Propagation

De Wiki
Aller à : navigation, rechercher
Propagation

Sommaire

Slave mode

The simplest way to propagate is to use the "*slave mode*". The final result (at the date of the end of the propagation will be stored in a [PATRIUS] SpacecraftState object.

```
System.out.println("Initial orbit:");
System.out.println("Date:"+iniOrbit.getDate());
System.out.println("SMA:"+iniOrbit.getA());

final SpacecraftState sc = test.propagateInSlaveMode();

System.out.println("Final orbit:");
System.out.println("Date:"+sc.getDate());
System.out.println("SMA:"+sc.getA());
```

Master mode

On the other side, if the user wants to store intermediate data all along the propagation, it is possible to use the "*master mode*". To do it, it will have to create an OutputConfig object as below, giving the output frames (inertial and local ones) as well as the output step (here 60s):

```
final OutputConfig output = new OutputConfig(FramesFactory.getEME2000(),
LOFType.LVLH, 60.);
```

Then, the propagation will be done by calling the propagateInMasterMode method.

Old fashion (before V11.4 but still available)

The list of SpacecraftState (see [PATRIUS] definition) could be recover using the getSpacecraftStateList() method. In the following example, we have no additional events (reason why of the null second argument).

The last argument allows to specify how results will be stored. We may choose between the following options:

- StorageType.**MEMORY** => only the spacecraft states in memory
- StorageType.**FILE_SC_BINARY** => only the spacecraft states but directly written in a SQLite file (to avoid to use too much RAM)
- StorageType.**FILE_COLUMNS** => all output variables (see here) are computed and stored in a SQLite file (since V11.1).

```
test.propagateInMasterMode(output, null, StorageType.MEMORY);
List<SpacecraftState> listOfSc = test.getSpacecraftStateList();
```

The user may now use this list ...

```
final int nb = listOfSc.size();
System.out.println("Amount of S/C states = "+nb);
System.out.println("Final orbit:");
System.out.println("Date:"+listOfSc.get(nb-1).getDate());
System.out.println("SMA:"+listOfSc.get(nb-1).getA());
```

New fashion (since V11.4)

- StorageType.**MEMORY_SC** (ex StorageType.**MEMORY**) ⇒ spacecraft states memory stored
- StorageType.**MEMORY_COL** ⇒ output variables memory stored; possibility to get them using getVarsList(), getVars(index), getVar(index,key) methods
- StorageType.**MEMORY_ALL** ⇒ **MEMORY_SC** et **MEMORY_COL**
- StorageType.**NONE** ⇒ no memory storage

- StorageType.**FILE_SC** (ex StorageType.**FILE_SC_BINARY**) ⇒ spacecraft states stored in a file
- StorageType.**FILE_COL** (ex StorageType.**FILE_COLUMNS**) ⇒ output variables stored in a file
- StorageType.**FILE_ALL** ⇒ StorageType.**FILE_SC** et StorageType.**FILE_COL**
- StorageType.**NONE** ⇒ no files

In order to differentiate memory vs file choice, the propagateInMasterMode() method can accept both states as in the examples below:

```
test.propagateInMasterMode(output, events, StorageType.MEMORY_SC,
StorageType.NONE);
```

⇒ only spacecraft states memory stored (equivalent to the old **MEMORY** state)

```
test.propagateInMasterMode(output, events, StorageType.NONE,
StorageType.File_COL);
```

⇒ no memory storage and output variables stored in a file (as the old **FILE_COLUMNS** state)

```
test.propagateInMasterMode(output, events, StorageType.MEMORY_SC,
StorageType.File_ALL);
```

⇒ spacecraft states stored both in memory and file as output variables will be stored in a file.

At last, to manage output variables as spacecraft states, these methods are available:

- getVariablesList(), getVariablesWithoutEventsList(), getVariablesOnlyEventsList()
- getVariables(index), getVariablesWithoutEvents(index), getVariablesOnlyEvents(index)
- getVariable (index, key), getVariableWithoutEvents(index, key), getVariableOnlyEvents(index, key)

Note that to get those lists of spacecraft states and/or variables, it is not necessary to have stored them in memory. Indeed, **PSIMU** is able to know if data has been directly stored in a file and, if it is the case, it will read in this file rather than in memory (using the methods described below).

Important remark : data stored in files are using units as described in the list here. On the contrary, when we get data directly from Java code, these data will be always in **SI** units ! .

Some new util methods

The following methods have been declared as "public":

- to read a SQLite format file and extract spacecraft states:

```
public static List<SpacecraftState> readSpacecraftStatesFromFile ( final
boolean isEphem, final boolean isEvent, final String outputDirectory, final
String nameFile, final String varName )
```

- to convert a spacecraft state in String and vice versa:

```
public static SpacecraftState fromBase64(final String spacecraftStateString)
public static String toBase64(final SpacecraftState currentState)
```

- to read a SQLite format file and extract a list of variables:

```
public static List<HashMap<String, Object>> readVariablesFromFile( final
boolean isEphem, final boolean isEvent, final String outputDirectory, final
String nameFile, final ArrayList<String> listOfVarnames )
```


Menu de navigation

Outils personnels

Espaces de noms

- Page
- Discussion

Variantes

Affichages

- Lire
- Voir le texte source
- Historique
- Exporter en PDF

Plus

Rechercher

Rechercher Lire

PSIMU

- Welcome
- Quick start
- News

GUI Mode

- Overall presentation
- Initial Orbit
- Earth features
- Vehicle
- Forces
- Maneuvers
- Attitude
- Integrator
- Events
- Output
- Console

Batch mode

- How to call it

Java interface

- Basic principle
- Data initialization
- Propagation
- Printing results
- Customize output variables

Evolutions

- Main differences between V11.7.3 and V11.7.4
- Main differences between V11.7.2 and V11.7.3
- Main differences between V11.7.1 and V11.7.2
- Main differences between V11.6.2 and V11.7.1
- Main differences between V11.5 and V11.6.2
- Main differences between V11.4.1 and V11.5
- Main differences between V11.4 and V11.4.1
- Main differences between V11.3 and V11.4
- Main differences between V11.2 and V11.3
- Main differences between V11.1 and V11.2
- Main differences between V11.0 and V11.1

Training

- Tutorials package for V11.7.x
- Tutorials package for V11.6
- Tutorials package for V11.5
- Tutorials package for V11.4
- Tutorials package for V11.3
- Tutorials package for V11.2
- Tutorials package for V11.0

Links

- CNES freeware server

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 16 décembre 2020 à 11:42.

- Politique de confidentialité
- À propos de Wiki
- Avertissements
- Powered By MediaWiki