

XRLSim Model Specifications and User Interfaces

FY89 Enhancements

K. D. Young, E. Breitfeller, and J. P. Woodruff

**R Program
Weapon Concepts Group
Weapons Engineering Division**

December 1989



**Lawrence Livermore National Laboratory
University of California • Livermore, California • 94550**

MASTER

JMP

Table of Contents

Executive Summary.....	iv
Administrative Information.....	iv
Chapter One: XRLSim Model Specifications	
Weapon Platform Model Specifications.....	1-1
Platform Orientation.....	1-1
Platform Dynamics.....	1-1
Pointing Control.....	1-3
Thruster Dynamics and PWM Logic.....	1-3
Target Maneuver Detection	1-7
Chapter Two: XRLSim User Interfaces	
Vehicle	2-1
Platform (Thruster).....	2-2
Onboard_Track	2-7
Laser Lethality.....	2-7

List of Figures

Figure 1.	Inertial and vehicle coordination frames.	1-2
Figure 2.	Principal axis and LOS angles.	1-3
Figure 3.	The limiter function $y = Lim(x, xl)$	1-5
Figure 4.	The limiter function $y = N_H(x)$	1-5
Figure 5.	Pointing loop signal flow.....	1-6
Figure 6.	The deadband hysteresis function.	2-6

Executive Summary

The two chapters in this manual document the engineering development leading to modification of **XRLSim** — an Ada-based computer program developed to provide a realistic simulation of an x-ray laser weapon platform. Complete documentation of the FY88 effort to develop **XRLSim** was published in April, 1989, as UCID-21736: *XRLSIM Model Specifications and User Interfaces*, by L.C. Ng, D.T. Gavel, R.M. Shectman, P.L. Sholl, and J.P. Woodruff. The FY89 effort has been primarily to enhance the x-ray laser weapon-platform model fidelity. Chapter One of this manual details enhancements made to **XRLSim** model specifications during FY89. Chapter two provides the user with changes in user interfaces brought about by these enhancements. This chapter is offered as a series of deletions, replacements, and insertions to the original document to enable **XRLSim** users to implement enhancements developed during FY89.

Administrative Information

This effort was supported by R Program's Weapon Concepts Group at the Lawrence Livermore National Laboratory. The project leader of this study is Steve Sackett and the project engineer is K. David Young.

Chapter One

XRLSim Model Specifications

Weapon Platform Model Specifications

Platform Orientation

Four Euler parameters define the vehicle's orientation with respect to the inertial frame. Euler parameters, which are also referred to as quaternions and form a four vector given by $(\eta, \vec{q}_{3 \times 1})$, are used to describe the orientation of the vehicle relative to the earth. Given the initial orientation of the vehicle with respect to the earth, the Euler parameters propagate in time according to

$$\begin{pmatrix} \dot{\eta} \\ \dot{\vec{q}} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 0 & -\vec{\omega}^T \\ \vec{\omega} & -\vec{\omega}^\times \end{pmatrix} \begin{pmatrix} \eta \\ \vec{q} \end{pmatrix}, \quad \begin{bmatrix} \eta(t_0) \\ \vec{q}(t_0) \end{bmatrix} = \begin{pmatrix} \eta_0 \\ \vec{q}_0 \end{pmatrix} \quad (1)$$

where for any vector $\vec{a}_{3 \times 1} = (a_1, a_2, a_3)$

$$\vec{a}^\times = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix}, \quad (2)$$

and in this case,

$$\vec{\omega}^\times = \begin{pmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{pmatrix}, \quad (3)$$

The direction cosine matrix C_{vi} , which transforms coordinates from the inertial frame into the vehicle frame, is calculated directly from the Euler parameters as

$$C_{vi} = \begin{pmatrix} \eta^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 + \eta q_3) & 2(q_1q_3 - \eta q_2) \\ 2(q_1q_2 - \eta q_3) & \eta^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 + \eta q_1) \\ 2(q_1q_3 + \eta q_2) & 2(q_2q_3 - \eta q_1) & \eta^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix} \quad (4)$$

Platform Dynamics

The motion of the vehicle is described by three translational and three rotational degrees of freedom (DOF). The definition of the vehicle's body-axis system is shown in Fig. 1 for the case of vehicle alignment with the inertial frame. The rotational thrusters used to generate torques are mounted near the rear.

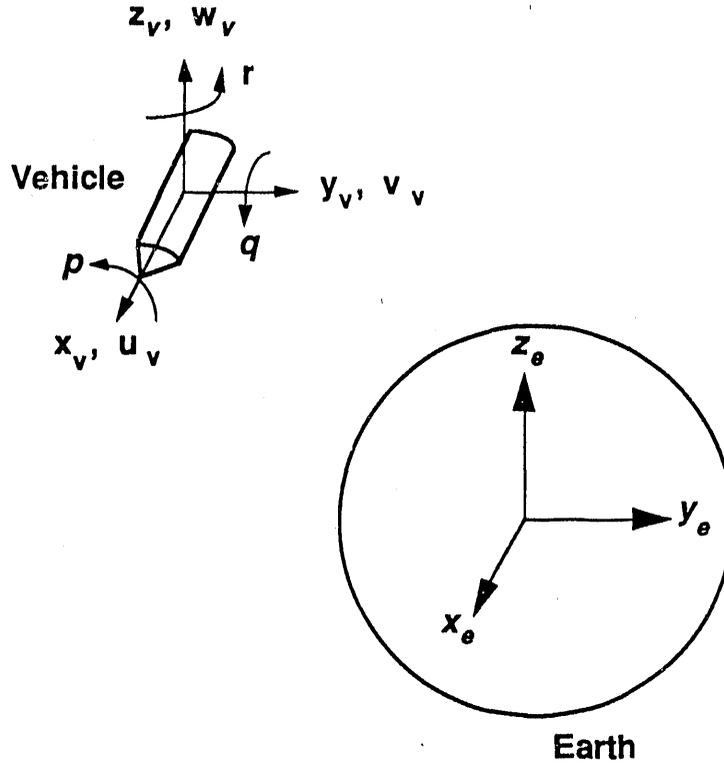


Figure 1. Inertial and vehicle coordination frames.

Principal axes are chosen as the vehicle's frame of reference. This axes set has the x-axis forward along the longitudinal axis, and the y-axis and z-axis that are chosen to form a right-handed coordinate system as shown in Fig. 1. The vehicle rolls, pitches, and yaws about its X_v , Y_v , and Z_v axes, respectively. The vehicle's rotational dynamics are described by:

$$I_{xx} \dot{p} - (I_{zz} - I_{yy})rq = f_p \ell_p \quad (5)$$

$$I_{yy} \dot{q} - (I_{xx} - I_{zz})rp = f_q \ell_q \quad (6)$$

$$I_{zz} \dot{r} - (I_{yy} - I_{xx})pq = f_r \ell_r \quad (7)$$

where ℓ_p , ℓ_q , and ℓ_r are the respective roll, pitch, and yaw moment arms, I_{xx} , I_{yy} , and I_{zz} are the principal moments of inertia about each axis, p , q , and r are the pitch, roll, and yaw rates, and f_p , f_q , and f_r are the roll, pitch, and yaw thruster forces.

Assuming a nonrotating spherical earth, the translational motion of the platform is expressed in the vehicle reference frame by,

$$m_v(\dot{u} - vr + wq) = -f_{gx} \quad (8)$$

$$m_v(\dot{v} + ur - wp) = -f_{gy} \quad (9)$$

$$m_v(\dot{w} - uq + vp) = -f_{gz} \quad (10)$$

where u , v , and w are the translational velocity along the vehicle's x , y , and z axes, the vehicle's total mass is m , and the gravitational force expressed in the vehicle frame is given by,

$$\vec{f}_g = (f_{gx}, f_{gy}, f_{gz}) = C_{ve} \frac{GM_e m_v}{|\vec{r}_v|^3} \vec{r}_v, \quad (11)$$

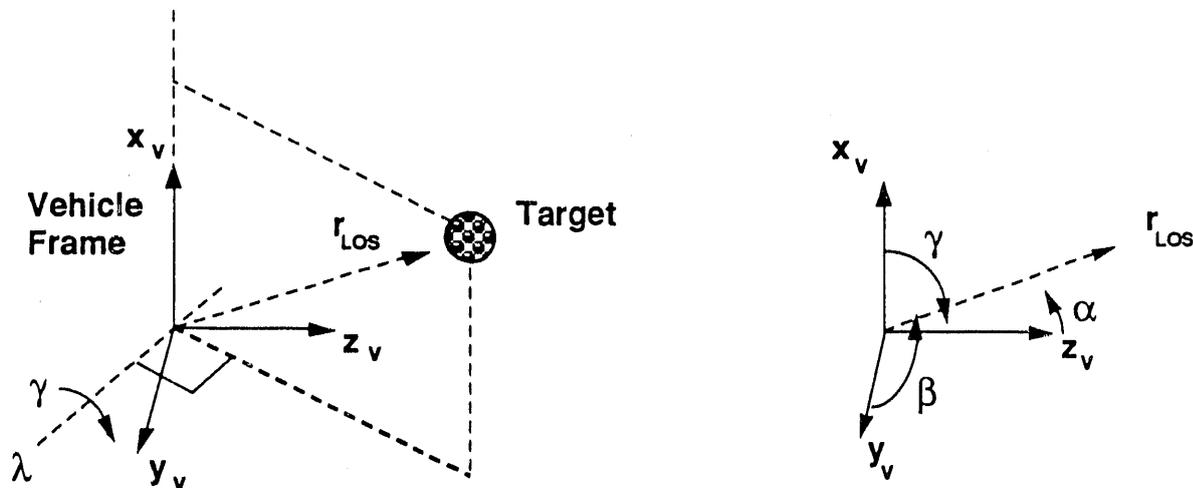
where \vec{r}_v is the inertial radial distance of the vehicle from the center of the earth, and GM_e is the gravitational constant.

Pointing Control

Pointing control is derived using Euler parameters. An axis of rotation λ and an angle of rotation γ are defined, which align the vehicle's longitudinal axis with its line of sight (LOS) of the target as shown in Fig. 2.

Given two coordinate systems C_1 and C_2 , which are unaligned by a principal axis of rotation $\vec{\lambda}$ and an angle of rotation γ , the Euler parameters of C_1 relative to C_2 are defined to be

$$\begin{aligned} \eta &= \cos\left(\frac{\gamma}{2}\right) \\ \vec{q} &= \sin\left(\frac{\gamma}{2}\right) \vec{\lambda} \end{aligned} \quad (12)$$



λ - axis of rotation
 γ - angle of rotation required
to align x_v with r_{LOS}

Figure 2. Principal axis and LOS angles.

From Fig. 2, it is seen that $\vec{\lambda}$ lies in the $Y_V - Z_V$ plane,

$$\vec{\lambda} = \left(0 - \frac{\cos \alpha}{\sin \gamma} \frac{\cos \beta}{\sin \gamma} \right)^T \quad (13)$$

where α , β , and γ are the angles between the LOS and the vehicle frame's axes, and where γ is the actual LOS angle error. Normalization of $\vec{\lambda}$ leads to the Euler parameters that describe the relative position of the vehicle to that of the target

$$\eta_{v \to t} = \cos \left(\frac{\gamma}{2} \right) \quad (14a)$$

$$\vec{q}_{v \to t} = - \begin{pmatrix} 0 \\ - \frac{\cos \alpha}{\sin \gamma} \\ \frac{\cos \beta}{\sin \gamma} \end{pmatrix} \sin \left(\frac{\gamma}{2} \right) \quad (14b)$$

These Euler parameters have the interpretation of vehicle pointing error in the vehicle frame. This error is necessarily expressed in the inertial frame for pointing control. Toward this end, two successive rotations about two different Euler axes are required: an Euler axis that transfers the vehicle from complete alignment with the inertial frame to its present position (i.e., the relative error from earth to vehicle), and an Euler axis that transfers the vehicle from its present position to its final desired position (i.e., the relative error from vehicle to LOS). The overall desired Euler parameters are, therefore,

$$\begin{pmatrix} \eta \\ \vec{q} \end{pmatrix}_{i \to t} = \begin{pmatrix} \eta \\ \vec{q} \end{pmatrix}_{i \to v} \otimes \begin{pmatrix} \eta \\ \vec{q} \end{pmatrix}_{v \to t} \quad (15)$$

where the notation $\vec{a} = \vec{b} \otimes \vec{c}$ represents the following vector operation:

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} b_0 & -b_1 & -b_2 & -b_3 \\ b_1 & b_0 & -b_3 & b_2 \\ b_2 & b_3 & b_0 & -b_1 \\ b_3 & -b_2 & b_1 & b_0 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} \quad (16)$$

The vehicle-attitude error can be expressed in the inertial frame as

$$\delta q = \eta_{i \to v} \vec{q}_{i \to t} - \eta_{i \to t} \vec{q}_{i \to v} - \vec{q}_{i \to v}^{\times} \vec{q}_{i \to t} \quad (17)$$

where the notation in (2) is used for $\vec{q}_{v \to t}^{\times}$, and ω_d is the LOS rate of the target relative to the vehicle. The pointing control law is

$$\begin{pmatrix} \ell_p f_p \\ \ell_q f_q \\ \ell_r f_r \end{pmatrix}_{cmd} = - \vec{\omega}^{\times} J \vec{\omega} - D(\vec{\omega} - \vec{\omega}_d) - K \delta q \quad (18)$$

where $J = \text{diag}(I_{xx}, I_{yy}, I_{zz})$, δq is the attitude error expressed in the inertial frame, and D , and K are diagonal matrices with positive elements, which are designed to meet pointing-response requirements.

Thruster Dynamics and PWM Logic

Second-order dynamic models are used for the thrusters. The state equations representing the thruster dynamics are given by

$$\begin{aligned} \dot{s}_1 &= \text{Lim}(s_2, s_{2l}) \\ \dot{s}_2 &= \omega_n^2 [N(f_c) - f] - 2\eta\omega_n s_2 \end{aligned} \quad (19)$$

where s_{2l} is a rate limit, f is the thruster force output, and s_{1l} is an optional force limit,

$$f = \text{Lim}(s_1, s_{1l}) \quad (20)$$

The functional N is the composition of the time delay function, N_{TD} , and the hysteresis function, N_H ,

$$N[y(t)] = N_{TD} \{N_H[y(t)]\} \quad (21a)$$

$$N_{TD}[y(t_k)] = y[t_k + T_d] \quad (21b)$$

The limiter function, Lim , and the hysteresis function N_H , are defined in Figs. 3 and 4, respectively.

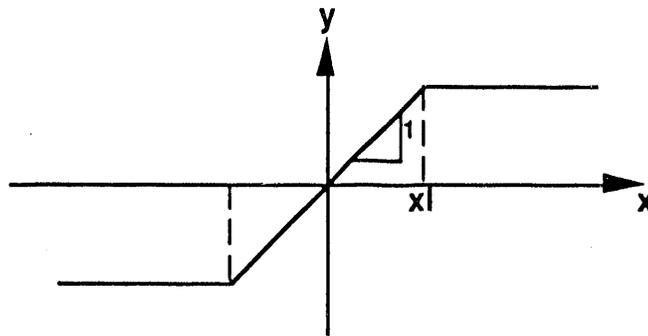


Figure 3. The limiter function $y = \text{Lim}(x, xl)$.

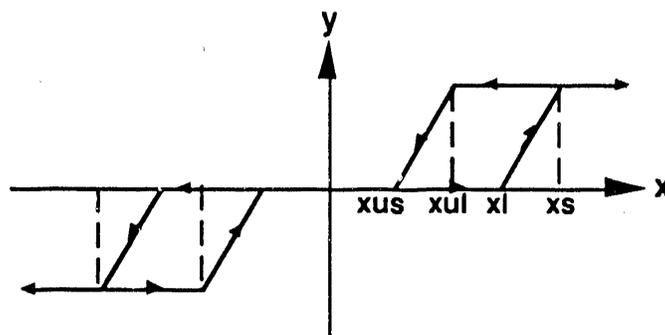


Figure 4. The hysteresis function $y = N_H(x)$.

The coupling of the pointing control law and the thruster dynamics is through a pulse-width-modulation (PWM) logic, which can be symbolically denoted by

$$f_{c,i} = PWM(f_{i,cmd}), \quad i = p, q, r \quad (22)$$

which is implemented according to the following logic: A continuous time signal $c(t)$ is pulse-width modulated by sampling at discrete instances, comparing the samples to two triangular waveforms, one positive $p(t)$ and one negative $n(t)$, and then setting on/off logic variables accordingly. The PWM output is computed as

$$\begin{aligned} c_{pwm} &= c_{on}(p_{flag} - n_{flag}) , \\ n_{flag} &= 1, \quad c(t) < n(t) \\ n_{flag} &= 0, \quad c(t) \geq n(t) \\ p_{flag} &= 1, \quad c(t) > p(t) \\ p_{flag} &= 0, \quad c(t) \leq p(t) \end{aligned} \quad (23)$$

where c_{on} is the full pulse value.

For each of the vehicle body axes, a pointing control loop is defined as shown in Fig. 5. The input to the PWM logic is the pointing control command in (18), the input to the thruster model is the output of the the PWM logic which is denoted as f_c in (19), and the thruster model output s_1 is the vehicle body axis force applied.

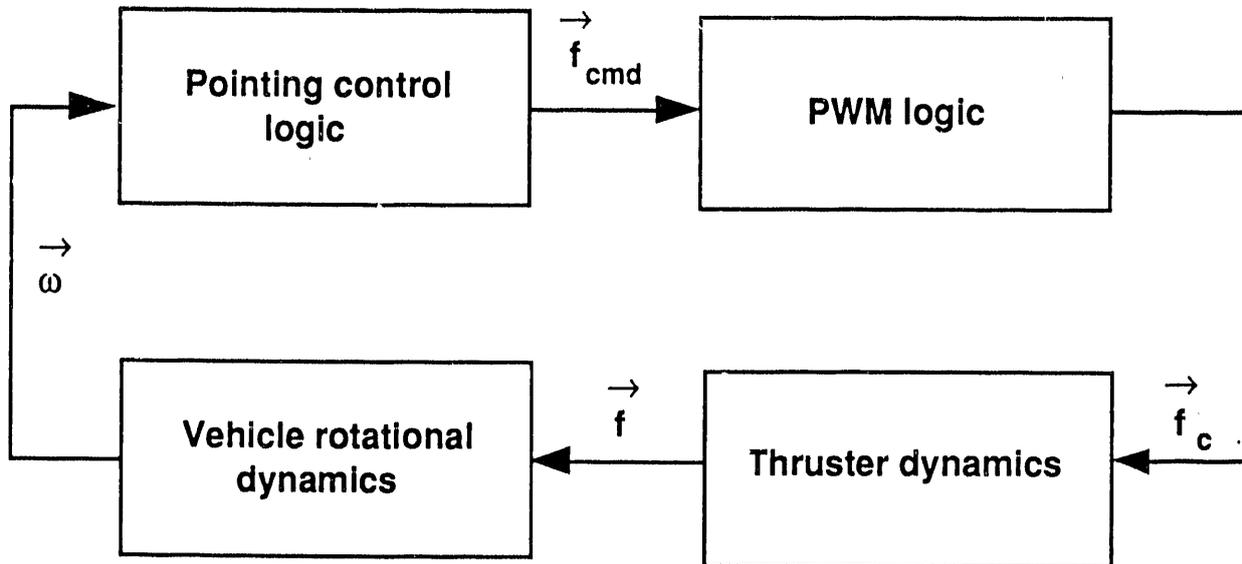


Figure 5. Pointing loop signal flow.

Target Maneuver Detection

A target maneuver detection algorithm has been implemented. It compares the measured elevation and azimuth angles with the predicted angles from the tracking filter which assumes a nominally non-maneuvering ballistic target. A weighted average of samples over a moving window is computed to estimate the prediction bias and the standard deviation. A maneuver detection is declared when the bias is greater than a selected threshold.

Let the actual measurements and the predicted measurements from the tracking filter be z_k , and \hat{z}_k , respectively, sampled at time t_k . An innovation process is defined by:

$$\begin{aligned}
 \varepsilon_k &= \hat{z}_k - z_k \\
 &= h(\hat{x}_k) - h(x_k) - w_k \\
 &= \left. \frac{\partial h}{\partial x} \right|_{x=\hat{x}_k} \delta x - w_k \\
 &= H_k(\hat{x}_k - x_k) - w_k
 \end{aligned} \tag{24}$$

where h is the functional related to the measurement equation, w_k is the measurement noise vector,

$$z_k = h(x_k) + w_k \tag{25}$$

and x_k denotes the sampled value of the target state vector whose components are the position and velocity of the target. The covariance of the innovation process is given by:

$$\begin{aligned}
 S_k &= E[\varepsilon_k \varepsilon_k^T] \\
 &= H_k P_k H_k^T + R_k,
 \end{aligned} \tag{26}$$

where P_k and R_k are the state covariance and the measurement noise covariance, respectively. The innovation bias vector is computed from averaging ε_n over m samples:

$$b_n = \frac{1}{m} \sum_n^{k=n-m+1} \varepsilon_k . \tag{27}$$

This can be also computed recursively using

$$b_n = b_{n-1} + \frac{1}{m} (\varepsilon_n - \varepsilon_{n-m}) . \tag{28}$$

The weighted mean square distance is

$$T_n = b_n^T S_n^{-1} b_n \tag{29}$$

Maneuver detection is based on comparing T_n to a threshold λ , i.e.,

$$\begin{cases} T_n \geq \lambda & \text{target maneuvers,} \\ T_n < \lambda & \text{target is ballistic} \end{cases} \tag{30}$$

For a 3-sigma maneuver-detection probability of 0.971, we choose $\lambda = 9$.

Chapter Two

XRLSim User Interfaces

This chapter provides the user with changes in user interfaces brought about by enhancements made to XRLSim model specifications during FY89. It is offered as a series of deletions, replacements, and insertions to the original document (UCID-21736) to enable XRLSim users to implement these enhancements.

3.3.3 Vehicle

**Delete the following:
(found on page 52 of UCID-21736)
from Sec. 3.3.3**

Input Name: System_Firing_Delay
Type: real
Units: seconds
Default: 0.0
Remarks: The time that elapses after receipt of the command `Fire_For_Effect` from the simulator, until beam energy is produced. Target lead-angle computation advances the estimated location of the target by a time interval $(\text{System_Firing_Delay} + 2 \text{ dr}/c)$ dr is estimated range to target; c is the speed of light.

Input Name: Angle
Type: real
Units: microradians
Default: 2π radians
Remarks: The full angle divergence of the beam energy in a directed-energy weapon.

Input Name: Distance
Type: real
Units: km
Default: infinity
Remarks: The lethal range of the beam energy in a directed-energy weapon. A target is destroyed if distance from platform to target is less than `Distance` *and* the target is within the included `Angle` when the beam arrives at the target's true position.

End Deletion

3.3.5 Platform

Completely replace Sec. 3.3.5 Platform
(from Page 55 and 56 of UCID-21736)
with the following section

The specification of the mathematical models for the Platform dynamics and control are described in Chapter 1 of this manual.

A Platform is a free-flying object that has control laws to point at a target point and to control its rotation rates on each axis. The object does not have any body forces aside from gravity; all the torques are balanced about the center of gravity so there is no acceleration of the center of gravity.

The state vector of the platform is

I-Frame (X, Y, Z)

V-Frame (Vx, Vy, Vz)

Omega (Roll, Pitch, Yaw)

Beta quaternion Euler parameter

Reaction_Rotor_Omega (Roll, Pitch, Yaw)[†]

Angular_Rate_Integral (Roll, Pitch, Yaw) ACS controller states^{††}

Roll_Thruster_State

Pitch_Thruster_State

Yaw_Thruster_State

Input Name: Name
Type: String
Units: none
Default: none
Remarks: Used for error messages.

Input Name: Ldx
Type: Real
Units: meters
Default: none
Remarks: Distance from centerline to pitch and yaw thrusters.

Input Name: Ldy
Type: Real
Units: meters
Default: none
Remarks: Distance from centerline to roll thrusters.

Input Name: Moment_of_Inertia
Type: 3 vector
Units: kg × m²
Default: none
Remarks: Moments of inertial about roll, pitch, yaw axes.

[†] These states are associated with reaction wheel control which is not fully implemented at present.

^{††} These states are used for testing only and have no meaning for most users.

The platform has identical thrusters on each of the roll, pitch and yaw axes. A single input Thruster defines the properties of the three thruster objects; thruster input is defined in Sec. 3.3.5.1 following this section.

Input Name: Lyapunov_Rate_Gain
Type: 3 vector
Units: <<?>>
Default: none
Remarks: The diagonal elements of the pointing control matrix D. The three constants control the roll, pitch, yaw pointing control, respectively.

Input Name: Lyapunov_Pointing_Gain
Type: 3 vector
Units: <<?>>
Default: none
Remarks: The diagonal elements of the pointing control matrix K. The three constants control the roll, pitch, yaw pointing control, respectively.

Input Name: ACS_Gain1†
Type: 3 vector
Units: <<?>>
Default: none
Remarks: The diagonal elements of the first attitude control constant. The three constants control the roll, pitch, yaw pointing control, respectively.

Input Name: ACS_Gain2†
Type: 3 vector
Units: <<?>>
Default: none
Remarks: The diagonal elements of the second attitude control constant. The three constants control the roll, pitch, yaw pointing control, respectively.

The platform's advance operator has a different effect dependent on whether the platform controller has the Sample_and_Hold property. In case Sample_and_Hold is false (i.e., no values are given for Control_Sample_Interval and Control_Sample_End_Time), the control law is integrated continuously with the dynamics. Pulse width modulation is not allowed when the continuous integration mode is selected.

Input Name: Control_Sample_Interval
Type: real
Units: seconds
Default: none
Remarks: A vector of control sample frequencies. Give a vector of no more than ten time-steps. Control sampling occurs at the interval given by the first Control_Sample_Interval until time reaches the first Control_Sample_End_Time, then at each successive Control_Sample_Interval in turn.

† ACS_Gain1 and ACS_Gain2 are associated with the PI_Controller algorithm, which is used only for testing. They should normally be defaulted or set to zero by the user. The PI_Controller algorithm should not be used with pointing thrust control.

Input Name: Control_Sample_End_Time
Type: real
Units: seconds
Default: none
Remarks: A vector of control sample end-times. Give a vector of no more than ten absolute times. Control sampling occurs at the interval given by the first Control_Sample_Interval until time reaches the first Control_Sample_End_Time, then at each successive Control_Sample_Interval in turn. The Control_Sample_End_Time vector must be monotonically increasing. The sample times must be greater than Vehicle.T_Eject so that the platform is free of the lifter rocket. The way to conform to these rules about starting the control late in the problem is to specify the first value of Control_Sample_Interval and the first value of Control_Sample_End_Time as the time when actual control should begin. For example, if Vehicle.T_Eject = 150, to control at 10 Hertz for 50 seconds, use the input Control_Sample_Interval = 150., 0.1 ; Control_Sample_End_Time = 150., 200.0

Control of the thrusters is optionally pulse width modulated (PWM) with input of pulse amplitude and synthesized frequency. In case of PWM control, the platform must be advanced using sample-and-hold semantics (not continuous pointing control) and the sample frequency must be at least four times the PWM synthesis frequency.

Input Name: PWM_Amplitude
Type: real
Units: newtons
Default: none
Remarks: The value of thrust that is assumed by the PWM controller; should be the same as the value of F_Bound of the thruster objects.

Input Name: PWM_Frequency
Type: real
Units: Hertz
Default: none
Remarks: The synthesized frequency that the PWM controller compares with the control law.

Input Name: Rotor_Moment_of_Inertia†
Type: 3 vector
Units: kg × m²
Default: none
Remarks: Unused (*there is no control law to operate rotors).

Input Name: Motor_Torque_Constant
Type:
Units:
Default:
Remarks: Unused (there is no control law to operate rotors).

Input Name: Motor_Damping_Coefficient
Type:
Units:
Default:
Remarks: Unused (there is no control law to operate rotors).

† Rotor_Moment_of_Inertia, Motor_Torque_Constant, and Motor_Damping_Coefficient are associated with reaction wheel control, which is not fully implemented at present.

Input Name: Integrator_Scale_Factor
Type: real
Units: none
Default: none
Remarks: If any value (which must be less than 1.0) is given, then the minimum and the initial integration interval used by DVerk (Hmin, HStart in DVerk documentation) is set to the advance interval times Integrator_Scale_Factor.

3.3.5.1 Thruster

The platform has three distinct thrusters that all have the same input options in a single run. The input parameters for these thrusters are given in a single block of input data.

The thrusters can select from any of the models described here. The default model is the ideal thruster, which has only the input F_Bound, the maximum thrust in newtons. If a positive input is given for Delay_Time, then the thruster responds to a command with some delay in seconds. If the input Cir_Freq is given, then the dynamic model is chosen and inputs must be given for Damp and S2_Bound. If the input Hysteresis_Model is given, then the deadband hysteresis model is given and its parameters (listed below) are required.

Input Name: Name
Type: String
Units: none
Default: none
Remarks: Used for error messages.

Input Name: F_Bound
Type: real
Units: newtons
Default: none
Remarks: s_{11} in the thruster model specification. Maximum thrust from thruster; required for all models.

Input Name: Cir_Freq
Type: real
Units: radians/second⁻¹
Default: none
Remarks: ω_n in the thruster model specification. Input to Cir_Freq selects the dynamic response model and requires input for Damp and S2_Bound.

Input Name: Damp
Type: real
Units: none
Default: none
Remarks: η in the thruster model specification. Used only in case of dynamic thruster model.

Input Name: S2_Bound
Type: real
Units: none
Default: none
Remarks: s_{21} in the thruster model specification. Used only in case of dynamic thruster model.

Input Name: Delay_Time
Type: real
Units: seconds
Default: none
Remarks: Selects the delay-line model for thruster.

Deadband Hysteresis models the response of a mechanical valve, which returns an output that is subject to a deadband (when the input is near zero) and, which saturates and returns an upper bound output when the input is larger than maximum. After saturating, the output follows a linear restoration curve that is different from the linear loading.

There is a different set of parameters for each of positive and negative inputs (but if negative parameters are not given, the default behavior is symmetric).

The deadband hysteresis function is graphically illustrated in Fig. 3-2. The arrowheads on the graphic depict the direction of a trajectory for response to positive input. With the object initially unsaturated, the response follows the trajectory to X_{load} , then rises to value Y_{sat} when X equals or exceeds X_{sat} ; until X has exceeded X_{sat} , the response function remains on this lower branch of the hysteresis loop.

After X has exceeded X_{sat} , the model is said to be saturated. The response of the saturated object is to return Y_{sat} until $X < X_{unload}$, then ramp down to $Y = 0$ when $X = X_{unsat}$. While the object is saturated, the response follows the upper branch of the hysteresis loop. After X has fallen below the value X_{unsat} , the object is once again unsaturated.

Response to negative input behaves just as the positive response, save that the parameters are optionally different if input, and the response to input $-1 * |x|$ is given by $-1 * |y|$.

Input Name: Hysteresis_Model
Type: integer
Units: none
Default: none
Remarks: Any input value selects the deadband hysteresis model and requires input for the following five parameters: X_{loadP} , X_{SatP} , $X_{UnloadP}$, X_{UnsatP} , Y_{SatP} . If the negative hysteresis loop is to be distinct from the positive loop, then give (positive) values for all of X_{loadN} , X_{SatN} , $X_{UnloadN}$, X_{UnsatN} , Y_{SatN} .

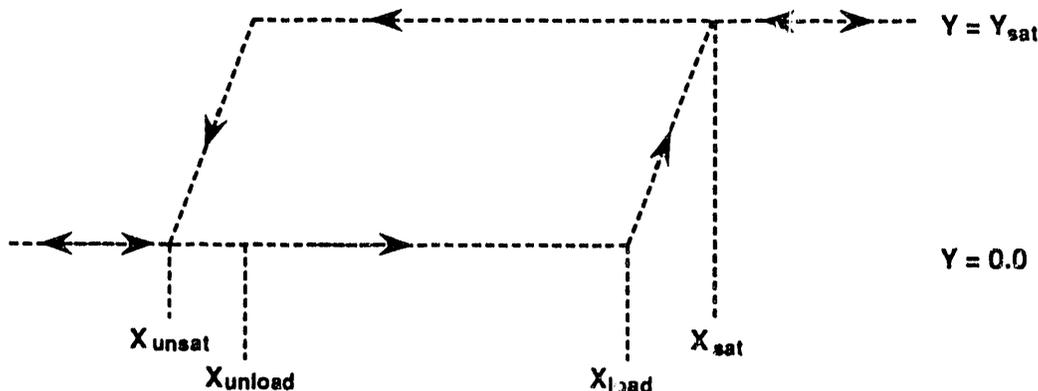


Figure 3-2. The deadband hysteresis function.

End Replacement

3.3.7 Onboard_Track

**Insert the following to the end of Sec. 3.3.7
on page 59 of UCID-21736**

The Kalman filter instance for onboard tracking has an optional model that allows target maneuvers to be detected.

The maneuver-detection algorithm is selected by specifying the number of samples over which the moving average bn is to be computed. When a maneuver is detected with probability greater than 3 sigma, the filter covariance is reset to the value that was specified when the filter was initialized.

Input Name: Maneuver_Detect_Span
Type: positive integer
Units: number
Default: zero
Remarks: If any positive value is given, then the maneuver detection algorithm is selected and the probability of target maneuver is computed with a moving average window of Maneuver_Detect_Span samples.

End Insertion

3.3.12 Laser Lethality

**Insert this new section
after Sec. 3.3.11
on page 65 of UCID-21736**

The XRLSim5A simulation may incorporate a directed-energy weapon. Algorithms have been developed to compute that weapon's effect on its target and certain inputs have been determined for the properties of potential targets. This section describes the input parameters employed to assess XRL lethality.

If the simulation object has an input for the variable `Fire_For_Effect_Time`, then the simulator sends a fire command to the weapon at that time. The weapon starts its firing countdown and expends its weapon after a time given by `Laser_Lethality` input `System_Firing_Delay`. When the weapon is expended, the `Damage_Assessment` model computes the time for speed-of-light propagation from the vehicle to the target and sets an event for the time when photons will arrive.

Input Name: System_Firing_Delay
Type: real
Units: seconds
Default: 0.0
Remarks: The time that elapses after receipt of the command `Fire_For_Effect` from the simulator until beam energy is produced.

When the weapon's energy arrives at the target, the effect on the target is computed by one of two models, which take their input from the category `Laser_Lethality`. If no value is given for the input `Gaussian_NDEW`, then the model of target damage is a simple one: if the target lies within the cone of effectiveness given by (`Angle`, `Distance`), then it is destroyed.

Input Name: `Angle`
Type: real
Units: microradians
Default: 2π radians
Remarks: The full-angle dispersion of the energy of the beam in a directed-energy weapon. A target is destroyed if the distance from the platform to the target is less than `Distance` *and* the target is within the included `Angle` when the beam arrives at the target's true position.

Input Name: `Distance`
Type: real
Units: km
Default: infinity
Remarks: The lethal range of the energy of the beam in a directed-energy weapon. A target is destroyed if the distance from the platform to the target is less than `Distance` and the target is within the included `Angle` when the beam arrives at the target's true position.

If any input is given for `Gaussian_NDEW`, then the effect of the weapon's energy deposited on the target is computed using algorithms developed for that purpose. Inputs to this model are:

Input Name: `Gaussian_NDEW`
Type: integer flag
Units: none
Default: flag is absent
Remarks: If any value is given, then the target damage model is selected, and input for `Beam_Energy` and `Beam_Divergence_Sigma` are required, along with target data.

Input Name: `Beam_Energy`
Type: real
Units: gigajoules (i.e., 10^9 joules)
Default: none
Remarks: Total energy for the laser beam.

Input Name: `Beam_Divergence_Sigma`
Type: real
Units: microradians (full angle)
Default: none
Remarks: The spread of the Gaussian distribution of the directed energy beam.

Input Name: `NDEW_Target_Filename`
Type: string
Units: readable file
Default: none
Remarks: Files containing target data are to be read.

End Insertion

END

DATE FILMED

12 / 11 / 90

