NumericalPropagationWithLiftAndDragAndMSISE2000

De Wiki
Aller à : navigation, rechercher
NumericalPropagationWithLiftAndDragAndMSISE2000

```
public class NumericalPropagationWithLiftAndDragAndMSISE2000 {

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

        // 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 = 6600.e+3;
        final double exc = 0.;
        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);

        // Mass model using an Assembly

        final AssemblyBuilder builder = new AssemblyBuilder();

        // Initial mass (mandatory to take into account mass for atmospheric
force computation)
```

```
        final double dryMass = 100.;
        builder.addMainPart("MAIN");
        builder.addProperty(new MassProperty(dryMass), "MAIN");

//SPECIFIC
        // Adding the AeroGSphere property for drag only
        final double cd = 2.0;
        final double cl = 0.2;
        final double sref = 10.;
        //builder.addProperty(new AeroGlobalProperty(cd, cl, new
ConstantFunction(sref)), "MAIN");
        builder.addProperty(new AeroGlobalProperty(cd, cl, new
Sphere(Vector3D.ZERO, FastMath.sqrt(sref/FastMath.PI))), "MAIN");

        final UpdatableFrame mainFrame = new UpdatableFrame(GCRF,
Transform.IDENTITY, "mainPartFrame");
        builder.initMainPartFrame(mainFrame);
//SPECIFIC

        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);

        // Adding additional state (change name add to set for V3.3)
        propagator.setMassProviderEquation(mm);

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

//SPECIFIC
        // Adding an attitude law
        final AttitudeLaw attitudeLaw = new LofOffset(LOFType.LVLH,
RotationOrder.ZYX, 0., 0., 0.);
        propagator.setAttitudeProvider(attitudeLaw);

        // Definition of the Earth ellipsoid for later atmospheric density
computation
        final Frame ITRF = FramesFactory.getITRF();
        final double AE = Constants.WGS84_EARTH_EQUATORIAL_RADIUS;
```

```
        final BodyShape EARTH = new OneAxisEllipsoid(AE,
Constants.WGS84_EARTH_FLATTENING, ITRF, "EARTH");

        // Adding atmospheric forces using MSISE2000 model
        SolarActivityDataProvider solarProvider = new
ConstantSolarActivity(100, 15);
        final MSISE2000InputParameters data = new
ClassicalMSISE2000SolarData(solarProvider);
        CelestialBody sunBody = CelestialBodyFactory.getSun();
        final Atmosphere atmosphere = new MSISE2000(data, EARTH, sunBody);

        final DragLiftModel dragLiftModel = new DragLiftModel(assembly);
        final ForceModel atm = new DragForce(atmosphere, dragLiftModel);
        propagator.addForceModel(atm);
//SPECIFIC

        // Propagating 5 periods
        final double dt = 5.*iniOrbit.getKeplerianPeriod();
        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("Mass = "+finalState.getMass("MAIN")+" kg");

    }

}
```

- 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

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 15 novembre 2022 à 08:40.

- Politique de confidentialité
- À propos de Wiki
- Avertissements

-