NumericalPropagationWithContinuousManeuver 4.4

De Wiki
Aller à : navigation, rechercher
NumericalPropagationWithContinuousManeuver 4.4

```
public class NumericalPropagationWithContinuousManeuver {

    public static void main(String[] args) throws PatriusException,
IOException, URISyntaxException {

        // Patrius Dataset initialization (needed for example to get the UTC
time)
        PatriusDataset.addResourcesFromPatriusDataset() ;

        // Recovery of the UTC time scale using a "factory" (not to duplicate
such unique object)
        final TimeScale TUC = TimeScalesFactory.getUTC();

        // Date of the orbit given in UTC time scale)
        final AbsoluteDate date = new AbsoluteDate("2010-01-01T12:00:00.000",
TUC);

        // Getting the frame with wich will defined the orbit parameters
        // As for time scale, we will use also a "factory".
        final Frame GCRF = FramesFactory.getGCRF();

        // Initial orbit
        final double sma = 7200.e+3;
        final double exc = 0.01;
        final double per = sma*(1.-exc);
        final double apo = sma*(1.+exc);
        final double inc = FastMath.toRadians(98.);
        final double pa = FastMath.toRadians(0.);
        final double raan = FastMath.toRadians(0.);
        final double anm = FastMath.toRadians(0.);
        final double MU = Constants.WGS84_EARTH_MU;

        final ApsisRadiusParameters par = new ApsisRadiusParameters(per, apo,
inc, pa, raan, anm, PositionAngle.MEAN, MU);
        final Orbit iniOrbit = new ApsisOrbit(par, GCRF, date);

//SPECIFIC
        // Creating a mass model (see also specific example)
        final AssemblyBuilder builder = new AssemblyBuilder();

        // Main part
        final double iniMass = 900.;
```

```java
        builder.addMainPart("MAIN");
        builder.addProperty(new MassProperty(iniMass), "MAIN");

        // Tank part (ergols mass)
        final double ergolsMass = 100.;
        final TankProperty tank = new TankProperty(ergolsMass);
        builder.addPart("TANK", "MAIN", Transform.IDENTITY);
        builder.addProperty(tank, "TANK");

        // Engine part
        final double isp = 300.;
        final double thrust = 400.;
        final PropulsiveProperty prop = new PropulsiveProperty(thrust, isp);
// au lieu de new PropulsiveProperty("PROP", thrust, isp);
        builder.addPart("PROP", "MAIN", Transform.IDENTITY);
        builder.addProperty(prop, "PROP");

        final Assembly assembly = builder.returnAssembly();
        final MassProvider mm = new MassModel(assembly);

        // We create a spacecratftstate
        final SpacecraftState iniState = new SpacecraftState(iniOrbit, mm);
//SPECIFIC

        // Initialization of the Runge Kutta integrator with a 2 s step
        final double pasRk = 2.;
        final FirstOrderIntegrator integrator = new
ClassicalRungeKuttaIntegrator(pasRk);

        // Initialization of the propagator
        final NumericalPropagator propagator = new
NumericalPropagator(integrator);
        propagator.resetInitialState(iniState);

        // Forcing integration using cartesian equations
        propagator.setOrbitType(OrbitType.CARTESIAN);

//SPECIFIC
        // Duration of the maneuver to get a 20 m/S boost
        final double G0 = 9.80665;
        final double duration = G0*isp*iniMass*(1. -
FastMath.exp(-20/(G0*isp)))/thrust;
        // Start and end thrust events
        final EventDetector eventStart = new
DateDetector(date.shiftedBy(10.));
        final EventDetector eventEnd = new
DateDetector(date.shiftedBy(10.+duration));
        // Creation of the continuous thrust maneuver
        final Vector3D direction = new Vector3D(1., 0., 0.);
        final ContinuousThrustManeuver man = new
ContinuousThrustManeuver(eventStart, eventEnd, prop, direction, mm, tank);
```

```
        // Adding a continuous thrust maneuver
        propagator.addForceModel(man);
        // Adding additional state (change name add to set for V3.3)
        propagator.setMassProviderEquation(mm);

        // Adding an attitude law (or attitude sequence : mandatory)
        final AttitudeLaw attitudeLaw = new LofOffset(LOFType.TNW,
RotationOrder.ZYX, 0., 0., 0.);
        propagator.setAttitudeProvider(attitudeLaw);

//SPECIFIC

        // Propagating 100s
        final double dt = 100.;
        final AbsoluteDate finalDate = date.shiftedBy(dt);
        final SpacecraftState finalState = propagator.propagate(finalDate);
        final Orbit finalOrbit = finalState.getOrbit();

        // Printing new date and semi major axis
        System.out.println();
        System.out.println("Initial semi major axis =
"+iniOrbit.getA()/1000.+" km");
        System.out.println("New date = "+finalOrbit.getDate().toString(TUC)+"
deg");
        System.out.println("Final semi major axis =
"+finalOrbit.getA()/1000.+" km");
        // Printing mass
        System.out.println();
        System.out.println("Dry Mass = "+finalState.getMass("MAIN")+" kg");
        System.out.println("Ergols Mass = "+finalState.getMass("TANK")+"
kg");

    }

}
```

User Manual

- User Manual 4.15
- User Manual 4.14
- User Manual 4.13
- User Manual 4.12
- User Manual 4.11
- User Manual 4.10
- User Manual 4.9
- User Manual 4.8
- User Manual 4.7
- User Manual 4.6.1
- User Manual 4.5.1
- User Manual 4.4
- User Manual 4.3
- User Manual 4.2
- User Manual 4.1
- User Manual 4.0
- User Manual 3.4.1
- User Manual 3.3

Tutorials

- Tutorials 4.15
- Tutorials 4.14
- Tutorials 4.13.5
- Tutorials 4.12.1
- Tutorials 4.8.1
- Tutorials 4.5.1
- Tutorials 4.4
- Tutorials 4.1
- Tutorials 4.0

Links

- CNES freeware server

Navigation

- Accueil
- Modifications récentes
- Page au hasard
- Aide

Outils

- Pages liées