NumericalPropagationWithUsedDV 4.4

De Wiki
Aller à : navigation, rechercher
NumericalPropagationWithUsedDV 4.4

```
public class NumericalPropagationWithUsedDV {

    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);

        // 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 spacecraftstate
        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>();

        // Event corresponding to the criteria to trigger the impulsive
maneuver
        final DateDetector eventImp = new DateDetector(date0.shiftedBy(10.));
        listOfEvents.add(eventImp);
        // Creation of the impulsive maneuver
        final double dv = 20.;
        final Vector3D deltaV = new Vector3D(dv, 0., 0.);
        final ImpulseManeuver imp = new ImpulseManeuver(eventImp, deltaV,
prop, mm, tank, LOFType.TNW);

        // Duration of the maneuver to reach the initial semi major axis
        final double duration = 49.4933;
        System.out.println(duration);
        // Creation of the continuous thrust maneuver
        final AbsoluteDate startDate =
```

```
date0.shiftedBy(iniOrbit.getKeplerianPeriod()-0.5*duration);
        final DateDetector eventStart = new DateDetector(startDate);
        final DateDetector eventEnd = new
DateDetector(startDate.shiftedBy(duration));
        listOfEvents.add(eventStart);
        listOfEvents.add(eventEnd);
        final Vector3D direction = new Vector3D(-1., 0., 0.);
        final ContinuousThrustManeuver man = new
ContinuousThrustManeuver(eventStart, eventEnd, prop, direction, mm, tank);

        // Creation of the sequence of maneuver
        ManeuversSequence seq = new ManeuversSequence(0., 0.);
        seq.add(imp);
        seq.add(man);

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

        // Dt to get information just before/after an event
        final double dt = 1.e-6;

        for (int i = 0; i < listOfEvents.size(); i++) {

            System.out.println("\nEVENT #"+i);

            System.out.println("Before ...");
            final AbsoluteDate dateBefore =
listOfEvents.get(i).getDate().shiftedBy(-dt);
            final SpacecraftState finalStateBefore =
propagator.propagate(dateBefore);
            printResults(dateBefore.toString(TUC), finalStateBefore, imp,
man);

            System.out.println("After ...");
            final AbsoluteDate dateAfter =
listOfEvents.get(i).getDate().shiftedBy(dt);
            final SpacecraftState finalStateAfter =
propagator.propagate(dateAfter);
            printResults(dateAfter.toString(TUC), finalStateAfter, imp, man);
        }

    }

    private static void printResults ( final String sdate, final
```

```
SpacecraftState sc,
            final ImpulseManeuver imp, final ContinuousThrustManeuver man )
throws PatriusException {
        System.out.println("  Date = "+sdate);
        System.out.println("  Impulsive Maneuver = "+imp.getUsedDV()+" m/s");
        System.out.println("  Continuous Maneuver = "+man.getUsedDV()+"
m/s");
        System.out.println("  Ergols Mass = "+sc.getMass("TANK")+" kg");
        System.out.println("  Semi major axis = "+sc.getA()/1000.+" km");
        System.out.println("  Eccentricity = "+sc.getE());
    }

}
```


Menu de navigation

Outils personnels

- 18.219.207.11
- Discussion avec cette adresse IP
- Créer un compte
- Se connecter

Espaces de noms

- Page
- Discussion

Variantes

Affichages

- Lire
- Voir le texte source
- Historique
- Exporter en PDF

Plus

Rechercher

Rechercher Lire

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 4 octobre 2019 à 14:32.