility model which conforms to the EADSIM mobility library interface.

5.8.5.2.2.1 CalcSpeedLL

```
DOUBLE CalcSpeedLL (ScenarioHeader *Scenario, DOUBLE dLat, DOUBLE dLon,
                    DOUBLE dXVec, DOUBLE dYVec, FLOAT DesiredSpeed,
                    DOUBLE CurrentSpeed, MobilityStruct *Mobility);
```

CalcSpeedLL calculates the actual speed a ground vehicle will move based on the vehicle characteristics, the soil type, and the terrain slope.

Table 5.8-1 MaxSpeedLL I/O Description

Variable	I/O	Data Type	Units	Description
Scenario	I	ScenarioHeader*	N/A	Pointer to scenario data
dLat	I	DOUBLE	Degrees	Latitude of current position
dLon	I	DOUBLE	Degrees	Longitude of current position
dXVec	I	DOUBLE	Meters	X component of ENU direction vector
dYVec	I	DOUBLE	Meters	Y component of ENU direction vector
DesiredSpeed	I	FLOAT	m/s	Speed at which to move this

				platform
CurrentSpeed	I	DOUBLE	m/s	Speed computed at previous scenario interval
Mobility	I	MobilityStruct*	N/A	Pointer to parameters describing this platform's detailed mobility model
(returns)	O	DOUBLE		Actual Speed

For this function, the input direction vector, $\vec{D}$, is required to be in a local (X, Y) coordinate system, oriented with X in the East direction and Y in the North direction. The current position $\vec{P}$, which is input in Latitude / Longitude (degrees) is, converted to Local map coordinates (X, Y) using the LLtoLocalX and LLtoLocalY terrain library functions described in section 5.8.5.3.6.

The desired speed is defined by the user for each waypoint as the waypoint speed. The actual speed is limited by the maximum speed at which this platform can move which is determined by the vehicle characteristics relative to the current soil type, slope, and acceleration limits. These vehicle characteristics are user-defined at the system level on the Detailed Mobility Definition window. The maximum speed which this platform can travel is degraded using the slope factor.

The elevation extracted from the DTED is used for calculating the slope. If a DTED is not available then the slope factor is zero. The slope is calculated in the following manner.

The direction vector $\vec{D}$ must be made into a unit vector.

$$\hat{D} = \frac{\vec{D}}{\left|\vec{D}\right|}$$

Next calculate a new position vector, $\vec{\mathrm{N}}$ 10 meters from the original position in the direction of travel is calculated as follows:

$$\vec{N} = \vec{P} + \hat{D}$$

The elevation of the original position (ElevP) and the new position ($ElevN$) is obtained then the slope is computed.

$$\mathrm{Slope} = (\mathrm{ElevN} - \mathrm{ElevP})/10$$

The slope factor is the ratio of the current slope in the direction of the next waypoint to the maximum slope.

$$S_{factor} = \frac{Slope}{MaxSlope}$$

where

S_{factor}	-	Slope factor used to degrade speed (none)
Slope	-	Current slope angle from level (degrees)
MaxSlope	-	Slope angle beyond which the platform speed is zero (degrees)

The speed is degraded using the slope factor.

$$S_{d\,max} = MaxSpeed - \left(MaxSpeed \cdot S_{factor}\right)$$

where

S_{dmax}	-	The degraded maximum speed at which to move the platform (m/s)
MaxSpeed	-	The maximum speed (m/s)
S_{factor}	-	Used to degrade speed (none)

If the slope factor is negative indicating a downhill slope then the degraded maximum speed at which to move the platform is equal to the maximum speed. If the degraded maximum speed which this platform can travel is greater than the desired speed then the platform is accelerated or decelerated to the desired speed. If the desired speed is zero or if it is greater than the degraded maximum speed then the platform is accelerated to the maximum speed. If the degraded maximum speed is less than or equal to zero then zero is returned for the actual speed. If the degraded maximum speed is equal to the current speed then the current speed is returned as the actual speed.

If the degraded maximum speed is greater than zero and is greater than the current speed then deceleration is applied. Deceleration is applied when the desired acceleration is negative causing the current speed to be reduced. The desired acceleration is computed as follows.

$$a_d = \frac{S_{des} - CurrentSpd}{\Delta t}$$

where

a_d	-	Desired acceleration (m/s^2)
S_d	-	The lower of desired speed and degraded maximum speed (m/s)
CurrentSpd	-	Speed computed at previous scenario interval (m/s)
Δt	-	Length of one scenario interval (s)

If the desired acceleration is negative then its magnitude is compared to the maximum deceleration. If the magnitude of the desired acceleration is less than the maximum deceleration then the platform can be decelerated to the desired speed in a single scenario interval and therefore the desired speed is returned as the actual speed. Otherwise the maximum deceleration is applied to the current speed to compute the actual speed.

$$S_{act} = CurrentSpd + MaxAccel \cdot \Delta t$$

where

S_{act}	-	Actual speed that is returned (m/s)
CurrentSpd	-	Speed computed at previous scenario interval (m/s)
MaxAccel	-	Maximum acceleration for this soil type (m/s^2)
Δt	-	Length of one scenario interval (s)

If the desired acceleration is positive then it is compared to the maximum acceleration. If the desired acceleration is greater than the maximum acceleration then the maximum acceleration is applied to the current speed to obtain the actual speed. If the desired acceleration is less than the maximum acceleration then the platform can be accelerated to the desired speed in a single scenario interval and therefore the desired speed is returned as the actual speed as described for the deceleration case.

5.8.5.2.2.2 ComputeRoadRouteLL

```
LONG ComputeRoadRouteLL (MobilityStruct *Mobility, ScenarioHeader *Scenario,
                         DOUBLE dLat1, DOUBLE dLon1, DOUBLE dLat2,
                         DOUBLE  Lon2, PointList** ppSubWaypoints);
```

ComputeRoadRouteLL calculates the most efficient path along roads from a starting location, (dLat1, dLon1), to an ending location (dLat2, dLon2). The algorithm searches for a continuous road path using only the road vertices. This algorithm assumes a road will exist within one kilometer of the starting and ending point and tests for this condition as a prerequisite for performing the search. If a suitable road path cannot be found then a failure flag is returned. This model is based on the A* search algorithm.

Table 5.8-2 ComputeRoadRouteLL I/O Description

Variable	I/O	Data Type	Units	Description
Scenario	I	ScenarioHeader*	N/A	Pointer to scenario data
Mobility	I	MobilityStruct*	N/A	Pointer to parameters describing this platform's detailed mobility model
dLat1	I	DOUBLE	Degrees	Start Latitude
dLon1	I	DOUBLE	Degrees	Start Longitude
dLat2	I	DOUBLE	Degrees	End Latitude
dLon2	I	DOUBLE	Degrees	End Longitude
ppSubWaypoints	I	PointList**	None	Pointer to intermediate waypoint structure
(returns)	O	LONG		0 = Success, >0 = Fail

The input latitude and longitudes are first converted to the local map coordinates. Let (X1, Y1) be the starting point and (X2, Y2) be the ending point of the search.

The nearest road point within a one kilometer radius of the starting point is found using the NearestRoadPoint terrain library function described in 5.8.5.3.1. Similarly the road point nearest the ending point is found. If road points do not exist within one kilometer of the starting and ending points, then the search is abandoned and the failure flag is returned.

Otherwise the path is chosen so that it enters the road at the road point nearest the starting point and leaves the road network at the road point nearest the ending point. The best cross-country path from the starting point to the entry of the road entry point is found using the using the mixed cross-country and road route algorithm. The best cross-country path from the road exit point to the end point in a similar manner. If these routes cannot be found, then the search is abandoned and the failure flag is returned. Finally, the road network from the road entry point to the road exit point is found using the road route algorithm.

5.8.5.2.2.3Road Route Algorithm

The road route algorithm searches for a continuous road path using only the road vertices, thereby reducing the computational load. First the nearest road point within a one kilometer radius of the starting point is found using the NearestRoadPoint terrain library function described in 5.8.5.3.1. Similarly the road point nearest the ending point is found. The route is computed such that the path will enter the road at the road point nearest the starting point and will leave the road network at the road point nearest the ending point. The best cross-country

path from the starting point to the road entry point using the cross-country and road route algorithm described in section 5.8.5.2.2.4. The best cross-country path from the road exit point to the end point is found in a similar fashion. If these routes cannot be found, then the route search is abandoned and a failure flag of one is returned. Finally, the road network from the road entry point to the road exit point is found. This method is similar to the cross-country and road route algorithm method, the only difference being that only the road segments are evaluated thus eliminating a large number of node iterations.

5.8.5.2.2.4Mixed Cross-Country and Road Route Algorithm

To start this process, the maximum possible speed is obtained from the detailed mobility model definition table for the Unknown soil type using a zero degree slope. Next the search routine is initialized by setting the start node to the current road location, setting the actual cost to zero and computing the remaining cost using the cost estimate algorithm. The cost estimate algorithm is used to calculate the weighted cost in units of time of the path from the start point (X1, Y1) to the end point (X2, Y2).

$$Cost = sqrt((X2 - X1) * (X2 - X1) + (Y2 - Y1) * Y2 - Y1)) / MaxSpeed$$

where

Cost	-	The amount of time estimate (s)
X1	-	X coordinate of the start point (m)
Y1	-	Y coordinate of the start point (m)
X2	-	X location of the end point (m)
Y2	-	Y location of the end point (m)

The initial cost estimate assumes a direct path from the start point (X1,Y1) to the end point (X2, Y2) using the maximum speed as calculated above.

5.8.5.2.2.5Node Generation

If the current position happens to be in the same place as the end position, then the route is complete. Otherwise, a list of potential nodes is created to which the vehicle can travel from the current position using the road node generation algorithm or the node generation algorithm as applicable. The road node generation algorithm is used if the route is restricted to roads only. The node generation algorithm is used if the route is not restricted to roads only.

The road node generation algorithm creates a list of possible paths a vehicle can take from its current position to adjacent road nodes on the way to its final destination. The current road and its vertices are input as arguments. The paths are restricted along roads. When using the road node generation algorithm cell boundaries are not considered when generating the node list. Instead, adjacent road vertices are added to the new node list. If the end node lies on the road

between the current point and an adjacent vertex, then the end node is also added to the new node list.

The node generation algorithm creates a list of possible paths a platform can take from its current position to adjacent nodes on the way to its final destination. The paths can be cross-country or along roads. The number of adjacent nodes returned depends on the current position relative to the grid cells. If the current position is inside a cell, then the 4 corner nodes of that cell are the adjacent nodes. If the current position is at a corner of a cell, then the corners of the 4 adjacent cells (8 corner nodes total) are the adjacent nodes. If the current position is on a horizontal side of a cell, then the corners of the cell above the point and the cell below the point are the adjacent nodes (6 corner nodes total). Similarly, if the current position is on a vertical side of a cell, then the corners of the cell to the right of the point and the cell to the left of the point are the adjacent cell (6 corner nodes total).

5.8.5.2.2.6Actual Cost

Using the list of possible paths, the actual cost for each node is systematically evaluated using the actual cost algorithm. The actual cost algorithm calculates the cost (time) of the path from the start point (X1, Y1) to the end point (X2, Y2).

If the start position $\vec{S}$ and the end position $\vec{E}$ are defined as follows.

$$\vec{S} = \begin{pmatrix} X1 \\ Y1 \\ 0 \end{pmatrix}$$

$$\vec{E} = \begin{pmatrix} X2 \\ Y2 \\ 0 \end{pmatrix}$$

The vector from the start to the destination is.

$$\vec{D} = \vec{E} - \vec{S}$$

The distance from the current point to the destination is magnitude of this vector.

$$Dis = \left|\vec{D}\right|$$

The Unit Vector along this path becomes.

$$\vec{U} = \vec{D} / \left|\vec{D}\right|$$

Since the actual ground conditions can change along this path, the actual ground conditions must be evaluated at short increments, Inc along this path. This increment is arbitrarily set at 10 meters, which corresponds to the minimum road width. This guarantees that no feature along the path will be missed.

If the distance remaining is greater than the 10 meter increment, then an intermediate point at this increment along the unit vector direction is evaluated. As these increments are evaluated, the current point is moved to the incremented position. The distance checked, T increases by the increment with each iteration, and the distance remaining decreases by the same amount.

Thus for any iteration of this evaluation, the current position, $\vec{C}$, and the next position, $\vec{N}$, are:

$$\vec{C} = \vec{S} + T * \vec{U}$$

$$\vec{N} = \vec{C} + Inc * \vec{U}$$

The elevations of the current and next point are retrieved from the database using the GetElevation function described in section 5.8.5.3.4. The slope at the current position is:

$$Slope = (Elev(\vec{N}) - Elev(\vec{C})) / Inc$$

The soil type for the next point is retrieved from the database using the GetSoil function described in 5.8.5.3.5.

Next the vehicle speed for this section, Speed, of the path is calculated using the CalcSpeedLL function. If this speed is less than or equal to zero, the destination is unreachable and the iteration stops. Otherwise, the cost and distance check are incremented based on the speed over this incremental section and the iteration continues.

$$Cost = Inc / Speed$$

$$T = T + Inc$$

If the distance remaining is less than the increment, Inc, then the final incremental cost is evaluated in the manner described above using the actual distance remaining instead of the increment. When the iteration loop terminates, the cost of the path is computed. The cost for the node is equal to the new cost of this node plus the accumulated costs of previous encountered to getting to this node. If this node has a lower cost than the previous node it is added to the node list. If the node results in an unreachable destination, or has a higher cost than the previous node, then the node is discarded. After all possible paths have been evaluated the resulting node list stored in ppSubWaypoints is the optimal path.

5.8.5.3 EADSIM Terrain Library

The EADSIM Terrain Library is used to retrieve the terrain features of the simulation area from the map database, initialize road and access road data, and perform coordinate transforms for the detailed mobility model. The library is implemented as a DLL on Windows and as a shared object on Linux and SUN. Only the routines used to describe the functionality of the mobility library are

documented in this section. It is not recommended nor is it beneficial for a user to attempt to create a different version of the terrain library.

5.8.5.3.1 NearestRoadPoint

LONG NearestRoadPoint(DOUBLE X1, DOUBLE Y1, DOUBLE Radius, DOUBLE *RoadX1, DOUBLE *RoadY1

This function determines if the closet road point (RoadX1, RoadY1) that exists within specified radius (Radius) of the input point (X1, Y1). It returns the closest road point to the starting point if the conditions are met.

Table 5.8-3 ConnectionCrossesMES I/O Description

Variable	I/O	Data Type	Units	Description
X1	I	DOUBLE	Meters	Starting point X coordinate
Y1	I	DOUBLE	Meters	Starting point Y coordinate
Radius	I	DOUBLE	Meters	Search radius from start pt.
RoadX1	O	DOUBLE	Meters	Closest Road point X coordinate
RoadY1	O	DOUBLE	Meters	Closest Road point Y coordinate
(returns)	O	LONG	None	0-point found, >0-no points

Starting at point (X1, Y1), identify the range of grid cells within the specified distance of the point. The boundaries are:

$$MinXGrid = Grid(X1) - Radius / GridXSpacing$$
$$MaxXGrid = Grid(X1) + Radius / GridXSpacing$$
$$MinYGrid = Grid(Y1) - Radius / GridYSpacing$$
$$MaxYGrid = Grid(Y1) + Radius / GridYSpacing$$

Where Grid( ) returns the grid cell associated the the X or Y coordinate, and GridXSpacing is the width of the grid cell and GridYSpacing is the height of the grid cell.

This forms the boundary of cells to search for road segments. For each cell, test for the existence of a road segment using the GridHasRoads function as described in section 5.8.5.3.2. If a road exists with the grid cell, then find the shortest distance from that road segment to the starting point (X1, Y1) using the DistanceToSegmentPoint function as described in section 5.8.5.3.3.

5.8.5.3.2 GridHasRoads

LONG GridHasRoads (LONG GridID);

This function determines if a road exists within the specified grid cell.

Table 5.8-4 GridHasRoads I/O Description

Variable	I/O	Data Type	Units	Description
GridID	I	LONG	None	Grid ID Number
(returns)	O	LONG	None	0-no roads, 1-has roads

5.8.5.3.3 DistanceToSegmentPoint

DOUBLE DistanceToSegmentPoint(DOUBLE X1, DOUBLE Y1, DOUBLE X2, DOUBLE Y2, DOUBLE X3, DOUBLE Y3);

This function determines if the closest distance between the point (X3, Y3) and the line specified by (X1, Y1) to (X2,Y2).

Table 5.7-1 ConnectionCrossesMES I/O Description

Variable	I/O	Data Type	Units	Description
X1	I	DOUBLE	Meters	Line start X coordinate
Y1	I	DOUBLE	Meters	Line start Y coordinate
X2	I	DOUBLE	Meters	Line end X coordinate
Y2	I	DOUBLE	Meters	Line end Y coordinate
X3	I	DOUBLE	Meters	point X coordinate
Y3	I	DOUBLE	Meters	point Y coordinate
(returns)	O	DOUBLE	Meters	Distance from point (X3, Y3) to line formed by (X1,Y1) and (X2, Y2)

Let (X3, Y3) represent a point and let a line exist from (X1,Y1) to (X2, Y2).

If (X1, Y1) and (X2, Y2) are in the same position, then the line is actually a point and the distance, Dist, is calculated by:

$$Dist = sqrt((X3 - X1) * (X3 - X1) + (Y3 - Y1) * (Y3 - Y1))$$

Otherwise define the following intermediate terms:

$$Numer = (X3 - X1) * (X2 - X1) + (Y3 - Y1) * (Y2 - Y1)$$

$$Denom = (X2 - X1) * (X2 - X1) + (Y2 - Y1) * (Y2 - Y1)$$

$$Ratio = Numer / Denom$$

If the ratio lies between (0,1) inclusive, the shortest distance, Dist, lies somewhere between the two endpoints of the line and the distance is calculated by:

$$Dist = fabs(((Y1 - Y3) * (X2 - X1) - (X1 - X3) * (Y2 - Y1)) / Denom) * sqrt(Denom)$$

Otherwise, the shortest distance is at one of the endpoints of the line. Calculate the distance from the point to each line endpoint and then choose the shortest distance. The squares of these distances are calculated as follows:

$$Dist1Sqd = (X3 - X1) * (X3 - X1) + (Y3 - Y1) * (Y3 - Y1)$$

$$Dist2Sqd = (X3 - X2) * (X3 - X2) + (Y3 - Y2) * (Y3 - Y2)$$

If ($Dist1Sqd < Dist2Sqd$)

Then $Dist = sqrt(Dist1Sqd)$

Otherwise $Dist = sqrt(Dist2Sqd)$

5.8.5.3.4 GetElevation

DOUBLE GetElevation(DOUBLE X, DOUBLE Y);

The elevation for the given position with local coordinate (X, Y) is calculated using the grid elevation data. Let the length / width of the grid cell be represented by GridSpacing. A grid of points and their corresponding elevation is defined in the map database.

Table 5.8-5 GetElevation I/O Description

Variable	I/O	Data Type	Units	Description
X	I	DOUBLE	Meters	Local X coordinate
Y	I	DOUBLE	Meters	Local Y coordinate
(returns)	O	DOUBLE	Meters	Elevation

If the local coordinate lies outside the boundaries of the map, an elevation of 0.0 is returned.

If the point lies within the map boundaries, the coordinates (X,Y) are first converted to the grid indices (IX, IY).

$$IX = (long)(X / GridSpacing)$$

$$IY = (long)(Y / GridSpacing)$$

Next the location within the cell (MX, MY) is computed

$$MX = X \bmod GridSpacing$$

$$MY = Y \bmod GridSpacing$$

A test is made on the local coordinate to determine if the point coincides with a grid point, i.e. $MX = MY = 0$. If the coordinate coincides with the grid point, then the elevation of that grid point is returned.

In the case that the local coordinate lies within the map area and does not coincide with a grid location, then an interpolation is done using the point and the adjacent grid elevations. First the elevations of all four corners of the grid cell (ElevBL, ElevBR, ElevTL, ElevTR) are retrieved from the database. A weighting factor is computed for the relative location of MX and MY within the cell.

$$WX = MX / GridSpacing$$

$$WY = MY / GridSpacing$$

The X weighting factor (WX) is used to first interpolate the elevations of the point with respect to the top of the cell (ElevTop) and to the bottom of the cell (ElevBot)

$$ElevTop = (ElevTL * (1 - WX)) + (ElevTR * WX)$$

$$ElevBot = (ElevBL * (1 - WX)) + (ElevBR * WX)$$

Finally, the Y weighting factor (WY) is used to interpolate the elevation of point (Elev) with respect to the elevations along the top and bottom of the cell.

$$Elev = ElevBot * (1 - WY) + ElevTop * WY$$

5.8.5.3.5 GetSoil

USHORT GetSoil (DOUBLE X, DOUBLE Y);

The soil type for the given local coordinate (X, Y) is retrieved from the database. Let the length / width of the grid cell be represented by GridSpacing. A grid of points and their corresponding soil type is defined in the map database. If the coordinate lies outside the map boundaries, a soil type of 0 is returned. Otherwise the soil type of the nearest grid point is returned.

Table 5.8-6 GetSoil I/O Description

Variable	I/O	Data Type	Units	Description
X	I	DOUBLE	Meters	Local X coordinate
Y	I	DOUBLE	Meters	Local Y coordinate
(returns)	O	USHORT	None	Soil Type

The nearest grid point is computed as follows:

Let dX, dY represent point being tested (double) and let $iInc$ represent the nominal grid dimension (long).

The nearest node iX, iY is

$$iX = (dX + iInc/2)/iInc$$

$$iY = (dY + iInc/2)/iInc$$

VMAP road databases return either on road or unknown for the soil type.

5.8.5.3.6 Coordinate Transforms

The functions used for conversion between the local gridded map coordinate system and the latitude longitude coordinate system are listed below. The map is a flat earth 2-dimension (X, Y) coordinate system with the X direction toward the East and the Y direction to the North. The origin of the map is the Southwest (lower left) corner of the map. The coordinates are expressed in the meters from this origin. The map is laid out in gridded sections with the distance per grid cell hard coded as 125 meters.

5.8.5.3.6.1LLtoLocal

```
DOUBLE LLtoLocal (ScenarioHeader *Scenario, DOUBLE dLat, DOUBLE dLon,
                  DOUBLE *XCoord, DOUBLE *YCoord);
```

LLtoLocalX converts the Lat/Lon position to local map coordinates. Let the origin of the map correspond to the southwest corner of the map (MapSouthLat, MapWestLon).

Table 5.8-7 LLtoLocalX I/O Description

Variable	I/O	Data Type	Units	Description
Scenario	I	ScenarioHeader*	N/A	Pointer to scenario data
Lat	I	DOUBLE	Degrees	Latitude of current position
Lon	I	DOUBLE	Degrees	Longitude of current position
XCoord	O	DOUBLE	Meters	Local X Coordinate
YCoord	O	DOUBLE	Meters	Local Y Coordinate

First calculate the latitude offset of the position of the object to the origin of the map.

$$Offset = Lat - MapSouthLat$$

Adjust the offset to lie within -180 to 180 degrees.

$$Offset = f\ mod(Offset,360)$$
$$if(Offset > 180)$$
$$then(Offset = Offset - 360)$$

For the spherical earth model, the X coordinate is

$$X = Cos(DEGTORAD(Lat)) * ECircumference * Offset / 360$$

where

ECircumference is the circumference of the earth, 40031555.9 meters

5.8.5.3.6.2LocalXYtoLat

DOUBLE LocalXYtoLat (ScenarioHeader *Scenario, DOUBLE dX, DOUBLE dY);

The local map coordinate position (X, Y) is converted to the Latitude. Let the origin of the local coordinate system correspond to the Southwest corner of the map (MapSouthLat, MapWestLon).

Table 5.8-8 LocalXYtoLat I/O Description

Variable	I/O	Data Type	Units	Description
Scenario	I	ScenarioHeader*	N/A	Pointer to scenario data
dX	I	DOUBLE	Meters	Local X coordinate
dY	I	DOUBLE	Meters	Local Y coordinate
(returns)	O	DOUBLE	Degrees	Latitude

First calculate the factor for converting the meters to degrees of latitude.

$$LatFactor = 360 / ECircumference$$

where

ECircumference is the circumference of the earth, 40031555.9 meters.

Next apply this factor to the Y component of the coordinate.

$$Lat = (Y * LatFactor) + MapSouthLat$$

5.8.5.3.6.3LocalXYtoLon

DOUBLE LocalXYtoLon(ScenarioHeader *Scenario, DOUBLE dX, DOUBLE dY);

The local map coordinate of the position is converted to ECEF Longitude. Let the origin of the map correspond to the Southwest corner of the map (MapSouthLat, MapWestLon). The Longitude value depends on the earth model designated in the simulation.

Table 5.8-9 LocalXYtoLon I/O Description

Variable	I/O	Data Type	Units	Description
Scenario	I	ScenarioHeader*	N/A	Pointer to scenario data
dX	I	DOUBLE	Meters	Local X coordinate
dY	I	DOUBLE	Meters	Local Y coordinate
(returns)	O	DOUBLE	Degrees	Longitude

First calculate the factor for converting the meters to degrees of longitude.

$$LonFactor = 360 / (ECircumference * \cos(MapSouthLat))$$

where

ECircumference is the circumference of the earth, 40031555.9 meters.

Next apply this factor to the X component of the coordinate.

$$Lon = (X * LonFactor) + MapWestLon$$

5.9 SATELLITE FLIGHT MODELING

5.9.1 Satellite Flight Modeling

Satellites, whether natural or artificial, tend to follow an elliptical path around the planet to which they are bounded. EADSIM provides two models for satellite propagation. The first model uses the Tarpre2 propagation method to simulate the elliptical satellite flight path. This model is used to propagate satellites that are deployed in EADSIM using the Initial ECI State option. It uses the known current satellite state vector and a time delta to compute the new state vector of the satellite along its path. This new state vector then becomes the current known state vector. Tarpre2 allows an object to be propagated to any part of its elliptical path in a single step. The delta time may be entered as positive or negative, allowing the satellite to be propagated forward or backward. The current state vector is passed into Tarpre2, where it is updated appropriately for the given time delta.

The second model is the higher fidelity Simplified General Perturbations Satellite Orbit Model 4 (SGP4/SDP4). This model is used to propagate satellites that are deployed in EADSIM using the Two Line Element (TLE) option. The TLE format provides ephemeral data valid for the date/time specified in the TLE. The SGP4/SDP4 model uses the given ephemeral data to first compute the satellite's initial state vector. The SGP4/SDP4 model then allows the satellite's state vector to be propagated/updated appropriately for a given time delta. The ephemeral data may either be entered by hand directly via Scenario Generation or be imported from a file in standard TLE format.

5.9.2 Tarpre2 Methodology

Tarpre2 is capable of integrating orbital trajectories which are elliptic, hyperbolic, or parabolic without the need for the user to define the trajectory type. Within Tarpre2, the proper logic for a given trajectory is selected based on the object's velocity compared to the escape velocity for the object's altitude. If the object's velocity is less than the escape velocity, the object will be propagated along an elliptical path. Although Tarpre2 can handle any of the three orbital trajectories, only the elliptical propagation method will be discussed because EADSIM assumes satellite flight paths are elliptical.

As previously mentioned, the Tarpre2 module integrates a ballistic drag-free trajectory over a desired interval of time (dt). This is done using Keplerian equations of motion applied to the initial state of the object. All the equations, algorithms, and explanations can be found in *Methods of Orbit Determination* by Escobal or in *Fundamentals of Astrodynamics* by Bate, Mueller, and White. To determine the new position of the satellite, its eccentric anomaly must be found through an iterative process (see ***Figure 5.9-1***).

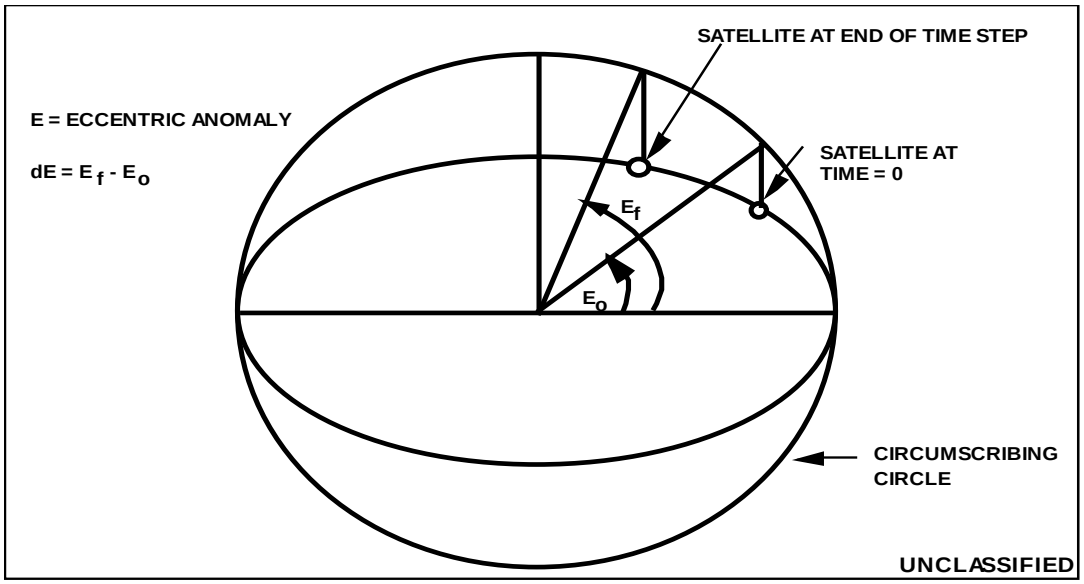

Figure 5.9-1 Eccentric Anomaly Definition

Tarpre2 requires the input of a position vector, velocity vector, and a delta time of flight. The position vector is used to calculate the magnitude of the satellite position at t = 0:

$$R = \left|\vec{R}_0\right|$$

where

R - magnitude of satellite position vector at t = 0 (m)

$\vec{R}_0$ - position vector of object from Earth center at t = 0 (m).

The velocity vector of the satellite is used along with the position vector to calculate the range rate from the Earth's center:

$$\dot{R} = \vec{R}_0 \bullet \vec{V}_0$$

where

$\dot{R}$ - range rate from Earth center at t=0 (m^2/sec)

$\vec{R}_0$ - position vector of object from Earth center at t = 0 (m)

$\vec{V}_0$ - satellite velocity vector at t = 0 (m/sec).

The velocity differential, α, is then calculated by:

$$= V_{mag}^2 - \frac{2 \times G_{mu}}{R}$$

where

α - difference between the square of the object's velocity and the square of the escape velocity for the object at that height (m^2/sec^2)

V_{mag} - magnitude of velocity at t=0, (m/sec)

G_{mu} - universal gravitational constant (m^3/sec^2)

R - magnitude of satellite position vector at t = 0 (m).

With R, $\dot{R}$, and the delta time, φ can be calculated:

$$\varphi = \left[\left(\frac{dt}{R}\right) - \left(\frac{\dot{R} \times dt^2}{2 \times R^3}\right)\right]$$

where

φ - amount of time given a positional change (sec/m)
dt - delta flight time, t_f - t_0 (sec)
R - magnitude of satellite position vector at t = 0 (m)
$\dot{R}$ - range rate from Earth center at t=0, (m^2/sec).

With the value for φ initialized, the iteration process can be started to find the new position and velocity of the satellite. The iteration process loops until the desired interval (dt) and the computed interval (dt_i) differ by less than the desired accuracy, currently set at 1.0 e^{-7}. Thus, the computed interval for each iteration is found by:

$$dt_i = (R \times S1) + (\dot{R} \times S2) + (G_{mu} \times S3)$$

where

$$S1 = \frac{\sin(\varphi \times \sqrt{\alpha})}{\sqrt{\alpha}}$$

$$S2 = \frac{(S0 - 1.0)}{\alpha}$$

$$S3 = \frac{(S1 - \varphi}{\alpha}$$

α - difference between the square of the object's velocity and the escape velocity for the object at that height (m^2/s^2)
φ - amount of time given a positional change (sec/m)
R - magnitude of satellite position vector at t = 0 (m)
$\dot{R}$ - range rate from Earth center at t=0, (m^2/sec)
G_{mu} - universal gravitational constant (m^3/sec^2).

A new range for each iteration is also calculated:

$$R_i = (R \times S0) + (\dot{R} \times S1) + (G_{mu} \times S2)$$

where

$$S0 = \cos(\varphi \times \sqrt{\alpha})$$

$$S1 = \frac{\sin(\varphi \times \sqrt{\alpha})}{\sqrt{\alpha}}$$

$$S2 = \frac{(S0 - 1.0)}{\alpha}$$

α - difference between the square of the object's velocity and the escape velocity for the object at that height (m^2/sec^2)
φ - amount of time given a positional change (s/m)

R - magnitude of satellite position vector at t = 0 (m)
$\dot{R}$ - range rate from Earth center at t=0, (m^2/sec^2)
G_{mu} - universal gravitational constant (m^3/sec^2).

The variable φ is then updated by:

$$\varphi = \varphi + \frac{(dt - dt_i)}{R_i}$$

where

dt - delta flight time, t_f - t_0 (s)
dt_i - iteration interval (s)
R_i - iteration range (m).

If the desired accuracy for (dt - dt_i) is reached, the iteration process is complete. If not, the loop is repeated until this accuracy is reached.

When the desired accuracy has been reached and the initial state vector and final range have been calculated, the new state vector is given by the closed form f and g series:

$$\vec{R} = \left(f \times \vec{R}_0\right) + \left(g \times \vec{V}_0\right)$$
$$\vec{V} = \left(\dot{f} \times \vec{R}_0\right) + \left(\dot{g} \times \vec{V}_0\right)$$

where

$\vec{R}$ - position of satellite at delta flight time (m)
$\vec{V}$ - velocity of satellite at delta flight time (m/sec)

$$f = 1.0 - \left(G_{mu} \times \frac{S2}{R}\right)$$
$$g = dt - \left(G_{mu} \times S3\right)$$
$$\dot{f} = \frac{(-G_{mu} \times S1)}{R_f \times R)}$$
$$\dot{g} = 1.0 - \left(G_{mu} \times \frac{S2}{R_f}\right)$$

$\vec{R}_0$ - range vector of object from Earth center at t = 0 (m)
$\vec{V}_0$ - satellite velocity vector at t = 0 (m/sec).

5.9.3 TLE Methodology

Satellites that are deployed using the Two Line Element (TLE) option are propagated using the SGP4/SDP4 propagation model.

5.9.4 Two Line Element (TLE) Format

Keplerian Elements in the TLE format may be downloaded from the U.S. Government Space-Track website: http://www.space-track.org. A basic description of the Two Line Element Format can be found on the U.S. Government Space-Track website: http://www.space-track.org/tle_format.html

Example TLE format: (Lines 0, 1 and 2 respectively)

```
Sat Name
1 #####U #####U   #####.########  .########  #####-#  #####-# #  ####
2 #####  ##.#### ###.#### ######## ###.#### ###.#### ##.###############
12345678901234567890123456789012345678901234567890123456789012345678901234567890
         1         2         3         4         5         6         7
```

Line 0

Columns	Example	Description
1-24	VANGUARD 1	The common name for the object.

Line 1

Columns	Example	Description
1	1	Line Number
3-7	00005	Object Identification Number/Satellite Number
8	U	Elset Classification
10-17	99087A	International Designator - Last two digits of launch year, Launch number of the year, Piece of the launch
19-32	02323.47021933	Epoch date – year and days
34-43	.00010257	1st Derivative of the Mean Motion with respect to Time
45-52	00000-0	2nd Derivative of the Mean Motion with respect to Time (decimal point assumed)
54-61	27583-4	BSTAR Drag Term
63	0	Element Set Type
65-68	513	Element Number
69	5	Checksum

Line 2

Columns	Example	Description
1	2	Line Number
3-7	00005	Object Identification Number/Satellite Number
9-16	52.6437	Inclination (degrees)
18-25	337.7730	Right Ascension of Ascending Node (degrees)
27-33	0006973	Eccentricity (decimal point assumed)
35-42	113.2323	Argument of Perigee (degrees)
44-51	295.9333	Mean Anomaly (degrees)
53-63	14.20306856	Mean Motion (revolutions/day)
64-68	32332	Revolution Number at Epoch
69	3	Checksum

Figure 5.9-2 Two Line Element Format description

5.9.5 Processing the Two Line Element (TLE) format

EADSIM processes TLE files in either of the formats found at the Space-Track website: Two-Line Format (no object names – Lines 1 and 2 only) and Three-Line Format (includes object names – Lines 0, 1 and 2). A TLE file can be used to deploy a satellite from the Edit Satellite window. The user can browse a directory for a TLE file. Once a file is selected, a list of the satellite numbers from the TLE file will be shown in a selection window. The user can then choose which satellite's TLE information will be used for the satellite being deployed in EADSIM. Once a set of TLE data is selected, each value of Line 1 and Line 2 will be placed in its corresponding field on the TLE portion of the Edit Satellite window in EADSIM.

Satellites deployed using TLE data are propagated using the SGP4/SDP4 propagation model detailed in "SPACETRACK REPORT NO. 3 Models for Propagation of NORAD Element Sets" and revised in "Revisiting Spacetrack Report #3". These documents can be found at: http://celestrak.com/NORAD/documentation/spacetrk.pdf and http://celestrak.com/publications/AIAA/2006-6753/AIAA-2006-6753.pdf respectively. This propagation model uses a prediction method which is compatible with the way in which the TLE elements are generated; therefore, this model retains maximum prediction accuracy.

The propagation model processes the TLE information and computes the satellite's initial ECI position and velocity vectors along with the Greenwich Sidereal Time (GST) and Greenwich Sidereal Angle (GSA). These values are computed at the absolute date/time specified in the TLE. GST is the amount of time that has elapsed since the most recent crossing of the prime meridian and the vernal equinox. GSA is the angle associated with the GST and is the angle between the vernal equinox and the prime meridian at the date/time specified in the TLE. At the beginning of each Julian day, the GSA equals zero, i.e., the ECI and ECEF coordinate frames are aligned. The GSA represents the rotation angle between the prime meridian and the vernal equinox and reflects the amount of rotation that has occurred since a reference universal time epoch (e.g., 1950, 1970, 2000). Therefore, this GSA provides the angle delta between the ECI and ECEF coordinate frames.

The initial ECI state valid for the date/time specified in the TLE file is used to compute the satellite longitude, using the ECEFtoLL function. See Appendix B10 for a description of the ECEFtoLL function. Then the GSA is subtracted from the computed longitude. This yields the correct satellite longitude for the date/time specified in the TLE. This longitude value is displayed in the ghosted Longitude field of the satellite deployment window. The satellite's ghosted Time field is populated with the number of seconds between start of simulation date/time and the date/time specified in the TLE. Since the satellite's Time field is related to the Scenario Start Date/Time, Time must be computed upon reading of laydowns, upon USE of the Edit Scenario and Edit Satellite windows. The Scenario Start Date is

entered by the user on the Edit Scenario window, its format is YYYY:MM:DD. This date is combined with the Zulu time to determine the absolute start of simulation (YYYY:MM:DD:HH:MM:SS).

5.9.6 TLE Satellite Positioning

If the absolute start of scenario date/time is different from the date/time specified in the satellite's TLE, the SGP4/SDP4 propagation model is used to forward or backward propagate the initial ECI position and velocity vectors to establish the satellite's ECI state at start of scenario. Upon start of scenario, the SGP4/SDP4 propagation model is used to propagate the satellite ECI state forward as the scenario progresses. At the start of scenario and at all times throughout scenario execution, the satellite's corresponding ECEF state vectors are computed using the ECItoECEF function with a time input equal to Current Simulation Time - Time + GST. See Appendix B10 for a description of the ECItoECEF function.

5.9.7 Satellite Definitions

Satellites deployed using the TLE option are defined by the ephemeral data contained within the TLE, see Section 5.9.4. An initial position vector, velocity vector and longitude are computed for the date/time specified in the TLE. Satellites deployed using the Initial ECI State option are defined by a user input initial position vector, velocity vector, and a longitude. The position and velocity vectors form the state vector.

For satellites deployed using the TLE option, Time is the number of seconds between start of simulation date/time and the date/time specified in the TLE. For satellites deployed using the Initial ECI State option, Time is user input. In both cases, Time is the time relative to the simulation start time for which the satellite's initial state vector is valid.

For satellites deployed using the Initial ECI State option, GST referred to in ***Figure 5.9-3*** and ***Figure 5.9-4*** is equal to zero. The initial state vector of the satellite is propagated forward or backward to compute the satellite's state vector at start of simulation as shown in ***Figure 5.9-3***. Then as the simulation progresses, the satellite's current state vector is propagated forward as shown in ***Figure 5.9-4***.

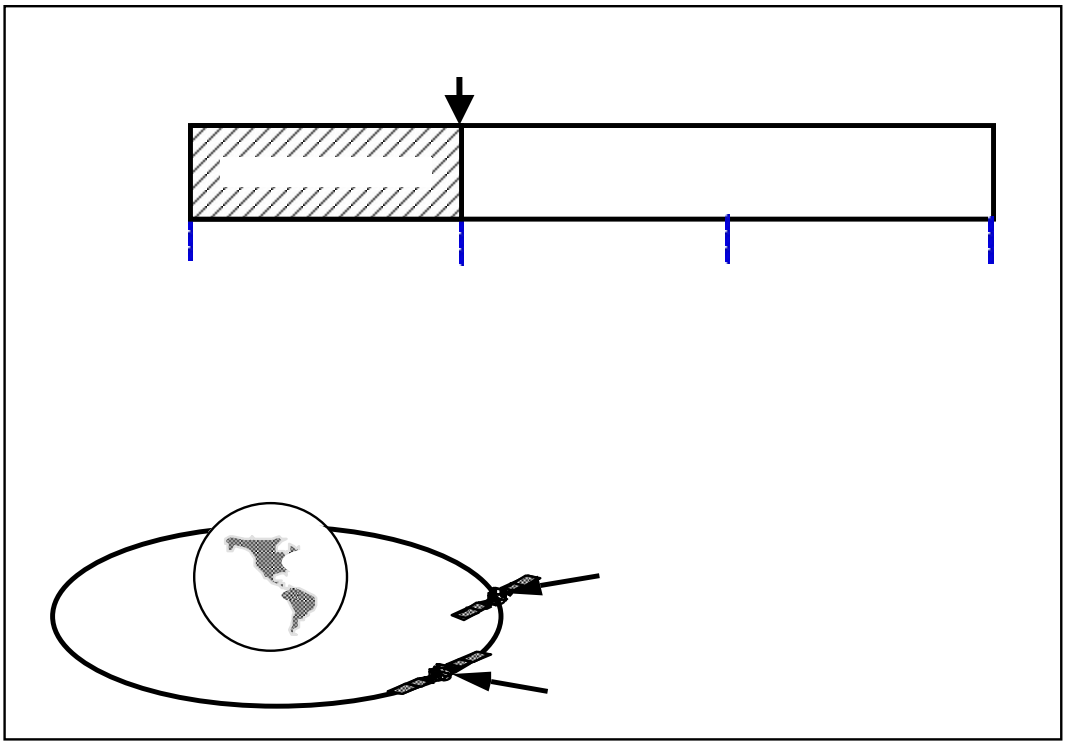

Figure 5.9-3 Establishing satellite state at start of simulation

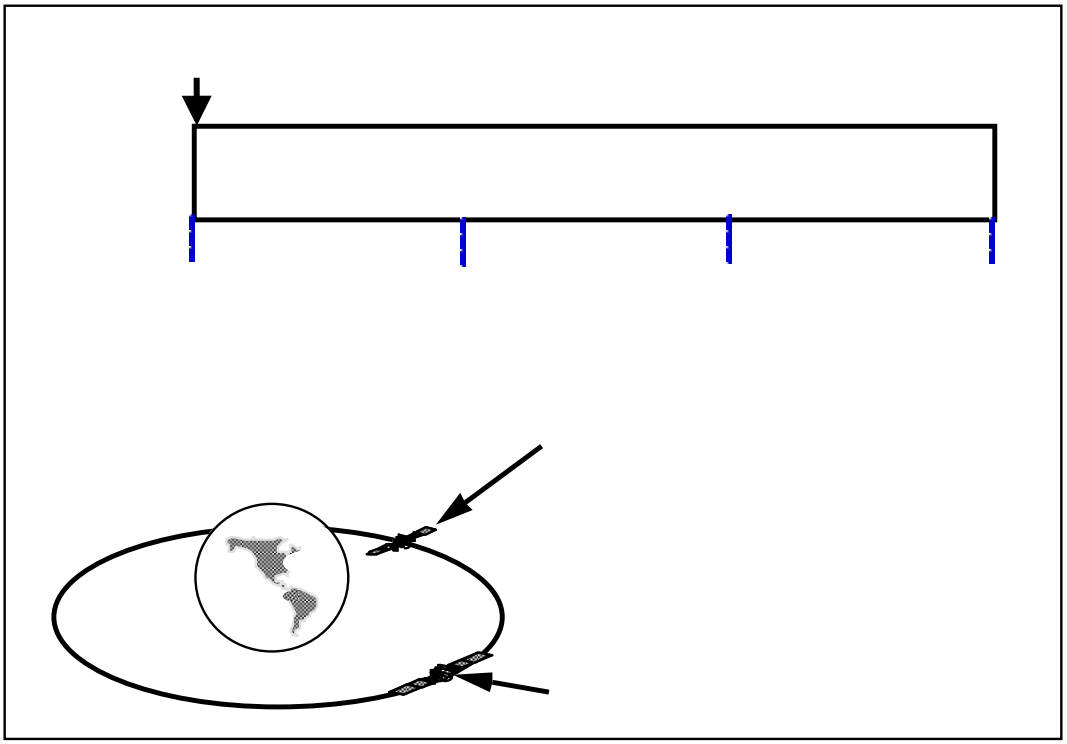

Figure 5.9-4 Satellite state update as simulation progresses

The satellites can be equipped with any combination of sensors, communications, or jamming devices. They can also be used as battle managers. Currently, the satellite cannot be used as a launching platform. The satellite can be detected by ground based or airborne sensors.

5.9.8 Satellite Model Coordinate System

A spherical rotating Earth is included with the body orientation through the Earth's center. The satellite state vector is computed in the ECI coordinate system, which has its origin at the Earth's center. The fundamental plane is along the equator with the positive x-axis pointing toward the vernal equinox and the positive z-axis pointing toward the north pole. See Subsection 6.4.2.1 for more information on the ECI coordinate frame.

The ECI coordinate system assumes that the x-axis aligns with the vernal equinox at time zero. In reality, the Earth has an actual daily rotation time of 23 hr 56 min and some seconds. Therefore, a satellite in orbit loses approximately 4 min of Earth rotation each day. The initial longitude input is used to account for this 4-min daily loss of rotation by the Earth. The initial longitude of the satellite is compared to the longitude calculated from the position vector of the satellite. If a difference exists, the satellite state vector is rotated by this amount to align the coordinate systems of the satellite and the model. Earth rotation is also taken into account for each time step, as illustrated in ***Figure 5.9-5***.

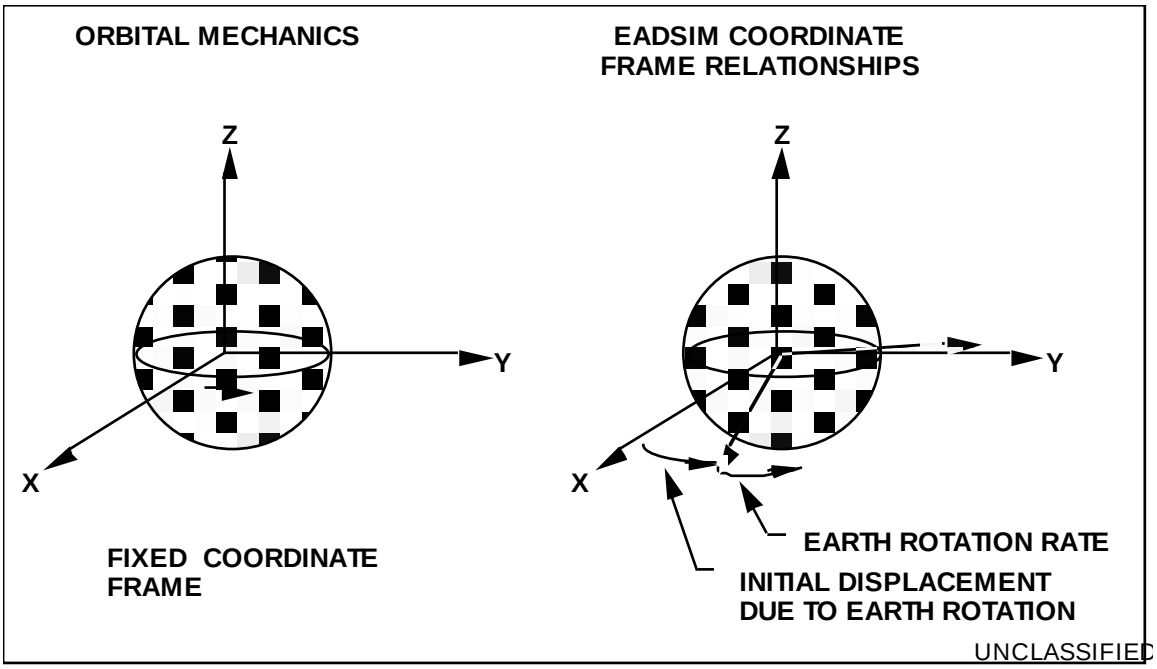

Figure 5.9-5 Satellite Coordinate Frames

5.10 INTERCEPTOR MISSILE FLIGHT

The Interceptor Flight Model uses common algorithms with the general missile flight modeling described in Section 5.6.

5.11 HANDOFF TO PRECISION GUIDED MUNITIONS (PGMS)

PGMs are modeled in EADSIM using captive platforms and complex weapons containing navigation elements. At the time of launch the navigation errors of the launching platform may be handed off to the PGM. If the host platform does not have a navigation element or if the Propagation option is not selected on the Edit Platform Processing Options window then the host platform's truth state vector is used as the initial perceived state vector of the weapon at the time of launch and the handoff errors are zero. The appropriate CEP, as determined by the methodology explained in Methodology Manual Section 4.4.5, is disabled when using navigation elements for navigation on weapons. For these cases, the error in the impact location will be the result of the guidance errors, modeled in a more explicit manner than the CEP.

5.11.1 Handoff Time Transfer Error

Perceived position and velocity information is assumed to be passed perfectly during handoff. This is not the case with the transfer of GPS time from the launch platform to the weapon. During handoff, a time delay serves to increase the error in GPS time that is passed to the weapon. This delay is known as the time transfer error and in reality is caused by data bus timing or latency errors. The time transfer error is applied to the loss period of the host platform at the time of handoff to get the initial loss period for the weapon. The time transfer error is defined on the GPS receiver for the weapon. The time transfer error on the GPS receiver of the host platform is not used. The time transfer error value can be computed by the user as follows.

$$t_{te} = \frac{\Delta t}{t_{osc_{accuracy}}}$$

Where

t_{te} - User input time transfer error (sec)
Δt - Bus latency (sec)
$t_{osc_{accuracy}}$ - Weapon oscillator accuracy (unitless)

Note that the time transfer error is the only user defined input value and that the above equation is only a reference for the user in computing this value. The loss period for the weapon is computed using the root sum square (RSS) of the time transfer error and the loss period of the host platform.

$$t_{lp} = \sqrt{t_{te}^2 + t_h^2}$$

Where

t_{lp}	-	Initial loss period of weapon (sec)
t_{te}	-	Time transfer error (sec)
t_h	-	Loss period of host platform at time of handoff (sec)

5.11.2 Hot vs. Cold Handoff

EADSIM models both hot and cold handoff at the time of PGM launch. This option is selected on the GPS Parameters (Receiver) window of the PGM that is to receive the handoff. When the hot handoff option is selected the overall GPS connectivity state of the PGM's GPS receiver is initialized to that of the launching platform at the time of launch. Therefore if the host platform has acquired and is tracking GPS in overall connectivity state 5 the PGM starts out tracking GPS in overall connectivity state 5 at the time of launch. The perceived state vector of the PGM is also initialized based on the perceived state vector of the launching platform, the handoff error, and on whether the Apply Error to Navigation option is set on the launching platform and the PGM. The initialization of perception for the PGM is described in section 5.11.3. If the host platform does not have GPS then the hot handoff option initializes the perceived state vector of the PGM but the PGM's receiver starts in the user-defined initial GPS connectivity state. If a PGM does not have GPS the hot handoff is always used.

The cold handoff option requires the PGM's GPS receiver to start in the user-defined initial overall GPS connectivity state. If the initial state is set to a not connected state then the PGM must acquire GPS connectivity at the time of launch. The perceived state vector is initialized by applying the user defined initial position and velocity errors to the true launch position when the cold handoff option is used. The cold handoff option is not valid for PGMs that do not have GPS. The cold handoff option in this case would best be represented using the simple CEP modeling without providing a navigation element to the released weapon.

5.11.3 PGM Handoff Error

The truth state vector of the PGM is initialized to the truth state vector of the launching platform at the time of launch. When a PGM is launched the handoff position and velocity errors are computed using the launching platform's GPS connectivity and navigation error parameters as described in section 5.4.1.2. These errors are logged for post processing and stored on the PGM. If the PGM acquires a GPS solution during fly out then the launching platform navigation handoff errors and the user defined initial cold handoff errors are erased.

If the Accumulate Error Only option is selected on an entity, then the perceived state vector is set equal to the truth state vector throughout its entire flight causing it to fly with perfect navigation. If the cold handoff option is selected

on a PGM the user defined initial position and velocity errors are stored at the time of launch and applied at the time of impact. For the hot handoff case, the handoff position and velocity errors are stored and then applied at the time of impact. The errored impact location is discussed in more detail in section 5.11.4.

If the Apply Error to Navigation option is selected for an entity then perception is computed as described in section 5.4.1.1. If the Apply Error to Navigation option is selected for a PGM then the perceived state vector is initialized using either the handoff errors or the user defined initial errors for cold handoff. The user defined initial errors for cold handoff are used when the cold handoff option is selected; otherwise the handoff errors are used. The errored impact location is a product of the PGM flying an errored flight path. However, the impact location must be adjusted as described in section 5.11.4. The perception, handoff, and flight calculations of a PGM with the Apply Error to Navigation option selected are independent of the Apply Error to Navigation option on the launching platform.

5.11.4 Errored Impact Location

If the Accumulate Error Only option is selected, the PGM flies directly to the true target location and then detonates at an errored impact location. If the Apply Error to Navigation option is selected the PGM flies using imperfect navigation towards the perceived targeted location using waypoint flight as described in section 5.6.1. The PGM schedules the intercept phase when it reaches its perceived last waypoint and detonates at the errored impact location.

If the PGM is using GPS at the time of impact then the position errors are computed as described in section 5.4.1.2.1.

If the PGM is using backup navigation at the time of impact, the navigation errors are computed at the time of impact and applied to the truth target location. The navigation errors include the navigation, initial, and handoff errors. The PGM error is summed with the initial error to obtain the total error. The initial error is the GPS error at the time when the entity started using backup navigation, the initial error generated from the user-defined initial error distributions for cold handoff for a PGM, or the handoff error for a PGM. The handoff errors are used if the PGM has not acquired GPS since launch and the hot handoff option is selected. If the cold handoff option is selected, the user-defined initial error distributions for cold handoff are used to generate the error with the same method used for generating GPS error. All handoff errors are set to zero if a PGM acquires GPS after handoff. Handoff errors are computed as described in section 5.11.3. If an entity goes from using GPS navigation to using backup navigation, then the GPS error at the time when the entity started using backup navigation is used as the initial error. The total position error is computed from the sum of the PGM position error and the initial position error in ENU coordinates.

$$\vec{P}_{err} = \vec{P}_{init} + \vec{V}_{init} \cdot t + \vec{P}_{entity}$$

where

$\vec{P}_{err}$ - Total position error

$\vec{P}_{init}$ - Initial position error from cold handoff, GPS, or handoff

$\vec{V}_{init}$ - Initial velocity error from cold handoff, GPS, or handoff

t - Time on backup navigation

$\vec{P}_{entity}$ - Entity backup navigation error

Additional components of error must also be considered when computing the errored impact position, whether the PGM is using GPS or backup navigation at the time of impact. The next component of error to consider is the vertical position error. Depending on direction of the error, this error will result in either a shortfall of the target or an overshoot of the target.

The vertical position error must be propagated onto the horizontal plane. First the amount of time until impact from the errored altitude is computed.

$$\frac{P_{errZ}}{V_{Zt}} = \Delta t$$

Where

P_{errZ} - Position error in up direction (m)

V_{Zt} - Truth velocity in up direction (m)

Δt - Time until impact (s)

The time until impact is then used to compute the horizontal distance in the east direction from the target the PGM lands due to the vertical position error.

$$P_{errX}(Z) = V_{Xt} \cdot \Delta t$$

Where

$P_{errX}(Z)$ - Position error in X or east direction due to vertical error (m)

V_{Xt} - True velocity in the X or east direction (m/s)

Δt - Time until impact (s)

The time until impact is then used to compute the horizontal distance in the north direction from the target the PGM lands due to the vertical position error.

$$P_{errY}(Z) = V_{Yt} \cdot \Delta t$$

Where

$P_{errY}(Z)$	-	Position error in Y or north direction due to vertical error (m)
V_{Yt}	-	True velocity in the X or north direction (m/s)
Δt	-	Time until impact (s)

The vertical error's contribution to the horizontal error is then applied to the horizontal error to get the total horizontal error. The Z or vertical component of the total horizontal error is zero.

$$\vec{P} = \vec{P}_{err} + \vec{P}_{err}(Z)$$

Where

$\vec{P}$	-	Total horizontal error in ENU coordinates (m)
$\vec{P}_{err}$	-	Horizontal position error (m)
$\vec{P}_{err}(Z)$	-	Vertical error contribution to horizontal error (m)

The position error is then rotated into ECEF coordinates.

$$\vec{P}_{ECEF} = \left[\vec{R}\right]^{T}\left[\vec{P}\right]$$

Where

$\vec{P}_{ECEF}$	-	Position error in ECEF coordinates
$\vec{R}$	-	Rotation matrix from ECEF to ENU computed from true targeted location
$\vec{P}_{ENU}$	-	Position error in ENU coordinates

The errored impact position is then computed using the total horizontal position error, the horizontal velocity error, and the amount of time using backup navigation. The Z or vertical component of the impact position error is zero.

$$\vec{P}_{impact} = \vec{P}_{t} + \vec{P}_{ECEF}$$

Where

$\vec{P}_{impact}$	-	Errored impact position in ECEF coordinates
$\vec{P}_{t}$	-	True targeted location in ECEF coordinates
$\vec{P}_{ECEF}$	-	Horizontal impact position error in ECEF coordinates

5.12 CLOUD MODELING

EADSIM has the capability to model a scripted description of a hazardous cloud. More detailed modeling is available via an HLA interface with PEGEM. Hazardous clouds are defined within EADSIM using the weapon event ruleset to allow the effects to be played through operation with an external cloud simulation, such as PEGEM.

5.12.1 Cloud Movement

The cloud state information is defined by a table of latitude, longitude, altitude, and speeds. This table defines the center of the cloud. Once activated, the cloud moves from its current position to the next position in the table. The initial location is calculated from the initial position specified in the table. The location is computed based on the latitude and longitude. The cloud is placed at the specified altitude above this location based on the terrain at this location, i.e. the specified altitude is an altitude Above Ground Level (AGL).

The cloud will begin moving towards the next specified location based on the speed identified for that location. The altitude rate necessary for the cloud to transition smoothly in altitude from one location to the next is computed. The position is then updated on each interval based on the input speed and a constant altitude rate change to go from one location to the next.

5.12.2 Cloud Extent

The cloud extent information is included in the same table defining the cloud location. One sigma lengths in x, y, and z in a coordinate system aligned with the velocity vector of the cloud define the cloud extent. The rate at which the sigma values are expanding is defined by the derivatives of x, y, and z in the same coordinate system. This information along with the defined total mass of the cloud is used to determine the concentration of the cloud out to two sigma lengths in x, y, and z. The sigma values along each axis expand as a function of time based on the following linear growth.

$$X_{i+1} = X_i + \dot{X}(T_{i+1} - T_i)$$
$$Y_{i+1} = Y_i + \dot{Y}(T_{i+1} - T_i)$$
$$Z_{i+1} = Z_i + \dot{Z}(T_{i+1} - T_i)$$

Where,

X_{i+1} = Current X axis one sigma length
X_i = Prior X axis one sigma length
$\dot{X}$ = X axis one sigma rate of growth
Y_{i+1} = Current Y axis one sigma length

Y_i = Prior Y axis one sigma length

$\dot{Y}$ = Y axis one sigma rate of growth

Z_{i+1} = Current Z axis one sigma length

Z_i = Prior Z axis one sigma length

$\dot{Z}$ = Z axis one sigma rate of growth

T_{i+1} = Current simulation time

T_i = Current simulation time – one interval