JSC-61777-x.xx.xx

JSC Engineering Orbital Dynamics LVLH Frame Model

Simulation and Graphics Branch (ER7)
Software, Robotics, and Simulation Division
Engineering Directorate

Package Release JEOD v5.1

Document Revision 1.0
July 2023

National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas

JSC-61777-x.xx.xx

JSC Engineering Orbital Dynamics LVLH Frame Model

Document Revision 1.0
July 2023

Jeff Morris

Simulation and Graphics Branch (ER7)
Software, Robotics, and Simulation Division
Engineering Directorate

National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas

Abstract

The JEOD LVLH Frame Model provides a simple standalone model for defining a reference frame with respect to the position and velocity vectors of an on-orbit object. This capability is extremely useful for certain commonly encountered situations in spaceflight, such as tracking the relative motion of multiple orbital vehicles. Making LVLH Frame Model distinct from the standard JEOD reference frames model allows the LVLH Frame Model to be anchored to objects that are not DynBodies.

When used in concert with the LvlhRelativeDerivedState derived class introducted with JEOD 3.2, LVLH Frame Model can be used to track relative motion in either rectilinear or curvilinear coordinates.

Contents

Chapter 1

Introduction

1.1 Model Description

The JEOD LVLH Frame Model enables the definition of a reference frame with respect to the position and velocity vectors of an on-orbit object and automatically handles the updates to that frame. This model is envisioned primarily to be used in tracking the relative motion of one or more orbital vehicles with respect to another. However, making the LVLH Frame Model distinct from the standard JEOD reference frames model allows the LVLH Frame Model to be anchored to (and therefore defined by the motion of) objects that are not DynBodies if so desired.

When used in concert with the LvlhRelativeDerivedState derived class introducted with JEOD 3.2, the LVLH Frame Model can be used to track relative motion in either rectilinear or curvilinear coordinates. However, the LVLH Frame Model simply defines the LVLH reference frame itself.

1.2 Document History

Author	**Date**	**Revision**	**Description**
Jeff Morris	Jan 2016	1.0	Initial Release

1.3 Document Organization

This document is formatted in accordance with the NASA Software Engineering Requirements Standard [2] and is organized into the following chapters:

Chapter 1: Introduction - This introduction contains three sections: description of model, document history, and organization. The first section provides the introduction to the LVLH Frame Model and its reason for existence. It also contains a brief description of the interconnections with other models, and references to any supporting documents. The second section displays the history of this document which includes author, date, and reason for each revision; it also lists the document that is parent to this one. The final section contains a description of the how the document is organized.

Chapter 2: Product Requirements - Describes requirements for the LVLH Frame Model.

Chapter 3: Product Specification - Describes the underlying theory, architecture, and design of the LVLH Frame Model in detail. It is organized into three sections: Conceptual Design, Mathematical Formulations, and Detailed Design.

Chapter 4: User Guide - Describes how to use the LVLH Frame Model in a Trick simulation. It contains information that will be useful to both users and developers of simulations. The LVLH Frame Model is not intended to be extended, so there is no discussion of how to do so.

Chapter 5: Inspections, Tests, and Metrics - The inspections, tests, and metrics describes the procedures and results that demonstrate the satisfaction of the requirements for the LVLH Frame Model.

Chapter 2

Product Requirements

This model shall meet the JEOD project requirements specified in the JEOD top-level document.

Requirement LvlhFrame_1: Project Requirements

Requirement:
This model shall meet the JEOD project-wide requirements specified in the JEOD v5.1 Top-Level Document.

Rationale:
This is a project-wide requirement.

Verification:
Inspection

Requirement LvlhFrame_2: Create and Manage Required Connections

Requirement:
The LVLH Frame Model shall create and manage all the connections and data structures necessary to enable successful model operation.

Rationale:
The model must be able to manage itself in order to function.

Verification:
Inspection, test

Requirement LvlhFrame_3: Define LVLH Frame

Requirement:
The LVLH Frame Model shall correctly calculate the rotational states of its LVLH reference frame with respect to the user-specified planetary inertial frame when requested.

Rationale:

This is the purpose of the LVLH Frame Model.

Verification:

Test

Chapter 3

Product Specification

3.1 Conceptual Design

The LVLH Frame Model is a relatively simple model, whose main function is to calculate the translational and rotation states of a standard JEOD reference frame with respect to a designated planet-centered inertial reference frame. However, it exists as a standalone model because the calculations that govern the orientation of a local-vertical, local-horizontal frame with respect to a planet are unique from other frames, yet common enough in spaceflight applications to merit them being encapsulated as their own model.

As implemented in JEOD, an LVLH reference frame is defined using the position and velocity vectors of the body or object with which it is associated. The vectors used are measured with respect to, and expressed in, a specified planet-centered inertial reference frame (does not have to be the frame in which the object is being integrated). The position vector defines the *negative* Z-axis, and the angular momentum vector (position cross velocity) defines the *negative* Y-axis. Consistent with the right-hand rule, the positive X-axis results from the vector cross of +Y and +Z to complete the orthogonal set.

3.2 Mathematical Formulations

The only calculations performed by the LVLH Frame Model are the ones used to determine the orientation and angular velocity of the LVLH frame with respect to the specified planet-centered inertial reference frame. It is not necessary to calculate its position or translational velocity, since those are by definition identical to those of the object to which it is anchored.

The three vectors that define LVLH are as follows:

- Opposite to the radial vector from planet center ($\hat{k} = -\hat{r}$)
- Opposite to the angular momentum vector ($\hat{j} = -\hat{r} \times \hat{v}$)
- Completing the right handed coordinate system ($\hat{i} = \hat{j} \times \hat{k}$)

The $\hat{\imath}$ axis also represents the projection of the velocity vector onto the instantaneous horizontal plane. For circular orbits, the $\hat{\imath}$ axis is tangent to the vehicle orbit, but that is a characteristic rather than a definition.

Suppose the position and velocity of an object are known in a given planet- centered inertial reference frame as:

$$\vec{r}_{inrtl} = [r_{x,inrtl}\ r_{y,inrtl}\ r_{z,inrtl}]$$

$$\vec{v}_{inrtl} = [v_{x,inrtl}\ v_{y,inrtl}\ v_{z,inrtl}]$$

It is desired to define an LVLH reference frame fixed at the object's center of mass, with orientation measured with respect to the given inertial frame. The resulting transformation matrix from inertial to LVLH would be:

$$\vec{x}_{lvlh} = T_{inrtl \to lvlh}\ \vec{x}_{inrtl}$$

$$T_{inrtl \to lvlh} = \begin{bmatrix} p_1 & p_2 & p_3 \\ -h_1 & -h_2 & -h_3 \\ -r_1 & -r_2 & -r_3 \end{bmatrix}$$

where r_i are the components of the unit radial vector, h_i are the components of the unit angular momentum vector, and p_i are the components resulting from the cross product of $\hat{r}$ with $\hat{h}$.

The LVLH frame is rotating with respect to the inertial frame at a rate of one revolution per orbit. Expressed in the LVLH frame, that relative angular velocity is oriented along the $-\hat{\jmath}$ axis only, by definition. Thus, the angular velocity is:

$$\vec{\omega}_{lvlh/inrtl:lvlh} = -|\omega|\hat{\jmath}$$

3.3 Detailed Design

See the Reference Manual[1] for a summary of member data and member methods for all classes.

3.3.1 Process Architecture

The process architecture for the LVLH Frame Model is trivial; the LVLH Frame Model comprises the usual *initialize* and *update* methods, as well as four methods for setting subject and planet names and pointers. Finally, there is the protected method *compute_lvlh_frame*, which calculates both the transformation matrix and angular velocity defined in the previous section.

3.3.2 Functional Design

This section describes the functional operation of the methods in each class.

The LVLH Frame Model contains only one class, LvlhFrame. It contains the following methods:

1. **initialize**

   This method sets up the model properly, does error checking on provided object and planet names, and makes connections to the reference frame tree.

2. **update**

   The *update* method calls *compute_lvlh_frame*, which determines the current orientation and angular velocity of the LVLH frame with respect to the specified planet. If the specified planet is the integration frame of the subject object with which the LVLH frame is associated, then *update* is effectively a pass-through. But if the subject is integrated in a different frame, then *update* calculates the object's state with respect to the planet before calling *compute_lvlh_frame*.

3. **set_subject_name**

   This method allows the user to set the subject object's name via function call.

4. **set_planet_name**

   This method allows the user to set the reference planet's name via function call.

5. **set_subject_frame**

   Allows the user to set the subject object's body reference frame pointer directly, via function call.

6. **set_planet**

   Allows the user to set the reference planet pointeri directly, via function call.

7. **compute_lvlh_frame**

   This method calculates the current orientation and angular velocity of the LVLH reference frame with respect to the reference planet's inertial frame.

Chapter 4

User Guide

The LVLH Frame Model is relatively straightforward to use. Simply include the object in one's Trick S_define, along with the *initialize* and *update* function calls, then provide the names of the subject object and planet either by using the provided setter methods or by doing so directly. The model should then perform correctly. The LVLH Frame Model is not intended to be extended via derived classes, as it is a simple, single-purpose model.

An example inclusion of the LVLH Frame Model in one's Trick simulation is as follows. In the S_define, include something similar to the following header, object declaration, and function calls:

```
##include "utils/lvlh_frame/include/lvlh_frame.hh"

public:
  LvlhFrame               lvlh_frame;

// Initialization jobs
P_DYN ("initialization") lvlh_frame.initialize (dyn_manager);
      ("initialization") lvlh_frame.update ();

// Environment class jobs
(1.0, "environment") lvlh_frame.update ();
```

Note that this example assumes a dynamics manager exists elsewhere in the simulation with the name "dyn_manager". Also notice that the *update* method should be run once at initialization, following the call to *initialize*, and then on a recurring basis as an "environment" job. Here, the frequency is once a second.

Once the object has been included in the simulation as above, something like the following should be added to the input file of the Trick simulation to enable proper model operation:

```
# Configure the LVLH reference frame.
vehicle.lvlh_frame.set_subject_name ("vehicle.composite_body")
vehicle.lvlh_frame.set_planet_name ("Earth")
```

This snippet assumes there is a planet that exists in the sim named “Earth”, as well as a DynBody with the name “vehicle”.

Chapter 5

Inspections, Tests, and Metrics

5.1 Inspection

Inspection LvlhFrame_1: Top-level Inspection

This document structure, the code, and associated files have been inspected, and together satisfy requirement LvlhFrame_1.

Inspection LvlhFrame_2: Connections Inspection

The model performs setup of itself during initialization, which includes establishing appropriate connections between its internal pointers and the corresponding specified structures outside of it, after error-checking the user inputs.

Therefore by inspection, the LVLH Frame Model partially satisfies requirement LvlhFrame_2.

5.2 Tests

This section describes the tests conducted to verify and validate that the LVLH Frame Model satisfies the requirements levied against it. All verification and validation test source code, simulations and procedures are archived in the JEOD directory `models/utils/lvlh_frame/verif`.

Test LvlhFrame_1: State Calculations

Background

The primary purpose of this test is to determine whether the LVLH Frame Modelcorrectly calculates the rotational states of its LVLH reference frame with respect to the user- specified planet's inertial frame. Implicitly, the test will also demonstrate that the model can properly manage its connections and internal data structures that are necessary for those calculations.

Test description

This test involves checking the calculations performed by the LVLH Frame Model to ensure that the transformation matrix and angular velocity of its LVLH frame is calculated correctly with respect to a given reference planet.

For this test, a simple Trick simulation of a single vehicle in orbit about a single planet was employed. The vehicle is provided to the LVLH Frame Model for use as the subject body for the LVLH reference frame, since an LVLH frame must be associated with a moving object in order to be mathematically valid. The planet in the simulation is of course provided to the LVLH Frame Model as the reference planet for the LVLH frame.

Once the state of the LVLH frame has been determined by the model, the output is then checked by an independent, non-Trick set of routines. The results are then compared against the Trick logfile for accuracy evaluation.

Test directory `SIM_LVLH_Frame`

Success criteria

A comparison of the output of the LVLH Frame Model and *compute_relative_state* should agree to within approximately numerical error.

Test results

All output data confirmed expectations, within numerical precision.

Applicable Requirements

This test completes the satisfaction of requirement LvlhFrame_2, and satisfies the requirement LvlhFrame_3.

5.3 Requirements Traceability

Table 5.1 summarizes the inspections and tests that demonstrate the satisfaction of the requirements levied on the model.

Table 5.1: Requirements Traceability

Requirement	**Traces to**
LvlhFrame_1 Project Requirements	Insp. LvlhFrame_1 Top-level Inspection
LvlhFrame_2 Create and Manage Required Connections	Insp. LvlhFrame_2 Connections Inspection
	Test LvlhFrame_1 State Calculations
LvlhFrame_3 Define LVLH Frame	Test LvlhFrame_1 State Calculations

5.4 Metrics

Table 5.2 presents coarse metrics on the source files that comprise the model.

Table 5.2: Coarse Metrics

	Number of Lines			
File Name	**Blank**	**Comment**	**Code**	**Total**
Total	0	0	0	0

Table 5.3 presents the extended cyclomatic complexity (ECC) of the methods defined in the model.

Table 5.3: Cyclomatic Complexity

Method	**File**	**Line**	**ECC**
jeod::LvlhType::LvlhType:: LvlhType (void)	include/lvlh_type.hh	117	1
jeod::LvlhFrame::LvlhFrame (void)	src/lvlh_frame.cc	48	1
jeod::LvlhFrame::~LvlhFrame (void)	src/lvlh_frame.cc	66	4
jeod::LvlhFrame::initialize (DynManager & dyn_ manager)	src/lvlh_frame.cc	91	7
jeod::LvlhFrame::update (void)	src/lvlh_frame.cc	186	2
jeod::LvlhFrame::set_subject_ name (const std::string & new_name)	src/lvlh_frame.cc	215	1
jeod::LvlhFrame::set_subject_ frame (RefFrame & new_ frame)	src/lvlh_frame.cc	226	1
jeod::LvlhFrame::set_planet_ name (const std::string & new_name)	src/lvlh_frame.cc	238	1
jeod::LvlhFrame::set_planet (BasePlanet &new_planet)	src/lvlh_frame.cc	250	1
jeod::LvlhFrame::compute_ lvlh_frame (const RefFrame Trans & rel_trans)	src/lvlh_frame.cc	262	1

Bibliography

[1] Generated by doxygen. *JEOD LVLH Frame Model Reference Manual*. National Aeronautics and Space Administration, Johnson Space Center, Software, Robotics & Simulation Division, Simulation and Graphics Branch, 2101 NASA Parkway, Houston, Texas, 77058, July 2023.

[2] NASA. NASA Software Engineering Requirements. Technical Report NPR-7150.2, NASA, NASA Headquarters, Washington, D.C., September 2004.