JSC-61777-models/environment/planet

# JSC Engineering Orbital Dynamics Planet Model

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

## Package Release JEOD v5.2

## Document Revision 1.0
## September 2024


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

JSC-61777-models/environment/planet

# JSC Engineering Orbital Dynamics
# Planet Model

## Document Revision 1.0
## September 2024

**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 Planet Model is primarily a container model in which information for a particular planetary body is stored. The parent BasePlanet object contains both an inertial and a planet-fixed reference frame, though the Planet Model plays no role in updating or maintaining either of them.

The derived-class Planet object is ellipsoidal-body specific, as it contains planetary shape constants that are only valid for an ellipsoid. The Planet class also contains a connection to the GravitySource structure associated with the same planetary body.

This document describes the implementation of the Planet Model including the model requirements, specifications, mathematical theory, and architecture. A user guide is provided to assist with implementing the model in Trick simulations. Finally, the methods and results of verification and validation of the model are included.

# Contents

# Chapter 1

# Introduction

## 1.1 Purpose and Objectives of the Planet Model

Modeling planets is an integral part of the functionality of JEOD v5.2, as many scenarios likely to be modeled using the package involve a vehicle or vehicles in orbit about some planetary body. The purpose of the Planet Model is to provide a single, consolidated location for the storage of, and connection to, all the related elements that can be considered to "define" a given planet mathematically. For an ellipsoidal planetary body, its shape constants are stored directly in the Planet Model along with a pointer to the gravity model for the same planet. For all planetary bodies, ellipsoidal or otherwise, the Planet Model houses the inertial and planet-fixed reference frames associated with it–though the Planet Model plays no role in updating or maintaining these two frames.

## 1.2 Context within JEOD

The following document is parent to this document:

- *JSC Engineering Orbital Dynamics* [3]

The Planet Model forms a component of the environment suite of models within JEOD v5.2. It is located at models/environment/planet.

## 1.3 Document History

| Author | Date | Revision | Description |
|---|---|---|---|
| Jeff Morris | Feb 2012 | 1.1 | Update for BasePlanet |
| Jeff Morris | April 2010 | 1.0 | Initial Version |

The Planet Model is a model that is new for JEOD v2.0.x, though it is partly derived from the

Miscellaneous Transformations and Planetary Functions model released as part of JEOD v1.5.x. Thus, this document derives lightly from that model's documentation, entitled JSC Engineering Orbital Dynamics Miscellaneous Transformations and Planetary Functions, most recently released with JEOD v1.5.2.

## 1.4 Document Organization

This document is formatted in accordance with the NASA Software Engineering Requirements Standard [4] and is organized into the following chapters:

**Chapter 1: Introduction** - This introduction contains four sections: purpose and objective, context within JEOD, document history, and document organization.

**Chapter 2: Product Requirements** - Describes requirements for the Planet Model.

**Chapter 3: Product Specification** - Describes the underlying theory, architecture, and design of the Planet Model in detail. It is organized into four sections: Conceptual Design, Mathematical Formulations, Detailed Design, and Version Inventory.

**Chapter 4: User's Guide** - Describes how to use the Planet Model in a Trick simulation. It is broken into three sections to represent the JEOD defined user types: Analysts or users of simulations (Analysis), Integrators or developers of simulations (Integration), and Model Extenders (Extension).

**Chapter 5: Verification and Validation** - Contains verification and validation procedures and results for the Planet Model.

# Chapter 2

# Product Requirements

This chapter identifies the requirements for the Planet Model.

*Requirement planet_1: Top-level Requirement*

**Requirement:**
This model shall meet the JEOD project requirements specified in the JEOD v5.2 top-level document.

**Rationale:**
This model shall, at a minimum, meet all external and internal requirements applied to the JEOD v5.2 release.

**Verification:**
Inspection

## 2.1 Data Requirements

This section identifies requirements on the data represented by the Planet Model. These as-built requirements are based on the Planet Model data definition header files.

*Requirement planet_2: Basic Planetary Body Data Encapsulation*

**Requirement:**
The base Planet Model object shall encapsulate the following data for the planetary body it represents:

*2.1 Planet Name* The name of the planet being defined.

*2.2 Reference Frames* The planet's inertial and planet-fixed reference frames.

**Rationale:**
Providing this basic capability in the Planet Model parent class allows its use with all planetary bodies, including irregularly shaped ones.

**Verification:**
Inspection

*Requirement planet_3: Ellipsoidal Planet Data Encapsulation*

**Requirement:**
For ellipsoidal planetary bodies, Planet Model shall additionally encapsulate the following via a provided derived class:

*3.1 Shape Constants* The set of constants that define the shape of the planet, consisting of the mean equatorial radius, mean polar radius, ellipsoid eccentricity, the square of the ellipsoid eccentricity, flattening coefficient, and the inverse of the flattening coefficient.

*3.2 Gravity Model Connection* A pointer to the planet's gravity model.

**Rationale:**
The primary purpose of the Planet Model is to serve as a container for these types of information considered to "define" a planet's shape for the purposes of JEOD v5.2.

**Verification:**
Inspection

## 2.2 Functional Requirements

This section identifies requirements on the functional capabilities provided by the Planet Model. These as-built requirements are based on the Planet Model source files.

*Requirement planet_4: Basic Model Registration*

**Requirement:**
The Planet Model shall perform the following actions when commanded to register, for all bodies represented:

*4.1 Register with Ephemerides Manager* Add itself to the Ephemerides Manager's list of planets, and its inertial and planet-fixed frames to the Ephemerides Manager's list of reference frames.

*4.2 Connect Reference Frames* Connect its two reference frames to each other and to the correct reference frame pointers in the corresponding ephemeris object.

**Rationale:**
All of these connections are needed for correct operation of the Planet Model.

**Verification:**
Inspection, Test

*Requirement planet_5: Ellipsoidal Planet Model Registration*

**Requirement:**
The Planet Model shall perform the following additional actions when commanded to register, for ellipsoidal bodies:

*5.1 Connect to Gravity Model* Find the gravity model corresponding to the same planet, then correctly cross-connect itself and the gravity manager together.

**Rationale:**
This additional connection is needed for correct operation of the Planet Model, for ellipsoidal planetary bodies.

**Verification:**
Inspection, Test

*Requirement planet_6: Initialize Ellipsoidal Planet Constants*

**Requirement:**
When commanded to initialize for an ellipsoidal planet, the Planet Model shall initialize the values of the set of constants that define the planetary shape, consisting of: the mean equatorial radius, mean polar radius, ellipsoid eccentricity, the square of the ellipsoid eccentricity, flattening coefficient, and the inverse of the flattening coefficient.

**Rationale:**
These values must be set for proper operation of any simulation that includes the ellipsoidal planet derived class of the Planet Model.

**Verification:**
Inspection, Test

# Chapter 3

# Product Specification

This chapter defines the conceptual design, the mathematical formulations, and the detailed design for the Planet Model. It also contains the version inventory for this release of the Planet Model.

## 3.1 Conceptual Design

The Planet Model is designed to be a container class which serves as the single consolidated JEOD v5.2 location for the storage of, or connection to, all the related elements that mathematically "define" a planetary body. It houses the inertial and planet-fixed reference frames for the planet, though the Planet Model plays no role in updating them. If the model is representing an ellipsoidal body, then based on user input, it also calculates and stores in itself the planetary shape constants for the planet. Also for ellipsoidal bodies, the model connects itself to the GravitySource and ephemeris structures associated with that planet.

## 3.2 Mathematical Formulations

While the Planet Model contains other information (a pointer to the ephemeris structure associated with the same planet, as well as the inertial and planet-fixed reference frames for it), it is only responsible for calculating something when it is being used to represent an ellipsoidal planetary body. When this is the case, it calculates the ellipsoidal shape constants found within it. The techniques used to solve for these values are discussed in this section.

From [8], the equations used to solve for the other planetary shape constants based on the ones provided by the user are as follows:

$$f = \frac{r_{eq} - r_{pol}}{r_{eq}} \tag{3.1}$$

$$r_{pol} = r_{eq} \times \sqrt{1 - e^2} \tag{3.2}$$

where

- $f$ is the planet's flattening coefficient,
- $r_{eq}$ is the planet's mean equatorial radius,
- $r_{pol}$ is the planet's mean polar radius, and
- $e$ is the planet's ellipsoid eccentricity.

Appropriate permutations of these equations are used depending on which constants the user provides. If none are provided, then all constants are automatically set to zero by the Planet Model. More details about how to specify appropriate combinations of the planetary constants are given in the User Guide (Chapter 4).

## 3.3 Detailed Design

The functionality of the Planet Model is contained within two classes, one being derived from the other. This section describes both classes in detail.

#### BasePlanet Class Design

The *base_planet.hh* file contains the basic Planet Model class, named "BasePlanet". BasePlanet contains the following parameters common to all planetary bodies:

- name: The name of the planetary object being defined,
- inertial: The planet-centered J2000 pseudo-inertial reference frame, of object type EphemerisRefFrame, associated with this object,
- alt_inertial: A secondary pseudo-inertial reference frame, of object type EphemerisRefFrame, associated with this object,
- pfix: The planet-centered, planet-fixed Cartesian reference frame, of object type EphemerisRefFrame, associated with this planet, and
- alt_pfix: A secondary planet-centered, planet-fixed Cartesian reference frame, of object type EphemerisRefFrame, associated with this planet.

In addition to the default constructor and the destructor, the BasePlanet class also contains the following member functions.

**register_planet** This member function registers the planet object and its reference frames with the Ephemerides Manager.

- Return: void - no returned value.
- Inout: EphemeridesManager & ephem_manager - the Ephemerides manager object for the current simulation.

**set_name** This member function sets the name of the planetary body being represented.

- Return: void - no returned value.
- In: const std::string & new_name - new name to be applied

**set_alt_inertial** This member function sets the fixed transformation from J2000 to alt_inertial.

- Return: void - no returned value.
- Inout: const double trans[3][3] - the transformation matrix from J2000 to alt_inertial

**set_alt_inertial** This member function sets the fixed transformation from J2000 to alt_inertial by using the celestial and ecliptic poles.

- Return: void - no returned value.
- Inout: const double cp[3] - the celestial pole
- Inout: const double ep[3] - the ecliptic pole

**set_alt_pfix** This member function sets the fixed transformation from pfix to alt_pfix.

- Return: void - no returned value.
- Inout: const double trans[3][3] - the transformation matrix from pfix to alt_pfix

**calculate_alt_pfix** This member function calculates the current transformation from J2000 to alt_pfix using the fixed transformation between pfix and alt_pfix.

- Return: void - no returned value.
- Inout: void - no input value.

### Planet Class Design

The *planet.hh* file contains the Planet Model class used to model ellipsoidal planetary bodies, which is named "Planet". Planet contains the following parameters that mathematically define the shape of an ellipsoidal planetary body:

- grav_body: A pointer to a GravitySource object representing the same planet being defined, which is linked to the proper GravBody when this class's register_model method is correctly invoked,
- r_eq: The mean equatorial radius for this planet, in meters,

- r_pol: The mean polar radius for this planet, in meters,
- e_ellipsoid: The ellipsoid eccentricity (NOT orbit eccentricity) of the planet being defined,
- e_ellip_sq: The square of the planet's ellipsoid eccentricity,
- flat_coeff: The planet's ellipsoid flattening coefficient, termed the reciprocal flattening parameter by Vallado, and
- flat_inv: The inverse of the ellipsoid flattening coefficient.

In addition to the default constructor, destructor, and the set_name function it inherits from BasePlanet, the Planet class also contains the following member functions.

**register_model** This member function invokes the register_planet function inherited from the BasePlanet class. In addition to the actions which that function performs, this function also establishes the connections between the Planet class and the gravity manager.

- Return: void - no returned value.
- Inout: GravitySource & grav_body_in - GravitySource object corresponding to the same planetary body.
- Inout: DynManager & dyn_manager - the Dynamics manager object for the current simulation. Note that DynManager is a class that inherits from EphemeridesManager.

**initialize** This member function initializes the internal shape constant member variables for the ellipsoidal planet being defined.

- return: void - no returned value
- void: function takes no inputs.

Further information about the design of this model can be found in the *Reference Manual* [1].

## 3.4 Inventory

All Planet Model files are located in the directory
`${JEOD_HOME}/models/environment/planet`. Relative to this directory,

- Header and source files are located in the model `include` and `src` subdirectories. Table 3.1 lists the configuration-managed files in these directories.
- Data files are located in the model `data` subdirectory. See table 3.2 for a listing of the configuration-managed files in this directory.

- Documentation files are located in the model `docs` subdirectory. See table 3.3 for a listing of the configuration-managed files in this directory.
- Verification files are located in the model `verif` subdirectory. See table 3.4 for a listing of the configuration-managed files in this directory.

Table 3.1: Source Files

| File Name |
|---|
| include/base_planet.hh |
| include/class_declarations.hh |
| include/planet.hh |
| include/planet_messages.hh |
| src/base_planet.cc |
| src/cmake_file_list.cmake |
| src/planet.cc |
| src/planet_messages.cc |

Table 3.2: Data Files

| File Name |
|---|
| data/include/earth.hh |
| data/include/jupiter.hh |
| data/include/mars.hh |
| data/include/moon.hh |
| data/include/planet_default_data.hh |
| data/include/sun.hh |
| data/src/cmake_file_list.cmake |
| data/src/earth.cc |
| data/src/jupiter.cc |
| data/src/mars.cc |
| data/src/moon.cc |
| data/src/sun.cc |

Table 3.3: Documentation Files

| File Name |
|---|
| docs/planet.pdf |
| docs/refman.pdf |

Continued on next page

Table 3.3: Documentation Files (continued from previous page)

| File Name |
|---|
| docs/tex/makefile |
| docs/tex/planet.bib |
| docs/tex/planet.sty |
| docs/tex/planet.tex |
| docs/tex/planetAbstract.tex |
| docs/tex/planetChapters.tex |
| docs/tex/planetSpec_Inventory.tex |
| docs/tex/planetVV_Metrics.tex |

Table 3.4: Verification Files

| File Name |
|---|
| verif/SIM_PLANET_VERIF/S_define |
| verif/SIM_PLANET_VERIF/S_overrides.mk |
| verif/SIM_PLANET_VERIF/DP_Product/DP_validation.xml |
| verif/SIM_PLANET_VERIF/Log_data/log_planet_verif.py |
| verif/SIM_PLANET_VERIF/SET_test/RUN_initplanet_eellip/input.py |
| verif/SIM_PLANET_VERIF/SET_test/RUN_initplanet_eellipsq/input.py |
| verif/SIM_PLANET_VERIF/SET_test/RUN_initplanet_flatcoeff/input.py |
| verif/SIM_PLANET_VERIF/SET_test/RUN_initplanet_invflat/input.py |
| verif/SIM_PLANET_VERIF/SET_test/RUN_initplanet_rpol/input.py |
| verif/SIM_PLANET_VERIF/SET_test/RUN_initplanet_spherical/input.py |
| verif/SIM_PLANET_VERIF/SET_test_val/RUN_initplanet_eellip/input.py |
| verif/SIM_PLANET_VERIF/SET_test_val/RUN_initplanet_eellip/log_planet_verif.header |
| verif/SIM_PLANET_VERIF/SET_test_val/RUN_initplanet_eellip/log_planet_verif.trk |
| verif/SIM_PLANET_VERIF/SET_test_val/RUN_initplanet_eellipsq/input.py |
| verif/SIM_PLANET_VERIF/SET_test_val/RUN_initplanet_eellipsq/log_planet_verif.header |
| verif/SIM_PLANET_VERIF/SET_test_val/RUN_initplanet_eellipsq/log_planet_verif.trk |

Continued on next page

Table 3.4: Verification Files (continued from previous page)

| File Name |
|---|
| verif/SIM_PLANET_VERIF/SET_test_val/RUN_initplanet_flatcoeff/input.py |
| verif/SIM_PLANET_VERIF/SET_test_val/RUN_initplanet_flatcoeff/log_planet_verif.header |
| verif/SIM_PLANET_VERIF/SET_test_val/RUN_initplanet_flatcoeff/log_planet_verif.trk |
| verif/SIM_PLANET_VERIF/SET_test_val/RUN_initplanet_invflat/input.py |
| verif/SIM_PLANET_VERIF/SET_test_val/RUN_initplanet_invflat/log_planet_verif.header |
| verif/SIM_PLANET_VERIF/SET_test_val/RUN_initplanet_invflat/log_planet_verif.trk |
| verif/SIM_PLANET_VERIF/SET_test_val/RUN_initplanet_rpol/input.py |
| verif/SIM_PLANET_VERIF/SET_test_val/RUN_initplanet_rpol/log_planet_verif.header |
| verif/SIM_PLANET_VERIF/SET_test_val/RUN_initplanet_rpol/log_planet_verif.trk |
| verif/SIM_PLANET_VERIF/SET_test_val/RUN_initplanet_spherical/input.py |
| verif/SIM_PLANET_VERIF/SET_test_val/RUN_initplanet_spherical/log_planet_verif.header |
| verif/SIM_PLANET_VERIF/SET_test_val/RUN_initplanet_spherical/log_planet_verif.trk |
| verif/SIM_PLANET_VERIF/SET_test_val_rh8/RUN_initplanet_eellip/input.py |
| verif/SIM_PLANET_VERIF/SET_test_val_rh8/RUN_initplanet_eellip/log_planet_verif.header |
| verif/SIM_PLANET_VERIF/SET_test_val_rh8/RUN_initplanet_eellip/log_planet_verif.trk |
| verif/SIM_PLANET_VERIF/SET_test_val_rh8/RUN_initplanet_eellipsq/input.py |
| verif/SIM_PLANET_VERIF/SET_test_val_rh8/RUN_initplanet_eellipsq/log_planet_verif.header |
| verif/SIM_PLANET_VERIF/SET_test_val_rh8/RUN_initplanet_eellipsq/log_planet_verif.trk |
| verif/SIM_PLANET_VERIF/SET_test_val_rh8/RUN_initplanet_flatcoeff/input.py |

Continued on next page

Table 3.4: Verification Files (continued from previous page)

| File Name |
|---|
| verif/SIM_PLANET_VERIF/SET_test_val_rh8/RUN_initplanet_flatcoeff/ log_planet_verif.header |
| verif/SIM_PLANET_VERIF/SET_test_val_rh8/RUN_initplanet_flatcoeff/ log_planet_verif.trk |
| verif/SIM_PLANET_VERIF/SET_test_val_rh8/RUN_initplanet_invflat/ input.py |
| verif/SIM_PLANET_VERIF/SET_test_val_rh8/RUN_initplanet_invflat/ log_planet_verif.header |
| verif/SIM_PLANET_VERIF/SET_test_val_rh8/RUN_initplanet_invflat/ log_planet_verif.trk |
| verif/SIM_PLANET_VERIF/SET_test_val_rh8/RUN_initplanet_rpol/ input.py |
| verif/SIM_PLANET_VERIF/SET_test_val_rh8/RUN_initplanet_rpol/log_ planet_verif.header |
| verif/SIM_PLANET_VERIF/SET_test_val_rh8/RUN_initplanet_rpol/log_ planet_verif.trk |
| verif/SIM_PLANET_VERIF/SET_test_val_rh8/RUN_initplanet_ spherical/input.py |
| verif/SIM_PLANET_VERIF/SET_test_val_rh8/RUN_initplanet_ spherical/log_planet_verif.header |
| verif/SIM_PLANET_VERIF/SET_test_val_rh8/RUN_initplanet_ spherical/log_planet_verif.trk |

# Chapter 4

# User Guide

This chapter discusses how to use the Planet Model in a Trick 7 simulation. Usage is treated at three levels of detail: Analysis, Integration, and Extension, each targeted at one of the three main anticipated categories of JEOD v5.2 users.

The Analysis section of the user guide is intended primarily for users of pre-existing simulations. It contains an overview of a typical S_define sim object that implements the Planet class of the Planet Model, but does not discuss how to edit it; the Analysis section also describes how to modify Planet Model variables after the simulation has compiled, such as via the input file. Since the Planet Model is primarily a container model, no discussion of variable logging will be given, as the variables likely to be logged are members of contained objects and thus are covered in other JEOD v5.2 model documentation.

The Integration section of the user guide is intended for simulation developers. It describes the necessary configuration of the Planet Model within an S_define file, and the creation of standard run directories. The Integration section assumes a thorough understanding of the preceding Analysis section of the user guide. Where applicable, the user may be directed to selected portions of the Product Specification (Chapter 3).

The Extension section of the user guide is intended primarily for developers needing to extend the capability of the Planet Model. Such users should have a thorough understanding of how the model is used in the preceding Integration section, and of the model specification (described in Chapter 3).

Note that the Planet Model depends heavily on the Ephemerides Manager or Dynamics Manager model, and the Gravity model; thus any simulation involving implementation of the Planet Model will also require the successful setup of either an EphemeridesManager object or a DynamicsManager object (depending on whether BasePlanet or Planet is being used in the simulation), as well as some type of a GravitySource object (see [6]) if using Planet. Note that the class DynManager inherits from EphemeridesManager; see [2] and [5] respectively for further information on each of these. See [6] for further information on the Gravity model. These model dependencies will be discussed as appropriate in the following sections.

## 4.1 Analysis

The Analysis and the Integration sections will assume, for the purposes of illustration, S_define objects of the following form:

```
sim_object {

   environment/time: TimeManager time_manager;

} time;


sim_object {

   dynamics/dyn_manager:     DynManager             dyn_manager;

} mngr;


sim_object {

   environment/gravity:       SphericalHarmonicsGravitySource   gravity_source
      (environment/gravity/data/earth_GGM02C.d);


   environment/planet:        Planet        planet
      (environment/planet/data/earth.d);


   P_ENV (initialization) environment/planet:
   earth.planet.register_model (
      Inout GravitySource &      grav_source = earth.gravity_source,
      Inout DynManager    &      dyn_manager = mngr.dyn_manager);

   P_BODY (initialization) environment/planet:
   earth.planet.initialize ( );

} earth;
```

Note that this code is only representative of sim objects necessary for the discussion of how to use the Planet Model, and does not hold a complete implementation. For full implementation details on the Time model, please see the JEOD v5.2 Time Representations Model documentation [7]; for full implementation details on the Dynamics Manager model, see the JEOD v5.2 Dynamics Manager Model documentation [2]; and for full implementation details on the Gravity model, please see the JEOD v5.2 Gravity Model documentation [6].

The input files for using the Planet Model are straight-forward. Basically, the user is only required to provide a valid name (i.e., matching exactly to the one used elsewhere in the simulation for the same planet) for the planet being modeled. Further inputs depend on whether the BasePlanet class or the Planet class is being used. For BasePlanet, nothing further is required. For Planet, the desired modeling fidelity affects what the user should provide. For example, setting or leaving all ellipsoidal planetary shape fields as zeros results in the model assuming a perfectly spherical planet with a radius and gravitational constant set equal to the values used in the Gravity Model. If the user desires a non-spherical planet, then at least one of the following should be specified:

- Flattening coefficient
- Inverse flattening coefficient
- Ellipsoid eccentricity
- The square of the ellipsoid eccentricity
- The mean polar radius

The model will calculate the remaining values from the first of these parameters it encounters at runtime based on the listed order.

An example of a non-spherical ellipsoidal planet initialization is the following set of input file commands, which are based on the assumed S_define above:

```
earth.planet.name = "Earth";
earth.planet.r_eq {km} = 6378.137;
earth.e_ellipsoid = 0.081819221;
```

Note that this set of example commands details initialization of the Planet Model as a representation of a non-spherical Earth using the ellipsoid eccentricity parameter to define the describing set of shape constants.

## 4.2 Integration

This section describes the process of implementing the Planet Model in a Trick simulation, and will use the same example S_define found in the Analysis section for illustration. Please again note that this code is only representative of sim objects necessary for this discussion, and does not describe a complete implementation. For full implementation details on the Time model, please see the JEOD v5.2 Time Representations Model documentation [7]. For full implementation details on the Dynamics Manager model, please see the JEOD v5.2 Dynamics Manager Model documentation [2]. For full implementation details on the Gravity model, please see the JEOD v5.2 Gravity Model documentation [6].

To successfully integrate the Planet Model into a Trick simulation, the S_define must contain instantiations of the following objects.

If using BasePlanet:

- A TimeManager object,
- An EphemeridesManager object, with which the Planet Model must register itself for proper operation, and
- A BasePlanet object.

If using Planet:

- A TimeManager object,
- A DynManager, with which the Planet Model must register itself for proper operation,
- A GravitySource object modeling the gravity of the same planet, to which the Planet Model will link itself at registration with the DynManager, and
- A Planet object.

As noted above, all of the non-Planet Model classes must be correctly initialized and set up for successful simulation operation, though discussion of such setup is outside of the scope of this document; documentation specific to each of the applicable models should be consulted for those details.

To use the Planet Model for modeling ellipsoidal planetary bodies, the main simulation object is the Planet class, which is instantiated in the example code above via the lines:

```
environment/planet:        Planet        planet
   (environment/planet/data/earth.d);
```

Note that this code includes a call to the "earth.d" default data file as a way of populating some of the model parameters with initial values. In addition to this file, there are also default data files shipped with the JEOD v5.2 Planet Model containing reference parameters for the sun, moon, mars, and Jupiter for the integrator's convenience, should the need for them arise.

After setting up the instantiation of a Planet object, the next step is to register the model with the Dynamics Manager for the simulation. This is done by invoking the Planet Model *register_model* member function as an initialization class job, as shown in the example above and reproduced here:

```
P_ENV (initialization) environment/planet:
earth.planet.register_model (
   Inout GravitySource &      grav_source  = earth.gravity_source,
   Inout DynManager    &      dyn_manager  = mngr.dyn_manager);
```

Note that this call requires a GravitySource and a DynManager as inputs, illustrating the previously mentioned dependencies of the Planet Model on other models. This registration step is the one in which the Planet Model registers itself with the Dynamics Manager, connects itself with the GravitySource object associated with the same planet (they must be named *identically* or the

linkage will fail and the simulation will terminate), and registers its internal planet-fixed and inertial reference frames with the DynManager.

The final step required to integrate the Planet Model into a Trick simulation for modeling ellipsoidal planetary bodies is to initialize the values of its planet shape constants using the Planet Model *initialize* member function:

```
P_BODY (initialization) environment/planet:
earth.planet.initialize ( );
```

This too is intended to be used as an initialization class job. It reads user inputs (via default data, input file, modified data, or some combination of the three) and calculates the shape constants for the modeled planet accordingly. Further information on providing the Planet Model with inputs can be found in the Analysis section of this chapter.

No further steps are required for integration of the Planet class of the Planet Model into a Trick simulation. As the model's purpose is primarily to serve as a container object with which other models can interface, the Planet Model itself performs no further calculations beyond the startup ones just described.

## 4.3 Extension

The Planet Model itself contains an example of how to extend the model, since the Planet class that is part of it derives from and extends the fundamental BasePlanet class. Thus the extender should use the Planet class as a guide for creating any new types of planetary body containers that should be needed.

# Chapter 5

# Verification and Validation

## 5.1 Verification

*Inspection planet_1: Top-level Inspection*

This document structure, the code, and associated files have been inspected, and together completely satisfy requirement planet_1.

*Inspection planet_2: Data Requirements Inspection*

By inspection, the data structures of the Planet Model completely satisfy requirements planet_2 and planet_3.

*Inspection planet_3: Functional Requirements Inspection*

By inspection, the as-written function code of the Planet Model satisfies requirements planet_4, planet_5, and planet_6.

## 5.2 Validation

For each test case, a Trick simulation was run with an input file tailored to that case. Each input file has its own associated run directory, named appropriately for its test case. These run directories are contained in the SET_test sub-directory of SIM_PLANET_VERIF.

*Test planet_1: Initialize From Ellipsoid Eccentricity*

**Purpose:**

This test case is designed to examine the ability of the Planet Model to register itself with

the Ephemerides Manager, and to test its ability to initialize the ellipsoidal planet shape constants given input in the form of ellipsoid eccentricity.

**Run directory:**
RUN_initplanet_eellip

**Requirements:**
By passing this test, this model satisfies requirements planet_4 and planet_5, and partially satisfies requirement planet_6.

**Procedure:**
Simple successful execution of a simulation containing a properly implemented instance of the Planet Model is sufficient to fully demonstrate the registration capability of the model. Implementing the simulation with an ellipsoid-specific instance will prove both the base and derived capabilities for the model, since the derived class inherits, and uses, all the capabilities of the base class.

To test the ability of the Planet Model to initialize the full JEOD v5.2 set of planet shape constants given ellipsoid eccentricity, values for Earth's ellipsoid eccentricity and mean equatorial radius were obtained from [8] and used to initialize the model. The expected values of the other parameters were then analytically determined and compared against the resulting Planet Model output to determine its calculation accuracy.

**Results:**
Table 5.1 shows the values used for each input parameter, and both the simulation and analytical results for this test. Note that since the simulation completed the run successfully, requirements planet_4 and planet_5 are demonstrated to be satisfied by this test.

| Input | Value | Output | Model Result | Analytical Result |
|---|---|---|---|---|
| $e$ | 0.081819221 | $e^2$ | 0.00669438 | 0.00669438 |
| $r_{eq}$ | 6378.137 $km$ | $f$ | 0.00335281 | 0.00335281 |
| | | $f^{-1}$ | 298.257 | 298.257 |
| | | $r_{eq}$ | 6378.137 $km$ | 6378.137 $km$ |
| | | $r_{pol}$ | 6356.752 $km$ | 6356.752 $km$ |

Table 5.1: Results for Ellipsoid Eccentricity Initialization

As this table shows, the model achieved the same results for the shape parameters as were obtained analytically for the listed inputs. (Note that the analytical results were truncated to the same number of significant digits as returned by the simulation to yield a consistent basis for comparison.) Thus, the model passed this test, partially satisfying requirement planet_6.

*Test planet_2: Initialize From Squared Ellipsoid Eccentricity*

**Purpose:**
This test case is designed to examine the ability of the Planet Model to initialize the planet shape constants given input in the form of squared ellipsoid eccentricity.

**Run directory:**
RUN_initplanet_eellipsq

**Requirements:**
By passing this test, this model partially satisfies requirement planet_6.

**Procedure:**
To test the ability of the Planet Model to initialize the full JEOD v5.2 set of planet shape constants given squared ellipsoid eccentricity, values for Earth's squared ellipsoid eccentricity and mean equatorial radius were developed from [8] and used to initialize the model. The expected values of the other parameters were then analytically determined and compared against the resulting Planet Model output to determine its calculation accuracy.

**Results:**
Table 5.2 shows the values used for each input parameter, and both the simulation and analytical results for this test.

| Input | Value | Output | Model Result | Analytical Result |
|---|---|---|---|---|
| $e^2$ | 0.006694384925 | $e$ | 0.0818192 | 0.0818192 |
| $r_{eq}$ | 6378.137 $km$ | $f$ | 0.00335281 | 0.00335281 |
| | | $f^{-1}$ | 298.257 | 298.257 |
| | | $r_{eq}$ | 6378.137 $km$ | 6378.137 $km$ |
| | | $r_{pol}$ | 6356.752 $km$ | 6356.752 $km$ |

Table 5.2: Results for Squared Ellipsoid Eccentricity Initialization

As this table shows, the model achieved the same results for the shape parameters as were obtained analytically for the listed inputs. (Note that the analytical results were truncated to the same number of significant digits as returned by the simulation to yield a consistent basis for comparison.) Thus, the model passed this test, partially satisfying requirement planet_6.

*Test planet_3: Initialize From Flattening Coefficient*

**Purpose:**
This test case is designed to examine the ability of the Planet Model to initialize the planet shape constants given input in the form of flattening coefficient.

**Run directory:**
RUN_initplanet_flatcoeff

**Requirements:**
By passing this test, this model partially satisfies requirement planet_6.

**Procedure:**
To test the ability of the Planet Model to initialize the full JEOD v5.2 set of planet shape constants given flattening coefficient, values for Earth's flattening coefficient and mean equatorial radius were obtained from [8] and used to initialize the model. The expected values of

the other parameters were then analytically determined and compared against the resulting Planet Model output to determine its calculation accuracy.

**Results:**

Table 5.3 shows the values used for each input parameter, and both the simulation and analytical results for this test.

| Input | Value | Output | Model Result | Analytical Result |
|---|---|---|---|---|
| $f$ | 0.003352813 | $e$ | 0.0818192 | 0.0818192 |
| $r_{eq}$ | 6378.137 $km$ | $e^2$ | 0.00669438 | 0.00669438 |
| | | $f^{-1}$ | 298.257 | 298.257 |
| | | $r_{eq}$ | 6378.137 $km$ | 6378.137 $km$ |
| | | $r_{pol}$ | 6356.752 $km$ | 6356.752 $km$ |

Table 5.3: Results for Flattening Coefficient Initialization

As this table shows, the model achieved the same results for the shape parameters as were obtained analytically for the listed inputs. (Note that the analytical results were truncated to the same number of significant digits as returned by the simulation to yield a consistent basis for comparison.) Thus, the model passed this test, partially satisfying requirement planet_6.

*Test planet_4: Initialize From Inverse Flattening Coefficient*

**Purpose:**

This test case is designed to examine the ability of the Planet Model to initialize the planet shape constants given input in the form of inverse flattening coefficient.

**Run directory:**

RUN_initplanet_invflat

**Requirements:**

By passing this test, this model partially satisfies requirement planet_6.

**Procedure:**

To test the ability of the Planet Model to initialize the full JEOD v5.2 set of planet shape constants given inverse flattening coefficient, values for Earth's inverse flattening coefficient and mean equatorial radius were developed from [8] and used to initialize the model. The expected values of the other parameters were then analytically determined and compared against the resulting Planet Model output to determine its calculation accuracy.

**Results:**

Table 5.4 shows the values used for each input parameter, and both the simulation and analytical results for this test.

As this table shows, the model achieved the same results for the shape parameters as were obtained analytically for the listed inputs. (Note that the analytical results were truncated to the same number of significant digits as returned by the simulation to yield a consistent basis for comparison.) Thus, the model passed this test, partially satisfying requirement planet_6.

| Input | Value | Output | Model Result | Analytical Result |
|---|---|---|---|---|
| $f^{-1}$ | 298.2570 | $e$ | 0.0818192 | 0.0818192 |
| $r_{eq}$ | 6378.137 $km$ | $e^2$ | 0.00669438 | 0.00669438 |
| | | $f$ | 0.00335281 | 0.00335281 |
| | | $r_{eq}$ | 6378.137 $km$ | 6378.137 $km$ |
| | | $r_{pol}$ | 6356.752 $km$ | 6356.752 $km$ |

Table 5.4: Results for Inverse Flattening Coefficient Initialization

*Test planet_5: Initialize From Mean Polar Radius*

**Purpose:**

This test case is designed to examine the ability of the Planet Model to initialize the planet shape constants given input in the form of mean polar radius.

**Run directory:**

RUN_initplanet_rpol

**Requirements:**

By passing this test, this model partially satisfies requirement planet_6.

**Procedure:**

To test the ability of the Planet Model to initialize the full JEOD v5.2 set of planet shape constants given mean polar radius, values for Earth's mean polar radius and mean equatorial radius were developed from [8] and used to initialize the model. The expected values of the other parameters were then analytically determined and compared against the resulting Planet Model output to determine its calculation accuracy.

**Results:**

Table 5.5 shows the values used for each input parameter, and both the simulation and analytical results for this test.

| Input | Value | Output | Model Result | Analytical Result |
|---|---|---|---|---|
| $r_{pol}$ | 6356.75160 | $e$ | 0.0818206 | 0.0818206 |
| $r_{eq}$ | 6378.137 $km$ | $e^2$ | 0.0066946 | 0.0066946 |
| | | $f$ | 0.00335292 | 0.00335292 |
| | | $f^{-1}$ | 298.247 | 298.247 |
| | | $r_{eq}$ | 6378.137 $km$ | 6378.137 $km$ |

Table 5.5: Results for Mean Polar Radius Initialization

As this table shows, the model achieved the same results for the shape parameters as were obtained analytically for the listed inputs. (Note that the analytical results were truncated to the same number of significant digits as returned by the simulation to yield a consistent basis for comparison.) Thus, the model passed this test, partially satisfying requirement planet_6.

*Test planet_6: Initialize As Spherical Planet*

**Purpose:**

This test case is designed to examine the ability of the Planet Model to initialize the planet shape constants given only zeros for input.

**Run directory:**

RUN_initplanet_spherical

**Requirements:**

By passing this test, this model partially satisfies requirement planet_6.

**Procedure:**

To test the ability of the Planet Model to initialize the full JEOD v5.2 set of planet shape constants given only zeros as input, all shape parameters were initialized to zeros at the start of the run. The resulting Planet Model output was then examined to determine whether the model did indeed properly initialize as a spherical planet with mean equatorial radius equal to the value obtained from its associated GravitySource object, as was expected.

**Results:**

Table 5.6 shows the values used for each input parameter, and both the simulation and analytical results for this test.

| Parameter | Input Value | Model Result | Analytical Result |
|---|---|---|---|
| $e$ | 0.0 | 0.0 | 0.0 |
| $e^2$ | 0.0 | 0.0 | 0.0 |
| $f$ | 0.0 | 0.0 | 0.0 |
| $f^{-1}$ | 0.0 | 0.0 | 0.0 |
| $r_{eq}$ | 0.0 $km$ | 6378.137 $km$ | 6378.137 $km$ |
| $r_{pol}$ | 0.0 $km$ | 6378.137 $km$ | 6378.137 $km$ |

Table 5.6: Results for Spherical Planet Initialization

As this table shows, the model performed as expected for the given case; it obtained the mean equatorial radius from the associated GravitySource and used that value for both $r_{eq}$ and $r_{pol}$, leaving everything else zeros to result in a representation of a spherical earth. (Note that the analytical results were truncated to the same number of significant digits as returned by the simulation to yield a consistent basis for comparison.) Thus, the model passed this test, partially satisfying requirement planet_6.

## 5.3 Requirements Traceability

Table 5.7: Requirements Traceability

| **Requirement** | **Inspection and Testing** |
|---|---|
| planet_1 - Top-level Requirement | Insp. planet_1 |
| planet_2 - Basic Data Encapsulation | Insp. planet_2 |
| planet_3 - Ellipsoidal Data Encapsulation | Insp. planet_2 |
| planet_4 - Basic Model Registration | Insp. planet_3<br>Test planet_1 |
| planet_5 - Ellipsoidal Model Registration | Insp. planet_3<br>Test planet_1 |
| planet_6 - Initialize Ellipsoidal Constants | Insp. planet_3<br>Test planet_1<br>Test planet_2<br>Test planet_3<br>Test planet_4<br>Test planet_5<br>Test planet_6 |

## 5.4 Metrics

### 5.4.1 Code Metrics

Table 5.8 presents coarse metrics on the source files that comprise the model.

Table 5.8: Coarse Metrics

| | Number of Lines | | | |
|---|---|---|---|---|
| **File Name** | **Blank** | **Comment** | **Code** | **Total** |
| include/base_planet.hh | 30 | 111 | 42 | 183 |
| include/class_declarations.hh | 11 | 61 | 8 | 80 |
| include/planet.hh | 27 | 96 | 30 | 153 |
| include/planet_messages.hh | 19 | 81 | 18 | 118 |
| src/base_planet.cc | 24 | 69 | 74 | 167 |
| src/cmake_file_list.cmake | 2 | 0 | 9 | 11 |
| src/planet.cc | 27 | 56 | 152 | 235 |
| src/planet_messages.cc | 15 | 30 | 8 | 53 |
| data/include/earth.hh | 7 | 44 | 12 | 63 |
| data/include/jupiter.hh | 7 | 44 | 12 | 63 |
| data/include/mars.hh | 7 | 44 | 12 | 63 |
| data/include/moon.hh | 7 | 44 | 12 | 63 |
| data/include/planet_default_data.hh | 6 | 41 | 13 | 60 |
| data/include/sun.hh | 7 | 44 | 12 | 63 |
| data/src/cmake_file_list.cmake | 2 | 0 | 11 | 13 |
| data/src/earth.cc | 10 | 14 | 15 | 39 |
| data/src/jupiter.cc | 10 | 14 | 15 | 39 |
| data/src/mars.cc | 13 | 20 | 18 | 51 |
| data/src/moon.cc | 13 | 23 | 21 | 57 |
| data/src/sun.cc | 11 | 17 | 15 | 43 |
| **Total** | 255 | 853 | 509 | 1617 |

Table 5.9 presents the extended cyclomatic complexity (ECC) of the methods defined in the model.

Table 5.9: Cyclomatic Complexity

| Method | File | Line | ECC |
|---|---|---|---|
| jeod::BasePlanet::std::set_ name (const std::string & name_in) | include/base_planet.hh | 133 | 1 |
| jeod::BasePlanet::set_alt_ inertial (const double trans[3][3]) | src/base_planet.cc | 45 | 4 |
| jeod::BasePlanet::set_alt_ inertial (const double cp[3], const double ep[3]) | src/base_planet.cc | 71 | 1 |
| jeod::BasePlanet::set_alt_pfix (const double trans[3][3]) | src/base_planet.cc | 93 | 2 |
| jeod::BasePlanet::calculate_ alt_pfix () | src/base_planet.cc | 111 | 1 |
| jeod::BasePlanet::register_ planet (BaseEphemerides Manager & ephem_ manager) | src/base_planet.cc | 126 | 2 |
| jeod::Planet::register_model ( GravitySource & grav_ source_in, BaseDynManager & dyn_manager) | src/planet.cc | 44 | 2 |
| jeod::Planet::initialize () | src/planet.cc | 73 | 18 |

# Bibliography


[1] Generated by doxygen. *Planet Model Reference Manual*. National Aeronautics and Space Administration, Johnson Space Center, Automation, Robotics & Simulation Division, Simulation and Graphics Branch, 2101 NASA Parkway, Houston, Texas, 77058, September 2024.

[2] Hammen, D. Dynamics Manager Model. Technical Report JSC-61777-dynamics/dyn_manager, NASA, Johnson Space Center, Houston, Texas, September 2024.

[3] Jackson, A., Thebeau, C. JSC Engineering Orbital Dynamics. Technical Report JSC-61777-docs, NASA, Johnson Space Center, Houston, Texas, September 2024.

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

[5] Thompson, B. Ephemerides Model. Technical Report JSC-61777-environment/ephemerides, NASA, Johnson Space Center, Houston, Texas, September 2024.

[6] Thompson, B. and Morris., J. Gravity Model. Technical Report JSC-61777-environment/gravity, NASA, Johnson Space Center, Houston, Texas, September 2024.

[7] Turner, G. Time Model. Technical Report JSC-61777-environment/time, NASA, Johnson Space Center, Houston, Texas, September 2024.

[8] David A. Vallado with contributions by Wayne D. McClain. *Fundamentals of Astrodynamics and Applications, Second Edition*. Microcosm Press, El Segundo, CA, 2004.