User Manual 3.4.1 Kinematics

De Wiki
Aller à : navigation, rechercher
User Manual 3.4.1 Kinematics

Sommaire

Introduction

Scope

The purpose is to extend the current Orekit attitude package with classes and methods to compute and process kinematics operations.

Javadoc

The kinematics objects are available in the package org.orekit.attitudes.kinematics in the Orekit library.

Library	Javadoc
Orekit	Package org.orekit.attitudes.kinematics

Links

Modèle:SpecialInclusion prefix=$theme sub section="Links"/

Useful Documents

Modèle:SpecialInclusion prefix=$theme sub section="UsefulDocs"/

Package Overview

The following diagram represents the main classes of the `org.orekit.attitudes.kinematics` package:

Features Description

Quaternions differentiation

A quaternion can be differentiated using various methods:

- The quaternion differentiation's formula [math]\dot{Q}_{S/R} = \frac{1}{2}Q_{S/R}\Omega_{S/R}^{[S]}[/math].

This formula connects time derivative of the quaternion q(t) with the vector of angular velocity W(t).

- When the quaternion is represented by a set of differentiable functions (Fourier series or polynomials), a direct analytic computation can be performed to get its derivative;
- Numerical differentiation (using finite difference, Ridders differentiation method, ...)

Quaternion integration

A rotation can be integrated, knowing its spin time function, using these methods:

- Wilcox method, with available approximations orders 1 to 4
- Edwards method, that corresponds to an order 3 approximation

Spin computation

The spin can be represented by a vector (instantaneous spin), or by a time-dependent function. The computation of the instantanous value of the spin can be made by one of the following methods:

- estimate the spin from two rotations corresponding to two different dates;
- using the kinematics equation when a quaternion and its derivative are known;
- compute the spin from the Euler or Cardan angles and their derivatives;
- using a direct analytic computation (when the function representing its variation over time is available).

Time-dependent orientation function

The interface `OrientationFunction` and the abstract class `AbstractOrientationFunction` have been added to the kinematics package in order to represent a time-dependent orientation function.

These classes provide the following functionalities:

- return the orientation at a date, by the `getOrientation(AbsoluteDate)` method; it is an abstract method because it is specific to the orientation (or attitude) law the user wants to implement;
- return the function representing the derivative of the quaternion components, by the `derivative()` method. A numerical differentiation method is used to compute the derivatives, nevertheless the classes inheriting the `AbstractOrientationFunction` can override this method and return the analytical function representing the derivatives, when possible.
- compute the function representing the spin using the [math]\Omega_{S/R}^{[S]} = 2Q_{S/R}\dot{Q}_{S/R}[/math] formula, by the `computeSpinFunction()` method. The derivatives of the quaternion are computed thanks to the `derivative()` method.
- estimate the function representing the spin using two rotations corresponding to two different dates (the velocity is supposed to be constant during the time interval between the two rotations), by the `estimateRateFunction(dt)` method.
- return the spin at a date, by the `computeSpin(AbsoluteDate)`, or `estimateRate(AbsoluteDate, dt)` methods (they call the `computeSpinFunction()` and `estimateRateFunction(dt)` methods)

Time-dependent spin function

The interface `Vector3DFunction` and the abstract class `AbstractVector3DFunction` have been added to the kinematics package in order to represent a time-dependent vector 3D function; a vector 3D function is used to represent the spin, or the n-th derivative of the spin.

These classes provide the following functionalities:

- return the vector at a date, by the `getVector3D(AbsoluteDate)` method; it is an abstract method because it is specific to the spin (or derivative of the spin) the user wants to implement;
- return the function representing the n-th derivative of the vector, by the `nthDerivative(order)` method. A numerical differentiation method is used to compute the derivatives, nevertheless the classes inheriting the `AbstractVector3DFunction` can override this method and return the analytical function representing the derivatives, when possible.

This method uses the Commons Math classes contained in the differentiation package, in particular the `DerivativeStructure` class, which allows to compute the n-th derivatives of a function at a point;

- return the integral of the function by the `integral(x0, xf)` method. A numerical integration method is used to compute the integral, nevertheless the classes inheriting the `AbstractVector3DFunction` can override this method and return the analytical function representing the integral, when possible.

The `DynamicsElements` object is created from a spin function and it gathers the n-th derivatives of spin at a given date; an attribute of this class is contained in the `Attitude` class.

Differentiate a quaternion

As a time-dependent quaternion can be represented by the `OrientationFunction` interface, the derivative of the quaternion can be computed using the `derivative()` method. As this method is implemented by the user, he can choose to use a numerical differentiation method or to compute the

exact derivative, when possible.

The derivative of a quaternion can be also computed using the kinematics equation: [math]\dot{Q}_{S/R} = \frac{1}{2}Q_{S/R}\Omega_{S/R}^{[S]}[/math]. This equation is implemented by the following method in the kinematics toolkit:

```
public static Quaternion differentiateQuaternion(final Quaternion q, final
Vector3D spin)
```

Integrate a quaternion

The rotation integration is to be made by the following methods of the `KinematicsToolkit` class :

- Wilcox method :

```
Rotation finalRot = KinematicsToolkit.integrate(IntegrationType.WILCOX_1,
initOrientation, initDate, finalDate, spin, dt);
```

The order integer is the approximation order : the ones available are 1 to 4.

- Edwards method :

```
Rotation finalRot = KinematicsToolkit.integrate(IntegrationType.EDWARDS,
initOrientation, initDate, finalDate, spin, dt);
```

Differentiate and integrate a vector

In a similar way to the orientation function, a new class has been added in order to represent a time-dependent 3-D vector function. This class implements the `Vector3DFunction` interface, and it contains the methods to compute the derivative of the vector or its integral over a time period. The computation can be done using a numerical method or it can be analytical, when possible.

Compute the spin

The Kinematics package contains some methods aiming to compute the instantaneous spin. These methods are listed hereafter:

- Estimation: two rotations corresponding to two different dates are used to compute the spin (the velocity is supposed to be constant during the time interval between the two rotations).

This function is implemented by the following method in the kinematics toolkit:

```
public static Vector3D estimateSpin(final Rotation start, final Rotation end,
final double dt)
```

.

- [math]\Omega_{S/R}^{[S]} = 2Q_{S/R}\dot{Q}_{S/R}[/math], when knowing the derivative of the quaternion.

This equation is implemented by the following kinematics toolkit method:

```
public static Vector3D computeSpin(final Quaternion q, final Quaternion qd)
```

.

- Using the derivative of the Euler or Cardan angles:
  - Euler (not to be used when the angles are small):

[math]\Omega_{S/R}^{[S]} = \left[\begin{array}{c} \dot{\psi}\sin(\theta)\sin(\phi) + \dot{\theta}\cos(\phi) \\ \dot{\psi}\sin(\theta)\cos(\phi)- \dot{\theta}\sin(\phi) \\ \dot{\psi}\cos(\theta) + \dot{\phi} \end{array} \right][/math]

  - Cardan (only to be used when the angles are small):

[math]\Omega_{S/R}^{[S]} = \left[\begin{array}{c}-\dot{\psi}\sin(\theta)+\dot{\phi} \\ \dot{\psi}\cos(\theta)\sin(\phi) + \dot{\theta}\cos(\phi) \\ \dot{\psi}\cos(\theta)\cos(\phi) - \dot{\theta}\sin(\phi) \end{array} \right][/math]

These formula are implemented by the following kinematics toolkit method:

```
public static Vector3D computeSpin(final double[] ang, final double[] angd,
final RotationOrder order)
```

.

- Analytic computation: if the spin is represented by a function, its instantaneous value must be obtained by direct analytic computation (the `IVector3DFunction` classes have been created for this purpose)

Getting Started

TBD

Contents

Interfaces

Interface	Summary	Javadoc
OrientationFunction	This interface represents a generic univariate orientation function.	...
Vector3DFunction	This interface represents a generic univariate 3-D vector function.	...

Classes

Class	Summary	Javadoc
AbstractOrientationFunction	This class represents an orientation function.	...
AbstractVector3DFunction	This class represents a spin function, or a spin n-th derivative function.	...

- User Manual 3.4.1 Attitude

Menu de navigation

Outils personnels

- 3.144.31.64
- 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 2 mars 2018 à 08:39.

- Politique de confidentialité
- À propos de Wiki
- Avertissements

-