NumericalPropagationWithCustomEvent 4.5.1

NumericalPropagationWithCustomEvent 4.5.1

```
public class NumericalPropagationWithCustomEvent {

    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 = 0.02;
        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(180.);
        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);

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

        // 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
        // Definition of the custom event
        EventDetector event = new EventDetector() {

            private static final long serialVersionUID = 1L;
            public double g(SpacecraftState s) throws PatriusException {
                // We want to raise the event when Lv = 45 deg
                final double delta = s.getLv() - FastMath.toRadians(45.);
                return delta;
            }

            public Action eventOccurred(SpacecraftState s, boolean
increasing,
                    boolean forward) throws PatriusException {
                System.out.println("Event occured at date :
"+s.getDate().toString(TUC)+" (LM = "+FastMath.toDegrees(s.getLv())+")");
                return Action.CONTINUE;
            }

            public boolean shouldBeRemoved() {
                return false;
            }
            public SpacecraftState resetState(SpacecraftState oldState)
                    throws PatriusException {
                return null;
            }
            public void init(SpacecraftState s0, AbsoluteDate t) {
            }
            public double getThreshold() {
                return AbstractDetector.DEFAULT_THRESHOLD;
            }
            public int getSlopeSelection() {
                return 0;
            }
            public int getMaxIterationCount() {
                return 20;
            }
            public double getMaxCheckInterval() {
                return AbstractDetector.DEFAULT_MAXCHECK;
            }
```

```
            @Override
            public EventDetector copy() {
                return null;
            }

        };

        // Adding the event to the propagator
        propagator.addEventDetector(event);
//SPECIFIC

        // Propagating on several orbits
        final double dt = 5.*iniOrbit.getKeplerianPeriod();
        final AbsoluteDate finalDate = date.shiftedBy(dt);
        propagator.propagate(finalDate);

    }

}
```


Menu de navigation

Plus

Rechercher

Rechercher Lire

PATRIUS

- Welcome

Evolutions

- 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 4.13
- 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 4.5.1
- 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 09:05.

- Politique de confidentialité
- À propos de Wiki
- Avertissements

-