JEOD Overview

Introduction to JEOD
Standard JEOD S_modules
A Quick Simulation

What is JEOD?

- “JSC Engineering Orbital Dynamics”
- Model suite originally began as part of Trick
- Simulates orbital dynamics and on-orbit space environment
- Complete redesign when moved to C++ for version 2.x
- Many models extensible, allowing adaptability and customization
- Class C and CMMI Level 3 software

Capabilities

- High-fidelity simulation of spacecraft orbits
- Gravity:
  - Comes with gravity fields of selectable complexity for Earth, Moon, & Mars
    - Users can add others
  - Point-mass gravity provided for all other bodies
  - Example user extension: JPL polyhedral gravity model
- Earth atmosphere
- Radiation pressure
- Selectable integration methods (most common Runge-Kutta 4, but many others available. Also, users can add more.)
- Multiple bodies concurrently
- Separately orbiting, or attachable/detachable
- Can be orbiting different planets/bodies simultaneously
- Use separate integrators and/or integration step-size for each

Selected Known Use Cases

- ISS and Visiting Vehicle simulations in Low Earth Orbit
- Earth-Moon L2 point station with vehicle rendezvousing from LEO
- Lunar Orbiting Platform-Gateway simulations
- Phobos, Deimos, and asteroid proximity missions
- Ascent/descent simulations for various planets

JEOD S_modules

- A standard set of S_modules is included with JEOD
  - Location: $JEOD_HOME/lib/jeod/JEOD_S_modules
- Furnishes several frequently used combinations of models and settings, e.g.
  - Dynamics:
    - Typical use, initialization-only, support for multiple integration groups
  - Environment
    - Environment management, various planetary and time configurations
  - Vehicles
    - Basic starting point for a vehicle, and for a vehicle subject to atmospheric effects.
- Is possible to construct complete simulations using primarily, or even exclusively, the standard S_modules

Standard S_module Simulation

- Exercise: construct a complete LEO simulation using only standard JEOD S_modules
  - Sun, Earth, Moon
  - Non-spherical Earth gravity
  - Two vehicles
  - Default priority settings and full dynamics manager

Simulation Exercise: S_define

```
// Definite a reasonable job calling interval
#define DYNAMICS      1.0

// Include the default Trick and JEOD objects
#include "sim_objects/default_trick_sys.sm"
#include "jeod_sys.sm"
#include "default_priority_settings.sm"

// Include the appropriate time object:
#include "time_TAI_UTC_UT1_TT_GMST.sm"

// Include the dynamics object suitable for integrating
#include "dynamics.sm"

// Include planets and DE4xx ephemeris
#include "environment.sm"
#include “sun_basic.sm”
#include "earth_GGM02C_MET_RNP.sm"
#include “moon_basic.sm”

// Include two basic vehicle objects
#include "vehicle_basic.sm"
VehicleSimObject vehicle2 (dynamics.dyn_manager);

// Set the integration
IntegLoop sim_integ_loop (DYNAMICS) dynamics;
```

Simulation Exercise: Input Deck

```
# Use Runge-Kutta 4 integrator
rk_integrator = trick.RK4IntegratorConstructor()
dynamics.dyn_manager_init.integ_constructor = rk_integrator

# Set up vehicle 1 in Earth orbit
vehicle.dyn_body.name = "veh1"
vehicle.dyn_body.integ_frame_name = "Earth.inertial"
vehicle.dyn_body.translational_dynamics = True
vehicle.dyn_body.rotational_dynamics = True

execfile("Modified_data/time.py")
execfile("Modified_data/vehicle_mass_props.py")
execfile("Modified_data/vehicle1_state.py")
execfile("Modified_data/vehicle1_grav_controls.py")

dynamics.dyn_manager.add_body_action( vehicle.mass_init )
dynamics.dyn_manager.add_body_action( vehicle.trans_init )
dynamics.dyn_manager.add_body_action( vehicle.rot_init )

# Set up vehicle 2 in Earth orbit
vehicle2.dyn_body.name = "veh2"
vehicle2.dyn_body.integ_frame_name = "Earth.inertial"
vehicle2.dyn_body.translational_dynamics = True
vehicle2.dyn_body.rotational_dynamics = True

execfile("Modified_data/time.py")
execfile("Modified_data/vehicle_mass_props.py")
execfile("Modified_data/vehicle2_state.py")
execfile("Modified_data/vehicle2_grav_controls.py")

dynamics.dyn_manager.add_body_action( vehicle2.mass_init )
dynamics.dyn_manager.add_body_action( vehicle2.trans_init )
dynamics.dyn_manager.add_body_action( vehicle2.rot_init )

trick.sim_services.exec_set_terminate_time(6000.0);
```