NumericalPropagationWithOrbitalIncrement Maneuvers 4.5.1

NumericalPropagationWithOrbitalIncrementManeuvers 4.5.1

```
public class NumericalPropagationWithOrbitalIncrementManeuvers {

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

        Locale.setDefault(Locale.US);

        // 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 date0 = 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 ecc = 0.;
        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 KeplerianParameters par = new KeplerianParameters(sma, ecc,
inc, pa, raan, anm, PositionAngle.MEAN, MU);
        final KeplerianOrbit iniOrbit = new KeplerianOrbit(par, GCRF, date0);
        final double period0 = iniOrbit.getKeplerianPeriod();
        final double sma0 = iniOrbit.getA();

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

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

```
        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);

        // 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);

        final ArrayList<DateDetector> listOfEvents = new
ArrayList<DateDetector>();
        final ArrayList<ImpulseManeuver> listOfMan = new
ArrayList<ImpulseManeuver>();
        final ArrayList<String> listOfManLabels = new ArrayList<String>();
        final boolean error = false;

        // Event corresponding to the criteria to trigger the Da impulsive
maneuver
        double da = 10.e3;
        final DateDetector eventDa = new DateDetector(date0);
        listOfEvents.add(eventDa);
        final ImpulseManeuver impDa = new ImpulseDaManeuver(eventDa, da,
prop, mm, tank);
```

```
        listOfMan.add(impDa);
        listOfManLabels.add("Da impulsive maneuver");
        final double newa = sma0 + da;
        final double newPeriod =
2.*FastMath.PI*FastMath.sqrt(newa*newa*newa/MU);

        // Event corresponding to the criteria to trigger the Da/De impulsive
maneuver ... but an impossible one !
        da = -10000.;
        double de = -0.001386962552;
        final DateDetector eventDaDe1 = new
DateDetector(date0.shiftedBy(0.5*newPeriod));
        listOfEvents.add(eventDaDe1);
        final ImpulseManeuver impDaDe1 = new ImpulseDeManeuver(eventDaDe1,
de, da, prop, mm, tank, error);
        listOfMan.add(impDaDe1);
        listOfManLabels.add("DaDe impossible impulsive maneuver");

        // Event corresponding to the criteria to trigger the Da/De impulsive
maneuver
        da = -10000.;
        de = -0.001386962552;
        final DateDetector eventDaDe2 = new
DateDetector(date0.shiftedBy(newPeriod));
        listOfEvents.add(eventDaDe2);
        final ImpulseManeuver impDaDe2 = new ImpulseDeManeuver(eventDaDe2,
de, da, prop, mm, tank, error);
        listOfMan.add(impDaDe2);
        listOfManLabels.add("DaDe impulsive maneuver");

        // Event corresponding to the criteria to trigger the Di impulsive
maneuver
        final DateDetector eventDi = new
DateDetector(date0.shiftedBy(newPeriod+period0));
        listOfEvents.add(eventDi);
        double di = FastMath.toRadians(0.1);
        final ImpulseManeuver impDi = new ImpulseDiManeuver(eventDi, di,
prop, mm, tank, error);
        //final ImpulseManeuver impDi = new ImpulseDiManeuver(eventDi, di,
0., prop, mm, tank, error);
        listOfMan.add(impDi);
        listOfManLabels.add("Di impulsive maneuver");

        // Event corresponding to the standard criteria
        final DateDetector eventStd = new
DateDetector(date0.shiftedBy(newPeriod+2.*period0));
        listOfEvents.add(eventStd);
        final Vector3D deltaV = new Vector3D(10., 0., 0.);
        final ImpulseManeuver impStd = new ImpulseManeuver(eventStd, deltaV,
prop, mm, tank, LOFType.TNW);
        listOfMan.add(impStd);
```

```
        listOfManLabels.add("Std impulsive maneuver");

        // Creation of the sequence of maneuver
        ManeuversSequence seq = new ManeuversSequence(0., 0.);
        for (ImpulseManeuver impulseManeuver : listOfMan) {
            seq.add(impulseManeuver);
        }

        // Adding the maneuver sequence to the propagator
        seq.applyTo(propagator);
        // Adding additional state
        propagator.setMassProviderEquation(mm);

        printResults(iniOrbit, listOfMan, "Initial conditions", false);

        // For propagating just after maneuvers
        final double dt = 1.e-6;

        for (int i = 0; i < listOfMan.size(); i++) {
            final SpacecraftState finalState =
propagator.propagate(listOfEvents.get(i).getDate().shiftedBy(dt));
            KeplerianOrbit kep = new KeplerianOrbit(finalState.getOrbit());
            printResults(kep, listOfMan, listOfManLabels.get(i), true);

        }

    }

    private static void printResults (final KeplerianOrbit kep, final
ArrayList<ImpulseManeuver> listOfMan,
            final String manLabel, final boolean isUsedDeltaV ) throws
PatriusException {

        String str = "";
        if ( !isUsedDeltaV ) {
            str = "Initial ";
        }

        System.out.println();
        System.out.println(manLabel);
        System.out.println("Initial date =
"+kep.getDate().toString(TimeScalesFactory.getUTC()));
        System.out.println(String.format("%sSemi major axis = %8.3f km", str,
kep.getA()/1000.));
        System.out.println(String.format("%seccentricity    =    %8.6f", str,
kep.getE()));
        System.out.println(String.format("%sinclination     = %8.3f deg",
str, FastMath.toDegrees(kep.getI())));
        System.out.println(String.format("%sAOL             = %8.3f deg",
str, FastMath.toDegrees(kep.getPerigeeArgument()+kep.getTrueAnomaly())));
        for (int i = 0; i < listOfMan.size(); i++) {
```

```
            if ( isUsedDeltaV ) {
                System.out.println("Maneuver #"+i+" =
"+listOfMan.get(i).getUsedDV() + " m/s");
            } else {
                System.out.println("Maneuver #"+i+" =
"+listOfMan.get(i).getDeltaVSat() + " m/s");
            }
        }

    }

}
```


Menu de navigation

PATRIUS

- Welcome

Evolutions

- Main differences between V4.15 and V4.14
- Main differences between V4.14 and V4.13
- Main differences between V4.13 and V4.12
- Main differences between V4.12 and V4.11
- Main differences between V4.11 and V4.10
- Main differences between V4.10 and V4.9
- Main differences between V4.9 and V4.8
- Main differences between V4.8 and V4.7
- Main differences between V4.7 and V4.6.1
- Main differences between V4.6.1 and V4.5.1
- Main differences between V4.5.1 and V4.4
- Main differences between V4.4 and V4.3
- Main differences between V4.3 and V4.2
- Main differences between V4.2 and V4.1.1
- Main differences between V4.1.1 and V4.1
- Main differences between V4.1 and V4.0
- Main differences between V4.0 and V3.4.1

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
- Suivi des pages liées
- Pages spéciales
- Adresse de cette version
- Information sur la page
- Citer cette page

- Dernière modification de cette page le 17 août 2020 à 09:11.

- Politique de confidentialité
- À propos de Wiki
- Avertissements

-