Exercises Manual

to accompany

JEOD Training Course

for JEOD version 5.0

Gary Turner (updated by Christopher Sullivan)
Odyssey Space Research
August, 2016 (updated July, 2022)

Introduction

As users work through the course presentation, they will periodically be referred to this Exercise Manual.

In the electronic course materials, there is a folder called *Exercises.* Most of the exercises in this manual involve building simulations. Inside the Exercises folder are a few directories that contain some hints and files to get you started on the early exercises. Each exercise identifies the directory in which the user should be working. Because many of these simulation-building exercises build on previous exercises, it is important to work in the specified directory in order to be able to find your work again when it is needed for later exercises.

Inside the *Exercises* folder, you should also find the *Solutions* folder. Inside *Solutions* is a solution manual, along with all the necessary simulation and data files. These solutions can be useful for:

- Checking your work for recommended good practice. There are often multiple ways to solve each problem, not all equally efficient.
- Getting hints when you get stuck. The solution manual in particular steps through the process for building the S_define and data files for each problem.
- For getting reliable foundational material for later problems. Especially if you are jumping into a topic in the middle of the course, it can be frustrating to have to build an entire simulation from scratch. Building off a solid foundation allows users to focus their attention on integrating the model at hand.
- For quickly grabbing ready-made simulations for testing code outside of this course. If a user needs, for example, an Earth-Moon-Sun simulation to test a navigation algorithm, the appropriate solution can be ripped from these course materials.

Warning about availability of Solutions - there is very limited educational benefit from you copy-and-pasting the solutions. Try not to look at the solutions for the problem you are currently working unless you get stuck.

Table of Contents

Exercise 1 - Familiarity with the S_define

If you are unfamiliar with Trick, review the Trick tutorial. This course will depend heavily on Trick-based simulations. The exercise solutions for the JEOD 5.0 course use Trick-19.

If your familiarity with Trick is a little rusty, review the Trick Primer (Appendix A), or the Trick tutorial.

Open the S_define file in Exercises/SIM_01. For the purpose of this exercise, ignore the two system modules, *default_trick_sys.sm* and *jeod_sys.sm*

1. Identify the Simulation objects
2. Review the contents of the simulation modules included in the S_define.
3. Order the function calls (only those that are called directly from the S_define) based on the sequence in which they are called. Indicate where input-file data and Modified data get introduced. Stop when all of the regularly scheduled jobs have been run at least twice.

Exercise 2 - Investigating Abstract Classes

This exercise uses the Time model, in *models/environment/time*

1. Identify the abstract classes and the pure virtual methods in the Time model.
2. Identify the classes that inherit from the abstract classes.
3. TimeStandard is used as a basis for multiple clocks, can a TimeStandard be instantiated, or just its subclasses?

Exercise 3 - Tracing Inheritance

This exercise uses the BodyAction model, in *models/dynamics/body_action*

A 2-vehicle attach process can be scheduled with the use of a *BodyAttachAligned* instance(this is found in dynamics/body_action).

This class has a data element called *subject.core_properties.inertia.*

1. Trace this element, and identify what it represents.

Exercise 4 - Identifying the Integration Algorithm

Identify the type of integration / name of the integration algorithm that will be used for integrating the state of the vehicle in Exercises/SIM_04.

Exercise 5 - Simple Simulation

This exercise quickly produces a working simulation. Review the S_define provided in Exercise 1 for hints.

NOTE – these early simulations will produce a warning message when they run. The warning message is alerting us to the fact that we do not have a gravity model, a basic inclusion in any realistic JEOD simulation. For these early tutorial-simulations, we do not need a gravity model so the warning may be ignored.

Work in Exercises/SIM_05.

1. Create a new S_define file in Exercises/SIM_05. Using the JEOD S_modules, create a new minimal simulation that is capable of integrating a vehicular state.
   (a) HINT – need only minimal time capabilities, a dynamics object that is capable of integrating state, a vehicle, and an integration statement.
2. Open the input file in SIM_05/RUN_test:
   (a) Change the vehicle mass to 500 kg.
   (b) Start the vehicle with an initial state of:
      i. position: (10, 0, 0) m
      ii. velocity: (-2, 0, 0) m/s
      iii. Orientation: Some rotation about some axis
      iv. Angular velocity: such that it returns to 0 orientation at t=5 seconds.
3. Compile and run your simulation.
4. Verify that the transformation matrix is, indeed, identity at t = 5s.

Exercise 6 - Simple Simulation, adding clocks

Starting from SIM_05, this exercise adds the concept of time.

Although it is possible to just add to your existing simulation, we are going to maintain a new S_define for each exercise.

1. Copy the contents of SIM_05 into SIM_06. Go to SIM_06
2. Edit the S_define to add:
   (a) a TAI clock
   (b) a UTC clock, with calendar representation.
3. Edit the RUN_05/input.py file so that it runs for at least one hour (change the stop time)
4. Edit the input file to start the simulation at some UTC time; remember to add the clock dependencies.
   (a) **Hint:** TAI can update from Dyn, UTC from TAI
5. Edit the log data files to add the UTC calendar outputs for seconds, minutes, and hours.

6. Run your simulation, check that the minute and hour values are discrete, and cycling appropriately.

Exercise 7 - Two vehicle Simulation

Starting from SIM_05, this exercise adds a second vehicle.

1. Copy the contents of SIM_05 into SIM_07. Go to SIM_07
2. Add a second vehicle to your simulation.
3. Configure the vehicle, specifying its inertia tensor relative to the vehicle's structural axes.
4. Experiment with initial conditions.
5. Plot the translational states of both vehicles on the same graph.

Exercise 8 - Creating a Simple Spherical-Earth Orbiter

Starts with Exercise SIM_05. Work in SIM_08. This simulation should put a vehicle in an Earth orbit, with the only environmental effects being Earth gravity.

1. Copy the contents of SIM_05 into SIM_08 (Note – SIM_08 has a file already in its Modified_data directory, do not overwrite this file). Go to SIM_08.
2. Rename RUN_test to RUN_Spherical
3. Add an environment simulation-object to the S_define, include the appropriate models.
4. Add a planet simulation-object to the S_define to represent Earth. Use a spherical gravity field.
5. Remember to edit the DynManagerInit mode in the input file. Which mode should we be working in?
6. Add a file to to Modified_data directory for defining gravity controls for your vehicle, and include it from the input file found in RUN_Spherical.
7. In the new file in Modified data, define your gravity controls such that the vehicle sees spherical Earth gravity only.
8. Initialize your vehicle state by including *Modified_data/veh_state_ex8.py* from your input file (Note – you do not need to understand what is in *veh_state_ex8.py*, it will be covered later). Be sure to delete any old vehicle-state information from your input file.
9. Compile your simulation and run it for 6000 s.

Exercise 9 - Creating a Simulation with Two Earths

Starts with Exercise SIM_08. Work in SIM_09. This is a designed-to-fail exercise. As we progress into more challenging terrain, you should be made aware that behind the scenes, JEOD is performing multiple verifications that your simulation is constructed in a physically meaningful way. While the

compilation of the simulation checks that the code is legal, it cannot interpret whether the simulation is structured appropriately. Hopefully, after this exercise, you will be more comfortable with the idea that most of the easy-to-make errors in building a simulation have been anticipated, and somewhat meaningful error messages built into JEOD to help you identify what you did wrong.

1. Copy the contents of SIM_08 into SIM_09 and go to SIM_09.
2. After the earth S_module has been defined, instantiate another instance of *EarthSphericalSimObject (*note – one instance is instantiated automatically inside the S_module). The second instance of the simulation-object must have a different name.
3. Both of your simulation-objects will have a Planet object *(planet*) with the default name *Earth* (this is populated from default data). Run the simulation.
4. In the input file, change the name one of your planets to *Erth*, or some other misspelling, or some fictional planet name. Run the simulation.

Exercise 10 - Complications of Adding a General Gravity Model

Starts with Exercise SIM_08. Work in SIM_10. This simulation replaces the spherical gravitational-field model with a non-spherical gravitational-field model.

WARNING!!!

A SIMPLE DROP-IN REPLACE OF THE GRAVITY FIELD IS INVALID.

THE SIMULATION WILL NOT BUILD (or if it does, it will not be properly configurable)

CAN YOU DETERMINE WHY?

1. Copy the contents of SIM_08 into SIM_10 and go to SIM_10.
2. Edit the S_define to replace “earth_basic.sm” with “earth_GGM05C.sm”.
3. Try to build this simulation. Can you identify why it failed? Here are the errors:
   (a) use of undeclared identifier LOW_RATE_ENV
   (b) ‘LOW_RATE_ENV’ was not declared in this scope
   (c) ‘class JeodTimeBaseSimObject’ has no member named ‘time_tt’
   (d) ‘class JeodTimeBaseSimObject’ has no member named ‘time_ut1’
   (e) ‘class JeodTimeBaseSimObject’ has no member named ‘time_gmst’

Explanation of error messages:

1. The non-spherical Earth Gravity Model is planet-fixed.
   (a) That is, the gravity field rotates with the planet!
   (b) Hence, the Gravity Model needs a planet-fixed reference frame.
   (c) Note that this is unnecessary for spherical gravity because of the spherical symmetry.
2. We have an earth-based planet-fixed frame, *pfix*, already in the simulation. However, until the simulation provides data on its orientation, it cannot be used correctly. The model that is

needed to initialize and update the orientation of the *pfix* frame is called Rotation, Nutation, and Precession, or RNP.

3. Therefore, correctly usage of the GGM05C gravity model requires that the RNP model also runs at the same time. So the RNP model is included in the GGM05C simulation-object. However, the RNP model requires additional settings, including a setting for the rate at which its full update should be run (LOW_RATE_ENV), and the clocks UT1, TT, and GMST.
4. Just dropping in the earth_GGM05C.sm sim-object does not provide these additional capabilities. We need to do a little more work to get the RNP model up and running first. That is the next topic.

Exercise 11 - Creating a Spherical-Earth Orbiter using a Non-spherical Gravity model

Continues from the work in SIM_10.

1. Copy the contents of SIM_10 into SIM_11. Go to SIM_11
2. Replace the time-model instance with one that provides – at a minimum – UT1, TT, and GMST.
3. Define a rate for low-rate-environment jobs. This is the rate at which the slowly-evolving environment can be updated. For the RNP model, this includes the deltas to earth-orientation resulting from Nutation, Precession, and Polar Motion (but not Rotation, which is handled at a higher rate). A rate of 100s should be more than adequate.
4. Compile and run the simulation.
5. You should notice that there are two more things we missed so far.
   (a) The RNP model adds earth-rotation as a derivative-class job, so earth must now be added to the integration loop to get that job correctly scheduled. Go ahead and add it and recompile.
   (b) The addition of absolute clocks requires the specification of the absolute time at which the sim is to be run. You will need to initialize the time.
6. Compare the results to those from SIM_08. The difference is that SIM_08 used a spherical gravity model, while this sim uses a spherical setting of a more capable non-spherical gravity field.

Exercise 12 - Creating a Non-spherical-Earth Orbiter

Starts with Exercise SIM_11. Work in SIM_12.

In exercise 11, we added the non-spherical gravity model, but only used it on its simplest setting – spherical gravity. In this exercise, we will modify the settings to fully activate the non-spherical components.

1. Copy SIM_11 to SIM_12. Go to SIM_12.
2. Copy the gravity controls file to a new file in Modified_data, and adjust it to make the gravity

non-spherical, 4x4 field.

3. Create a new directory, RUN_4x4 and:
   (a) copy the input file over from RUN_Spherical,
   (b) edit the input file to include your new gravity controls file.
4. Make a new RUN directory, RUN_200x200, change the gravity to 200x200.
5. Run the spherical, 4x4 and the 200x200 cases. Note – this run should take a few seconds longer than the others.
6. Compare your positional data from the three runs to see the significance of the non-spherical components of the field.

Exercise 13 - Multiple Gravitational Fields

Starts with Exercise SIM_12. Work in SIM_13.

1. Copy the contents of SIM_12 to SIM_13. Go to SIM_13
2. Add 2 simulation objects, comparable to *earth*, called *sun*, and *moon*. Both *sun* and *moon* will only need spherical gravity for this exercise.
3. Add the Ephemeris model to your simulation so that it keeps track of the relative positions of the three Planet objects.
   (a) HINT Be careful! While replacing the sim-module is easy, correct operation of the extended sim-module has multiple dependencies that the old one did not have. Think about which mode you should be running in, and what is required to run in that mode.
4. Edit your vehicle object to add gravity controls for the 2 new Planet objects.
   (a) HINT There is no pre-configured simulation-module for this one!
5. You should have RUN_directories copied in for spherical Earth gravity, 4x4 Earth gravity and 200x200 Earth gravity. Make additional RUN directories for each of the following cases:
   (a) 4x4 Earth and spherical Sun,
   (b) 4x4 Earth and spherical Moon,
   (c) 4x4 Earth, spherical Sun and spherical Moon,
   (d) 200x200 Earth, spherical Sun, and spherical Moon.

   (You should now have 7 run directories)
6. Either create new gravity controls files, or edit directly in the input files to specify the appropriate gravity controls per run situation.
   (a) So, for the Earth and Sun run, make Moon inactive, etc.
   (b) Remember to add your new gravity controls to the *vehicle.dyn_body.grav_interaction* list.

7. Compare your positional data for the following pairs:
   (a) 4x4 Earth alone versus 4x4 Earth with Sun
   (b) 4x4 Earth alone versus 4x4 Earth with Moon
   (c) 4x4 Earth with Sun and Moon versus 4x4 Earth with Sun only
   (d) 4x4 Earth with Sun and Moon versus 4x4 Earth with Moon only
   (e) 200x200 Earth alone versus 200x200 Earth with Sun and Moon

Exercise 14 - Lunar Orbiting Vehicles

Stand-alone simulation.

1. Create a new simulation, at SIM_14
2. Put a vehicle in orbit around the moon. Implement a simple spherical lunar gravity model, then add terrestrial and solar gravity (as perturbing effects).

   NOTE - because the moon accelerates so rapidly (so is a very poor approximation to an inertial reference frame), it is strongly recommended to add at least terrestrial gravity to all lunar simulations, unless running short-duration simulations near the surface. It is also a good idea to make the Ephemerides model run at the derivative rate to keep the relative positions of Earth and Moon as up-to-date as possible.

Exercise 15 - Gravity Gradient

Start from Exercise SIM_08. Work in SIM_15.

1. Copy the contents of SIM_08 to SIM_15. Take care not to overwrite the two files in the Modified_data folder.
2. Add gravity gradient to the simulation
3. Turn gravity gradient on in the controls.
4. Make two runs of the simulation, with the mass properties for the vehicle populated by each of the two mass properties files in Modified_data. Name the two runs appropriately.
5. Contrast the evolution of the state of the vehicle between runs to observe the effect of gravity gradient torque.

Exercise 16 - Solid-Body Tides

Start from Exercise SIM_12. Work in SIM_16.

1. Copy the contents of SIM_12 to SIM_16. Go to SIM_16.

2. Add the Solid-Body Tides model to the appropriate simulation-objects.
3. Make the gravity field 8x8 (or any other resolution – other than spherical)
4. Make two runs of the simulation, one with the solid-body tides on and one off.
5. Compare the evolution of the state of the vehicle between runs to observe the effect of solid-body tides.

Exercise 17 - Interpretation of Variables

During a simulation, the following values are logged. Assume there is a *DynBody* object called *d_body*, and a *MassBody* object called *m_body* in the vehicle simulation-object

vehicle.d_body.composite_body.state.trans.position has value [1.0, 0.0, 0.0]

vehicle.d_body.mass.composite_properties.position has value [-1.0, 0.0, 0.0]

vehicle.m_body.composite_properties.position has value [-1.0, 0.0, 0.0]

1. What do these variable values tell you about the objects in the simulation?
2. Can you determine the relative positions of *m_body* and *d_body*?
3. Can you determine the relative orientation of the structure_points of *m_body* and *d_body?*

Exercise 18 - Adding Mass Points and Internal Reference Frames

Starting from SIM_07, this exercise adds a mass-point to each of the two vehicles.

1. Copy the contents of SIM_07 to SIM_18. Go to SIM_18.
2. Add a Mass Point to the first vehicle, with the following parameters:
    1. position at (-1.0, 0.0, 0.0) m in the structural frame
    2. Oriented with a 180-degree rotation on the structure-frame z-axis, relative to the structure-frame.
3. Add a Mass Point to the second vehicle, with the following parameters:
    1. Position at (0.0, 1.0, 0.0) m in the structural frame
    2. Oriented with a 90-degree rotation on the structural-frame z-axis, relative to the structural-frame.
4. Set the orientation of the body-frame of the first vehicle to be equivalent to the structure-frame of the same.
5. Set the orientation of the body-frame of the second vehicle to be a 90 degree rotation about the negative-z-axis of the structure frame of the same.
6. Set the positions of the body-frame to be equal to those of the structure-frame for both vehicles.

7. Output the position and orientation of the mass point of the second vehicle expressed in and relative to the body frame of the same.
    1. Note – this part is very difficult. The solutions manual contain a hint before providing the solution.

Exercise 19 - Interpreting the Mass Tree - Depleting a Tank Mass

Stand-alone simulation. The simulation provided in SIM_19 represents two attached vehicles, A and B, with B attached to A. Each vehicle has an attached *MassBody* representing its fuel tank, called *tank.*

1. Sketch the mass tree for this configuration, and calculate the core and composite masses of the four masses in the simulation.
2. Print the mass tree to confirm your values.
3. Change the mass of *tankB* to 50% of its original value. Update the mass properties throughout all masses in the simulation
4. Verify that the mass properties have updated appropriately.

Exercise 20 - Exercise with Mass Points and Body Ref Frames

Starting from SIM_18, this exercise computes the state of the reference frames associated with the mass points added in SIM_18.

1. Copy the contents of SIM_18 to SIM_20. Go to SIM_20.
2. To make this a little easier, change the vehicle states to stop them from rotating.
3. Compute the initial state of the body-frame of the second vehicle relative to, and expressed in, the body frame of the first vehicle.
4. Using these data, manually predict the translational state of the added mass-point on the second vehicle relative to (and expressed in):
    1. body frame of the first vehicle
    2. the mass-point added to the first vehicle.
5. Verify that when adding the Mass Points in Exercise 18, they were added as BodyRefFrame instances by computing the state of the additional mass-point on the second vehicle relative to the additional mass-point on the first vehicle.
6. Compare your manually-calculated answer from part (4) to the JEOD-generated answers for the same relative states.

Exercise 20.1 - Sensors, Body Reference Frames, and Mass Points

The first half of this exercise was rather abstract, but now we will see the same principles in action in more realistic application

1. Create a new RUN directory, RUN_sensors. For this run, set *vehicle* rotating again.
2. Copy the contents of *Modified_data* to a new directory *Modified_data_sensors*. We will work with these Modified-data files for the second half of his exercise.
3. Name the mass-point on *vehicle2, "reflector"*.
4. Make case-frames:
    1. Make 2 additional *MassPoint* instances on *vehicle* (for a total of 3).
    2. Name the first mass-point on *vehicle, "lidar-case"*
    3. Name the second mass-point on *vehicle, "sun-sensor-case"*
    4. Name the third mass-point on vehicle, *"imu-case"*
5. Compute states:
    1. Compute the state of the *vehicle2*'s reflector in the *vehicle's lidar-case* frame (hint: see question 4.2 in the first part of this exercise)
    2. Now, we could add a Sun to the sim and get the position of the sun in the *sun-sensor-case* frame, or we could use what we already have, and pretend. Let's do the latter, the construct is the same. Compute the state of the *Space.inertial* frame relative to the *"sun-sensor-case"* frame.

    Hint – the vehicles are integrated in the frame named *"Space.inertial"*, so the frame is already identified as *vehicle.dyn_body.integ_frame*. If we had a Sun in our sim, it would be the state of *sun.planet.inertial* relative to the *sun-sensor-case* frame.

    3. Compute the state of the *imu-case* frame relative to *vehicle's* body-frame
    4. Compute the inertial state of the *imu-case* frame.

Exercise 21 - A New Two-vehicle Simulation

We have been building simulations on old simulations, returning to older simulations, rebuilding, returning ... and this is all getting a little complicated. So, we are going to start over. It is legitimate to use previous simulation code in your solution to this problem, but you have to figure out which simulations to use. This simulation will be built in three stages.

Exercise 21.1 - Mass Properties

1. Build a new S_define suitable for simulating two-vehicles in (otherwise) empty space.
2. Vehicle A is represented as a uniform cube of side 2.0 m, mass 3.0 kg, and with a structural-frame origin 2.0 m above the center of one cube-face (i.e. 2.0 m along the normal to the face). Make the structural-frame and body-frame aligned, with the body-frame origin on the +x axis of the structural frame.

    The inertia tensor for a cube of side *L* about its center of mass, with axes passing through

the facet centroids is: $I = \frac{ML^2}{6}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

3. To Vehicle A, add a MassPoint at (4.0, 0.0, 0.0) m in the structural frame of vehicle A. Orient the MassPoint such that it is aligned with the structural and body frames.
4. Vehicle B is represented as a uniform cube of side 2.0 m, mass 9.0 kg, and with a structural-frame origin at the center of one cube-face. Make the structural-frame and body-frame aligned, with the body-frame origin on the +x axis of the structural frame.
5. To Vehicle B, add a MassPoint at (0.0, 0.0, 0.0) m in the structural frame of vehicle B. Orient the *MassPoint* with a 180 degree yaw relative to the body and structural frames about the structural z-axis.
6. Sketch the vehicle configurations.

Exercise 21.2 - Setting the Attach and Detach Processes

1. Schedule two events:
   (a) Attach vehicle A to vehicle B at t = 5.0 seconds using the two mass points defined in the previous exercise
   (b) Detach vehicle A from vehicle B at t = 15.0 seconds.

Exercise 21.3 - Setting the Vehicle States and Testing

1. Set the state of Vehicle A to the following:
   (a) Position of center of mass is (0.0, 0.0, 0.0) in inertial frame
   (b) Velocity of center of mass is (0.0, 0.0, 0.0) in inertial frame
   (c) Orientation of body reference frame to be a negative-90 degree rotation about the inertial z-axis relative to the inertial frame.
   (d) Angular rate of body frame is (0.0, 0.0, 0.0) relative to inertial frame
2. Set the state of Vehicle B to the following:
   (a) Position of center of mass is (-10.0, -2.0, 0.0) m in inertial frame
   (b) Velocity of center of mass is (2.0, 0.0, 0.0) m/s in inertial frame
   (c) Orientation of body reference frame to be a negative-90 degree rotation about the inertial z-axis relative to the inertial frame.
   (d) Angular rate of body frame is (0.0, 0.0, 0.0) relative to the inertial frame.
3. Plot the core and composite state of both bodies throughout the simulation.
   (a) Pay particular attention to the velocity and angular velocity states.

4. Identify what changes are being made to the state, and why they are occurring.
5. Repeat, giving VehicleB an initial angular rate of (20.0, 0.0, 0.0) degrees/s relative to the inertial frame, expressed in the body frame.

Exercise 22 - Review of Exercise 19

Return to Exercise 19 for this exercise.

1. Look at the following values for each *MassBody* in the simulation:
   (a) structure_point.T_parent_this
   (b) core_properties.T_parent_this
   and justify the differences.

Exercise 23 - Initializing with Orbital Elements

For this exercise, it will be necessary to review the documentation on the *DynBodyInitOrbit* body-action, both the model documentation (*models/dynamics/body_action/docs/body_action.pdf*) and in the API.

1. Develop a simple Earth-orbital simulation in which the vehicle is in an orbit with the following orbital elements, based on the Earth-centered inertial reference-frame:
   (a) semi-major axis = 6700 km
   (b) inclination = 52 degrees
   (c) eccentricity = 0.01
   (d) argument of periapsis = 60 degrees
   (e) longitude of ascending node = 25 degrees
   (f) time since periapsis = 0.0 seconds
2. Plot the inertial position of the vehicle over at least three orbits

Exercise 24 - Initializing from LVLH

Starting from SIM_23, this simulation adds a vehicle, the state of which is initialized relative to the LVLH states of the existing vehicle and of itself.

1. Copy the content of SIM_23 into SIM_24. Go to SIM_24.
2. Add a second vehicle to the simulation.
3. Initialize its translational state with a position that is offset from the original vehicle by (-200, 0, 0) m, and with a relative velocity of (10, 0, 0) m/s in the LVLH frame of the original vehicle..
4. Initialize its rotational state with respect to its own LVLH frame such that it is pitched up by ninety degrees (rotation = +90 degrees on y-axis).

5. Initialize the rotational state of the first vehicle to be aligned with the Earth inertial reference frame, and with zero relative angular velocity.
6. Compare and contrast the inertial positions and attitudes of the two vehicles.

Exercise 25 - Add an Atmosphere

Starting with SIM_12 (non-spherical-Earth orbiter, with RNP), this simulation adds the MET atmosphere to Earth

1. Copy the content of SIM_12 into SIM_25. Go to SIM_25. We will be using the spherical gravity model.
2. Remove the 200x200 and 4x4 run directories and the non-spherical-gravity Modified-data file.
3. Add the MET atmosphere to the Earth simulation-object
   (a) Hint – this can be done by replacing a lot of the S_define with one of the included S_modules.
4. Add an atmosphere state to the vehicle.
5. Plot the density, pressure, and temperature of the atmosphere, and the altitude and longitude of the vehicle.
6. Adjust the orbit, make a comparative prediction about the atmosphere in this new orbit, and repeat step 5. Compare your new results to your prediction.

Exercise 26 - Setting up a Surface Model

Continues from Exercise 25; while it is possible to set up a surface model on a vehicle in empty space, it is difficult to test the implementation. This Exercise will start with setting up the surface model, and continue with using that surface model to generate an aerodynamic drag profile.

1. Copy the content of SIM_25 into SIM_26. Go to SIM_26. If you have more than one RUN_* directory, keep the primary run and delete the others.
2. Set up a SurfaceModel implementation to accommodate an Aerodynamics Interaction Surface.
3. Make a data file to construct a box:
   (a) use FlatPlate facets;
   (b) use the following dimensions:
      i. Length = 2.0 m,
      ii. Width = 1.5 m,
      iii. Height = 1.0 m.
   (c) Make some out of "white" and some out of "black".

(d) Give them some sensible temperature (all the same).

4. Compile and run your simulation. Confirm (by comparison with Exercise 25) that the addition of the Surface Model has no effect on the propagation of the vehicle dynamic state.

Exercise 27 - Add Aerodynamic Drag

Continues from Exercise 26.

1. Copy the content of SIM_26 into SIM_27. Go to SIM_27.
2. Create an *interactions* object in the S_define.
3. Add an instance of *AerodynamicDrag* to the *interactions* object.
4. Create an *AeroSurface* from your *SurfaceModel* implementation.
5. Add the call to *aero_drag* as a scheduled-class job. Your vehicle has the following properties:
   (a) inertial velocity,
   (b) T_inertial_struct,
   (c) mass,
   (d) center_grav.

   Find their respective IDs for entry in the call to the *aero_drag* method.
6. Add the collect statements to collect the force and torque to your vehicle.
7. For your box, define Aero Parameters for “white” and “black”.
   (a) Black:
      i. Diffuse reflection,
      ii. calculate drag coefficient.
   (b) White:
      i. Mixed reflection,
      ii. 80% specular reflection,
      iii. calculate drag coefficient.
8. Make two new run directories - *RUN_gravonly*, and *RUN_aero.*
9. Using the input file from SIM_26 as a basis,
   (a) in the *RUN_gravonly/input.py* input file, turn aerodynamic drag off.
   (b) in the *RUN_aero/input.py* input file, turn aerodynamic drag on.
10. Remove the run directories copied in from SIM_26.
11. Run both cases – i.e. with aerodynamic drag turned on and off

12. Compare and contrast the vehicle dynamic state and applied force values between *RUN_aero* and *RUN_gravonly*.

Exercise 28 - Add Radiation Pressure

Continues from Exercise 27.

1. Copy the content of SIM_27 into SIM_28. Go to SIM_28.
2. Add the radiation model to the interactions object.
3. Add the radiation forces to the collect mechanism.
4. Add the Radiation Data Logging capability.
5. Add a Sun simulation-object. Now there are multiple planets, so be careful about the simulation configuration; this is a big step. Do not be concerned with the gravitational forces from anywhere other than Earth. Do not be concerned with shadowing effects.
6. Edit the surface model so that it comprises FlatPlateThermal facets instead of FlatPlate facets, but is otherwise identical.
7. Define Radiation Parameters for “white” and “black”.
   (a) Black:
      i. albedo = 0.2,
      ii. diffuse = 0.95,
      iii. emissivity = 0.8,
      iv. heat capacity per area = 200 J K^{-1} m^{-2}.
   (b) White:
      i. albedo = 0.7,
      ii. diffuse = 0.7,
      iii. emissivity = 0.3,
      iv. heat capacity per area = 500 J K^{-1} m^{-2}.
8. Make all facets thermally active.
9. Make sure that you have created a RadiationSurface from your new surface.
10. Edit the input files in *RUN_gravonly* and *RUN_aero* so that radiation pressure is turned off in them.
11. Run these two simulations and compare with the data from Exercise 27. Confirm that your changes have had no effect on the vehicle.
12. Create two new run directories, *RUN_rad* and *RUN_both*. In *RUN_rad,* turn aerodynamic drag off and radiation pressure on; in *RUN_both*, turn aerodynamic drag and radiation pressure on.

13. Run both new runs.
14. Compare and contrast the results of the four runs.
15. Compare the power absorbed and resulting temperature variations of your 6 facets.

Exercise 29 - Utilizing the Contact Model

This is a stand-alone simulation, built from scratch. It uses multiple components that have already been covered. Referring to any of the earlier exercise solutions, particularly Exercise 21, could be useful in getting the simulation established.

1. Set up a simulation with two vehicles unattached and approaching one another. Create 4 run directories – *RUN_no_contact*, *RUN_contact*, *RUN_stiff*, and *RUN_stiff_fast*
2. Configure the vehicle states such that they will collide in a head-on collision at a relative speed of 1.0 m/s.
3. Provide each vehicle with a surface containing a single flat-plate-facet of size 10.0m suitable for creating a contact-facet from.
   Note - with an integration step of 1.0s, and a relative speed of 1.0m/s, an interaction length of 10.0m will provide 10 integration steps, which should be sufficient to avoid "fly-through" problems.
   (a) Configure the facets such that they are facing one another.
4. Implement the Contact Model in a new *interactions* simulation-object.
   (a) Configure a contact-pair for the two facets in the simulation using the following values for the contact-pair data:
      i. spring_k = $1.0\times10^{2}\ N\ m^{-1}$
      ii. damping_b = $20.\ N\ s\ m^{-1}$
      iii. mu = 0.4
   (b) Configure a contact-pair with a stiffer response, in which the spring-constant is 3 orders of magnitude larger.
5. Be sure to collect the contact forces.
6. Create an input file in the *RUN_contact* directory to run the nominal configuration as outlined above (with contact forces enabled).
7. In RUN_no_contact, disable the contact model and run the simulation.
8. In RUN_stiff, increase the spring_k by 3 orders of magnitude, and run.
9. Compare the results from these three simulations
10. In RUN_stiff_fast, repeat the RUN_stiff simulation-run but with the simulation running 3 orders of magnitude faster in the vicinity of the collision.
11. Compare your results between RUN_contact and RUN_stiff_fast.

Exercise 30 - Articulating Surfaces

This is a stand-alone simulation, built from scratch. It comprises three sections. There may be useful components from earlier exercises.

Exercise 30.1 - Re-orienting an Array

1. Set up a basic simulation again, with a baseline time and dynamics model. For this simulation, we do not need any integration or compute_derivatives calls in the dynamics object.
2. Set up a vehicle, called *vehicle* with an attached MassBody, called *array*.
3. Attach *array* and *vehicle* so that the *vehicle* body-frame, *vehicle* structure-frame, *array* body-axes, and *array* structure-axes are all co-located and aligned.
   (a) **Hint –** use a BodyAction process to attach *array* to *vehicle*.
4. Generate a surface model, comprising two FlatPlate facets - one for *vehicle* and one for *array*.
   (a) **WARNING –** for the purposes of this exercise. we will want to monitor the normals to these FlatPlate objects. That cannot be accomplished if we instantiate the surface model elements from a Modified Data file – as we have done previously – because those data are not available for logging. Instead of creating new FlatPlate objects in the Modified data files, declare them directly in the S_define. The Modified data file will still be used to populate these facets and add them to the surface.
5. Associate one facet with *array* and one with *vehicle*. Make *vehicle* the master frame and activate articulation.
6. Initialize both facets so that they are co-located on the x-axis (say, at 1.0 m) and co-aligned with normals along the x-axis.
7. Periodically, re-orient *array* with respect to *vehicle*, such that the x-axes are no longer aligned, and reattach *array* and *vehicle*.
8. Verify that the normal vectors behave as follows:
   (a) For the *vehicle* facet, the normal remains constant.
   (b) For the *array* facet, the normal should move around in response to the reconfigured orientation of the *array* MassBody.
9. Verify that the facet position behaves as expected.

Exercise 30.2 - Verifying the Re-orientation

1. Reconfigure the structural origin of the vehicle by moving the bodies *(MassBodyInit* data) and facets the same distance (say, 2.0 m) along the y-axis.

   Note – the facets retain their positions in the body frame, but have a new position in the redefined structural frame.
2. Run the same orientation sequence that was tried in PART 1.7

3. Confirm, to your own satisfaction, that the rotations associated with redefining the orientation are interpreted as being about the origin of the structural reference frame.

Exercise 31 - Smooth Rotation, with Radiation Pressure

Continues from Exercise 30.

1. Use the configuration from either exercise 30.1 or exercise 30.2.
2. Change the simulation to single-planet mode, and add Sun. This simulation will still not need any integration or compute_derivatives calls in the dynamics object.
3. Position the vehicle so that the facet is pointing towards Sun.
4. Add a radiation pressure model to the simulation.
5. Slowly rotate the array facet about an axis perpendicular to the Sun-facet vector, so that the facet rotates into shadow and back to illumination again.
6. Plot the force generated by the radiation pressure model during this operation.

Exercise 32 - Simple Simulation of Vehicle with Thrusters

This is a stand-alone simulation, built from scratch. There may be useful components from earlier exercises.

1. Implement a simple simulation of a vehicle in empty space with a zero initial state.
2. Add a constant force to the vehicle, representing a thruster firing.
3. Plot the state of the vehicle.

Exercise 33 - Revisiting an Orbital-Elements Initialized Simulation; Output Orbital-Elements

In Exercise 23, we initialized a vehicle using Orbital Elements, and plotted the state of the vehicle in inertial coordinates. In this exercise, we will build on that, adding orbital elements as an output from the propagated state.

1. Copy SIM_23 to SIM_33. Go to SIM_33.
2. Include the GGM05C gravity model for Earth.
3. Add Orbital Elements outputs for the vehicle.
4. Make two run-directories, one with spherical gravity, and one with a 4x4 gravity model.
5. Plot the following data:
   (a) Eccentricity
   (b) Semi-major axis
   (c) inclination
   (d) true anomaly
6. Compare and contrast the evolution of the orbital elements in the two scenarios.

Exercise 34 - Derived States Exercise - Orbital Elements and Planet Fixed States

This exercise continues with the scenario developed in Exercise 33.

1. Copy SIM_33 to SIM_34. Go to SIM_34
2. Add a second vehicle, *vehicleB*, to the simulation.
   (a) Initialize *vehicleB* to be 50m below v*ehicleA*, and moving with the same velocity.
   (b) Initialize the rotational states of both vehicles to be zero in their respective LVLH frames (hint – see Exercise 24).
3. Add *OrbitalElements* and *PlanetFixed* derived states such that it is possible to output:
   (a) the state of both vehicles in terms of *OrbitalElements*,
   (b) the position of both vehicles in a spherical *PlanetFixed* representation, and
   (c) the position of both vehicles in an elliptical *PlanetFixed* representation.
4. Compile and run the simulation, comparing the new output data to that from Exercises 33 and 34.

Exercise 35 - Derived States Exercise – Relative State

In this exercise, we will look at the relative state between two vehicles. You may want to refer to Exercise 34 for some ideas on how to set this up.

1. Write a simulation to represent two independent vehicles (*vehicleA*, *vehicleB*) in similar Earth-orbits. The simulation can operate in *SinglePlanet* mode and Earth may be considered spherical.
2. Instantiate a Relative State to provide the state of the body-frame of *vehicleB* relative to the body-frame of *vehicleA*.
3. Initialize the vehicles with the following state definitions:
   (a) Initialize Vehicle A in a circular equatorial orbit with semi major axis = $6.7 \times 10^3 km$ (hint – see Exercise 23)
   (b) Initialize Vehicle B to be 50m below *VehicleA*, and moving with the same velocity.
   (c) Initialize the rotational states of both vehicles to be zero in their respective LVLH frames.
4. Compile and run the simulation to ensure satisfactory setup.
5. Watch the inertial position, velocity, and orientation of the vehicles over a period equal to a little longer than one complete orbit.
6. Plot and evaluate the relative state between the vehicles.

Exercise 36 - Derived States Exercise - Define a LVLH Reference Frame

This simulation does not do much, but it will be developed in later exercises. You may want to refer to Exercises 34 and 35 for some ideas.

1. Write a simulation to represent two independent vehicles in similar Earth-orbits. The simulation can operate in *SinglePlanet* mode and Earth may be considered spherical.
2. Instantiate a LVLH reference frame based on each of the vehicles.
3. Initialize the vehicles with the following state definitions (hint – this is identical data to Exercise 35, but may benefit from a different implementation):
   (a) Initialize Vehicle A in a circular equatorial orbit with semi major axis = $6.7 \times 10^3 km$ (hint – see Exercise 23)
   (b) Initialize Vehicle B to be 50m below *VehicleA*, and moving with the same velocity.
   (c) Initialize the rotational states of both vehicles to be zero in their respective LVLH frames.
4. Compile and run the simulation to ensure satisfactory setup.
5. Compare the results with those from Exercise 35.

Exercise 37 - Derived States Exercise - Relative LVLH State

This exercise continues with the scenario developed in Exercise 36.

1. Copy SIM_36 to SIM_37. Go to SIM_37
2. Create relative-lvlh states to provide the state of each vehicle in the rectilinear LVLH reference frame of the other.
3. Plot the two new position vectors.
4. Create a new run to provide the state of each vehicle in the curvilinear (rather than rectilinear) LVLH reference frame of the other.
5. Compare the results between the two runs.

Exercise 38 - Solar Beta

The initialization of this simulation has much in common with that from Exercise 33.

1. Create a simulation of a single vehicle in a circular, equatorial orbit about a spherical Earth. Include Sun and Earth in the simulation.
2. Add the calculations of the Solar Beta Angle Model for this vehicle.
3. Run the simulation for at least one year ($3.14 \times 10^7 s$) and watch the solar-beta-angle change as the orientation of Earth's equator varies with the seasons.

Exercise 39 - Managing Relative States with the Relkin Manager

This exercise continues with the scenario developed in earlier exercises.

1. Copy SIM_37 to SIM_39. Go to SIM_39.
2. Add a *MassPoint* to each vehicle, with a translational offset of 1.0m from the center of mass along the structural x-axis.
3. For each vehicle, orient the body frame to be a 90-degree rotation on the z-axis relative to the respective structure frame. Maintain the alignment of the body frames with their local LVLH.
4. For simplicity, return to the circular equatorial orbit of Exercise 33. This will assist in maintaining the alignment between LVLH and body-frames because LVLH also rotates at a constant rate in a circular orbit.
5. Add a *RelativeKinematics* manager to the simulation.
6. Add the following relative states to the existing relative states, and register all relative states (including those from Exercise 36) with the *RelativeKinematics* manager:
   (a) VehicleA MassPoint w.r.t. VehicleB center-of-mass in the LVLH frame associated with VehicleB's body-frame.
   (b) VehicleB MassPoint w.r.t. VehicleB center-of-mass in the LVLH frame of VehicleB.
   (c) VehicleA MassPoint w.r.t. VehicleB MassPoint in the structural reference frame of

VehicleB.

7. Generate all listed relative states.
8. Provide Vehicle B with a small rotation rate on the body y-axis (LVLH y-axis) and repeat the simulation. Watch particularly for changes in the state between the two mass-points.

Exercise 40 - Managing Relative States with the Relkin Manager, part II

This exercise continues with the previous Relkin Manager exercise.

1. Copy SIM_39 to SIM_40. Go to SIM_40. Remove the rotation run.
2. Add the following relative states to the Relkin manager:
   (a) As in SIM_39
   (b) As in SIM_39
   (c) As in SIM_39
   (d) VehicleB center-of-mass w.r.t. VehicleA center-of-mass in VehicleA's body-frame.
   (e) VehicleA center-of-mass w.r.t. VehicleB center-of-mass in the LVLH frame associated with VehicleB's body-frame.
   (f) VehicleB center-of-mass w.r.t. VehicleA center-of-mass in the LVLH frame associated with VehicleA's body-frame.
   (g) VehicleA center-of-mass w.r.t. VehicleB structure-point in the curvilinear LVLH frame associated with VehicleB's structure-frame.
   (h) VehicleB center-of-mass w.r.t. VehicleA structure-point in the curvilinear LVLH frame associated with VehicleA's structure-frame.
   (i) Vehicle B center-of-mass w.r.t. VehicleA structure-point in the rectilinear LVLH frame associated with VehicleA's structure-frame.
3. Initialize Vehicle B to be 5km below and 10km behind vehicleA, as measured in vehicleA's body-centered rectiilinear-LVLH frame.
4. Position the center of mass of both vehicles to be at [0.0, 5.0, 0.0]*m* in the respective structural frame.
5. Generate all listed relative states. Compare, in particular:
   (a) position from state (d) -vs- position from state (f)
   (b) position from state (f) -vs- position from state (i)
   (c) position from state (h) -vs- position from state (i)

Exercise 41 - Multiple Integration Groups

This exercise introduces the concept of multiple integration groups. The real strength of multiple groups comes in situations involving multiple vehicles with independent integration requirements (e.g.

A simulation including an earth-orbiter and an earth-moon transfer vehicle, or a moon-orbiter and a lunar-lander). For this exercise, however, we will keep things simple and integrate 3 identical vehicles in 3 different ways.

1. Create a simple 3-vehicle simulation, with each vehicle having the same initial state around a spherical Earth (hint – see Exercise 7) with initial state:
   a. position = [7500, 0, 0] km
   b. velocity = [0, 7, 0] km/s
   c. attitude aligned with inertial
   d. zero angular-rate
   e. body-frame and structural-frame aligned.
2. Create 3 integration groups, populate each group with 1 vehicle.
3. Use an RK4 integrator for all 3 groups. Assign integration steps of 0.005s (200Hz), 1s, and 100s to the three groups.
4. Record the position of the vehicle at 10Hz for 500s; plot the data to observe the stair-stepping seen on the low-frequency integration.

Note – because this is a particularly simple case (designed to illustrate only the usage of the integration groups), the driving force is regular and symmetric. Consequently, it will take a while before any noticeable difference appears in the data between these three vehicles (when considering only those instances at which all three states are calculated). However, if this exercise were extended to include non-spherical gravity, then differences would start to be observable earlier. Of the three integration rates, 1Hz will provide the most accurate integration for this type of scenario; differences observed with the 200Hz and 100s integration rates are a result of additional numerical errors.

5. At 400s, move *vehicle* into the medium-rate integration group. Plot the data to observe the transition.

Exercise 42 - Cis-lunar 2-vehicle Simulation and Ephemerides Rate Investigation

This simulation is a refresher of a number of pieces that have been put together over the duration of this course. The input files and Modified-data are provided; in some cases these are taken from real mission data. The exercise is to create a 2-vehicle simulation, with one vehicle (*moon_vehicle*) being integrated in Moon-inertial and another (*earth_vehicle*) in Earth-inertial.

The secondary task is to investigate the effect on vehicle states of setting the Ephemerides model to run at the derivative rate, and to have the lunar-orientation update at the derivative rate.

There are four runs provided in this exercise corresponding to the combinations of Ephemerides model update rates (dynamics or derivative) and the lunar-orientation special update (on or off).

It might be expected that updating the lunar orientation at the derivative rate would be redundant when the Ephemerides model is also updating at the derivative rate, but this is a good opportunity to verify that.

1. Build a simulation that includes:
   a) Earth, Sun, Moon
   b) GGM05C gravity model for Earth
   c) LP150Q gravity model for Moon
   d) To the moon sim-object, add the *De4xxEphemeris::propagate_lunar_rnp( )* job at the derivative rate. This can be turned off at the input-file level.
   e) Two vehicles with gravitational effects from all 3 bodies
2. Have the two vehicles be integrated in different frames but the same group (already done for you, can you see where?)
3. Log the positions and velocities of both vehicles and compare the evolution between the 4 cases.