

User Manual 3.3 Multi Propagation

De Wiki

Aller à : [navigation](#), [rechercher](#)

[User Manual 3.3 Multi Propagation](#)

Sommaire

- [1 Introduction](#)
 - [1.1 Scope](#)
 - [1.2 Javadoc](#)
 - [1.3 Links](#)
 - [1.4 Useful Documents](#)
 - [1.5 Package Overview](#)
 - [1.5.1 Multi Numerical Propagator](#)
 - [1.5.2 Three propagation modes](#)
- [2 Features Description](#)
 - [2.1 Multi numerical propagator](#)
- [3 Contents](#)
 - [3.1 Interfaces](#)
 - [3.2 Classes](#)
 - [3.3 Using the multi numerical propagator](#)
 - [3.3.1 Create propagator from FirstOrderIntegrator](#)
 - [3.3.2 Add initial states](#)
 - [3.3.3 Add additional equations](#)
 - [3.3.4 Propagate the attitude](#)
 - [3.3.5 Set the central attraction coefficient](#)
 - [3.3.6 Add force model](#)
 - [3.3.7 Add event detector](#)

Introduction

Scope

This section describes the multi propagator provided by the Patrius library. At this time, only numerical propagator is available for multi spacecraft propagation.

Javadoc

All the classes related to numerical propagation are in the `fr.cnes.sirius.patrius.propagation.numerical.multi` package of the Patrius library. The classes related to events detection are in the package `fr.cnes.sirius.patrius.propagation.events.multi` of the Patrius library. But the interface for multi event detectors is in `org.orekit.propagation.events.multi` package of the Orekit-addons library.

Library

Javadoc

Orekit addons	Package org.orekit.propagation.events.multi
Patrius	Package fr.cnes.sirius.patrius.events.multi
Patrius	Package fr.cnes.sirius.patrius.propagation.events.multi
Patrius	Package fr.cnes.sirius.patrius.propagation.numerical.multi
Patrius	Package fr.cnes.sirius.patrius.propagation.precomputed.multi
Patrius	Package fr.cnes.sirius.patrius.propagation.sampling.multi

Links

Other useful links can be found here :

- [Classical propagation chapter](#)
- [SpacecraftState chapter](#)

Useful Documents

None as of now.

Package Overview

Multi Numerical Propagator

The multi propagator architecture is copied from [Propagator](#) interface and [NumericalPropagator](#) class, and added to PATRIUS library.



Three propagation modes

Like single spacecraft propagation, the multi spacecraft propagation can be performed in different propagation modes : slave, master and ephemeris generation. The architecture of the classes responsible for theses propagation modes are presented below :



Features Description

Multi numerical propagator

The PATRIUS library offers a [multi numerical propagator](#). It aims at propagating several [org/orekit/propagation/SpacecraftState.html SpacecraftState](#) at the same time using numerical propagation. The N [SpacecraftState](#) are propagated during the same time interval using the same [FirstOrderIntegrator](#).

Each state is identified with an ID of type String. All states can be completely different but they should have the same internal date.

As for the mono-satellite numerical propagation, it is possible to define the following global propagation parameters:

- The type of orbital parameters and position angle

- The propagation mode

For each state, it is possible to configure the following elements:

- The central attraction coefficient
- The force models
- The tolerances used for orbital parameters and the tolerances applied on additional states.
- The additional states equations associated with the additional states
- The attitude providers

It is possible to define event detectors applied on a specific state or global event detectors applied on several states.

Contents

Interfaces

Interface	Summary	Javadoc
MultiPropagator	This interface provides a way to propagate several states at any time.	...
MultiModeHandler	Common interface for all multi propagator mode handlers initialization.	...

Classes

Class	Summary	Javadoc
SpacecraftState	This class is the representation of a complete state holding orbit, attitude for forces and for events computation and additional states at a given date.	...
MultiNumericalPropagator	This class propagates several SpacecraftState using numerical integration.	...
AttitudeEquation	This class represents attitude differential equations.	...
MultiIntegratedEphemeris	This class stores sequentially generated orbital parameters for later retrieval.	...

Using the multi numerical propagator

Here is presented a basic instantiation of the multi numerical propagator.

Create propagator from FirstOrderIntegrator

The [MultiNumericalPropagator](#) should be created by giving a [FirstOrderIntegrator](#).

```
final MultiNumericalPropagator propagator = new
MultiNumericalPropagator(integrator);
```

The integrator could be declared using absolute and relative tolerances represented by :

- a scalar value

```
final double abstolScal = 1.0e-10;
final double reltolScal = 1.0e-10;
```

```
final FirstOrderIntegrator integratorScal = new
DormandPrince853Integrator(0.001, 200, abstolScal, reltolScal);
```

- a vector

```
final double[] abstolVec = { 1e-5, 1e-5, 1e-5, 1e-8, 1e-8, 1e-8};
final double[] reltolVec = { 1e-10, 1e-10, 1e-10, 1e-10, 1e-10, 1e-10};
final FirstOrderIntegrator integratorVec = new
DormandPrince853Integrator(0.1, 60., abstolVec, reltolVec);
```

The orbit tolerances (vector or scalar tolerances) given to the integrator are used as default tolerances. The orbit tolerances could be defined in vector type using the method ***setOrbitTolerance***.

The integrator could be changed by using the method ***setIntegrator(final FirstOrderIntegrator integrator)***.

Add initial states

Initial states could be added to the propagator using the method ***addInitialState(SpacecraftState, String)***. Each added state is defined by a unique ID. An error is raised if :

- the input spacecraft ID is empty or null.
- a state with the same ID was already added to the propagator.
- the date associated with the added state is different from the date associated with the states previously added. (All states are propagated from the same date).

Note that, the initial state should be added before adding an additional equation, an attitude provider, a force model, an event ... etc ... associated with this state. Otherwise, the input spacecraft Id will not be recognized.

The map of the added states could be retrieved with ***getInitialStates()***

Add additional equations

As for the [NumericalPropagator](#), additional equations corresponding to the additional states of each states should be added to the propagator.

These additional equations are given to the multi numerical propagator for a specific state thanks to the following methods :

- ***addAdditionalEquation(AdditionalEquation, String)***
- ***addAttitudeEquation(AttitudeEquation, String)***
- ***setMassProviderEquation(MassProvider, String)*** (This method should be called only once, mass provider must be mass provider used in force models)

The method ***setAdditionalStateTolerance(String, double[], double[])*** may be used to add variation tolerance values (absolute and relative) to a specific additional state.

For each state, the additional states should correspond with the added additional equations (same name, same size).

See [ORB_PRO_UseP mono numerical propagation](#) chapter for more details.

The following example shows how to propagate two basic SpacecraftState :

```
// Initial date
final AbsoluteDate date = AbsoluteDate.J2000_EPOCH;

// Constants
final double mu = Constants.EGM96_EARTH_MU;
final Frame gcrf = FramesFactory.getGCRF();
final String STATE1 = "STATE1";
final String STATE2 = "STATE2";
final String BODY = "body";

// First initial state
final Orbit orbit1 = new KeplerianOrbit(7500000, 0.001, 0.40, 0, 0,
0,PositionAngle.MEAN, gcrf, date, mu);
final MassProvider massModel = new SimpleMassModel(1000., BODY);
final SpacecraftState state1 = new SpacecraftState(orbit1, massModel);

// Second initial state
final Vector3D position = new Vector3D(7.0e6, 1.0e6, 4.0e6);
final Vector3D velocity = new Vector3D(-500.0, 8000.0, 1000.0);
final Orbit orbit2 = new EquinoctialOrbit(new PVCoordinates(position,
velocity), gcrf, date, mu);
final AttitudeProvider law = new
ConstantAttitudeLaw(FramesFactory.getEME2000(), Rotation.IDENTITY);
final Attitude attitude = law.getAttitude(orbit2, date, gcrf);
final SpacecraftState state2 = new SpacecraftState(orbit2, attitude);

// Add initial states to the propagator
propagator.addInitialState(state1, STATE1);
propagator.addInitialState(state2, STATE2);

// Add additional equation associated with the mass model of the first state
// Mass provider must be mass provider used in force models.
propagator.setMassProviderEquation(massModel, STATE1);
propagator.setAdditionalStateTolerance("MASS_" + BODY, new double[]{1e-7},
new double[]{1e-7}, STATE1);

// Set attitude provider associated with the second state
propagator.setAttitudeProvider(law, STATE2);

// propagation
final Map<String, SpacecraftState> finalStates =
propagator.propagate(date.shiftedBy(propagationDuration));
```

Propagate the attitude

As for the [NumericalPropagator](#), to propagate the [Attitude](#) of a specific state, two treatments could

be applied :

- compute the attitude with an [AttitudeProvider](#) :***setAttitudeProvider(AttitudeProvider, String)***. It is possible to deal with :
 - a single attitude by calling ***setAttitudeProvider(AttitudeProvider, String)*****
 - two attitudes by calling ***setAttitudeProviderForces(AttitudeProvider, String)*** or ***setAttitudeProviderEvents(AttitudeProvider, String)***. It is not possible to call ***setAttitudeProvider(AttitudeProvider, String)*** and ***setAttitudeProviderForces(AttitudeProvider, String)*** (or ***setAttitudeProviderEvents(AttitudeProvider, String)***)
- propagate the attitude as a 7-dimension additional state :
addAttitudeEquation(AttitudeEquation, String).

In practical terms, the user has to build the additional equation by extending the abstract class [AttitudeEquation](#), call ***addAttitudeEquation(AttitudeEquation, String)***.

See [ORB_PRO_UseP mono numerical propagation] chapter for more details.

Set the central attraction coefficient

For each state, the central attraction coefficient used in the integration process could be defined with the method ***setMu(double, String)***.

If for a specific state, this method was not called, the central attraction coefficient of the orbit is used in the integration process.

The central attraction coefficient and the frame used in the integration process could be retrieved for a specific state using ***getMu(String)*** and ***getFrame(String)***.

```
propagator.setMu(mu, STATE1);  
final double muState1 = propagator.getMu(STATE1);  
final Frame frameState1 = propagator.getFrame(STATE1);
```

Add force model

For each state, a force model could be added to the list of forces used at each step by the propagator.

```
propagator.addForceModel(model, STATE1);
```

See [force model](#) chapter for more details.

Add event detector

The user could add :

- an [EventDetector](#) associated with a specific state using ***addEventDetector(EventDetector, String)***
- a [MultiEventDetector](#) associated with several states added to the propagator using ***addEventDetector(MultiEventDetector)***

See [events presentation](#) and [multi events presentation](#) chapters for more details.

Récupérée de

« http://patrius.cnes.fr/index.php?title=User_Manual_3.3_Multi_Propagation&oldid=1321 »

Catégorie :

- [User Manual 3.3 Orbit Propagation](#)

Menu de navigation

Outils personnels

- [3.128.226.128](#)
- [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

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 1 mars 2018 à 14:02.

- [Politique de confidentialité](#)
- [À propos de Wiki](#)
- [Avertissements](#)

- 