rpOML

MATLAB Orbital Mechanics Library
[C: 11JAN22] [Readme LM: 16FEB22]

*****This library is a work in progress and will be frequently updated*****

Overview

This library of MATLAB functions and scripts is designed around CU Boulder's ASEN5050 and ASEN6008 courses. The intent is to catalog common astrodynamics functions and tools that are used in trajectory design and orbital mechanics. Optionally, the JPL CSPICE MICE Toolkit can be loaded through this library. Currently, rpOML is not dependent on CSPICE, but in future development, it is likely that astrodynamics functions from JPL's library will be utilized here. The toolkit is also useful to process JPL published constants, frames, and ephemerides. rpOML primarily consists of 2-Body orbital mechanics functions and interplanetary mission design related algorithms.

Installation

1. Clone the following repository
2. In the download there is a file called `rpOMLstart.m` . Open this file in MATLAB and follow the commented block of code at the top to set the correct path(s).
3. [Optional] The JPL CSPICE MICE Toolkit can be loaded in with this library. To do so, follow the instructions in the file from step (2).

Using the Library

1. Copy/Paste `rpOMLstart.m` to your project's working directory.
2. At the top of your MATLAB script add either of the following lines to initialize rpOML:
    a. Without CSPICE MICE: `rpOMLstart();`
    b. With CSPICE MICE: `rpOMLstart('cspice',true);`

- Note: Step (2) will clear the CSPICE kernel and reinitialize it every time it is called. Consider this when attempting to use `rpOMLstart()` in functions or subroutines.

Scripts/Functions Directory

Most functions have description block comments at the top that can be called by the MATLAB console window command:

```
help <name_of_function>
```

Assumptions, warnings, TODO, inputs, outputs, references, and other data is all noted at the top of each function in this comment block. Below is a summarized table of all the scripts and functions in this library.

Scripts and functions with an asterisk after the name* indicate that they are commented and have a help description available.

Example Scripts (./examples/)

File Name	Description
ex1_statesandconstants.m*	Introduction to 2-body orbital dynamics state creation and propagation using integration or Kepler's method
[WIP] ex2_nbp.m	N-Body Problem simulation example
ex3_lambert_transferTypes.m*	This script is intended to show how the transfer time of flight varies between two bodies as a function of the universal variable for Hyperbolic, Parabolic, and Elliptical Type I-IV trajectories
ex4_porkchopplot1.m*	Ballistic Type I and II porkchop plot between Earth and Mars (example 1)
ex5_porkchopplot2.m*	Ballistic Type I and II porkchop plot between Earth and Mars (example 2)
ex6_launchvehicleperf.m*	This script is intended to show how to use the launch vehicle performance function.

Code Verification Scripts/Data (./codeverif/)

File Name	Description
test_lambert0rev.m*	Verify 0 Revolution Lambert Algorithm
test_planetarystates.m*	Verify Meeus Algorithm to compute planetary inertial states in MJ200EC
test_planetdata()*	Test cases data to verify planetary states

Astrodynamics Functions

*(Functions in **blue** are used more often and are especially useful)*

Constants, States, Frames, and Conversions

File Path and Name	Description
./constants()	Function to load in planetary constants and conversions info including: Mu, radius, Orbit SMA, Orbit Period, Grav Constant, AUtoKM, and more.
./conv_ele2state()	Converts classical orbital elements (a, e, i , o w, ta) to a cartesian state in the inertial and perifocal frames (includes the required rotation matrix)
./conv_state2ele()	Converts inertial state vector to classical orbital elements
./create_state()*	Creates a full state structure with all cartesian coordinates and orbital elements. Full orbit properties are given in the output MATLAB structure.
./dcm_iot2rth()*	Given inc., RAAN, true anomaly, and arg. of peri. (in radians) find DCM

“Get” Functions (convenience functions and planetary states)

File Path and Name	Description
./getDatePastDate()*	Given a calendar date and number of days elapsed, find the new calendar date.
./getJulianDate()*	Wrapper to MATLAB’s Julian date conversion from calendar date.
./getMeeusData()*	Returns an array of planetary coefficients for the Meeus algorithm.
./getStateCircularPlanar()*	Returns Cartesian State w.r.t Time (CIRCULAR-PLANAR ORBITS ONLY)
./getStatePlanet()*	Returns MJ2000EC inertial cartesian state for given julian date(s) vector using the Meeus algorithm (future support w/ NAIF JPL Ephemerides DE files).

Equations of Motion (./eoms/)

File Path and Name	Description
eom2BP()*	2-Body Equations of Motion (Integrate with ODE45)
eomNBP()*	N-Body Equations of Motion (Integrate with ODE45)

Launch Vehicle Performance

File Path and Name	Description
./lvperformance()*	Finds mass delivered for various LVs given a departure C3.

Two-Body Problem (./twobp/)

File Path and Name	Description
getAnomalyandDt()	Gets Eccentric, Mean Anomaly, and Time Since Periapsis
kepElipAnomaly()*	Newton Iterative Process to find F and theta given Mh and e
kepHypAnomaly()*	Iterative Process to find F and theta given Mh and e
prop2bp()	Propagate state using 2-BP EOMs and optionally plot trajectory
propKepElip()*	Propagate using Kepler's Equation of an Ellipse
propKepHyp()*	Propagate using Kepler's Equation of a Hyperbola

Lambert's Problem (./lambert/)

File Path and Name	Description
lambert0rev()*	Universal variable Lambert algorithm for Type I and II trajectories
lambertgeo()*	Geometric Lambert's Algorithm (considerably slower but more intuitive than the Universal variable algorithm)
lambert_getc2c3()*	Gets the C2 and C3 constants for the universal variable algorithm
lambert_getPsiNRev()*	Compute Bounds on Psi given 'n>0' Number of Revolutions
lambert_getTOFfromPsi()*	Finds the associated time of flight given a value of the universal variable
plotLambertTransfer()	[GEO ALG. ONLY] Plots geometric lambert solver's answer

Porkchop Plotting (./porkchop/)

File Path and Name	Description
pkchp_porkchop()	Iteratively solves the Lambert's problem for Type I and II trajectories to create C3, arrival Vinf, and TOF contour plots
pkchp_plt()	Plots result from previous function See example: "ex4_porkchopplot1" for porkchop plot setup.