User Manual 4.14 Frames

User Manual 4.14 Frames

Introduction

Scope

In Patrius, a frame is represented by the class Frame. The different frames are organized as a tree whose root is the ICRF. A frame is defined by a transformation with respect to its parent. A frame factory enables us to build easily some current frames such as GCRF, EME2000, ITRF... Models and data defining transformations between frame is defined in [FDY_FRCON_Home Frames configuration]

Javadoc

The frames are available in the package `fr.cnes.sirius.patrius.frames`.

Library	Javadoc
Patrius	Package fr.cnes.sirius.patrius.frames
Patrius	Package fr.cnes.sirius.patrius.frames.transformations

Links

- Frames architecture (some of this information may be obsolete)

Useful Documents

None as of now.

Package Overview

None.

Features Description

Frames

The following frames are available in PATRIUS:

- Reference frames :
  - ICRF,
  - ICRF translated on the Earth-Moon Barycenter (EMB),
  - GCRF,
  - CIRF,
  - TIRF,
  - ITRF,
  - EME 2000,

    - Ecliptic J2000,
    - MOD with and without EOP corrections,
    - TOD with and without EOP corrections,
    - GTOD with and without EOP corrections,
    - G50,
    - VEIS 1950,
    - and more ...
- Local orbital frames
- Vehicule frame (see AttitudeFrame in the [ATT_ALW_Home attitude laws] section)
- Topocentric frames
- CelestialBody frame (frame centerd on a celestial body)

To use any of the reference frames, the user must use the FramesFactory like so :

```
Frame eme2000 = FramesFactory.getEME2000();
```

PATRIUS implements a frame configuration mechanism. To use it, see the [FDY_FRAME_Home#HTheFramesConfiguration Frames Configuration] section in this page.

See FramesFactory javadoc for a complete list of public methods.

Note that, since PATRIUS 4.13, referential can be defined for any frame using the method `Frame.setReferential(Frame)`. By default, the referential is the same as the frame. Changing the referential will only impact the projection of the velocities in the destination frame.

Topocentric frame

A topocentric frame can be instantiated with the class TopocentricFrame. This class is based on the BodyShape and GeodeticPoint classes.

TopocentricFrame

This class provides several constructors. The simplest one returns an East oriented topocentric frame. The second one, that works with an angle, returns a North oriented topocentric frame if the angle is 0, or an image of this frame through the rotation around the zenith direction of the given angle (counterclockwise).

```
// Reference frame = ITRF
Frame frameITRF = FramesFactory.getITRF();

// Elliptic earth shape
OneAxisEllipsoid earthSpheric = new OneAxisEllipsoid(6378136.460, 0.,
frameITRF);

// Geodetic point at which to attach the frame
final EllipsoidPoint point = new EllipsoidPoint(earthSpheric ,
LLHCoordinatesSystem.ELLIPSODETIC, FastMath.toRadians(0.),
FastMath.toRadians(30.), 0., "point ");

// Build the east frame
TopocentricFrame topoEast = new TopocentricFrame(point, "lon 30 lat 0");
```

```
// Build a frame rotated by 30° from the North frame
TopocentricFrame topo30 = new TopocentricFrame(point,
FastMath.toRadians(30.), "lon 30 lat 0");
```

It can provide a number of directions, such as West, North or Nadir. It provides methods that compute elevation, azimuth and the range of a point located by a cartesian vector.

```
// get the zenith or the Nadir
Vector3D north = topoEast.getZenith();
Vector3D nadir = topoEast.getNadir();

// compute the azimuth of a position
Vector3D position = new Vector3D(1.0, 2.0, 3.0);
double azimuth = topo30.getAzimuth(position, earthSpheric.getBodyFrame(),
date);
```

Finally, being a Frame, it can be used to automatically transform a position/velocity to any frame (the inverse is also possible).

```
//get the transformation from the topocentric East to the ITRF
Transform eastTOitrf = topoEast.getTransformTo(frameITRF,
AbsoluteDate.FIFTIES_EPOCH_UTC);

//apply the transformation
PVCoordinates pointPVeast= new PVCoordinates(new Vector3D(0.,-1., 1),
Vector3D.ZERO);
PVCoordinates pointPVitrf= eastTOitrf.transformPVCoordinates(pointPVeast);
```

We can instantiate a TopocentricFrame without GeodeticCoordinates using either cartesian coordinates and a zenith direction or cartesian coordinates associated with a Body shape. This allows to define accurate Topocentric Frame on bodies which are not ellipsoids (e.g. Phobos).

TopocentricCoordinates

The topocentric coordinates are defined by the elevation, the azimuth and the distance from the center of the local topocentric frame [R2].

The azimuth is the angle between the local North and the projection of the line which joins the center of the topocentric frame and the satellite in the horizontal plane. This angle ranges from 0 to 2[math]\pi[/math]. This angle is oriented by the axis opposite to the Zenith. On the figure, the angle g is called the bearing, it is linked to the azimuth by : g = 2[math]\pi[/math]- azimuth.

The elevation is the angle between the line which joins the center of the topocentric frame and the satellite and the projection of this line in the horizontal plane. This angle ranges from-[math]\pi[/math]/2 to [math]\pi[/math]/2. This angle is oriented by the image of the West axis by the rotation of angle azimuth and around Zenith axis. On the figure, s is the elevation angle.

If the elevation equals [math]{\pi\over2}[/math] or [math]-{\pi\over2}[/math], the azimuth is undefined.

The object TopocentricCoordinates contains also the first derivatives of these elements (elevation rate, azimuth rate and range rate).

We can transform the position of the satellite expressed in Cartesian coordinates into TopocentriCoordinates and vice versa.

We can also transform PVCoordinates into TopocentricCoordinates and vice versa.

```
// North topocentric frame
final GeodeticPoint point = new GeodeticPoint(FastMath.toRadians(43.604482),
FastMath.toRadians(1.443962), 0.);
final TopocentricFrame topoNorth = new TopocentricFrame(earthSpheric, point,
0., "north topocentric frame");
final AbsoluteDate date = new AbsoluteDate(new DateComponents(2008, 04, 07),
TimeComponents.H00, TimeScalesFactory.getUTC());
// Topocentric coordinates
TopocentricPosition topoCoord = new
TopocentricPosition(FastMath.acos(FastMath.sqrt(2. / 3.)),
FastMath.toRadians(315), FastMath.sqrt(3));
// Cartesian coordinates (position only)
Vector3D position = topoNorth.transformFromTopocentricToPosition(topoCoord);
```

```
// North topocentric frame
final EllipsoidPoint point = new EllipsoidPoint(ellipsoid,
LLHCoordinatesSystem.ELLIPSODETIC, FastMath.toRadians(43.604482),
FastMath.toRadians(1.443962), 0., "point ");
final TopocentricFrame topoNorth = new TopocentricFrame(point, 0., "north
topocentric frame");
final AbsoluteDate date = new AbsoluteDate(new DateComponents(2008, 04, 07),
TimeComponents.H00, TimeScalesFactory.getUTC());
// Cartesian coordinates (position only)
Vector3D position = new Vector3D(1,1,1);
// Topocentric Coordinates
TopocentricPosition topoCoord =
topoNorth.transformFromPositionToTopocentric(position, topoNorth, date);
```

CardanMounting

The Cardan mounting is defined by two angles (X and Y) and the distance from the center of the local topocentric frame [R2].

The X angle is the angle between the Zenith and the projection of the line which joins the center of the topocentric frame and the satellite in the vertical plane (plane defined by the Zenith and the West axis). This angle ranges from-[math]\pi[/math] to [math]\pi[/math]. This angle is oriented by the South axis. On the figure, x is the X angle.

The Y angle is the angle between the line which joins the center of the topocentric frame and the satellite and the projection of this line in the vertical plane (plane defined by the Zenith and the West axis). This angle ranges from-[math]\pi[/math]/2 and [math]\pi[/math]/2. It is oriented by the image of the West axis by the rotation of angle X and around North axis. On the figure, y is the Y angle.

If the Y angle equals [math]\pi[/math]/2 or-[math]\pi[/math]/2, the X angle is undefined.

The object CardanMounting contains also the first time derivatives of these elements (X angle rate, Y angle rate and range rate).

We can transform the position of the satellite expressed in Cartesian coordinates into CardanMounting and vice versa.

We can also transform PVCoordinates into cardanMounting and vice versa.

```
// North topocentric frame
final EllipsoidPoint point = new EllipsoidPoint(ellipsoid,
LLHCoordinatesSystem.ELLIPSODETIC, FastMath.toRadians(43.604482),
FastMath.toRadians(1.443962), 0., "point ");
final TopocentricFrame topoNorth = new TopocentricFrame(point, 0., "north
topocentric frame");
final AbsoluteDate date = new AbsoluteDate(new DateComponents(2008, 04, 07),
TimeComponents.H00,
TimeScalesFactory.getUTC());
// Cardan mounting
CardanMountPosition cardanCoord = new
CardanMountPosition(FastMath.toRadians(45), FastMath.acos(FastMath.sqrt(2. /
3.)), FastMath.sqrt(3));
// Cartesian coordinates (position only)
Vector3D position = topoNorth.transformFromCardanToPosition(cardanCoord);
```

```
// North topocentric frame
final EllipsoidPoint point = new EllipsoidPoint(ellipsoid,
LLHCoordinatesSystem.ELLIPSODETIC, FastMath.toRadians(43.604482),
FastMath.toRadians(1.443962), 0., "point ");
final TopocentricFrame topoNorth = new TopocentricFrame(point, 0., "north
topocentric frame");
final AbsoluteDate date = new AbsoluteDate(new DateComponents(2008, 04, 07),
TimeComponents.H00, TimeScalesFactory.getUTC());
// Cartesian coordinates (position only)
Vector3D position = new Vector3D(1,1,1);
// Cardan mounting
cardanCoord = topoNorth.transformFromPositionToCardan(position, topoNorth,
date);
```

CelestialBodyFrame

This frame is a wrapper for frames centered on celestial bodies. All celestial bodies-centered frames

are instances of CelestialBodyFrame (including geocentric frames).

"H0- n" frame

The "H0- n" frame is a pseudo-inertial frame; its parent frame is the GCRF. It uses the frozen transformation of the ITRF with respect to the GCRF frame at the date "H0- n", combined with a rotation by a fixed angle around the Z axis of the ITRF frame.

TwoDirectionFrame

The `TwoDirectionFrame` is a generic reference frame that can be build starting from two directions and two axis given as input. The specific reference frame inherits from the generic `Frame` and uses an internal implementation of a `TransformProvider` in order to define a specific `Transform` starting from a specific date.

The `Transform` is obtained as follows :

- a translation obtained starting from the `PVCoordinatesProvider` and the `Frame` given as input to the constructor ;
- a rotation obtained starting from the directions defined towards the axis of the reference frame ;
- a rotation rate which is built a finite difference method with the method `AngularCoordinates.estimateRate(…)` : the finite difference step for the computation can be configurable by the used when creating the reference frame.

Getting Started

TBD

Contents

Interfaces

Classes

Class	Summary	Javadoc
PredefinedFrame	Base class for the predefined frames that are managed by FramesFactory.	…
CelestialBodyFrame	Class for frames centered on celestial bodies.	…
Frame	Tridimensional references frames class.	…
FramesFactory	Factory for predefined reference frames.	…
HelmertTransformation	Transformation class for geodetic systems.	…
LocalOrbitalFrame	Class for frames moving with an orbiting satellite.	…
TopocentricFrame	The topocentric frame.	…
Transform	Transformation class in three dimensional space.	…
H0MinusNProvider	Provider for the "H0-n" frame.	…
H0MinusNFrame	"H0-n" frame.	…
TwoDirectionFrame	A generic frame the can be built starting from two directions and two axes.	…

FrameConvention	Enumeration of all frame conventions used in Patrius.	...
SynodicFrame	Synodic frame: frame linked to the 3-body problem. In practise this frame generalizes the concept of synodic frame (not necessarily linked to the 3-body problem).	...
TranslatedFrame	Frame realizing a translation from a parent frame (axis direction remains unchanged).	...
GCRFProvider	Provider for the "GCRF" frame (from its parent frame, the EMB frame).	...
EMBProvider	Provider for the "EMB" frame (from its parent frame, the ICRF frame).	...

Tips & Tricks

IERS frames transformations

This chapter sums up the up and downsides of the use of the IERS frames. Its purpose is to help the user avoid some traps and know what to expect as to the numerical precision of the computations. Further explanations about the IERS frames can be found on the IERS website**[R1]**.

Tips

The ICRF frame is the frame colinear to GCRF frame and centered on solar system barycenter.

One can compute the transformations between two frames with a specific frames configuration passed as a parameter as opposed to the current frame configuration (that is obtained with the `FramesFactory.getConfiguration()` method). Given a date and a user specific config, the method to call is :

```
frame1.getTransformTo(frame2, date, config);
```

One can also compute the jacobian matrix of the conversion between two frames. Given a date epoch, the jacobian matrix of the conversion from frame1 to frame2 is computed by the method `getTransformJacobian` called by :

```
frame1.getTransformJacobian(frame2, epoch);
```

Traps

The first thing to check when working with IERS frames is the presence of the Earth Orientation Parameters bulletins it requires. Indeed, if the c04 bulletins of the correct year cannot be found in the data, Patrius will use the ones stored internally to compute the transformations. To be sure that Patrius will use the good files, the user has first to point out the directory which contains these files. The fact that it doesn't warn the user that it does not have the parameters for the time of the simulation may lead to error, and thus the deviation compared to a reference could be enormous without the user being aware. One more thing to know about these bulletins is that they have to be continuous, meaning that the transformation won't work (and will throw an exception) if there are missing years or month in the files.

Make sure to use IAU 2000 EOP data, otherwise nutation corrections are null.

Usage of Frames

The methods to be used in order to instanciate the IERS Frames are :

- FramesFactory.getGCRF()
- FramesFactory.getCIRF()
- FramesFactory.getTIRF()
- FramesFactory.getITRF()

PATRIUS

- Welcome

Evolutions

- 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 4.14
- 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 4.14
- 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 5 septembre 2024 à 13:21.

- Politique de confidentialité
- À propos de Wiki
- Avertissements

-