JSC-61777-models/environment/gravity

JSC Engineering Orbital Dynamics Gravity Model

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

Package Release JEOD v5.1

Document Revision 1.3
July 2023

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

JSC-61777-models/environment/gravity

JSC Engineering Orbital Dynamics Gravity Model

Document Revision 1.3
July 2023

Jeff Morris, Blair Thompson

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 Gravity Model provides a framework for representation of the gravity effects of planetary bodies. The basic model is generic and extensible to allow implementation of any common type of gravity field representation method. The basic model also provides the simple point-mass gravitational attraction calculations common to all Newtonian gravity representations to reduce redundant effort when extending the model.

An extension of the basic model that uses a recursive, stable, and singularity– free spherical harmonics algorithm to model gravity fields is also provided as part of JEOD. This algorithm computes the gravitational acceleration acting on an object due to the gravity of planetary bodies; it can also include the effects of solid body tides in those calculations if desired. The model also outputs the gradient of the gravitational acceleration, which can be used for calculating gravitational torque or other microgravity effects.

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

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Purpose and Objectives of the Gravity Model

The Gravity Model provides a framework for representation of the gravity effects of planetary bodies. The basic model is generic and extensible to allow implementation of any common type of gravity field representation method. The basic model also provides the simple point-mass gravitational attraction calculations common to all Newtonian gravity representations to reduce redundant effort when extending the model.

For increased computational efficiency, the basic gravity framework provided allows the gravitational calculations to be controlled for each simulated object independent of other objects. For example, the same planetary body can be specified to have simple point-mass gravitational effects on one object, while simultaneously being specified to provide high-order effects on another. Gravitational attraction calculations can also be controlled per planetary body for each simulated object, so that some planets can be acting on an object as point-masses while others are simultaneously acting on the same object with non-point-mass effects.

An extension of the basic model that uses a recursive, stable, and singularity– free spherical harmonics algorithm to model gravity fields is also provided as part of JEOD. This algorithm computes the gravitational acceleration acting on an object due to the gravity of planetary bodies; it can also include the effects of solid body tides in those calculations if desired. The model also outputs the gradient of the gravitational acceleration, which can be used for calculating gravitational torque or other microgravity effects.

A basic set of data files has been included along with the provided spherical harmonics gravity implementation. These data files include the spherical harmonic gravity coefficients for Earth, the Moon, and other planetary bodies commonly included in simulations using JEOD. Model users can easily add additional sets of gravity coefficients for these or any other planetary body as required for their purposes.

1.2 Context within JEOD

The following document is parent to this document:

- *JSC Engineering Orbital Dynamics* [3]

The Gravity Model forms a component of the environment suite of models within JEOD v5.1. It is located at models/environment/gravity.

1.3 Document History

Author	Date	Revision	Description
Mitch Hollander	June 2021	4.0	Update nomenclature for JEOD 4.0 release
Jeff Morris	February 2012	1.3	Update for model generalization
Jeff Morris	October 2010	1.2	Expanded tidal gravity discussion; metrics
Blair Thompson	March 2010	1.1	Separate gradient degree/order controls
Blair Thompson	November 2009	1.0	Initial Version

1.4 Document Organization

This document is formatted in accordance with the NASA Software Engineering Requirements Standard [8].

The document comprises chapters organized as follows:

Chapter 1: Introduction -This introduction describes the objective and purpose of the Gravity Model.

Chapter 2: Product Requirements -The requirements chapter describes the requirements on the Gravity Model.

Chapter 3: Product Specification -The specification chapter describes the architecture and design of the Gravity Model.

Chapter 4: User Guide -The user guide chapter describes how to use the Gravity Model.

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 Gravity Model.

Chapter 2

Product Requirements

This chapter describes the requirements for the Gravity Model.

Requirement Gravity_1: Top-level

Requirement:
This model shall meet the JEOD project requirements specified in the JEOD v5.1 top-level document.

Rationale:
This model shall, at a minimum, meet all external and internal requirements applied to the JEOD v5.1 release.

Verification:
Inspection

Requirement Gravity_2: Extensibility

Requirement:
The Gravity Model shall be extensible.

Rationale:
Extensibility allows custom, or multiple, methods of representing gravitational effects to be used in a simulation that employs the Gravity Model.

Verification:
Inspection

2.1 Data Requirements

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

Requirement Gravity_3: Generic Model Data

Requirement:

The Gravity Model shall provide a set of data fields denoting active status and other generic settings to facilitate use of the model in simulations.

Rationale:

The primary function of the basic Gravity Model framework is to provide these generic settings and interfaces that apply to all gravity representation methods.

Verification:

Inspection

Requirement Gravity_4: Spherical Harmonics Gravitational Data

Requirement:

In the provided spherical harmonics gravity model extension, gravitational potential shall be modeled by spherical harmonic expansion. The data for the extension shall consist of unit-less gravity coefficients and associated constants.

4.1 Gravity coefficients shall be fully normalized (cf. [2, 6, 14]).

4.2 All zero-order S coefficients shall equal zero ($S_{n0} = 0$).

4.3 Coefficient C_{00} *shall be exactly one* ($C_{00} = 1$), *and all first-degree coefficients shall be exactly zero* ($C_{10} = C_{11} = S_{11} = 0$).

Rationale:

The functions in the provided spherical harmonics gravity model extension utilize fully normalized gravity coefficients. The functions begin computing with degree two ($n = 2$) coefficients. The degree-zero and degree-one coefficients are assumed to be the values listed above.

Verification:

Inspection

2.2 Functional Requirements

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

Requirement Gravity_5: Generic Model Functionality

Requirement:

The Gravity Model shall provide basic functionality required to enable all extensions of the model to properly interface with JEOD.

5.1 The Gravity Model shall provide a set of generic initialization functions.

5.2 The Gravity Model shall provide a set of generic update functions.

5.3 The Gravity Model shall provide simple point-mass gravitational calculations.

Rationale:

The primary function of the basic Gravity Model framework is to provide these generic functional interfaces that apply to all gravity representation methods.

Verification:

Inspection for subrequirements 1 and 2, Test for subrequirement 3.

Requirement Gravity_6: Spherical Harmonics Gravitational Acceleration

Requirement:

The model shall compute the inertial acceleration vector due to a planetary body at a specified position external to the body. The acceleration vector shall be computed in the planet-fixed coordinate frame and transformed into inertial coordinates. The model must be non-singular for polar orbits and stable for high degree and order gravity fields.

Rationale:

This describes the essential functionality of the spherical harmonics gravity model extension. The acceleration due to gravity will be transformed into inertial coordinates for integration in the spacecraft equations of motion.

Verification:

Test

Requirement Gravity_7: Spherical Harmonics Gravity Gradient

Requirement:

The model shall compute the gradient of the inertial gravity acceleration vector due to a planetary body at a specified position external to the body. The model shall provide the capability to specify the gravity gradient degree and order independently from the acceleration degree and order.

Rationale:

The gravity gradient is required for computing gravity torque and other microgravity purposes.

Verification:

Test

Requirement Gravity_8: Spherical Harmonics Solid Body Tides

Requirement:

This model shall include the first-order effects of solid body tides acting on a planetary body due to any other planetary body (i.e., moon acting on Earth, sun acting on moon, etc.)

Rationale:

Solid body tidal effects are required for high precision orbit modeling.

Verification:

Test

Chapter 3

Product Specification

3.1 Conceptual Design

3.1.1 Basic Model

The Gravity Model provides a framework for representation of the gravity effects of planetary bodies. The model is generic and extensible to allow implementation of any common type of gravity field representation method. The basic model also provides the simple point-mass gravitational attraction calculations common to all Newtonian gravity representations to reduce redundant effort when extending the model.

For increased computational efficiency, the basic gravity framework provided allows the gravitational calculations to be controlled for each simulated object independent of other objects. For example, the same planetary body can be specified to have simple point-mass gravitational effects on one object, while simultaneously being specified to provide high-order effects on another. Gravitational attraction calculations can also be controlled per planetary body for each simulated object, so that some planets can be acting on an object as point-masses while others are simultaneously acting on the same object with non-point-mass effects.

The generic gravity model can be easily extended to employ multiple types of gravity representation methods simultaneously, not only running in the same simulation but also acting upon the same object(s) at the same time. One such extension of the generic model has been provided as part of the JEOD package in the form of the spherical harmonics gravity implementation; this model extension will be the primary focus of much of this chapter.

3.1.2 Spherical Harmonics Implementation

The spherical harmonics gravity implementation is an extension of the basic gravity model that uses a recursive, stable, and singularity-free spherical harmonics algorithm to model gravity fields. This implementation can also include the effects of solid body tides in those calculations if desired. The model additionally outputs the gradient of the gravitational acceleration, which can be used for calculating gravitational torque or other microgravity effects.

The spherical harmonics implementation must be provided with a set of gravity coefficients corresponding to a particular gravity field to perform its calculations. These coefficients must be in fully normalized form. However, so long as the given file is properly formatted and the coefficients are normalized, any set of spherical harmonics gravity coefficients may be used with the extended model; no code modification is required.

The gravitational acceleration and gradient calculations in this implementation are adapted from the Ada code of Gottleib [1]. The solid body tide model is the first-order model of the International Earth Rotation and Reference Systems Service (IERS) [5].

3.2 Mathematical Formulations

3.2.1 Coordinate Systems

Gravitational fields are typically developed and expressed in spherical[1], planet-fixed coordinates. The symbols (r, λ, ϕ) are the spherical coordinates radius, longitude, and latitude (Figure 3.1). The

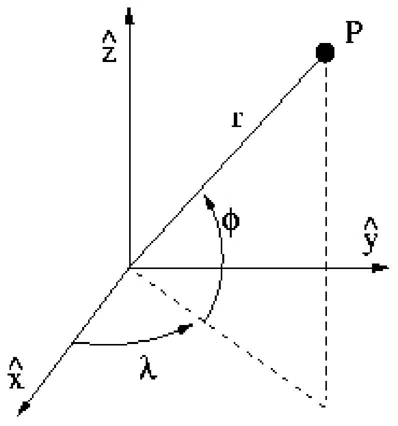

Figure 3.1: Spherical coordinates.

coordinate frames in which the fields are developed are typically planet-fixed, that is, fixed with respect to the planetary body and, therefore, are generally non-inertial.

Note: the spherical harmonics model extension is based on the Gottlieb [1] formulation, which uses rectangular (i.e., Cartesian) coordinates (x, y, z). The conversion from spherical to rectangular coordinates is inherent in the extension's routines.

Typcially, the equations of motion of a spacecraft are expressed in inertial coordinates. Therefore, a transformation is usually required to express the gravitational acceleration and gradient in inertial coordinates. This transformation is performed by the Gravity Model but the required transformation matrix is computed externally to the model. In general, the transformation matrix incorporates effects such as planetary rotation, nutation, and precession; it is often referred to as the *RNP matrix*[2]. For high precision work it is important to use the planet-fixed coordinate system

[1]Here the term *spherical* means coordinates base on a spherical body as opposed to an oblate elliptical body (i.e., geocentric vs. geodetic coordinates).

[2]Other effects, such as polar motion, can be included in the RNP matrix.

of the gravity field that is being used for each gravitational body.

3.2.2 Spherical Harmonics Gravitational Potential

A gravitational potential field model can be used to compute acceleration (i.e., force per mass) acting on an object. Most large planetary bodies are predominately spherical in shape, but are not perfect spheres of uniform mass distribution. The potential due to nonspherical bodies must be considered for precision orbit modeling; spherical harmonics have been found to be a useful way to model these non-ideal effects.

Gravitational potential can be expressed in spherical coordinates as [6, 14]

$$U(r,\lambda,\phi) = \frac{\mu}{r}\sum_{n=0}^{\infty}\sum_{m=0}^{n}\left(\frac{R}{r}\right)^n \bar{P}_{n,m}(sin\phi)(\bar{C}_{n,m}\cos m\lambda + \bar{S}_{n,m}\sin m\lambda) \tag{3.1}$$

where (r,λ,ϕ) are the spherical coordinates radius, longitude, and latitude (Figure 3.1), μ is the gravitational parameter ($\mu = GM$); n and m are model degree and order; R is the mean equatorial radius of the gravitational body; $\bar{P}_{n,m}$ is the fully normalized associated Legendre function of degree n and order m; and $\bar{C}_{nm}$ and $\bar{S}_{nm}$ are fully normalized unit-less gravity coefficients that are related to the mass distribution of the body [4, 6]. The overhead bar denotes a coefficient that is fully normalized using the relationship

$$\left\{\begin{array}{c}\bar{C}_{nm}\\ \bar{S}_{nm}\end{array}\right\} = \sqrt{\frac{(n+m)!}{(2-\delta_{0m})(2n+1)(n-m)!}}\left\{\begin{array}{c}C_{nm}\\ S_{nm}\end{array}\right\} \tag{3.2}$$

where $\delta_{0m} = 1$ when $m = 0$, and $\delta_{0m} = 0$ when $m \neq 0$.

The zero-degree term (C_{00}) represents the spherical surface upon which the higher degree terms are imposed. It is unity by definition ($C_{00} = 1$). All $\bar{S}_{nm}$ terms vanish for $m = 0$ because the sine term in equation 3.1 vanishes when $m = 0$ ($\sin 0 = 0$). It can be shown [6, 13] that the location of the center of mass $(\bar{x},\bar{y},\bar{z})$ of the gravitational body is related to the un-normalized first degree coefficients by

$$\begin{aligned}\bar{x} &= RC_{11}\\ \bar{y} &= RS_{11}\\ \bar{z} &= RC_{10}\end{aligned} \tag{3.3}$$

The normal practice is to define the spherical coordinate system such that the origin is located at the center of mass. Therefore, the first degree terms C_{10}, C_{11}, and S_{11} all equal zero. With these definitions of the zero-degree and first-degree terms, the spherical harmonic model becomes

$$U(r,\lambda,\phi) = \frac{\mu}{r}\left[1 + \sum_{n=2}^{\infty}\sum_{m=0}^{n}\left(\frac{R}{r}\right)^n \bar{P}_{n,m}(sin\phi)(\bar{C}_{n,m}\cos m\lambda + \bar{S}_{n,m}\sin m\lambda)\right] \tag{3.4}$$

For a large continuous body of mass (i.e., a planet) the gravity coefficients are related to the mass distribution by

$$\begin{aligned}C_{nm} &= \frac{2-\delta_{0m}}{M}\frac{(n-m)!}{(n+m)!}\int\left(\frac{s}{R}\right)^n P_{nm}\sin(\phi')\cos(m\lambda')\rho(s)d^3s\\ S_{nm} &= \frac{2-\delta_{0m}}{M}\frac{(n-m)!}{(n+m)!}\int\left(\frac{s}{R}\right)^n P_{nm}\sin(\phi')\sin(m\lambda')\rho(s)d^3s\end{aligned} \tag{3.5}$$

where (s, λ', ϕ') are the spherical coordinates of each infinitesimal particle of mass, $\rho(s)$ is density as a function of position within the body of mass, and d^3s indicates volume integration over the entire body of mass [6].

3.2.3 Gravitational Acceleration

The acceleration in rectangular coordinates due to gravity is found by computing the gradient of the gravitational potential.

$$\bar{a} = \nabla U \tag{3.6}$$

To satisfy the functional requirement that the spherical harmonics extension of the Gravity Model be non-singular and stable, acceleration is efficiently computed in rectangular coordinates using recursion equations for the Legendre polynomials. The technique used is that of Gottlieb as detailed in Fast Gravity, Gravity Partials, Normalized Gravity, Gravity Gradient Torque, and Magnetic Field: Derivation, Code, and Data [1].

Simple point-mass gravitational acceleration is calculated by the base Gravity Model using the standard two-body equation for gravitational attraction.

3.2.4 Gravity Gradient

The gravity gradient is a 3x3 Jacobian (matrix) that is the partial derivative of the inertial gravitational acceleration vector with respect to the inertial position vector (rectangular coordinates).

$$\nabla \bar{a} = \frac{\partial \bar{a}}{\partial \bar{r}} = \begin{bmatrix} \partial a_x/\partial x & \partial a_x/\partial y & \partial a_x/\partial z \\ \partial a_y/\partial x & \partial a_y/\partial y & \partial a_y/\partial z \\ \partial a_z/\partial x & \partial a_z/\partial y & \partial a_z/\partial z \end{bmatrix} \tag{3.7}$$

The spherical harmonics extension of the Gravity Model uses Gottlieb's method for computing gravity gradient [1]. Gottlieb's method computes the gradient due to the full gravity field. It does not make any simplifying assumptions such as a spherical or simple oblate gravitational body.

The base Gravity Model computes spherical gravity gradient using the analytic solutions for the partial derivatives of the simple point-mass acceleration.

The trace of the gravity gradient matrix (sum of the diagonal terms) is the Laplacian of the gravitational potential and is equal to zero for any gravity field.

$$\nabla^2 U = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} = \frac{\partial a_x}{\partial x} + \frac{\partial a_y}{\partial y} + \frac{\partial a_z}{\partial z} = 0 \tag{3.8}$$

where U is the gravitational potential (equation 3.4).

The gravity gradient can be used to compute gravity torque. Consult the JEOD Gravity Torque Model document for detail [11]. Gravity gradient can also be used for microgravity and other similar purposes.

3.2.5 Solid Body Tides

In general, the large planetary bodies of the solar system are not perfectly rigid. They deform in response to the gravitational forces of other large bodies acting upon them, resulting in temporal variations of mass distribution known as *tides*. Deformation results in a change in the mass distribution of the planetary body which, in turn, causes a change in its spherical harmonics gravity coefficients (equations 3.5). The term *solid body tide* refers to the large-scale tidal response of the entire planetary body, which is different from the tidal response of oceans and atmospheres.

The solid body tide calculations in the spherical harmonics extension of the Gravity Model is the first-order model of the International Earth Rotation and Reference System Service (IERS) Conventions 2003 [5]; note that solid body tides are not calculated by the base Gravity Model since it models planetary bodies as simple point-masses that are not capable of deformation. The model is expressed as a change in the normalized gravity coefficients as

$$\Delta\bar{C}_{nm} - i\Delta\bar{S}_{nm} = \frac{k_{nm}}{2n+1}\sum_{j}\frac{\mu_j}{\mu}\left(\frac{R}{r_j}\right)^{n+1}\bar{P}_{nm}\left(\sin\phi_j\right)e^{-im\lambda_j} \tag{3.9}$$

where

k_{nm} is the elastic Love number of the central body

j is the index of each tidal producing body

μ_j is the gravitational parameter of the tidal producing body

μ is the gravitational parameter of the central body

R is the equatorial radius of the central body

r_j is the distance to the tidal producing body (assume point mass)

ϕ_j is the spherical latitude of the tidal producing body

λ_j is the spherical longitude of the tidal producing body

Euler's formula ($e^{ix} = \cos x + i\sin x$) can be used to write separate equations for the gravity coefficient changes.

$$\Delta\bar{C}_{nm} = \frac{k_{nm}}{2n+1}\sum_{j=2}^{n}\frac{\mu_j}{\mu}\left(\frac{R}{r_j}\right)^{n+1}\bar{P}_{nm}\left(\sin\phi_j\right)\cos m\lambda_j \tag{3.10}$$

$$\Delta\bar{S}_{nm} = \frac{k_{nm}}{2n+1}\sum_{j=2}^{n}\frac{\mu_j}{\mu}\left(\frac{R}{r_j}\right)^{n+1}\bar{P}_{nm}\left(\sin\phi_j\right)\sin m\lambda_j \tag{3.11}$$

In general, the tidal effect on the $\bar{C}_{20}$ gravity coefficient does not average to zero over time. The non-zero mean effect is called the *permanent tide* or the *zero frequency tide*. The effect can be included in the *a priori* gravity field model. If it is, care must be taken to remove this effect when modeling solid body tides. Otherwise, the permanent tide effect will be erroneously included twice in the tide model. Whether or not the permanent tide is included in a particular gravity field is usually specified in supporting documentation. For Earth, the permanent tide effect can be removed using [5, 10]

$$\Delta \bar{C}_{20} = -4.4228 \times 10^{-8}(-0.31460)k_{20} \tag{3.12}$$

3.3 Detailed Design

The general algorithm used by the spherical harmonics extension of the Gravity Model is that of Gottlieb as detailed in Fast Gravity, Gravity Partials, Normalized Gravity, Gravity Gradient Torque, and Magnetic Field: Derivation, Code, and Data [1]. Each simulated spacecraft has gravity controls that select the size (degree and order) of the gravity field for each planetary body acting on the spacecraft. Each spacecraft also controls the gradient and tide effects of each body. The degree and order controls for the gravity gradient can be set independently from the degree and order of the gravity acceleration controls[3]. However, the gradient degree and order must be less than or equal to the acceleration degree and order. If the gradient controls are not specified by the user, the Gravity Model will default to the gradient of a spherical body as calculated by the base model regardless of the gravity control settings for acceleration.

3.3.1 Third Body Gravity

Third body gravity describes the gravitational acceleration of a vehicle V toward a third body B, as expressed in a frame F whose origin is also accelerating gravitationally toward the same third body. Let $\boldsymbol{\rho}$, $\boldsymbol{r}$ and $\boldsymbol{d}$ be the vectors from the origin of the accelerating frame to B, from the origin to V, and from B to V. (Note that $\boldsymbol{d} = \boldsymbol{r} - \boldsymbol{\rho}$.)

The gravitational acceleration of the vehicle in an inertial frame I toward the body with gravitational parameter μ is

$$\boldsymbol{a}_{V\to B,I} = -\mu \frac{\boldsymbol{d}}{d^3} \tag{3.13}$$

The origin of the frame is also accelerating gravitationally toward B:

$$\boldsymbol{a}_{O\to B,I} = \quad \mu \frac{\boldsymbol{\rho}}{\rho^3} \tag{3.14}$$

The difference between the above two vectors yields the acceleration of the vehicle toward the body in the accelerating frame:

$$\boldsymbol{a}_{V\to B,F} = -\mu \left(\frac{\boldsymbol{d}}{d^3} + \frac{\boldsymbol{\rho}}{\rho^3} \right) \tag{3.15}$$

Equation (3.15) can suffer from precision loss when $r \ll \rho$ as the vectors $\frac{\boldsymbol{d}}{d^3}$ and $\frac{\boldsymbol{\rho}}{\rho^3}$ are close to additive inverses of one another in this case.

[3]This required a functional modification to Gottlieb's original algorithm.

Development of a Robust Expression

The goal is to develop an equivalent expression that avoids precision loss. The development follows that of Battin in *Introduction to the Mathematics and Methods of Astrodynamics*. Factoring out $1/d^3$ from equation (3.15) and using $\boldsymbol{d} = \boldsymbol{r} - \boldsymbol{\rho}$ yields

$$\boldsymbol{a}_{V\to B,F} = -\frac{\mu}{d^3}\left(\boldsymbol{r} + \boldsymbol{\rho}\left(\frac{d^3}{\rho^3} - 1\right)\right) \tag{3.16}$$

The above remains problematic because $\frac{d^3}{\rho^3}$ will be very close to one in the case $r \ll \rho$. The solution is to express $f \equiv \frac{d^3}{\rho^3} - 1$ in a manner that does not involve the subtraction. Multiplying f by one, where one is written as $\frac{\frac{d^3}{\rho^3}+1}{\frac{d^3}{\rho^3}+1}$ results in

$$\begin{aligned} f &= \frac{d^3}{\rho^3} - 1 \\ &= \frac{\left(\frac{d^3}{\rho^3} - 1\right)\left(\frac{d^3}{\rho^3} + 1\right)}{\frac{d^3}{\rho^3} + 1} \\ &= \frac{\left(\frac{d^2}{\rho^2}\right)^3 - 1}{\left(\frac{d^2}{\rho^2}\right)^{3/2} + 1} \end{aligned} \tag{3.17}$$

To simplify the calculation, denote q and δ by

$$\begin{aligned} q &\equiv \frac{d^2}{\rho^2} - 1 \\ &= \frac{\rho^2 - 2\,\boldsymbol{r}\cdot\boldsymbol{\rho} + r^2}{\rho^2} - 1 \\ &= \frac{\boldsymbol{r}\cdot(\boldsymbol{r} - 2\boldsymbol{\rho})}{\rho^2} \end{aligned} \tag{3.18}$$

$$\begin{aligned} \delta &\equiv (q+1)^3 - 1 \\ &= q(3 + q(3+q)) \end{aligned} \tag{3.19}$$

With this, equation (3.17) becomes

$$\begin{aligned} f &= \frac{(q+1)^3 - 1}{(q+1)^{3/2} + 1} \\ &= \frac{\delta}{1 + \sqrt{1+\delta}} \end{aligned} \tag{3.20}$$

Finally, equation (3.16) becomes

$$\boldsymbol{a}_{V\to B,F} = -\frac{\mu}{\rho^3}\frac{\boldsymbol{r} + f\boldsymbol{\rho}}{1+f} \tag{3.21}$$

Note that the above avoids precision loss because of how q, δ, and f are calculated in equations (3.18) to (3.20). Note also that equation (3.21) avoids having to calculate the vector $\boldsymbol{d}$.

3.3.2 Reference Manual

The complete API for the Gravity Model can be found in the *Reference Manaual* [9].

3.4 Inventory

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

- Model header and source files are located in model `include` and `src` subdirectories. See table **??** for a list of these configuration-managed files.
- Model documentation files are located in the model `docs` subdirectory. See table **??** for a list of the configuration-managed files in this directory.

Chapter 4

User Guide

The User Guide is divided into three instruction sets:

1. **Instructions for Simulation Users**. This instruction-set contains a description of how to modify Gravity Model variables after the simulation has compiled, including a discussion of the input file; an overview of how to interpret (but not edit) the S_define file; and a sample of some of the typical variables that may be logged.
2. **Instructions for Simulation Developers**. This instruction-set builds on the previous set, and adds information on the necessary configuration of the Gravity Model within an S_define file, and the creation of standard run directories.
3. **Instructions for Model Developers**. This instruction-set builds on the previous set, and adds information on the potential for extending the model to perform tasks that are beyond its current abilities.

Note that the Gravity Model has been reworked to allow for other types of gravity representations than spherical harmonics, the default for JEOD v5.1 remains the spherical harmonics extension of the model. Thus, this implementation is the primary one discussed in this chapter, though the Model Developers section briefly discusses how to implement an alternative gravity representation.

4.1 Instructions for Simulation Users

4.1.1 Spherical Harmonics Gravity Fields

The gravity field to be used for each gravitational body is specified in the Trick sim object for each body in the S_define file. The specified spherical harmonics coefficients are loaded into the Gravity Model at sim initialization (i.e., each time the sim is run). The gravity fields provided with the current version of JEOD are found in the **environment/gravity/data** directory. Other gravity fields can be easily added to this directory by JEOD users[1]. Care must be taken to ensure

[1]As explained in section 3.2.1, each gravity field is developed in a specific planet-fixed coordinate system. To support high precision analysis, new gravity fields should be introduced with their corresponding planet-fixed frames.

the gravity coefficients are in the correct format. Whether using the existing data files or adding new ones, users should note that for all spherical harmonics gravity fields, the first three terms of SphericalHarmonicsGravitySource.Cnm are always the same and have thus been hard-coded into the model. So, any value entered for them in the data file is ignored by the gravity routines. The same holds true for the first three terms of SphericalHarmonicsGravitySource.Snm.

Example code is given for Trick users; non-Trick users should follow a similar pattern using their appropriate syntax. An example from an S_define sim object that selects the GGM02C gravity field for Earth is:

```
##include "environment/gravity/include/spherical_harmonics_gravity_source.hh"
##include "environment/gravity/data/include/earth_GGM02C.hh"

  SphericalHarmonicsGravitySource   gravity_source;
  SphericalHarmonicsGravitySource_earth_GGM02C_default_data
                                    gravity_source_default_data;
  ("default_data") gravity_source_default_data.initialize ( &gravity_source );
```

4.1.2 Gravity Controls

The effects of gravity for each gravitational body in each sim are controlled on a per-spacecraft basis. Each spacecraft object has gravity controls that determine whether the gravity field is spherical or nonspherical, the size (degree/order) of a nonspherical field, whether the gravity gradient is computed, and whether the full nonspherical field is computed or just the perturbing component of the field. Each of these controls is available for each spacecraft and for each active gravitational body. The controls are intended to be set by the user via the simulation input file.

An example declaration of the gravity controls in an S_define file that includes Earth, the sun, and the moon would be:

```
##include "environment/gravity/include/spherical_harmonics_gravity_controls.hh"

  SphericalHarmonicsGravityControls earth_grav_ctrl;
  SphericalHarmonicsGravityControls sun_grav_ctrl;
  SphericalHarmonicsGravityControls moon_grav_ctrl;
```

Assuming these example gravity controls are declared in a Trick sim object named sv_dyn, a sample gravity controls setup for Earth in an input file would then be:

```
  sv_dyn.earth_grav_ctrl.source_name     = "Earth";
  sv_dyn.earth_grav_ctrl.active          = True;
  sv_dyn.earth_grav_ctrl.spherical       = False;
  sv_dyn.earth_grav_ctrl.degree          = 70;
  sv_dyn.earth_grav_ctrl.order           = 70;
  sv_dyn.earth_grav_ctrl.gradient        = True;
  sv_dyn.earth_grav_ctrl.gradient_degree = 5;
  sv_dyn.earth_grav_ctrl.gradient_order  = 5;
```

```
sv_dyn.earth_grav_ctrl.perturbing_only = False;
sv_dyn.earth_grav_ctrl.battin_method   = False;
```

If a spherical gravity field has been selected for a particular body, then degree and order are ignored. If degree and order are not specified when nonspherical gravity is selected, then the entire gravity field (maximum degree and order available) is used by default. This could cause a sim to run much slower than it would otherwise run.

The gradient control must be "True" in order to invoke the JEOD Gravity Torque Model. The gradient degree and order must be less than or equal to the acceleration degree and order. In this example, the gradient degree and order are both 5, and the acceleration degree and order are both 70. If the gradient degree and order are not specified, the Gravity Model will compute the spherical body gradient by default. Spherical body gradient can also be invoked by setting the gradient degree and order both to 0. See JEOD Gravity Torque Model [11] for more information on gravity torque.

The perturbing_only control will default to false if not otherwise specified. This will be the correct option for most sims running JEOD (i.e., spacecraft trajectory simulations). This option was included because certain gravity field analysis or advanced integration methods may require only the perturbing portion of the gravity field to be computed.

The battin_method control determines whether an alternative formulation of third body gravity will be used. The alternate method provides better numerical accuracy at a slight cost in complexity. For a detailed description, see 3.3.1. The default value is false.

4.1.3 Solid Body Tides

Solid body tide effects are modeled as temporal variations in the gravity coefficients of a gravitation body due to the gravity of one or more other gravitational bodies (such as the sun or moon). The tidal causing bodies are selected via a default data file in the simulation S_define. After the solid body tides have been properly included in the S_define (see the next section for how this is done), the sim user can "turn off" or "turn on" solid body tides from the input file by setting the "active" control to "True" or "False."

An example declaration of a solid body tide control in an S_define file is:

```
##include "environment/gravity/include/spherical_harmonics_delta_controls.hh"
  SphericalHarmonicsDeltaControls sbtide_ctrl;
```

Again assuming that this control was declared in a Trick sim object named sv_dyn, an example solid body tide control setup in the input file would be:

```
earth.sb_tide.dC20 = 0.0;
sv_dyn.sbtide_ctrl.active = True;
sv_dyn.sbtide_ctrl.first_order_only = True;
sv_dyn.sbtide_ctrl.degree = 2;
sv_dyn.sbtide_ctrl.order = 0;
sv_dyn.sbtide_ctrl.grav_effect = earth.sb_tide;
sv_dyn.sbtide_ctrl.grav_source   = earth.gravity_source;
```

4.1.4 Data Logging

An example line from a log file for recording gravitational acceleration of a spacecraft is:

```
for ii in range(0,3):
 dr_group.add_variable(""+VEH_OBJ+".body.grav_interaction.grav_accel["+str(ii)+"]")
```

4.2 Instructions for Simulation Developers

4.2.1 Gravity Model

The top-level structure for the Gravity Model is normally declared as part of a space environment sim object in the S_define file for a simulation, such as:

```
##include "environment/gravity/include/gravity_manager.hh"
  GravityManager gravity;
```

An initialization job is also required as part of the same sim object:

```
P_ENV  ("initialization") gravity.initialize_model (
   internal_dynamics->manager );
```

Each planetary body must declare a gravity body object (note that the spherical harmonics coefficients are normally loaded into this object as default data) and call two initialization-related jobs. An example that uses the GRACE GGM02C gravity field for Earth is:

```
##include "environment/gravity/include/spherical_harmonics_gravity_source.hh"
##include "environment/gravity/data/include/earth_GGM02C.hh"

  SphericalHarmonicsGravitySource   gravity_source;
  SphericalHarmonicsGravitySource_earth_GGM02C_default_data
                                   gravity_source_default_data;
  ("default_data") gravity_source_default_data.initialize ( &gravity_source );

  P_ENV  ("initialization") gravity_source.initialize_body ( );
  P_ENV  ("initialization") internal_env->gravity.add_grav_source(
     gravity_source );
```

Each spacecraft object in the simulation will have its own set of gravity controls that specify how these gravity bodies affect it. Implementation of these controls requires declaration of the controls themselves. In the S_define, the set of controls are implemented as follows:

```
##include "environment/gravity/include/spherical_harmonics_gravity_controls.hh"

  SphericalHarmonicsGravityControls earth_grav_ctrl;
  SphericalHarmonicsGravityControls sun_grav_ctrl;
  SphericalHarmonicsGravityControls moon_grav_ctrl;
```

The following corresponding lines would then go in the input file:

```
   sv_dyn.body.grav_interaction.add_control(sv_dyn.earth_grav_ctrl)
   sv_dyn.body.grav_interaction.add_control(sv_dyn.sun_grav_ctrl)
   sv_dyn.body.grav_interaction.add_control(sv_dyn.moon_grav_ctrl)
```

The use of the settings within these controls was discussed in the previous section and will not be repeated here.

4.2.2 Gravity Interaction

The gravity effects acting on an individual spacecraft are captured in a gravity interaction class to which gravity controls are added as shown above. The gravity interaction class offers a method to sort its gravity controls so that their summation is performed in order of increasing magnitude. This is a recommended practice for multi-planet simulations since the additional overhead is minuscule, and it can significantly improve accuracy. To sort at ten minute intervals, add the following line to your Trick 10 vehicle sim object.

```
(600, "environment") body.grav_interaction.sort_controls ();
```

4.2.3 Solid Body Tides

To include solid body tides in a simulation, an object of type SphericalHarmonicsSolidBodyTides must be included as part of the S_define in the relevant planet-related sim object; its companion initialization object SphericalHarmonicsSolidBodyTidesInit must also be included. For example, as part of the Earth-related sim object, one could include:

```
 ##include "environment/gravity/include/spherical_harmonics_solid_body_tides.hh"
 ##include "environment/gravity/include/spherical_harmonics_solid_body_tides_init.hh"
 ##include "environment/gravity/data/include/earth_solid_tides.hh"

   SphericalHarmonicsSolidBodyTides       sb_tide;
   SphericalHarmonicsSolidBodyTidesInit   sbtide_init;
   SphericalHarmonicsSolidBodyTidesInit_earth_solid_tides_default_data
                                          sbtide_init_default_data;
   ("default_data") sbtide_init_default_data.initialize ( &sbtide_init );
```

More information about these two objects can be found in JEOD Gravity Model Reference Manual [9]. A call to job "add_deltacoeff" must accompany these items (in the same sim object) for proper functioning of the solid body tides:

```
   P31  ("initialization") gravity_source.add_deltacoeff (
      sbtide_init,
      internal_dynamics->manager,
      sb_tide );
```

Each spacecraft object in the simulation will have its own set of controls that specify how the solid body tides affect it. Similar to the controls for each gravity body, implementation of the solid body tides controls requires declaration of the controls themselves in the S_define:

```
##include "environment/gravity/include/spherical_harmonics_delta_controls.hh"
  environment/gravity: SphericalHarmonicsDeltaControls sbtide_ctrl;
```

The relevant input file entry that corresponds would then be:

```
  sv_dyn.earth_grav_ctrl.add_deltacontrol(sv_dyn.sbtide_ctrl)
```

Once these elements are included, the Gravity Model will automatically incorporate the solid body tide calculations as appropriate based on the relevant control settings. Control of the solid body tides model was described in the previous section.

4.3 Instructions for Model Developers

There are two ways to extend the Gravity Model: by incorporating additional spherical harmonics gravity fields other than those provided as default data in the **environment/gravity/data** directory, and by implementing different gravity representations other than spherical harmonics.

If incorporating a new spherical harmonics gravity field, care must be taken to ensure the gravity coefficients are in the correct format. All of the coefficients must be in fully normalized form (see Chapter 3).

If implementing a new gravity representation, one must have familiarity with the C++ programming language. The existing spherical harmonics extension of the Gravity Model might also prove to be a helpful guide for the implementation process.

For the new model extension, two derived classes should be created: one that inherits from GravitySource, and one that inherits from GravityControls. The derived GravitySource-based class should contain any gravity representation– related data; for example, the analogous class for the spherical harmonics extension contains all of the Gottlieb parameters and spherical harmonics coefficients necessary to perform the model's gravitational calculations. The GravitySource-based class should also be primarily a container class, since all gravitational calculations are performed in the GravityControls-based class.

The base GravityControls class has four methods: "initialize_control", "reset_control", "gravitation", and protected method "calc_nonspherical". As their names imply, the first two methods contain basic class initialization and reset functionality that will be common to all gravity model representations. The model developer can optionally extend either, both, or neither method as best suits his needs without affecting overall Gravity Model integrity, so long as any extensions still call the base-class method(s). For reference, the spherical harmonics implementation extends initialize_control (while still invoking the base method), but it leaves reset_control untouched.

The base-class "gravitation" method converts the given position vector into the proper frame and then performs the universal two-body point-mass gravity calculation. This method is not intended to be overridden.

Base-class method "calc_nonspherical" is the one method that *must* be overridden for successful model extension. The base method exists only for inheritance and will throw an error if invoked. The overriding method should contain any nonspherical gravity calculations for the new model extension. If properly implemented, the derived-class method will automatically be called by the Gravity Model when an instance of the new model extension is employed.

Chapter 5

Inspections, Tests, and Metrics

5.1 Inspections

This section describes the inspections conducted on the Gravity Model to examine its compliance with the inspection requirements levied against it.

Inspection Gravity_1: Top-level Inspection

This document structure, the code, and associated files have been inspected, and together satisfy requirement Gravity_1.

Inspection Gravity_2: Extensibility Inspection

The successful spherical harmonics extension of the model demonstrates satisfaction of requirement Gravity_2.

Inspection Gravity_3: Generic Model Data Inspection

The model code has been inspected and satisfies requirement Gravity_3.

Inspection Gravity_4: Spherical Harmonics Data Inspection

The spherical harmonics code has been inspected and it satisfies requirement Gravity_4.

Inspection Gravity_5: Generic Model Functionality Inspection

The model code has been inspected, and it partially satisfies requirement Gravity_5.

5.2 Tests

This section describes the verification and validation tests conducted on the Gravity Model to examine its satisfaction of the test requirements levied against it. The tests described in this section are archived in the JEOD directory `models/environment/gravity/verif`.

Test Gravity_1: Spherical Harmonics Gravity Acceleration Computation

Purpose:

The purpose of this test is to examine the spherical harmonics gravitational accleration calculations against an analytic solution.
SIM directory: models/environment/gravity/verif/SIM_grav_accel_verif
Run directory: RUN_01

Applicable Requirements:

By passing this test, the Gravity Model satisfies requirement Gravity_6.

Procedure:

The gravity coefficients of a system of i discrete point masses (m_i) can be obtained by reformulating equations 3.5 as (see Thompson [12])

$$\begin{aligned} C_{nm} &= \frac{2-\delta_{0m}}{M_\oplus}\frac{(n-m)!}{(n+m)!}\sum_i \left(\frac{s_i}{R_\oplus}\right)^n P_{nm}\sin(\phi_i')\cos(m\lambda_i')m_i \\ S_{nm} &= \frac{2-\delta_{0m}}{M_\oplus}\frac{(n-m)!}{(n+m)!}\sum_i \left(\frac{s_i}{R_\oplus}\right)^n P_{nm}\sin(\phi_i')\sin(m\lambda_i')m_i \end{aligned} \tag{5.1}$$

The coefficients can be normalized using equation 3.2. They can be read into a gravity model, and the resulting acceleration can be compared with the acceleration resulting directly from the point masses in order to verify the spherical harmonic algorithms are working properly.

Point Mass System

Because the goal of this work was to test the implementation of spherical harmonic acceleration algorithms, the gravity coefficients used did not need to truly represent any particular planet. However, for familiarity it was decided to choose a system of point masses that would produce accelerations on the same scale that would be encountered by a low-Earth-orbiting satellite. For the chosen point mass system, the constants of the JGM-3 (or GGM02C) gravity field were used [7]:

$$\begin{aligned} \mu &= 398600.4415\ km^3/s^2 \\ R_\oplus &= 6378.1363\ km \end{aligned} \tag{5.2}$$

Knowing that $\mu = GM_\oplus$ and assuming a value for G of $6.673\times10^{-20}\ km^3/(kg\cdot s^2)$, the total mass of the system of point masses was chosen to be $M_\oplus = \mu/G$.

It was decided that testing the spherical harmonic algorithms to degree and order four (4x4) would be sufficient. If comparing the acceleration computed from the spherical harmonic algorithms to that computed from the point mass model showed differences on the order of $\sim 10^{-15}$ or better (the approximate limit of double-precision computer arithmetic), then it was assumed by induction that the spherical harmonic algorithms were correctly implementing coefficients higher than degree four. To avoid truncation errors, the gravity coefficients to degree and order five (5x5) were computed and used. However, a point mass system was selected such that the degree five coefficients were several orders of magnitude less than the degree four coefficients.

A system of 12 point masses was configured by experimentation. The total mass of the system was approximately equal to the mass of the Earth roughly divided among all of the points (see Table 5.1). The first six point masses were chosen to result in a center of mass at the origin and equal mass in the "northern" and "southern" hemispheres. The north/south symmetry of mass caused the odd zonal ($m = 0$) coefficients to be zero. This was unsuitable for evaluation of the spherical harmonic algorithms, so the final six point masses were included such that more mass was positioned in the southern hemisphere (similar to Earth). These points were added to the system in pairs such that the center of mass remained at the origin of the coordinate system in spite of the unequal north/south mass distribution. In terms of radial distance from the origin, the center of mass of two points is

$$c.o.m = \frac{r_1 m_1 + r_2 m_2}{m_1 + m_2} \tag{5.3}$$

This assumes the masses are situated in diametrically opposing locations with respect to the origin. Setting equation 5.3 to zero and solving for r_2 yields (dropping the negative sign)

$$r_2 = \frac{m_1}{m_2} r_1 \tag{5.4}$$

This is the radial position of the second point mass in each pair that kept the center of mass at the origin.

Table 5.1 shows the 12 point masses that were selected. This system resulted in degree-one gravity coefficients on the order of $\sim 10^{-20}$, indicating that the center of mass of the system was very nearly located at the origin (see equation 3.3). Therefore, the degree-one terms were considered negligible and forced to be exactly zero for all computations. This was necessary because the Gravity Model spherical harmonic algorithms make this assumption and begin computing with the degree-two terms (see equation 3.4).

The fully normalized gravity coefficients that represent the point mass system were computed using equations 5.1 and 3.2. They are shown in Table 5.2. Again, the degree-one terms were forced to be exactly equal to zero, and the degree-five terms are several orders of magnitude smaller than the degree-four terms. The entire 5x5 set of coefficients was used during testing, but the degree-five terms were found to have no impact on the acceleration computations at the level of precision to which the tests were conducted ($\sim 10^{-15}$).

To compare the acceleration computed from the spherical harmonic algorithms to the point mass system, the magnitude of the acceleration due to the ith point mass can be computed from

$$a_i = \frac{G m_i}{r_i^2} \tag{5.5}$$

The acceleration magnitude is multiplied by the unit vectors from the point of interest to the position of each point mass to obtain the acceleration vectors. The resulting acceleration vector components are summed to produce the total acceleration vector resulting from the

Table 5.1: Point Masses

Point	Mass (kg)	Lat. (deg)	Lon. (deg)	Radius (m)
1	$M_\oplus/12$	45.0	0.0	4000.0
2	$M_\oplus/12$	45.0	120.0	4000.0
3	$M_\oplus/12$	45.0	240.0	4000.0
4	$M_\oplus/12$	-45.0	180.0	4000.0
5	$M_\oplus/12$	-45.0	300.0	4000.0
6	$M_\oplus/12$	-45.0	60.0	4000.0
7	$0.8M_\oplus/12$	23.0	73.0	4000.0
8	$1.2M_\oplus/12$	-23.0	253.0	(0.8/1.2)4000.0
9	$0.6M_\oplus/12$	77.0	303.0	4000.0
10	$1.4M_\oplus/12$	-77.0	123.0	(0.6/1.4)4000.0
11	$0.6M_\oplus/12$	51.0	12.0	4000.0
12	$1.4M_\oplus/12$	-51.0	192.0	(0.6/1.4)4000.0

Table 5.2: Fully Normalized Gravity Coefficients

n	m	$\bar{C}_{nm}$	$\bar{S}_{nm}$
0	0	1.000000000000000E+00	0.000000000000000E+00
1	0	0.000000000000000E+00	0.000000000000000E+00
1	1	0.000000000000000E+00	0.000000000000000E+00
2	0	3.340052631095518E-08	0.000000000000000E+00
2	1	1.656759356170754E-08	9.855566979885050E-09
2	2	-8.176775024351829E-09	9.269248429468815E-09
3	0	1.759117523226707E-12	0.000000000000000E+00
3	1	3.832276836797579E-12	-1.471910512326285E-12
3	2	8.885825377841406E-14	1.828488643978194E-12
3	3	-1.081882767554426E-12	-9.045529521634635E-13
4	0	-9.608634091968111E-15	0.000000000000000E+00
4	1	1.532755741714554E-15	-3.539212041089171E-15
4	2	1.514697168098408E-15	4.836572455402800E-16
4	3	1.212195117185278E-14	-1.135418848925473E-15
4	4	1.098630790621571E-15	-1.950021729407439E-15
5	0	7.689778516079726E-19	0.000000000000000E+00
5	1	5.103903289685398E-19	-1.270409281937703E-18
5	2	1.288578200168744E-18	-3.356781430612713E-19
5	3	4.127243225918561E-19	3.200323268358486E-19
5	4	6.153203543016434E-19	-6.131745252702638E-19
5	5	7.718691941720859E-19	1.485405551920983E-19

point mass system. Computing the unit vectors requires conversion from spherical coordinates to Cartesian coordinates using

$$\begin{aligned} x &= r\cos(\phi)\cos(\lambda) \\ y &= r\cos(\phi)\sin(\lambda) \\ z &= r\sin(\phi) \end{aligned} \tag{5.6}$$

For completeness, the value of π that was used during all computations and testing was $\pi = 3.14159265358979$.

Results:

A set of test points was selected somewhat randomly to compare the acceleration computed by the spherical harmonic algorithms to the point mass acceleration. All test points were selected to have a radial distance of 6778000.0 meters, approximating the altitude of a typical low-Earth-orbiting satellite. The results in Table 5.3 show that the spherical harmonic accelerations exactly match the point mass accelerations on the order of $\sim 10^{-15}$. Only four of the 27 acceleration components computed showed any difference, and these differences were of the order $\sim 10^{-15}$. Some of the test points were chosen to be located at possible singularity points such as over the poles and the equator (i.e., points 1, 4, etc.). No problems of this type were discovered.

Table 5.3: Results comparing acceleration vectors computed by the spherical harmonic algorithms vs. the acceleration vectors computed from the point mass model. All accelerations are body-fixed; no rotation was included in the computations.

Test Point	Lat. (deg)	Lon. (deg)	Sph. Harm. Accel. (m/s^2)	Pt. Mass Accel. (m/s^2)	Difference
1	90.0	0.0	0.000000493153815	0.000000493153815	0E-15
			0.000000293186434	0.000000293186434	0E-15
			-8.676303443954591	-8.676303443954589	2E-15
2	73.0	9.0	-2.505474755998081	-2.505474755998081	0E-15
			-0.396827911350609	-0.396827911350609	0E-15
			-8.297190422630861	-8.297190422630862	1E-15
3	13.0	100.0	1.468009707514858	1.468009707514857	1E-15
			-8.325494148828971	-8.325494148828971	0E-15
			-1.951742612340920	-1.951742612340920	0E-15
4	0.0	0.0	-8.676300496540250	-8.676300496540250	0E-15
			0.000000275785602	0.000000275785602	0E-15
			0.000000492927440	0.000000492927440	0E-15
5	0.0	180.0	8.676300497030235	8.676300497030235	0E-15
			-0.000000275833219	-0.000000275833219	0E-15
			-0.000000493021800	-0.000000493021800	0E-15
6	0.0	270.0	-0.000000275805400	-0.000000275805400	0E-15
			8.676301226569601	8.676301226569601	0E-15
			-0.000000293309608	-0.000000293309608	0E-15
7	-26.0	210.0	6.753447406310072	6.753447406310070	2E-15
			3.899104312479454	3.899104312479453	1E-15
			3.803439650290294	3.803439650290293	1E-15
8	-87.0	350.0	-0.447184691105532	-0.447184691105532	0E-15
			0.078850356049630	0.078850356049630	0E-15
			8.664412766767059	8.664412766767063	4E-15
9	-90.0	180.0	-0.000000492794709	-0.000000492794709	0E-15
			-0.000000293324361	-0.000000293324361	0E-15
			8.676303443685409	8.676303443685409	0E-15
10	-90.0	160.0	-0.000000486448821	-0.000000486448821	0E-15
			-0.000000289546566	-0.000000289546566	0E-15
			8.620253461628579	8.620253461628582	3E-15

Test Gravity_2: Spherical Harmonics Gravity Gradient

Purpose:

The purpose of this test is to examine the spherical harmonics gravity gradient calculations.

Applicable Requirements:

By passing this test, the Gravity Model satisfies requirement Gravity_7.

Procedure:

The gravity gradient feature of the Gravity Model was tested by the JEOD Gravity Torque Model. See the JEOD Gravity Torque Model document for details [11].

Results:

The gravity gradient calculations were found to operate correctly.

Test Gravity_3: Spherical Harmonics Solid Body Tides

Purpose:

This test was conducted to verify that the solid body tide model was computing and applying temporal changes in the C20 Earth gravity coefficient consistent with the order described in the IERS Conventions [5]. The test also served to test the simple point-mass gravitational acceleration calculations found in the base Gravity Model. SIM directory: models/environment/gravity/verif/SIM_tide_verif
Run directory: RUN_01

Applicable Requirements:

By passing this test, the Gravity Model satisfies requirement Gravity_8, and partially satisfies requirement Gravity_5.

Procedure:

The solid body tides calculations were enabled in a test case with spherical harmonics degree and order for Earth both set to eight, and the sun and Moon set to be modeled as simple point masses.

Results:

The test run produced changes in the C20 Earth gravity coefficient on the order of $1E - 09$, which was the expected value. Therefore, the solid body tides calculations were found to operate correctly. The test also calculated values of gravitational acceleration for the sun and moon consistent with the analytic solution, indicating that the simple point-mass gravity calculations were also operating correctly.

Test Gravity_4: CSR/GRACE Comparison

Table 5.4: CSR Test Point Locations (in Earth-Centered, Earth-Fixed Coordinates)

Point	x (m)	y (m)	z (m)	magnitude (m)
1	7218634.798289895	18998.64159785956	1938152.473366886	7474321.662162215
2	−7218634.798289895	18998.64159785956	1938152.473366886	7474321.662162215
3	−7218634.798289895	−18998.64159785956	1938152.473366886	7474321.662162215
4	−7218634.798289895	−18998.64159785956	−1938152.473366886	7474321.662162215
5	−7218634.798289895	18998.64159785956	−1938152.473366886	7474321.662162215

Purpose:

The purpose of this test was to demonstrate that the JEOD Gravity Model was computing accelerations that matched those computed by an independent, well established model. SIM directory: models/environment/gravity/verif/SIM_csr_compare
Run directory: RUN_01

Applicable Requirements:

By passing this test, the Gravity Model satisfies requirement Gravity_6.

Procedure:

Perturbing acceleration and potential from the 70x70 GRACE GG2M02C gravity model at select test point locations were provided by the Center for Space Research (CSR) at the University of Texas, Austin. The Gravity Model was run at the same test point locations (Table 5.4) and the computed acceleration components and gravitational potential were compared to CSR values. The results of the comparisons are shown in Table 5.4.

Results:

The results of the potential comparisons are shown in Table 5.5, and the acceleration comparisons are shown in Table 5.6. Based on the test point position magnitudes provided by CSR, only the first 12 digits were considered significant. The two tables show an excellent match between the CSR and JEOD gravity computations.

Table 5.5: Perturbing Potential Comparisons (m^2/s^2)

Point	CSR	JEOD	Diff
1	16958.96604841	16958.96604841	0E-08
2	16882.51269785	16882.51269784	1E-08
3	16881.43374446	16881.43374446	0E-08
4	17035.10717458	17035.10717458	0E-08
5	17037.52048293	17037.52048292	1E-08

Table 5.6: Perturbing Acceleration Comparisons (m/s^2)

Point	x CSR JEOD Diff	y	z
1	$-5.48059571666E-03$ $-5.48059571666E-03$ $0.0E-14$	$-3.29750891335E-05$ $-3.29750891334E-05$ $1.0E-16$	$-5.83469494098E-03$ $-5.83469494098E-03$ $0.0E-14$
2	$5.41224638245E-03$ $5.41224638245E-03$ $0.0E-14$	$1.44408452851E-05$ $1.44408452851E-05$ $0.0E-16$	$-5.88929182974E-03$ $-5.88929182974E-03$ $0.0E-14$
3	$5.41212832075E-03$ $5.41212832074E-03$ $1.0E-14$	$4.23509704180E-05$ $4.23509704180E-05$ $0.0E-16$	$-5.88884615843E-03$ $-5.88884615842E-03$ $1.0E-14$
4	$5.53737040719E-03$ $5.53737040719E-03$ $0.0E-14$	$7.84095111595E-05$ $7.84095111594E-05$ $1.0E-16$	$5.85680272817E-03$ $5.85680272817E-03$ $0.0E-14$
5	$5.53912757967E-03$ $5.53912757966E-03$ $1.0E-14$	$4.86095494368E-05$ $4.86095494368E-05$ $0.0E-16$	$5.85722493893E-03$ $5.85722493892E-03$ $1.0E-14$

5.3 Requirements Traceability

Table 5.7 summarizes the inspections and tests that demonstrate the satisfaction of the requirements levied on the model.

Table 5.7: Requirements Traceability

Requirement	**Traces to**
Gravity_1 Top-level	Insp. Gravity_1 Top-level Inspection
Gravity_2 Extensibility	Insp. Gravity_2 Extensibility Inspection
Gravity_3 Generic Model Data	Insp. Gravity_3 Generic Model Data Inspection
Gravity_4 Spherical Harmonics Gravitational Data	Insp. Gravity_4 Spherical Harmonics Data Inspection
Gravity_5 Generic Model Functionality	Insp. Gravity_5 Generic Model Functionality Inspection Test Gravity_3 Spherical Harmonics Solid Body Tides

Continued on next page

Table 5.7: Source Files (continued from previous page)

Requirement	**Traces to**
Gravity_6 Spherical Harmonics Gravitational Acceleration	Test Gravity_1 Spherical Harmonics Gravity Acceleration Computation Test Gravity_4 CSR/GRACE Comparison
Gravity_7 Spherical Harmonics Gravity Gradient	Test Gravity_2 Spherical Harmonics Gravity Gradient
Gravity_8 Spherical Harmonics Solid Body Tides	Test Gravity_3 Spherical Harmonics Solid Body Tides

5.4 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**
Total	0	0	0	0

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::GravityControls::accel_mag_less_ptr (const GravityControls* a, const GravityControls* b)	include/gravity_controls.hh	249	1
jeod::GravityManager::std::get_bodies ()	include/gravity_manager.hh	159	1
jeod::SphericalHarmonicsGravityControls::disable_min_radius_warnings ()	include/spherical_harmonics_gravity_controls.hh	262	1
jeod::GravityControls::GravityControls ()	src/gravity_controls.cc	62	1
jeod::GravityControls::~GravityControls ()	src/gravity_controls.cc	87	1
jeod::GravityControls::initialize_control (GravityManager & grav_man)	src/gravity_controls.cc	99	5
jeod::GravityControls::reset_control (BaseDynManager& dyn_manager)	src/gravity_controls.cc	146	10
jeod::GravityControls::gravitation (const double integ_pos[3], unsigned int integ_frame_idx, double body_grav_accel[3], double dgdx[3][3], double Pot[1])	src/gravity_controls.cc	207	4
jeod::GravityControls::gravitation (const RefFrame& point_of_interest, unsigned int integ_frame_idx, double body_grav_accel[3], double dgdx[3][3], double& pot)	src/gravity_controls.cc	258	5

Continued on next page

Table 5.9: Cyclomatic Complexity (continued)

Method	File	Line	ECC
jeod::GravityControls::calc_spherical (const double integ_pos[3], const double posn[3], const GravityInteg Frame & grav_source_frame, double body_grav_accel[3], double dgdx[3][3], double& pot)	src/gravity_controls.cc	333	5
jeod::GravityControls::calc_relativistic (const Ref Frame& point_of_interest, const double rel_pos[3], const double rel_vel[3], double perturbing_accel[3])	src/gravity_controls.cc	418	4
jeod::GravityIntegFrame:: GravityIntegFrame (void)	src/gravity_integ_frame.cc	41	1
jeod::GravityIntegFrame::~ GravityIntegFrame (void)	src/gravity_integ_frame.cc	55	1
jeod::GravityInteraction:: GravityInteraction (void)	src/gravity_interaction.cc	57	1
jeod::GravityInteraction::~ GravityInteraction (void)	src/gravity_interaction.cc	75	1
jeod::GravityInteraction::set_integ_frame (const EphemerisRefFrame & ref_frame, const BaseDyn Manager & dyn_manager)	src/gravity_interaction.cc	86	1
jeod::GravityInteraction::add_control (GravityControls * control)	src/gravity_interaction.cc	103	2
jeod::GravityInteraction:: remove_control (Gravity Controls* control)	src/gravity_interaction.cc	129	2
jeod::GravityInteraction:: initialize_controls (BaseDyn Manager & dyn_manager, GravityManager & grav_manager)	src/gravity_interaction.cc	152	2

Continued on next page

Table 5.9: Cyclomatic Complexity (continued)

Method	**File**	**Line**	**ECC**
jeod::GravityInteraction:: reset_controls (BaseDyn Manager& manager)	src/gravity_interaction.cc	174	2
jeod::GravityInteraction::sort_ controls (void)	src/gravity_interaction.cc	190	3
jeod::GravityManager::JEOD_ DECLARE_ATTRIBUTES (GravitySource)	src/gravity_manager.cc	52	1
jeod::GravityManager::~ GravityManager (void)	src/gravity_manager.cc	68	1
jeod::GravityManager::find_ grav_source (const std:: string & source_name)	src/gravity_manager.cc	79	4
jeod::GravityManager::add_ grav_source (GravitySource & source)	src/gravity_manager.cc	112	3
jeod::GravityManager:: initialize_model (BaseDyn Manager & manager)	src/gravity_manager.cc	145	1
jeod::GravityManager:: initialize_state (const Base DynManager & manager)	src/gravity_manager.cc	161	2
jeod::GravityManager:: gravitation (const double integ_pos[3], Gravity Interaction & grav)	src/gravity_manager.cc	189	3
jeod::GravityManager:: gravitation (const Ref Frame& point, Gravity Interaction& grav)	src/gravity_manager.cc	235	3
jeod::GravitySource::JEOD_D ECLARE_ATTRIBUTES (GravityIntegFrame)	src/gravity_source.cc	47	1
jeod::GravitySource::~Gravity Source (void)	src/gravity_source.cc	67	2

Continued on next page

Table 5.9: Cyclomatic Complexity (continued)

Method	**File**	**Line**	**ECC**
jeod::GravitySource:: initialize_state (const std:: vector¡EphemerisRefFrame *¿ & integ_frames, const GravityManager & gravity_ manager JEOD_UNUSED)	src/gravity_source.cc	79	3
jeod::SphericalHarmonics GravityControls::calc_ nonspherical (const double integ_pos[3] __attribute__ ((unused)	src/spherical_harmonics_calc_ nonspherical.cc	47	42
jeod::SphericalHarmonics DeltaCoeffs::Spherical HarmonicsDeltaCoeffs (void)	src/spherical_harmonics_ delta_coeffs.cc	52	1
jeod::SphericalHarmonics DeltaCoeffs::~Spherical HarmonicsDeltaCoeffs (void)	src/spherical_harmonics_ delta_coeffs.cc	69	3
jeod::SphericalHarmonics DeltaCoeffs::initialize (SphericalHarmonicsDelta CoeffsInit & var_init, Base DynManager& dyn_ manager JEOD_UNUSED)	src/spherical_harmonics_ delta_coeffs.cc	86	5
jeod::SphericalHarmonics DeltaCoeffs::update (SphericalHarmonicsGravity Controls & controls JEOD_ UNUSED)	src/spherical_harmonics_ delta_coeffs.cc	122	1
jeod::SphericalHarmonics DeltaCoeffsInit::Spherical HarmonicsDeltaCoeffsInit (void)	src/spherical_harmonics_ delta_coeffs_init.cc	42	1
jeod::SphericalHarmonics DeltaCoeffsInit::~Spherical HarmonicsDeltaCoeffsInit (void)	src/spherical_harmonics_ delta_coeffs_init.cc	55	1

Continued on next page

Table 5.9: Cyclomatic Complexity (continued)

Method	File	Line	ECC
jeod::SphericalHarmonicsDeltaControls::SphericalHarmonicsDeltaControls (void)	src/spherical_harmonics_delta_controls.cc	46	1
jeod::SphericalHarmonicsDeltaControls::~SphericalHarmonicsDeltaControls (void)	src/spherical_harmonics_delta_controls.cc	63	1
jeod::SphericalHarmonicsGravityControls::SphericalHarmonicsGravityControls (void)	src/spherical_harmonics_gravity_controls.cc	53	1
jeod::SphericalHarmonicsGravityControls::~SphericalHarmonicsGravityControls (void)	src/spherical_harmonics_gravity_controls.cc	80	5
jeod::SphericalHarmonicsGravityControls::initialize_control (GravityManager & grav_manager)	src/spherical_harmonics_gravity_controls.cc	116	4
jeod::SphericalHarmonicsGravityControls::add_deltacontrol (SphericalHarmonicsDeltaControls * delta_control)	src/spherical_harmonics_gravity_controls.cc	181	8
jeod::SphericalHarmonicsGravityControls::get_degree (void)	src/spherical_harmonics_gravity_controls.cc	233	1
jeod::SphericalHarmonicsGravityControls::get_order (void)	src/spherical_harmonics_gravity_controls.cc	245	1
jeod::SphericalHarmonicsGravityControls::get_degree_order (unsigned int &current_degree, unsigned int &current_order)	src/spherical_harmonics_gravity_controls.cc	257	1
jeod::SphericalHarmonicsGravityControls::get_grad_degree (void)	src/spherical_harmonics_gravity_controls.cc	274	1

Continued on next page

Table 5.9: Cyclomatic Complexity (continued)

Method	**File**	**Line**	**ECC**
jeod::SphericalHarmonicsGravityControls::get_grad_order (void)	src/spherical_harmonics_gravity_controls.cc	286	1
jeod::SphericalHarmonicsGravityControls::get_grad_degree_order (unsigned int &curr_grad_degree, unsigned int &curr_grad_order)	src/spherical_harmonics_gravity_controls.cc	298	1
jeod::SphericalHarmonicsGravityControls::set_degree (unsigned int new_degree)	src/spherical_harmonics_gravity_controls.cc	315	1
jeod::SphericalHarmonicsGravityControls::set_order (unsigned int new_order)	src/spherical_harmonics_gravity_controls.cc	332	1
jeod::SphericalHarmonicsGravityControls::set_degree_order (unsigned int new_degree, unsigned int new_order)	src/spherical_harmonics_gravity_controls.cc	349	1
jeod::SphericalHarmonicsGravityControls::set_grad_degree (unsigned int new_grad_degree)	src/spherical_harmonics_gravity_controls.cc	369	1
jeod::SphericalHarmonicsGravityControls::set_grad_order (unsigned int new_grad_order)	src/spherical_harmonics_gravity_controls.cc	386	1
jeod::SphericalHarmonicsGravityControls::set_grad_degree_order (unsigned int new_grad_degree, unsigned int new_grad_order)	src/spherical_harmonics_gravity_controls.cc	403	1
jeod::SphericalHarmonicsGravityControls::check_validity (void)	src/spherical_harmonics_gravity_controls.cc	423	11
jeod::SphericalHarmonicsGravityControls::update_deltacoeffs (void)	src/spherical_harmonics_gravity_controls.cc	529	3

Continued on next page

Table 5.9: Cyclomatic Complexity (continued)

Method	File	Line	ECC
jeod::SphericalHarmonicsGravityControls::sum_deltacoeffs (void)	src/spherical_harmonics_gravity_controls.cc	550	8
jeod::SphericalHarmonicsGravitySource::SphericalHarmonicsGravitySource (void)	src/spherical_harmonics_gravity_source.cc	56	1
jeod::SphericalHarmonicsGravitySource::~SphericalHarmonicsGravitySource (void)	src/spherical_harmonics_gravity_source.cc	87	5
jeod::SphericalHarmonicsGravitySource::initialize_body (void)	src/spherical_harmonics_gravity_source.cc	126	10
jeod::SphericalHarmonicsGravitySource::find_deltacoeff (const SphericalHarmonicsDeltaCoeffs & delta_coeff)	src/spherical_harmonics_gravity_source.cc	260	4
jeod::SphericalHarmonicsGravitySource::add_deltacoeff (SphericalHarmonicsDeltaCoeffsInit & var_init, BaseDynManager & dyn_manager, SphericalHarmonicsDeltaCoeffs & var_effect)	src/spherical_harmonics_gravity_source.cc	289	2
jeod::SphericalHarmonicsSolidBodyTides::SphericalHarmonicsSolidBodyTides (void)	src/spherical_harmonics_solid_body_tides.cc	55	1
jeod::SphericalHarmonicsSolidBodyTides::~SphericalHarmonicsSolidBodyTides (void)	src/spherical_harmonics_solid_body_tides.cc	65	1

Continued on next page

Table 5.9: Cyclomatic Complexity (continued)

Method	**File**	**Line**	**ECC**
jeod::SphericalHarmonicsSolidBodyTides::initialize (SphericalHarmonicsDeltaCoeffsInit & var_init, BaseDynManager & dyn_manager)	src/spherical_harmonics_solid_body_tides.cc	76	1
jeod::SphericalHarmonicsSolidBodyTides::update (SphericalHarmonicsGravityControls & controls JEOD_UNUSED)	src/spherical_harmonics_solid_body_tides.cc	93	2
jeod::SphericalHarmonicsSolidBodyTidesInit:: SphericalHarmonicsSolidBodyTidesInit (void)	src/spherical_harmonics_solid_body_tides_init.cc	44	1
jeod::SphericalHarmonicsSolidBodyTidesInit::~ SphericalHarmonicsSolidBodyTidesInit (void)	src/spherical_harmonics_solid_body_tides_init.cc	54	5
jeod::SphericalHarmonicsTidalEffects::JEOD_DECLARE_ATTRIBUTES (Planet)	src/spherical_harmonics_tidal_effects.cc	60	1
jeod::SphericalHarmonicsTidalEffects::~SphericalHarmonicsTidalEffects (void)	src/spherical_harmonics_tidal_effects.cc	83	5
jeod::SphericalHarmonicsTidalEffects::initialize (SphericalHarmonicsDeltaCoeffsInit & gen_var_init, BaseDynManager & dyn_manager)	src/spherical_harmonics_tidal_effects.cc	110	12
jeod::SphericalHarmonicsTidalEffects::update (SphericalHarmonicsGravityControls&)	src/spherical_harmonics_tidal_effects.cc	219	1

Continued on next page

Table 5.9: Cyclomatic Complexity (continued)

Method	**File**	**Line**	**ECC**
jeod::SphericalHarmonicsTidalEffectsInit::SphericalHarmonicsTidalEffectsInit (void)	src/spherical_harmonics_tidal_effects_init.cc	42	1
jeod::SphericalHarmonicsTidalEffectsInit::~SphericalHarmonicsTidalEffectsInit (void)	src/spherical_harmonics_tidal_effects_init.cc	57	1

Bibliography

[1] Gottlieb, R.G. Fast Gravity, Gravity Partials, Normalized Gravity, Gravity Gradient Torque and Magnetic Field: Derivation, Code and Data. Technical Report NASA Contractor Report 188243 (JSC 23762), National Aeronautics and Space Administration, Johnson Space Center, Houston, Texas, February 1993.

[2] Heiskanen, W. and Moritz, H. *Physical Geodesy.* W.H. Freeman and Company, San Francisco, California, 1967.

[3] Jackson, A., Thebeau, C. JSC Engineering Orbital Dynamics. Technical Report JSC-61777-docs, NASA, Johnson Space Center, Houston, Texas, July 2023.

[4] Kaula, W.M. *Theory of Geodesy: Applications of Satellites to Geodesy.* Blaisdell Publishing Company, Waltham, Massachusetts, 1966.

[5] McCarthy, D.D. and Petit, G. (eds.). *IERS Conventions(2003)*. IERS Technical Note 32, International Earth Rotation and Reference System Service, 2004.

[6] Montenbruck, O. and Gill, E. *Satellite Orbits: Models, Methods, and Applications.* Springer-Verlag, Berlin, 2000.

[7] Montenbruck, O. and Gill, E. *Satellite Orbits.* Springer, Netherlands, 2005.

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

[9] National Aeronautics and Space Administration, Johnson Space Center, Software, Robotics & Simulation Division, Simulation and Graphics Branch, 2101 NASA Parkway, Houston, Texas, 77058. *JEOD Gravity Model Reference Manual*, July 2023.

[10] Seidelmann, P.K. *Explanatory Supplement to the Astronomical Almanac.* University Science Books, Sausalito, California, 2006.

[11] Thompson, B. Gravity Gradient Torque Model. Technical Report JSC-61777-interactions/gravity_torque, NASA, Johnson Space Center, Houston, Texas, July 2023.

[12] Thompson, B., Hammen D., Jackson, A., Crues, E. Validation of Gravity Acceleration and Torque Algorithms for Astrodynamics. *Proceedings of the 18th AAS/AIAA Space Flight Mechanics Meeting*, Galveston, Texas, Jan. 2008, pp.1593-1604.

[13] Torge, W. *Geodesy.* Walter de Gruyter, Berlin, 3rd revised edition, 2001.

[14] Vallado, D.A. *Fundamentals of Astrodynamics and Applications.* Microcosm Press, El Segundo, California, 2nd edition, 2001.