JSC-61777-x.xx.xx

JSC Engineering Orbital Dynamics SPICE 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-x.xx.xx

JSC Engineering Orbital Dynamics SPICE Model

Document Revision 1.0
September 2024

Jeff Morris, Robert O. Shelton

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 SPICE Model provides equivalent functionality to the existing DE4xx ephemerides and Earth orientation (RNPJ2000) models, while expanding upon their capabilities. As those models can, it can supply high-fidelity ephemerides for the Sun, planets, Pluto, and the Moon, as well as high-fidelity RNP for the Earth and Moon. However, the SPICE Model can also provide high-fidelity ephemeris and simple orientation models for all of the planets, natural satellites (moons), asteroids, comets, and other objects and locations of interest in the solar system. Thus, the SPICE Model offers significant advantages over existing JEOD implementations of DE4xx and RNPJ2000:

- Compatibility – The SPICE Toolkit, which is implemented by the SPICE Model, is an agency standard supported and maintained by the Jet Propulsion Laboratory (JPL).
- Extended coverage – JPL offers a vast selection of data files which can provide custom ephemerides and orientation information for most objects and locations of interest in the solar system.
- Custom data – JPL offers data files which are fitted for specific applications where high accuracy or other special characteristics are required. Additionally, with the proper know-how, users can create their own SPICE-compatible data files which would then be usable with the JEOD SPICE Model.

Users are encouraged to convert existing simulations to use the SPICE Model as it may prove necessary in the future to deprecate and ultimately remove the classic JEOD implementations of DE4xx and RNPJ2000.

Contents

Chapter 1

Introduction

1.1 Model Description

The SPICE Model is a first step toward replacement of custom JEOD code with an agency standard library (SPICE). The SPICE Model provides identical functionality to the existing JEOD models with the single exception of the high fidelity JEOD model for Mars orientation. In addition, the SPICE Model includes high fidelity ephemerides and simple (International Astronomical Union) orientation models for planets, natural satellites, asteroids, comets, and many other objects and locations of interest in the solar system. Moreover, JPL maintains archives of the data files (called kernels) that drive the ephemerides and orientation models of the SPICE Model, which are updated as higher fidelity models become available. This has the effect of "future-proofing" JEOD while removing the requirement to maintain and update over 60,000 lines of code in the legacy ephemerides and RNP models.

1.2 Document History

Author	Date	Revision	Description
Jeff Morris, Robert O. Shelton	August, 2017	1.0	Initial Release

1.3 Document Organization

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

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

lists the document that is parent to this one. The final section contains a description of the how the document is organized.

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

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

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

Chapter 5: Inspections, Tests, and Metrics - The final chapter describes the procedures and results that demonstrate the satisfaction of the requirements for the SPICE Model.

Chapter 2

Product Requirements

2.1 Project Requirements

Requirement Spice_1: Top-level Requirement

Requirement:
The SPICE Model shall meet the JEOD project-wide requirements specified in the JEOD v5.2 Top-Level Document.

Rationale:
This is a project-wide requirement.

Verification:
Inspection

Requirement Spice_2: Solar System Body Ephemeris Representation

Requirement:
The SPICE Model shall provide the capability to use the JPL SPICE library to represent the ephemerides of any solar system body for which a corresponding data set is provided.

Rationale:
The existing JEOD DE4xx model is limited in the number of solar system bodies it can represent. SPICE provides a standard way of providing the same bodies, as well as virtually any other solar system body desired, so long as the data is available for it.

Verification:
Test

Requirement Spice_3: Solar System Body Orientation Representation

Requirement:
The SPICE Model shall provide the capability to use the JPL SPICE library to represent the

orientation of any solar system body for which a corresponding data set is provided.

Rationale:

Existing JEOD models provide only Earth, Moon, and Mars orientation. SPICE provides a standard way of providing these plus many others, so long as the data is available for it.

Verification:

Test

Chapter 3

Product Specification

3.1 Conceptual Design

NASA's Navigation Ancillary Information Facility (NAIF) offers NASA flight projects and NASA funded researchers an observation geometry information system called "SPICE." Among other things, SPICE provides ephemerides for hundreds of natural and man-made objects in the solar system as well as orientation data for planets and moons which have orientation models.

The JSC Engineering Orbital Dynamics (JEOD) software includes a reference frame model which relies on the concept of the frames having parents, where each frame is defined with respect to its parent. The goal then is to use SPICE to extend JEOD's built-in frame concept to any celestial object of interest in a seamless way. To this end, it is necessary to derive both the relational and mathematical components of a JEOD frame from information provided by SPICE.

3.2 Mathematical Formulations

In order for a given reference frame B (which need not initially be known to JEOD) to properly mesh with the JEOD reference frame system, it is necessary to define the following quantities:

- The parent frame, denoted generically as A. This needs to be a frame that is already part of the JEOD reference frame tree and which is also known to SPICE.
- The position and velocity of the origin of frame B expressed in A.
- T_parent_this. This is the 3×3 transformation from A to B.
- ang_vel_this. This is the angular velocity of B with respect to A, expressed in B.

3.2.1 Nomenclature

Subscripts will indicate the reference frame to which a quantity is relative and the coordinate system in which it is expressed. For example, $\vec{x}_{B|A:A}$ represents a vector position of frame B with respect to frame A expressed in frame A.

3.2.2 Relevant Functions from CSPICE

The SPICE kernels and functions provide means of determining the information necessary to define B as a JEOD reference frame. This is accomplished by interfacing with the C-language implementation of SPICE, known as CSPICE.

Translation In order to obtain $\vec{x}_{B|A:A}$ and $\dot{\vec{x}}_{B|A:A}$, the position and velocity of the origin of B expressed in A, JEOD employs the CSPICE function spkez_c. This function provides the position and velocity of a "target" with respect to an "observer" in a specified coordinate system at a given time. For JEOD, the coordinate system to use is J2000, since JEOD tracks the origins of all celestial bodies in J2000. The values $\vec{x}_{B|A:A}$ and $\dot{\vec{x}}_{B|A:A}$ are returned by the function in a single 6-array from which JEOD will then need to extract them. The call to spkez_c by JEOD is:

```
spkez_c (int Frame_B_ID,
         double ephemeris_time,
         "J2000",
         "none",
         int Frame_A_ID,
         double state[6],
         double *lt);
```

The arguments for the function call are explained as follows:

- The Frame_x_ID arguments are the SPICE-defined integer codes for A and B. For a description of these codes, see section 3.2.4.
- The "ephemeris_time" parameter is the time at which the relative state between A and B is desired, in the Barycentric Dynamic Time (TDB) time scale.
- Parameter "J2000" identifies the coordinate system in which position and velocity should be expressed.
- The "none" specifies that no aberration correction is to be applied, which is consistent with the geometric interpretation of the ephemeris understood by JEOD.
- Parameter "state" is the previously mentioned 6-array from which the position and velocity will be extracted following the function call.
- The last parameter is unused by JEOD and thus is populated by a dummy variable. It is the light travel time.

Upon completion of the call to spkez_c, JEOD then extracts the desired quantities $\vec{x}_{B|A:A}$ and $\dot{\vec{x}}_{B|A:A}$ from "state" by:

$$\begin{aligned} \vec{x}_{B|A:A} &= \text{state}[i] \text{ for } i = 0 \ldots 2 \\ \dot{\vec{x}}_{B|A:A} &= \text{state}[i] \text{ for } i = 3 \ldots 5 \end{aligned} \tag{3.1}$$

Orientation and Rotation The C implementation of SPICE (CSPICE) includes the function sxform_c which retrieves a 6×6 matrix mapping position and velocity expressed in a given frame to its representation in a different frame. This matrix accounts for orientation and relative rotation of the frames. (In the context of ephemeris, this matrix is the mapping between the standard JEOD pseudo-inertial frame (JEOD_Planet_Name.inertial) centered on a celestial body and its planet-fixed counterpart.) The sxform_c function assumes that both frames share a common origin. The relation between the frames can be written as:

$$\begin{bmatrix} \vec{x}_B \\ \dot{\vec{x}}_B \end{bmatrix} = M_{B|A} \begin{bmatrix} \vec{x}_A \\ \dot{\vec{x}}_A \end{bmatrix} \tag{3.2}$$

where $\begin{bmatrix} \vec{x}_A \\ \dot{\vec{x}}_B \end{bmatrix}$ and $\begin{bmatrix} \vec{x}_B \\ \dot{\vec{x}}_B \end{bmatrix}$ are position and velocity of a point expressed in frame A and B respectively.

The matrix $M_{B|A}$ can be partitioned into four 3×3 sub-matrices to facilitate extracting the orientation and angular velocity between the standard pseudo-inertial and planet-fixed frames of the celestial body of interest:

$$M_{B|A} = \begin{bmatrix} T_{B|A} & 0 \\ -\Omega_{B|A:B} T_{B|A} & T_{B|A} \end{bmatrix} \tag{3.3}$$

where $T_{B|A}$ is the 3×3 transformation from frame A to frame B (T_parent_this in JEOD, because in the JEOD reference frame tree the planet-fixed frame for a body is child to the standard pseudo-inertial frame for the same body), and $\Omega_{B|A:B}$ is the skew-symmetric matrix satisfying

$$\Omega_{B|A:B} \vec{x} = \vec{\omega}_{B|A:B} \times \vec{x} \tag{3.4}$$

where $\vec{\omega}_{B|A:B}$ is the vector angular velocity of B with respect to A expressed in B (ang_vel_this in JEOD). From 3.3 it follows that

$$T_{i,jB|A} = M_{i,j} \text{ for } i,j = 0 \ldots 2 \tag{3.5}$$

and

$$\Omega_{B|A:B} = -M_{i,j} T^t_{B|A} \text{ for } i = 3 \ldots 5, j = 0 \ldots 2 \tag{3.6}$$

Finally, $\vec{\omega}_{B|A:B}$ is realized as

$$\vec{\omega}_{B|A:B} = \begin{bmatrix} -\Omega_{1,2B|A:B} \\ \Omega_{0,2B|A:B} \\ -\Omega_{0,1B|A:B} \end{bmatrix} \tag{3.7}$$

The call to sxform_c by JEOD is:

```
sxform_c (char * from,
          char * to,
          double ephemeris_time,
          double trans[6][6]);
```

The parameter "from" should be set to "J2000" in order to obtain the relation between the standard pseudo-inertial frame centered at the origin of the body of interest and the associated planet-fixed frame. This is because in JEOD all standard pseudo-inertial frames are J2000-oriented.

Next, "to" is the name of B, the planet-fixed frame, which JEOD constructs consistent with SPICE nomenclature. The time parameter "ephemeris_time" is the time at which the rotation state between A and B is desired in the TDB time scale. Finally, "trans" is the previously discussed 6×6 matrix $M_{B|A}$.

3.2.3 Integration with the JEOD Reference Frame System

Following the calls to spkez_c and sxform_c described in the previous subsections, all information needed to create a new JEOD frame representing B is now available. The data members in the new Frame_B correspond to the quantities obtained in 3.1, 3.5 and 3.7 as follows:

$$\begin{aligned} \text{Frame\_B.state.trans.position} &= \vec{x}_{B|A:A} \\ \text{Frame\_B.state.trans.velocity} &= \dot{\vec{x}}_{B|A:A} \\ \text{Frame\_B.state.rot.T\_parent\_this} &= T_{B|A} \\ \text{Frame\_B.state.rot.ang\_vel\_this} &= \vec{\omega}_{B|A:B} \end{aligned} \tag{3.8}$$

3.2.4 NAIF Integer ID Codes

Every object in the SPICE inventory is assigned an integer identification code. These codes follow some simple rules which are exploited in order to determine parent-child relationships in the JEOD reference frame tree. Specifically, the solar system and planetary barycenters are numbered $0 \ldots 9$ in order of distance from the sun. For purposes of this identification system, the Pluto-Charon barycenter ID is 9. Each planet is assigned the code $x99$ where $x = 1 \ldots 9$. The Sun is numbered 10. The planets Mercury and Venus are moonless, thus there is no distinction between 1 and 199 or 2 and 299. Codes larger than 999 are reserved for comets and asteroids. Planetary moons are ordered from inner to outer, so, for example, the code for Phobos is 401, and the code for Deimos is 402. The code system treats Pluto as a planet, thus Pluto is 999 and Charon is 901. For full documentation, see NAIF Documentation.

With this information, it is possible to write simple rules to determine the code of the parent of any solar system object using the object's own code.

- $Code > 999$ (comet or asteroid): $Parent = 0$ (Solar System Barycenter).
- $Code = 199$ or 299 (Mercury or Venus): $Parent = 0$ (Solar System Barycenter).
- Otherwise, $Parent = \left\lfloor \frac{Code}{100} \right\rfloor$.

3.3 Detailed Design

3.3.1 Process Architecture

The SPICE Model is an extension of the JEOD ephemeris model framework and is organized into three primary classes. The first is SpiceEphemeris, which extends the EphemerisInterface class and thus is the orchestrator of the entire SPICE Model. The second is SpiceEphemPoint, which

is a child class of EphemerisPoint. There must be a separate SpiceEphemPoint for each celestial body for which SPICE is to update the translational state in a simulation. The final class is SpiceEphemOrientation, which extends EphemerisOrientation. It performs a similar function as SpiceEphemPoint, but for the rotational state of celestial bodies; one must exist in the sim for each body that SPICE is to keep updated with respect to rotational state.

The usual types of methods used for such things as initializing and updating the model exist for the SPICE Model, just as for most JEOD models. Since this is a model to interface with SPICE, much of the functionality involves either creating connections to SPICE or obtaining data from it in accordance with the setup implemented in the simulation. Since the SPICE Model fits into the existing JEOD ephemeris model framework, the Dynamics Manager takes care of orchestrating the state updates automatically.

3.3.2 Functional Design

This section describes the functional operation of the methods of the SPICE Model.

The SPICE Model contains the classes SpiceEphemeris, SpiceEphemPoint, and SpiceEphemOrientation. While there are several methods for each class, both new and inherited, this section will focus on only a few key SpiceEphemeris methods. Discussion of these methods will provide good insight into the inner workings of the SPICE Model. For an exhaustive treatment of the methods for all three classes, see the Reference Manual[1].

The SpiceEphemeris class contains the following methods, among others:

1. **add_planet_name**

   This is an S_define or input file method which is used to add a new SpiceEphemPoint to the SPICE Model's list of ones to keep updated.

2. **add_orientation**

   This is an S_define or input file method which is used to add a new SpiceEphemOrientation to the SPICE Model's list of ones to keep updated.

3. **initialize_model**

   This is an S_define-level method, which sets the SPICE Model up properly. It has several subroutines and performs critical tasks such as loading the given SPICE kernels, finding the desired celestial bodies and orientations within them, figuring out what barycenters need to exist in the simulation based upon the points and orientations loaded, creating said barycenters, and making connections to the appropriate ephemeris point and orientation objects in the SPICE kernels.

4. **ephem_build_tree**

   This method is called automatically by the Dynamics Manager whenever it is determined that the reference frame tree needs to be rebuilt.

5. **ephem_update**

   This method is called automatically by the Dynamics Manager and is the routine that actually performs the state updates for the loaded SpiceEphemPoints and SpiceEphemOrientations.

Chapter 4

User Guide

The User Guide chapters for many JEOD models contain sections for simulation users, simulation builders, and model extenders. This chapter will primarily describe the verification sim provided for the SPICE Model and how to obtain SPICE and the data files to use with it. Thus, most of the content is intended for simulation users. However, there is also a short section at the end of the chapter which describes briefly how to use the SPICE Model in a Trick and JEOD simulation more generally, which should be useful for simulation builders. The SPICE Model was not designed to be extended, therefore, there is no section describing extension.

The verification directory for the SPICE Model includes two simulations: the SPICE Model verification sim based on the SPICE ephemeris toolkit, and a classic JEOD DE4xx-based ephemerides sim. These exercise comparable capabilities, though it will be demonstrated how to expand SIM_spice to provide capabilities not available from the DE4xx model. Both simulations should basically run "out-of-the-box", except that the SPICE simulation ("SIM_spice") requires the C implementation of SPICE library, named CSPICE, in order to run.

4.1 The CSPICE Library

SIM_spice requires an external library in order to access the capabilities of SPICE. In order to run SIM_spice, it is necessary to download and compile the SPICE toolkit, then define the environment variable "JEOD_SPICE_DIR" to point to the directory where it was compiled.

The default build for the SPICE toolkit creates the static library

`${JEOD_SPICE_DIR}/lib/cspice.a.`

SIM_spice includes a file containing make overrides (S_overrides.mk) which uses the environment variable `$JEOD_SPICE_DIR` to locate the SPICE library. Unfortunately, the default build process for SPICE does not prepend the "lib" prefix to the name of the library file cspice.a. To fix this, either rename the file or provide an appropriately named symbolic link. The shell commands for each alternative are given below.

```
mv ${JEOD_SPICE_DIR}/lib/cspice.a ${JEOD_SPICE_DIR}/lib/libcspice.a
```

```
ln -s ${JEOD_SPICE_DIR}/lib/cspice.a ${JEOD_SPICE_DIR}/lib/libcspice.a
```

4.2 Obtaining SPICE kernels

This section describes the process used to obtain SPICE data files such as the ones for Sun, Earth, Moon, and Mars used in SIM_spice. Two major sources of SPICE data will be discussed: NAIF and Horizons.

4.2.1 NASA's Navigation and Ancillary Information Facility (NAIF)

Data files which are used by the SPICE toolkit are called "kernels". To obtain many pre-made kernels including those for the Sun and planets, go to the NAIF homepage. Follow the links to Data → Generic Kernels → Generic Kernels. The resulting page should show a list of directories and a file named "aareadme.txt". At this point it would be a good idea to read the file, which contains a concise explanation of the various generic spice kernels.

Kernels for planetary ephemerides which describe translational motion of astronomical objects are referred generically in SPICE as "spk" (SPICE Planetary Kernel) files. There are special high-fidelity versions of such files which are in binary format and have the extension ".bsp". The spk file used in SIM_spice for ephemerides of the Sun and planets is named "de421.bsp" and can be found in the directory spk/planets/a_old_versions. The DE421 ephemeris file was used in order to provide a consistent comparison with what is available with the classic JEOD ephemerides model, but any other file containing analogous data could have been used.

Kernel files for natural satellites of other planets can be found in the folder spk/satellites. For example, data for Phobos and Deimos, the moons of Mars, are contained in the file "mar097.bsp." Planets with large moon systems are broken into separate files; however, since Mars has only two moons, there is only a single file for the Martian system. The "readme" files in that folder can assist in determining which files are needed for a given object.

Besides de421.bsp, the other files needed for SIM_spice are the high-precision orientation models for Earth and Moon. These can be found in the "pck" directory on the "Generic Kernels" page. The specific files used are "earth_000101_171024_170805.bpc" and "moon_pa_de421_1900-2050.bpc." These files provide the high-fidelity orientation models for Earth and Moon. The directory contains other versions which may be a better match for some situations. The file "aareadme.txt' contains descriptions of each file which can aid in determining which best suits the desired application. The files used with SIM_spice were chosen because they provide capabilities similar to JEOD's existing RNP model.

Note that the text kernels "pck00010.tpc" and "moon_080317.tf" are also found in this directory. "moon_080317.tf " includes definitions for standard lunar reference frames while "pck00010.tpc" contains low fidelity (IAU) orientation models for planets and natural satellites in the solar system. The orientation model for Mars that SPICE offers is one of these relatively low fidelity IAU models; however, it agrees reasonably well with the JEOD model for Mars orientation.

4.2.2 The Jet Propulsion Laboratory Horizons System

While the NAIF website includes ephemerides for Sun, Planets, natural satellites, and barycenters, as well as orientation models for planets and satellites, it is generally necessary to obtain kernel files for objects such as comets and asteroids from the JPL Horizons system. There are both web (Horizons main page) and telnet interfaces to Horizons; however, high-fidelity binary files are only available through the telnet interface. The following example describes the process for obtaining a high-fidelity ephemeris file for the asteroid Itokawa.

First type the following from the command line:

```
telnet ssd.jpl.nasa.gov 6775
```

The first task is to find the body of interest. In the case of Itokawa, this is very easy. Just type "Itokawa" at the "Horizons>" prompt. Once Horizons finds Itokawa, press Return to confirm. The screen will fill with information about Itokawa, and at the bottom the user will be prompted for input. Type "s" to request an "spk" file. The next step will request an email address. Enter it and confirm. Then, Horizons will ask whether a text file is desired. Since binary is desired instead, respond with "n".

Next, the start and stop dates for the file are needed. For instance, SIM_spice starts and ends on Jan. 30, 2009, so any range which includes that day would be fine.

Finally, the system creates a binary spk file and provides instructions on how/where to retrieve it by anonymous ftp. The name of the file is a somewhat random string of characters; the user will likely find it helpful to rename it something more human-friendly after retrieval, such as "itokawa.bsp". Follow the instructions and retrieve the file.

4.3 SIM_spice

Once SPICE has been downloaded and installed, SIM_spice is ready to be run. The simulation directory already contains the SPICE kernels needed to operate the basic simulation which includes Sun, Earth, Moon, and Mars. These files were retrieved using the process described in section 4.2.1.

The standard RUN case delivered with SIM_spice is named "RUN_01". That run will create files that log the position and velocity of Sun, Earth, Moon and Mars in Solar System Barycenter coordinates. The rotational states of Earth, Moon, and Mars are also logged in their respective files. The exact same information is logged by SIM_de4xx and can be used for comparison.

The generalized version of SIM_spice includes two bodies, Phobos and Itokawa, which are not part of the JEOD DE4xx model. In order to run the generalized sim, one merely needs to uncomment a few lines in the input deck and download the necessary files from NAIF and Horizons. The appropriate file for Phobos is named "mar097.bsp" and is located in the directory "spk/satellites" on the NAIF Generic Kernels page; see section 4.2.1 for further information on retrieval. The appropriate file for Itokawa can be retrieved from the JPL Horizons system; the process is thoroughly described in section 4.2.2.

Once those files are retrieved, place them in the data directory of SIM_spice. Then open the file SET_test/RUN_01/input.py and uncomment line 19. The additional bodies will now be actively

updated upon subsequent re-runs of SIM_spice.

(Aside: If one is interested in seeing how the SPICE files are unpacked and the resulting JEOD reference frame tree is built, then uncomment line 7 of SET_test/RUN_01/input.py. Note that doing so uncorks a spew of other debug output from JEOD as well, so it might be the kind of thing best left for debugging purposes.)

4.4 Using SPICE in a Trick Simulation

In order to have planets or other celestial bodies in a Trick simulation, one should instantiate a SimObject for each object. The typical architecture includes the following:

- A SimObject to manage the overall environment.
- Planetary SimObjects, one for each celestial body, containing Planet and GravitySource classes to represent the associated celestial body's reference frames, characteristics, and gravity.
- "Default data" classes to initialize the Planet and GravitySource objects within each planetary SimObject with the characteristics of the associated celestial body.

4.4.1 Planetary SimObjects

Planetary SimObjects are the S_define level classes which encapsulate a planet or other celestial object. JEOD provides several predefined S_modules of this sort for Earth, Moon, Mars, and the Sun; they can be found in `${JEOD_HOME}/lib/jeod/JEOD_S_modules`.

In addition to the standard JEOD S_modules for Earth, Moon, Sun, and Mars, SIM_spice also includes two custom S_modules, itokawa_basic.sm and phobos_basic.sm, which are used in the generalized simulation. These S_modules were built by inheriting from a generic planet module found in the directory `${JEOD_HOME}/lib/jeod/JEOD_S_modules/Base` which is there for that purpose. Thus, itokawa_basic.sm and phobos_basic.sm are excellent prototypes for any new S_modules that may need to be created for celestial objects driven by SPICE.

4.4.2 Planetary Initialization Classes

JEOD provides default data classes to initialize Planet and GravitySource objects for Earth, Moon, Sun, Mars, and Jupiter. These standard JEOD initialization classes are found in the "data" directories for the planet and gravity models respectively. For the purposes of demonstrating the capabilities of the SPICE model via SIM_spice, analogous initialization classes were created for Phobos and Itokawa. However, these classes are only for demonstration and should not be used when accurate planet or gravity models are needed.

4.4.3 Working with SPICE Ephemerides

Starting with JEOD version 3.3, the standard JEOD module library includes an S_module named environment_spice.sm which defines an environment SimObject for the SPICE ephemerides. Thus, including SPICE in a sim is as simple as adding the line

```
#include "JEOD_S_modules/environment_spice.sm"
```

to an S_define file.

Unlike the classic DE4xx model, the SPICE Model has no concept of activation or deactivation of an individual celestial object once declared. If a SPICE ephemeris object exists in a sim, then SPICE assumes that the object should be active. So, all that is necessary in order for a sim to contain celestial bodies driven by the SPICE Model is to define their S_modules, and include environment_spice.sm as above.

The rest of what is needed is supplied in the input deck. The following examples assume one is using the standard JEOD S_module environment_spice.sm, which instantiates a SimObject named "env" containing the data member "spice", which is of type "SpiceEphemeris" (i.e., the SPICE Model).

First, introduce the kernel files downloaded from NAIF, containing data for Sun, Earth, Moon, and Mars:

```
env.spice.metakernel_filename = "data/kernels.tm"
```

The file kernels.tm is a so-called "metakernel" file, meaning it contains a list of kernels for SPICE to load. This file can be used as a template for tailoring to one's own needs. For the generalized sim containing Phobos and Itokawa, please refer to the file data/more_kernels.tm.

Next, load the planetary ephemerides (see Modified_data/spice.py):

```
env.spice.add_planet_name("Sun")
env.spice.add_planet_name("Earth")
env.spice.add_planet_name("Moon")
env.spice.add_planet_name("Mars")
```

Finally, request orientation calculations for Earth, Moon, and Mars:

```
env.spice.add_orientation("Earth")
env.spice.add_orientation("Moon")
env.spice.add_orientation("Mars")
```

Chapter 5

Inspections, Tests, and Metrics

5.1 Inspection

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

Inspection Spice_1: Top-level Inspection

This document structure, the code, and associated files have been inspected, and together satisfy requirement Spice_1.

5.2 Tests

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

Test Spice_1: Ephemeris and Rotation

Background
The purpose of this test is to demonstrate the ability of the SPICE Model to represent both the ephemerides and orientation of any solar system body.

Test description
This test utilizes two simulations – one that utilizes the SPICE Model for ephemeris and rotational state representation of several solar system bodies, and one that uses the legacy JEOD De4xx ephemeris and RNP models for analogous calculations for a subset of the same bodies. The results of the first will be compared against the second for the bodies that are common to both, in order to build confidence in the correct operation of the SPICE Model even for the bodies that are not.

The SPICE-only simulation contains the solar system bodies Sun, Earth, Moon, Mars, Itokawa (an asteroid), and Phobos (a moon of Mars). It can provide ephemeris data for all of them, as well as rotation data for Earth, Moon, and Mars.

The simulation using legacy JEOD models contains the bodies Sun, Earth, Moon, and Mars. It provides ephemeris for all of them, along with rotation for Earth, Moon, and Mars. The legacy JEOD models are unable to provide ephemeris or rotation information for Itokawa or Phobos, which illustrates the primary benefit of the new SPICE Model: the ability to include many more solar system bodies in a simulation.

Test directories `SIM_spice and SIM_de4xx`

Success criteria

A comparison of the output of the SPICE-only simulation with the legacy simulation should show good agreement between the two for both translational and rotational states of the bodies being compared. Due to the nature of the data and models being employed, translation should agree to within approximately machine precision, while rotation will be less accurate but still agree to near machine precision.

Test results

All output data confirmed expectations.

Applicable Requirements

This test satisfies the requirements Spice_2 and Spice_3.

5.3 Requirements Traceability

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

Table 5.1: Requirements Traceability

Requirement	**Traces to**
Spice_1 Top-level Requirement	Insp. Spice_1 Top-level Inspection
Spice_2 Solar System Body Ephemeris Representation	Test Spice_1 Ephemeris and Rotation
Spice_3 Solar System Body Orientation Representation	Test Spice_1 Ephemeris and Rotation

5.4 Metrics

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

Table 5.2: Coarse Metrics

	Number of Lines			
File Name	**Blank**	**Comment**	**Code**	**Total**
include/spice_ephem.hh	60	153	84	297
include/spice_ephem_orient.hh	21	80	27	128
include/spice_ephem_point.hh	25	89	33	147
src/cmake_file_list.cmake	2	0	11	13
src/spice_ephem.cc	142	322	540	1004
src/spice_ephem_orient.cc	25	68	52	145
src/spice_ephem_point.cc	18	53	34	105
Total	293	765	781	1839

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

Table 5.3: Cyclomatic Complexity

Method	File	Line	ECC
jeod::SpiceEphemeris::add_planet_name(std::add_planet_name (std::string planet_name)	include/spice_ephem.hh	122	1
jeod::SpiceEphemeris::add_orientation(std::add_orientation (std::string object_name)	include/spice_ephem.hh	128	1
jeod::SpiceEphem Orientation::std::set_spice_frame_name (const std:: string & new_name)	include/spice_ephem_orient.hh	103	1
jeod::SpiceEphemeris::Spice Ephemeris ()	src/spice_ephem.cc	61	1
jeod::SpiceEphemeris::~Spice Ephemeris ()	src/spice_ephem.cc	73	3
jeod::SpiceEphemeris:: activate ()	src/spice_ephem.cc	101	2
jeod::SpiceEphemeris:: deactivate ()	src/spice_ephem.cc	117	1
jeod::SpiceEphemeris:: timestamp ()	src/spice_ephem.cc	125	1
jeod::std::get_name ()	src/spice_ephem.cc	134	1
jeod::SpiceEphemeris:: initialize_model (const Time Manager & time_manager, EphemeridesManager & ephem_manager)	src/spice_ephem.cc	143	2
jeod::SpiceEphemeris::simple_restore ()	src/spice_ephem.cc	188	1
jeod::SpiceEphemeris:: initialize_time (const Time Manager & time_manager)	src/spice_ephem.cc	196	2
jeod::SpiceEphemeris::load_spice_files ()	src/spice_ephem.cc	223	2
jeod::SpiceEphemeris:: process_spk ()	src/spice_ephem.cc	249	9

Continued on next page

Table 5.3: Cyclomatic Complexity (continued)

Method	File	Line	ECC
jeod::SpiceEphemeris:: process_orientations ()	src/spice_ephem.cc	398	3
jeod::SpiceEphemeris:: populate_item (Ephemeris Item & item, const std:: string & object_name)	src/spice_ephem.cc	414	1
jeod::SpiceEphemeris::create_ new_ephem_point (std:: string object_name, const std::string & spice_name)	src/spice_ephem.cc	426	2
jeod::SpiceEphemeris::create_ new_ephem_orientation (std::string jeod_name)	src/spice_ephem.cc	463	1
jeod::SpiceEphemeris:: initialize_items ()	src/spice_ephem.cc	479	4
jeod::SpiceEphemeris:: introduce_item (Ephemeris Item & item)	src/spice_ephem.cc	514	2
jeod::std::spice_2_jeod (std:: string spice_name)	src/spice_ephem.cc	537	1
jeod::std::jeod_2_spice_pfix (std::string jeod_name)	src/spice_ephem.cc	552	3
jeod::SpiceEphemeris::name_ barycenter_frames ()	src/spice_ephem.cc	572	2
jeod::SpiceEphemeris::add_ barycenter (int id)	src/spice_ephem.cc	597	2
jeod::SpiceEphemeris::create_ barycenters ()	src/spice_ephem.cc	623	6
jeod::SpiceEphemeris:: determine_root_node ()	src/spice_ephem.cc	670	5
jeod::SpiceEphemeris::ephem_ initialize (Ephemerides Manager &)	src/spice_ephem.cc	725	1
jeod::SpiceEphemeris::ephem_ activate (Ephemerides Manager &)	src/spice_ephem.cc	740	1
jeod::SpiceEphemeris::add_ descendants_r (SpiceEphem Point * parent)	src/spice_ephem.cc	752	4

Continued on next page

Table 5.3: Cyclomatic Complexity (continued)

Method	File	Line	ECC
jeod::SpiceEphemeris::ephem_build_tree (Ephemerides Manager & ephem_manager)	src/spice_ephem.cc	784	4
jeod::SpiceEphemeris::ephem_update ()	src/spice_ephem.cc	832	4
jeod::SpiceEphemeris::find_spice_id (int id_to_find)	src/spice_ephem.cc	852	3
jeod::SpiceEphemeris::find_parent_id (int obj_id)	src/spice_ephem.cc	871	4
jeod::SpiceEphemeris::update_trans ()	src/spice_ephem.cc	903	6
jeod::SpiceEphemeris::update_rot ()	src/spice_ephem.cc	964	2
jeod::SpiceEphemeris::mute_spice_errors ()	src/spice_ephem.cc	975	1
jeod::SpiceEphem Orientation::SpiceEphem Orientation ()	src/spice_ephem_orient.cc	41	1
jeod::SpiceEphem Orientation::update (double time_tdb, double time_dyn)	src/spice_ephem_orient.cc	49	3
jeod::SpiceEphem Orientation::validate (double time_tdb)	src/spice_ephem_orient.cc	95	1
jeod::SpiceEphem Orientation::get_spice_transformation (double time_tdb, double trans6x6[6][6])	src/spice_ephem_orient.cc	110	2
jeod::SpiceEphemPoint::Spice EphemPoint ()	src/spice_ephem_point.cc	37	1
jeod::SpiceEphemPoint::set_status (SpiceEphemPoint:: Status new_status)	src/spice_ephem_point.cc	45	1
jeod::SpiceEphemPoint::get_status ()	src/spice_ephem_point.cc	54	1
jeod::SpiceEphemPoint::set_spice_id (int new_id)	src/spice_ephem_point.cc	63	1

Continued on next page

Table 5.3: Cyclomatic Complexity (continued)

Method	File	Line	ECC
jeod::SpiceEphemPoint::get_spice_id ()	src/spice_ephem_point.cc	72	1
jeod::SpiceEphemPoint::set_parent_id (int new_id)	src/spice_ephem_point.cc	81	1
jeod::SpiceEphemPoint::get_parent_id ()	src/spice_ephem_point.cc	90	1

Bibliography

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

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