Siconos Project

Software User Manual

Version : 1.3
Status : validated
Date : September 29, 2004
Document Code : um!

INRIA

Identification

Document Title :	Software User Manual
Document Code :	**um!**

About the document and its author(s)

Nature :	The purpose of this document is to describe how to use the **siconos!** platform. It combines tutorials and reference information to meet the needs of both end-users and platform developers.
Language :	English
Author(s) :	Jean-Michel Barbier, Jean-Baptiste Charlety, Jérémie Blanc-Tranchant, Alexandre Ravoux
Possible remarks :	
References :	**srd!**, **ddd!**, **esd!**

Current Version

Date :	September 29, 2004
Current version number :	1.3
Status :	◯ in progress ⊗ validated Approved by : Vincent Acary

About the document developing process

Date :	May 25, 2004
Current version number :	1.0
Validated by :	Jérémie Blanc-Tranchant
Document change record since last version :	First issue

Date :	June 20, 2004
Current version number :	1.1
Validated by :	Jérémie Blanc-Tranchant
Document change record since last version :	Second issue

Date :	September 3, 2004
Current version number :	1.2
Validated by :	Vincent Acary
Document change record since last version :	new chapter "How to give data to the platform" correction of the XML tags

Date :	September 29, 2004
Current version number :	1.3
Validated by :	Jean-Michel Barbier
Document change record since last version :	New information stored in the XML to describe the SiconosModel

Contents

Chapter 1

Introduction

1.1 Intended Readership

Two main types of users have been identified during the software requirements analysis phase : end users and developers (see **SRD!**).

Typically, end-users are scientists who use the platform to make simulations of **nsds!**. They have knowledge about mechanics or electronics, but don't need to be expert in computer sciences. On the contrary, platform and plugins developers have to be skilled in object oriented languages and **XML!** files format.

1.2 Applicability Statements

This document apply to the first version of **SICONOS!**.

1.3 Purpose

This project is a work-package of European project **SICONOS!**.
Besides the standard features which are required for a software of scientific computing, the objectives of this project are of the following :

- To provide a common framework (formalisms and solver tools) for non smooth problems present in various scientific fields : applied Mathematics, Mechanics, Robotics, Electrical networks, etc.
- To be able to rely on existing developments. The platform will not re-implement the dedicated tools, which are already used for the modelling of specific systems in various fields,but will provide a framework to their integration.
- To support exchanges and comparisons of methods between researchers.
- To disseminate the know-how to other fields of research and industry.
- To take into account the diversity of users (end users, algorithms developers, framework builders, industrial).
- To set up standards in terms of modelling of such systems.

- To ensure software quality by the use of modern software design methods.

This manual has to be a reference document concerning the using of the **SICONOS!** platform. For the end-users, it provides tutorials and basic examples about all the functionalities offered by the platform to realize a simulation of **NSDS!**. For developers, the manual details the commands of the platform, how to create, add and test a plugin. It references sections of other documents of **SICONOS!** which concern data structures, implementation choices, etc.

1.4 How to use this document

The document explain first the platform architecture. This section describe all the directory and their contents provide with the platform. After, this document explain how to install and execute the platform. In the next section, an explication of the **SICONOS! XML!** file format and how to write it is given. Finally, a description of the possible errors is given

1.5 Conventions

Convention to show a display on the screen :

```
% make build
build successful
```

Convention to show a user command :

```
make install
```

Convention to show a file or a directory :
lib/
README.txt

Chapter 2

Overview

The SICONOS! platform is dedicated to the modelling and simulation of NSDS!. For modelling part, several canonical model for NSDS!, relations and non-smooth laws can be used. For the simulation part, robust time integration method (time stepping, event driven) of the dynamics and numerical solvers for non smooth laws can be used.

The SICONOS! platform can be used as a library.

The numerical routine for computation is provide by plugin. Each user can develop his own plugin and use it

The SICONOS! platform can take as input file an XML! file which describe the system. The platform can formalise the input file into canonical model, and analyse the coherence of the data. It can make an XML! file in order to save the state of the system during and / or after the simulation.

Expert user can modify the platform and add some functionality, because this platform is open source. Some documents will help developers (DDD!, ESD!, anp!, ...)

In this version of SICONOS!, for driving a simulation, the user have to write a command file (in C++). This file creates dynamical systems, interactions, ... run the simulation and collect informations.

Chapter 3

Platform Architecture

Two deliverables can be distributed, corresponding to the two main uses of the platform : a basic one for end-users, with only binaries, and a complete one with source files, to allow users to develop news functionalities for the software. Here we describe the architecture of the distribution for end users. Developers can refer to the **DDD!** for complete architecture.

3.1 Files and directories

The software *siconos* will be a set of C++ libraries driven by a main program. This kind of libraries are named in the following as "internal libraries" (i.e. libraries developed in **SICONOS!**, in opposition to the "external libraries" which are for example libXML2 or Lapack++).

In this distribution, the root directory is named *SICONOS/*. It contains the following sub-directories and files :

- *bin/* : contains binaries of the platform.
- *include/* : contains header files of internal libraries.
- *lib/* : contains the libraries of the platform.
- *share/* : contains some configuration files, necessary ton run a simulation with the platform : **XML!** schema, ...

Chapter 4

Instructions

4.1 Installation of the platform

4.1.1 get sources of the platform

You have to download the sources from our SCN server (see http://gforge.inria.fr/projects/siconos for instructions).

4.1.2 External libraries

The platform needs some external libraries to compile:

- Siconos Numerics
- libxml2
- lapack++
- lapack and blas
- nana

To know the required versions and where these libraries are available on the web, please look at the siconos gforge site (http://gforge.inria.fr/projects/siconos).

4.1.3 Generation of configure script

The platform version of SVN is a developer version and is not ready to be installed. You have to run a script to generate configuration script. This script is situated in Kernel directory.

```
% ./reconf.sh
```

4.1.4 Run configure script

By default, the platform is installed in /usr/local (you must be logged as root to be allowed to write some files in this directory).

```
% ./configure
```

To set another installation directory, use –prefix=path flag. for example :

```
% ./configure --prefix=/home/User/Siconos
```

There may be some parameters to set; to know them, run

```
% configure --help
```

If an external library is not installed in /usr or /usr/local, you have to give its path to configure script by the flag : –with-local[*library name*]=[*path of this library*].
For example :

```
% ./configure --with-localnana=/home/User/nana-2.5
```

Nota: *you have only to run configure script once, you can specify several flags together.*

4.1.5 compiling the platform

```
% make
```

This command compiles the platform and creates libraries.

4.1.6 Installing the platform

```
% make install
```

this command installs the platform in /usr/local by default, or in the directory specified during configure running.

4.2 Launch the platform

First of all, you have to set a `SICONOSPATH` environment variable to the complete path of the siconos install directory[1]. The input files are :

- a command file in C++(like *command.cpp*) which lead the execution[2] ;
- an **XML!** file which describes your system[3].
- all the plugin you want to use in a C file (like *myPlugin.c*). Its name must finish by ÍPlugin.cÍ

Run the command (we assume you are in the directory containing your command file):

```
%  siconos yourCommandFile.cpp
```

This will compile your plugins (if there are), your command file and link it to the platform. In your command file, you can for example create a model using an **XML!** file place in the *sample* directory.

Remark :

[1] for example "$HOME/SICONOS"
[2] too see an example of command file report to Annexe A
[3] it is optional. Anyway, you can put it in another directory

1. The siconos executable are placed in the same directory of your command file. If you want to run this file without the siconos script, you have to specify in environment variable `LD_LIBRARY_PATH` the paths of the shared libraries needed by the platform : plugins, lib directory of your siconos installation.

2. If you want to use a plugin which is not in the directory *sample* or in the directory *src/plugin*, you must modify the `LD_LIBRARY_PATH` environment variable. It tell where to find libraries and plugins, so don't forget to add the path to this plugin in this environment variable (see 4.5) Note that this plugin will NOT be compiled automatically by the siconos script.

4.3 Modify the platform

If you have the source files of the platform, you can modify them. After that, you must re-compile the platform with :

```
% make
```

4.4 Test the platform

If you modify the platform, you should run test to validate your modification. Do (cppUnit must be installed):

```
% make check
```

This will launch a test suite with CppUnit. If you add a class or a function, don't forget to make a test class.

All commands can be launch on the root directory (all, clean, check) or in a specified directory.

4.5 Make a plugin

To make a plugin, create a file *MyPlugin.c* and declare your functions with `extern "C"`. The name of the plugin file must end by *Plugin.c* : myPlugin.c, electronical_Plugin.c are correct, error_plugin.c or badName.c are not. If the name of your plugin does not respect this rule, the siconos script will not automatically compile and add it to `LD_LIBRARY_PATH` before running. Example for a vector field function :

```
extern "C" void vectorField(int sizeOfX, double time, double& xPtr, double& xdotPtr)
{
    /* input parameter : sizeOfX (size of the vector X);
     *                   time ; xPtr (pointer to X vector);
     * output parameter : xdotPtr (pointer to Xdot vector)
     */

/*.....*/
}
```

You must respect the parameter order for each function you declare. You can look at the *BasicPlugin.c* file in *src/plugin* directory to see examples. If you do not use the siconos script to compile the plugin : To compile the plugin, do (we assume tou have g++ 3.*.* and your OS is Linux):

```
%  g++ -c myPlugin.c
%  g++ -shared  myPlugin.o -o myPlugin.so
```

The plugin *myPlugin.so* is now created. To use it, you have to precise its name and the function name in the **XML!** file (see xml part). You can add the complete plugin path to the `LD_LIBRARY_PATH` environment variable. If you don't do this, you have to precise it in the **SICONOS! XML!** file.

Chapter 5

Input/Output Siconos XML files

This chapter allows to know how to understand and create Input/Output Siconos **XML!** files.

5.1 General type syntax

Certain Siconos **XML!** tags have complex types, we are going to see in this section.

5.1.1 Plugin type

Some tags are defined with a plugin type : it allows you to indicate functions you want to use in order to compute values.

The syntax to indicate a plugin is always the same : first the name of the library (without it extension), then the function name ; separated with a double point.
For example : 'libraryNameYouWantToUse**:**theLibraryFunctionYouWantToUse'

The plugin tags have no content ; only an attribute to indicate the plugin to use, and which the name is *plugin*.

For example : <TagPlugin **plugin**='myLib:myFunction'/>

5.1.2 Matrix and Vector syntax

Several Siconos tags are vector double or matrix double type. In a Siconos **XML!** file, it exists several ways to represent a vector or a matrix.

A computed representation is even possible (using plugin).

5.1.2.a Vector representation

In this section we consider that *myVector* is a **XML!** tag with vector type.

- **Give directly a vector double values**

If you want to give directly the values of a vector in a Siconos XML! file, you have just to precise double values in the content of the concerned XML! tag, and indicate the vector size (positive integer value) in an attribute whose name is *vectorSize*.

For example :

<myVector **vectorSize**='4'>
45.554 5465.5 99
54
</myVector>

Note that vector values must be separated with spaces, tabs, or broken lines.

- **Precise a file who contents vector values**

You can also set a vector with a vector represented in a ASCII file.

To precise a simple vector in a file, just write the 1 number (this first number represents the number of sub vector ; here one), then the size of your vector, and at last it double values (different values can/must be separated with spaces, tabs or broken lines).

For example :
myVectorFile.dat `1 5`
`1.2 21. 5 2e5 4.5454`

Therefore, you can use your file and precise it in a XML! tag ; just indicate in a *vectorFile* attribute of the vector tag the name of the concerned file :

<myVector **vectorFile**='myVectorFile.dat'/>

- **Precise a plugin who computes the vector value**

A last solution consists to give a plugin which will compute the value of the vector. Just indicate the plugin in a *vectorPlugin* attribute :

<myVector **vectorPlugin**='myLib:myFunction'/>

5.1.2.b Matrix representation

In this section we consider that *myMatrix* is a XML! tag with matrix type.

- **Give directly a matrix double values**

If you want to give directly the values of a matrix in a Siconos **XML!** file, first you have to indicate the matrix size (positive integer values) in the following attributes : *matrixRowSize* (the row size of the matrix) and *matrixColSize* (the column size of the matrix). Then, each row double values of the matrix have to be contained in a *row* tag.

For example :

<myMatrix **matrixRowSize**='3' **matrixColSize**='2'>
```
 <row>45.554 5.08 </row>
 <row>95.4 -54 </row>
 <row>6e-4 4e-5 </row>
```
</myMatrix>

Note that, like vector type, double values in row matrix content must be separated with spaces, tabs, or break lines.

- **Precise a file who contents matrix values**

You can also set a matrix with a matrix represented in a ASCII file. For it, you have just to indicate in a *matrixFile* attribute of the matrix tag, the name of the concerned file :

<myMatrix **matrixFile**='myMatrixFile.dat'/>

The syntax of matrix file is the following : first indicate the row size, then the column size, and, at last, the matrix double values.

For example :

```
2 3
5   12.2 -1
.6 .7 6e-7
```

- **Precise a plugin who computes the matrix value**

A last solution consists to give a plugin who will compute the value of the matrix. Just indicate the plugin in a *matrixPlugin* attribute :

<myMatrix **matrixPlugin**='myLib:myFunction'/>

5.1.3 Memory syntax

Some tags contain *Memory* tags which are themselves vector type (several representations are so allow). We say this kind of tag are memory type. This type serves to represent

old values of an element. The first *Memory* tag indicated is the oldest value, the last the youngest value.

For example :

```
<myVectorMemory>
   <Memory vectorFile='myVectorMemory1.dat'/>
   <Memory vectorSize="2">
        54.54 5454212.5445456456
   </Memory>
   <Memory vectorFile='myVectorMemory3.dat'/>
</myVectorMemory>
```

5.1.4 Siconos boolean type

Some attributes in a tag are boolean type and their values are 'true' or 'false', nothing else.

5.2 Siconos XML files syntax

5.2.1 Important remarks

5.2.1.a Order of the tags

You must respect the order of the tags in the XMl file like it is described further. However, there no order to respect for the tag of the dynamical systems, interactions, relations, non smooth laws, one-step integrators and one-step problem.

5.2.1.b Factorization of the tags

Because of there's no order in most of the tags, XML schema is not able to allow us to make factorization (for example for the dynamical systems, relations and non smooth laws). Moreover, when factorization could have been done, it was impossible because some tags were common but have different cardinalities.

5.2.2 General structure

The general structure of a Siconos **XML!** file (see A for an sample file):

```
SiconosModel
 |
 +-- Time
 |
 +-- NSDS
 |    |
 |    +-- DS_Definition
 |    |     |
 |    |     +-- LinearSystemDS
```

```
|     |     |     |
|     |     |     +-- BoundaryCondition
|     |     |
|     |     +-- LagrangianNLDS
|     |     |     |
|     |     |     +-- BoundaryCondition
|     |     |
|     |     +-- ...
|     |
|     +--Interaction_Definition
|           |
|           +-- Interaction
|           |     |
|           |     +-- DS_Concerned
|           |     |
|           |     +-- Interaction_Content
|           |           |
|           |           +-- LinearTIR
|           |           |
|           |           +-- NewtonImpactLaw
|           |
|           +-- ...
|
+-- Strategy
      |
      +-- TimeDiscretisation
      |
      +-- OneStepIntegrator_Definition
      |     |
      |     +-- Moreau
      |     |     |
      |     |     +-- DS_Concerned
      |     |
      |     +-- Adams
      |     |     |
      |     |     +-- DS_Concerned
      |     |
      |     +-- ...
      |
      +-- OneStepNSProblem
            |
            +-- LCP (or QP, Relay, ...)
            |     |
            |     +-- Interaction_Concerned
            |
            +-- Solver
                  |
                  +-- LcpSolving (or ContactFrictionPrimalSolving, ...)
```

5.2.3 Definition of each tag

5.2.3.a SiconosModel tag

This tag is the main tag of a Siconos XML! file. It has no attribute. Look 5.1 tab to see it content. The information about the model (Title, Author, Description, Date, SchemaXML) are required but can be left empty.

Tag name	Definition	Content Type	Attributes	Use
Title	Title of the model	String	None	Required
Author	Author of the model	String	None	Required
Description	Description of the model	String	None	Required
Date	Date of creation	String	None	Required
SchemaXML	Version of the XML schema used	String	None	Required
Time	Time data of the simulation	Abstract - see 5.2.3.b section	None	Required
NSDS	Non Smooth Dynamical System of the simulation	Abstract - see 5.2.3.c section	*bvp* : boolean type. This attribute must have 'true' value if the Non Smooth Dynamical System has boundaries conditions	Required
Strategy	Strategy used in the simulation	Abstract - see 5.2.3.l section	*type* : can take 2 values ; 'TimeStepping' or 'EventDriven' according to the strategy you want to use	Optional

Table 5.1: Content of the main tag : *SiconosModel*

5.2.3.b Time tag

This tag contains information about time of the simulation. Look 5.2 tab to see it content.

5.2.3.c NSDS tag

This tag contains the problem definition to simulate. Look 5.3 tab to see it content.

Tag name	Definition	Content Type	Attributes	Use
t0	Initial time of the simulation	Double	None	Required
T	Final time of the simulation	Double	None	Optional
t	Current time of the simulation	Double	None	Optional

Table 5.2: Content of tag : *Time*

Tag name	Definition	Content Type	Attributes	Use
DS_Definition	Contains the Dynamical Systems definition of the simulation	Abstract - see 5.2.3.d section	None	Required
Interaction_Definition	Contains the Interactions definition of the simulation	Abstract - see 5.2.3.f section	None	Optional

Table 5.3: Content of tag : *NSDS*

5.2.3.d DS_Definition tag

This tag content a list of tag defining Dynamical Systems. At least one Dynamical System (one *DS* tag) must be declared inside it - see 5.4 tab.

Tag name	Definition	Content Type	Attributes	Use
LagrangianNLDS	Content informations of a Dynamical System	Abstract - see 5.2.3.e section	*number* : an unique positive integer number / key of a Dynamical System	List - at least one *DS* tag
LagrangianTIDS	Content informations of a Dynamical System	Abstract - see 5.2.3.e section	*number* : an unique positive integer number / key of a Dynamical System	List - at least one *DS* tag
NonLinearSystemDS	Content informations of a Dynamical System	Abstract - see 5.2.3.e section	*number* : an unique positive integer number / key of a Dynamical System	List - at least one *DS* tag
LinearSystemDS	Content informations of a Dynamical System	Abstract - see 5.2.3.e section	*number* : an unique positive integer number / key of a Dynamical System	List - at least one *DS* tag

Table 5.4: Content of tag : *DS_Definition*

5.2.3.e DS tag contained in DS_Definition tag

The content of this tag is different according to their *type* attribute value : 'LagrangianNLDS', 'LagrangianTIDS', 'NonLinearSystemDS' or 'LinearSystemDS'. The DS_Definition regroups all the declarations of the following dynamical systems :*NonLinearSystemDS* (see 5.5), *LinearSystemDS* (see 5.6), *LagrangianTIDS* (see 5.2.3.e) and *LagrangianNLDS* (see 5.2.3.e).

Tag name	Definition	Content Type	Attributes	Use
Id	Content an identifier of the Dynamical System	String	None	Optional
n	The dimension of the system	Integer	None	Required
x0	x at the initial state	Vector	See Vector type	Required
x	The state of the Dynamical System	Vector	See Vector type	Optional
xDot	The x derivative of the Dynamical System	Vector	See Vector type	Optional
xMemory	The old x values	Memory	None	Optional
xDotMemory	The old xDot values	Memory	None	Optional
StepsInMemory	The number of values in memory	Positive Integer	None	Optional
vectorField	The plugin used to compute the vectorField	String	None	Required
computeJacobianX	The plugin used to compute the jacobianX	String	None	Optional
R	The input vector due to the non-smooth law	See Vector type	None	Optional
RMemory	The old r values	Memory	None	Optional
BoundaryCondition	The number of values in memory	Boundary Condition	see Boundary Condition type	Optional

Table 5.5: Content of tag : *DS* for all Dynamical System type

Tag name	Definition	Content Type	Attributes	Use
Id	Content an identifier of the Dynamical System	String	None	Optional
n	The dimension of the system	Integer	None	Required
x0	x at the initial state	Vector	See Vector type	Required
x	The state of the Dynamical System	Vector	See Vector type	Optional
xDot	The x derivative of the Dynamical System	Vector	See Vector type	Optional
xMemory	The old x values	Memory	None	Optional
xDotMemory	The old xDot values	Memory	None	Optional
StepsInMemory	The number of values in memory	Positive Integer	None	Optional
R	The input vector due to the non-smooth law	See Vector type	None	Optional
RMemory	The old r values	Memory	None	Optional
A	Contains the A matrix	Matrix	See Matrix type	Required
B	Contains B matrix	Matrix	See Matrix type	Required
BoundaryCondition	The number of values in memory	Boundary Condition	see BoundaryCondition type	Optional

Table 5.6: Content of tag : *DS* if it *type* attribute is 'LinearSystemDS'

Tag name	Definition	Content Type	Attributes	Use
Id	Content an identifier of the Dynamical System	String	None	Optional
n	The dimension of the system	Integer	None	Optional
x0	x at the initial state	Vector	See Vector type	Optional
x	The state of the Dynamical System	Vector	See Vector type	Optional
xDot	The x derivative of the Dynamical System	Vector	See Vector type	Optional
xMemory	The old x values	Memory	None	Optional
xDotMemory	The old xDot values	Memory	None	Optional
StepsInMemory	The number of values in memory	Positive Integer	None	Optional
R	The input vector due to the non-smooth law	See Vector type	None	Optional
RMemory	The old r values	Memory	None	Optional
M	The mass of the Dynamical System	Matrix (plugin)	See Matrix type	Required
ndof	The number of freedom degree	Integer	None	Required
q	The actual Dynamical System coordinates	Vector	See Vector type	Optional
q0	The initial Dynamical System coordinates	Vector	See Vector type	Required
qMemory	The old Dynamical System coordinates values	Memory	None	Optional
Velocity	The actual Dynamical System velocity	Vector	See Vector type	Optional
Velocity0	The initial Dynamical System velocity	Vector	See Vector type	Required

Tag name	Definition	Content Type	Attributes	Use
...	Continuation of the tags of a 'LagrangianNLDS'			
VelocityMemory	The old Dynamical System velocity values	Vector	See Vector type	Optional
Fint	The internal force	Vector	See Vector type	Required
Fext	The external force	Vector	See Vector type	Required
JacobianQFint	The Jacobian internal force	Matrix	See Matrix type	Required
JacobianVelocityFint	The Jacobian internal force derivative	Matrix	See Matrix type	Required
JacobianQQNLInertia	Contains the JacobianQQ matrix	Matrix	See Matrix type	Required
JacobianVelocityQNLInertia	Contains the JacobianQQ derivative matrix	Matrix	See Matrix type	Required
QNLInertia	The inertia vector	Vector	See Vector type	Optional
BoundaryCondition	The number of values in memory	Boundary Condition	see BoundaryCondition type	Optional

Table 5.7: Content of tag : *DS* if it *type* attribute is 'LagrangianNLDS'

Tag name	Definition	Content Type	Attributes	Use
Id	Content an identifier of the Dynamical System	String	None	Optional
n	The dimension of the system	Integer	None	Optional
x0	x at the initial state	Vector	See Vector type	Optional
x	The state of the Dynamical System	Vector	See Vector type	Optional
xDot	The x derivative of the Dynamical System	Vector	See Vector type	Optional
xMemory	The old x values	Memory	None	Optional
xDotMemory	The old xDot values	Memory	None	Optional
StepsInMemory	The number of values in memory	Positive Integer	None	Optional
R	The input vector due to the non-smooth law	See Vector type	None	Optional
RMemory	The old r values	Memory	None	Optional
M	The mass of the Dynamical System	Matrix(not a plugin)	See Matrix type	Required
ndof	The number of freedom degree	Integer	None	Required
q	The actual Dynamical System coordinates	Vector	See Vector type	Optional
q0	The initial Dynamical System coordinates	Vector	See Vector type	Required
qMemory	The old Dynamical System coordinates values	Memory	None	Optional
Velocity	The actual Dynamical System velocity	Vector	See Vector type	Optional
Velocity0	The initial Dynamical System velocity	Vector	See Vector type	Required

Tag name	Definition	Content Type	Attributes	Use
...	Continuation of the tags of a 'LagrangianNLDS'			
VelocityMemory	The old Dynamical System velocity values	Vector	See Vector type	Optional
Fext	The external force	Vector	See Vector type	Required
K	Contains the K matrix	Matrix	See Matrix type	Required
C	Contains the C matrix	Matrix	See Matrix type	Required
BoundaryCondition	The number of values in memory	BoundaryCondition	see BoundaryCondition type	Optional

Table 5.8: Content of tag : *DS* if it *type* attribute is 'LagrangianTIDS'

If the Non Smooth Dynamical System is a boundary value problem (see the *bvp* attribute of the *NSDS* tag), a *BoundaryCondition* tag must be added at the end of each *DS* tag content. *BoundaryCondition* tag has attribute *type* which can take the following values according to it nature : 'Linear', 'NLinear' (Not Linear), and 'Periodic'. Only one boundary condition type has content : 'Linear' ; see 5.9 tab.

Tag name	**Definition**	**Content Type**	**Attributes**	**Use**
Omega	Contains the Omega vector	Vector	See Vector type	Required
Omega0	Contains the Omega0 vector	Matrix	See Matrix type	Required
OmegaT	Contains the OmegaT vector	Matrix	See Matrix type	Required

Table 5.9: Content of tag : *BoundaryCondition* if it *type* attribute is 'Linear'

5.2.3.f Interaction_Definition tag

This abstract tag contains a list of tag describing interaction between Dynamical Systems ; a list (maybe empty) of *Interaction* tags - see 5.10 tab.

Tag name	**Definition**	**Content Type**	**Attributes**	**Use**
Interaction	Contains information about an interaction	Abstract - see 5.2.3.g section	*number* : an single integer number / key of an Interaction- NONE	Required

Table 5.10: Content of tag : *Interaction_Definition*

5.2.3.g Interaction tag

This tag contains the properties of an interaction - see 5.11 tab.

5.2.3.h Interaction_Content tag

This tag contains the properties of an interaction - see 5.12 tab.

5.2.3.i DS_Concerned tag contained in Interaction tag

This tag allows to define the Dynamical Systems couples concerned by an Interaction. A couple is defined with 2 Dynamical Systems numbers inside a *DS* tag (*number* and *interactsWithDS_Number* attributes) - See 5.13 tab.

Be careful : the number of the Dynamical System given in the *number* attribute must be lower than the number of the Dynamical System given in the *interactsWithDS_Number* attribute.

5.2.3.j Relation tag

Switch the type of the relation (precised in the *type* attribute), *Relation* tag content is different. If you want a to define a 'LinearTIR' relation, see 5.14 tab ; if you want define a 'LagrangianLinearR' relation see 5.15 tab; if you want define a 'LagrangianNonLinearR' relation, the content of the *Relation* tag is empty.

5.2.3.k NS_Law tag

Switch the type of the Law (precised in the *type* attribute), *NS_Law* tag content is different. If you want a 'RelayNSL' Law, the content of the *NS_Law* tag is empty ; if you want a 'ComplementarityConditionNSL' relation see 5.16 tab; if you want a 'NewtonImpactLawNSL' relation see 5.17 tab.

Tag name	Definition	Content Type	Attributes	Use
Status	Contains the status of the Interaction	Positive Integer	None	Required
Id	Contains an identifier of the Interaction	String	None	Required
DS_Concerned	Contains a list of *DS* tags which precise the couple of Dynamical Systems numbers concerned by the interaction	List none empty of *DS* tag - see 5.2.3.i section	none	Required
Interaction_Content	Contains the Relation and the NonSmoothLaw of the Interaction	Abstract - see 5.2.3.j section and see 5.2.3.k section		Required

Table 5.11: Content of tag : *Interaction*

Tag name	Definition	Content Type	Attributes	Use
LinearTIR	Contains information about the Relation Interaction	Abstract - see 5.2.3.j section		Required
LagranNonLinearR	Contains information about the Relation Interaction	Abstract - see 5.2.3.j section		Required
LagrangianLinearR	Contains information about the Relation Interaction	Abstract - see 5.2.3.j section		Required
ComplementraityConditionNSL	Contains information about the Law Interaction	Abstract - see 5.2.3.k section		Required
NewtonImpactLawNSL	Contains information about the Law Interaction	Abstract - see 5.2.3.k section		Required
RelayNSL	Contains information about the Law Interaction	Abstract - see 5.2.3.k section		Required

Table 5.12: Content of tag : *Interaction_ Content*

Tag name	Definition	Content Type	Attributes	Use
DS	Gives a couple of Dynamical Systems concerned by an Interaction	Empty	*number* : a number of a Dynamical defined in *DS_ Definition* tag - *interactsWithDS_ Number* : a number of a Dynamical System defined in *DS_ Definition* tag	List - at least one

Table 5.13: Content of tag : *DS_ Concerned tag contained in Interaction tag*

Tag name	Definition	Content Type	Attributes	Use
C	Contains the C matrix	Matrix	See Matrix type	Required
D	Contains the D matrix	Matrix	See Matrix type	Required
E	Contains the E matrix	Matrix	See Matrix type	Required

Table 5.14: Content of tag : *Relation* when the relation type is 'LinearTIR'

Tag name	Definition	Content Type	Attributes	Use
H	Contains the H matrix	Matrix	See Matrix type	Required
b	Contains the b vector	Vector	See Vector type	Required

Table 5.15: Content of tag : *Relation* when the relation type is 'LagrangianLinearR'

Tag name	Definition	Content Type	Attributes	Use
c	Contains the value after the non smooth event	Double	None	Required
d	Contains the value before the non smooth event	Double	None	Required

Table 5.16: Content of tag : *NS_Law* when the relation type is 'ComplementarityConditionNSL'

Tag name	Definition	Content Type	Attributes	Use
e	Contains the Newton coefficient of restitution	Double	None	Required

Table 5.17: Content of tag : *NS_Law* when the relation type is 'NewtonImpactLawNSL'

5.2.3.l Strategy tag

This tag contains the strategy information to use in order to simulate the problem described in *NSDS* tag. Look 5.18 tab to see it content.
Remark : if the *OneStepNSProblem* tag is not defined, *Interaction_Definition* tag content should be empty.

Tag name	Definition	Content Type	Attributes	Use
TimeDiscretisation	Properties of the time discretisation	Abstract - see 5.2.3.m section	None	Required
OneStepIntegrator_Definition	Contains the One Step Integrators definition of the strategy	Abstract - see 5.2.3.n section	None	Required
OneStepNSProblem	Contains the One Step Non Smooth Problem definition of the strategy	Abstract - see 5.2.3.q section	none	Optional

Table 5.18: Content of the tag : *Strategy*

5.2.3.m TimeDiscretisation

In this tag you can precise the Time Discretisation you want to use. Look 5.14 tab to see it content.

5.2.3.n OneStepIntegrator_Definition tag

This tag content a list of tag defining One Step Integrators. At least One Step Integrator (one *OneStepIntegrator* tag) must be declared inside it - see 5.20.

5.2.3.o OneStepIntegrator tag contained in OneStepIntegrator_Definition tag

The content of this tag is the same for all the OneStepIntegrator types (precise in *type* attribute) : 'Moreau', 'Adams' or 'LSODAR'. Look 5.21 tab in order to see it content.

5.2.3.p DS_Concerned tag contained in OneStepIntegrator tag

This tag ables to indicate the Dynamical Systems concerned by a One Step Integrator - See 5.22 tab.

Tag name	Definition	Content Type	Attributes	Use
h	Contains the h value	Positive Double	*isConstant* : is boolean type to precise if h is constant or not	Optional
N	Contains the N value	Positive Integer	None	Optional
tk	Contains the tk value	Vector	None	Optional
hMin	Contains the hMin value	Positive Double	None	Optional
hMax	Contains the hMax	Positive Double	None	Optional

Table 5.19: Content of tag : *TimeDiscretisation*

Tag name	Definition	Content Type	Attributes	Use
Adams	Contains informations about a One Step Integrator	Abstract - 5.2.3.o section		List - at least one
Moreau	Contains informations about a One Step Integrator	Abstract - 5.2.3.o section		List - at least one
LSODAR	Contains informations about a One Step Integrator	Abstract - 5.2.3.o section		List - at least one

Table 5.20: Content of tag : *OneStepIntegrator_Definition*

5.2.3.q OneStepNSProblem tag

This tag content information allowing to give the One Step Non Smooth Problem. According to the type of One Step Non Smooth Problem, the content of this tag is different, but some tags are even though common. You can see these common tags in 5.23 tab. The OneStepNSProblem tag is composed by 2 main tags. The one is a choice between LCP, QP and Relay, and the second is Solver, as we can see 5.23.

5.2.3.r Interaction_Concerned tag contained in LCP / QP / Relay tag

This tag ables to indicate the Interactions concerned by a One Step Non Smooth Problem - See 5.27 tab.

5.2.3.s Solver tag contained in OneStepNSProblem tag

Only one of the tags (see 5.28) must compose the Solver tag.

Tag name	Definition	Content Type	Attributes	Use
r	Contains the r value	Integer	None	Optional
DS_Concerned	Contains a list of *DS* tags which precise the couple of Dynamical Systems numbers concerned by the One Step Integrator	List none empty of *DS* tag - see 5.2.3.p section	none	Required

Table 5.21: Content of tag : *OneStepIntegrator*

Tag name	Definition	Content Type	Attributes	Use
DS	Ables to indicate a Dynamical System is concerned by a One Step Integrator	Empty	*number* : a number of a Dynamical defined in *DS_ Definition* tag	List - at least one

Table 5.22: Content of tag : *DS_ Concerned tag contained in OneStepIntegrator tag*

Tag name	Definition	Content Type	Attributes	Use
LCP	Modeling tools used for the data of the OneStepNSProblem	Abstract - 5.24 section	None	Choice (LCP, QP, Relay)
QP	Modeling tools used for the data of the OneStepNSProblem	Abstract - 5.25 section	None	Choice (LCP, QP, Relay)
Relay	Modeling tools used for the data of the OneStepNSProblem	Abstract - 5.26 section	None	Choice (LCP, QP, Relay)
Solver	Solving strategy used for computations	Abstract - 5.28 section	*lib* : the external library to use for computations (*optional*)	Required

Table 5.23: Content of tag : *OneStepNSProblem*

Tag name	Definition	Content Type	Attributes	Use
n	Contains the n value	Positive Integer	None	Optional
M	Contains the M matrix	Matrix	See Matrix type	Optional
q	Contains the q vector	Vector	See Vector type	Optional
Interaction_Concerned	Contains a list of *Interaction* tags which precise the Interactions numbers concerned by the One Step Non Smooth Problem	List none empty of *Interaction* tag - see 5.2.3.r section	none	Required

Table 5.24: Content of tag : *LCP*

Tag name	Definition	Content Type	Attributes	Use
n	Contains the n value	Positive Integer	None	Optional
Q	Contains the Q matrix	Matrix	See Matrix type	Optional
p	Contains the p vector	Vector	See Vector type	Optional
Interaction_Concerned	Contains a list of *Interaction* tags which precise the Interactions numbers concerned by the One Step Non Smooth Problem	List none empty of *Interaction* tag - see 5.2.3.r section	none	Required

Table 5.25: Content of tag : *QP*

Tag name	Definition	Content Type	Attributes	Use
Interaction_Concerned	Contains a list of *Interaction* tags which precise the Interactions numbers concerned by the One Step Non Smooth Problem	List none empty of *Interaction* tag - see 5.2.3.r section	none	Required

Table 5.26: Content of tag : *Relay*

Tag name	Definition	Content Type	Attributes	Use
Interaction	Ables to indicate an Interaction is concerned by one of the OneStepNSProblem	Empty	*number* : a number of an Interactiondefined in *Interaction_ Definition tag*	List - at least one

Table 5.27: Content of tag : *Interaction_ Concerned tag contained in LCP / QP / Relay tag*

Tag name	Definition	Content Type	Attributes	Use
LcpSolving	Contains the data to solve the problem with LCP methods	Abstract - 5.29 section	None	Choice
RelayPrimalSolving	Contains the data to solve the problem with Relay Primal methods	Abstract - 5.30 section	none	Choice
RelayDualSolving	Contains the data to solve the problem with Relay Dual methods	Abstract - 5.31 section	none	Choice
ContactFrictionPrimalSolving	Contains the data to solve the problem with Contact Friction Primal methods	Abstract - 5.32 section	none	Choice
ContactFritcionDualSolving	Contains the data to solve the problem with Contact Friction Dual methods	Abstract - 5.33 section	none	Choice

Table 5.28: Content of tag : *Solver*

Tag name	Definition	Content Type	Attributes	Use
...				

Table 5.29: Content of tag : *LcpSolving*

Tag name	Definition	Content Type	Attributes	Use
...				

Table 5.30: Content of tag : *RelayPrimalSolving*

Tag name	Definition	Content Type	Attributes	Use
...				

Table 5.31: Content of tag : *RelayDualSolving*

Tag name	Definition	Content Type	Attributes	Use
...				

Table 5.32: Content of tag : *ContactFrictionPrimalSolving*

Tag name	Definition	Content Type	Attributes	Use
...				

Table 5.33: Content of tag : *ContactFrictionDualSolving*

Chapter 6

How to give data to the platform

This chapter allows to know how to launch the platform with the needed data. It is important to understand that all the users can do correspond to the API's methods.

6.1 Loading from an XML file

It is the simpliest way to use the platform. You only need to write a correct XML file (see **??**XML)) and to call the XML file loading function. To do that, follow the next steps (the example given correspond to C++ language):

- Declaration of a Model : **Model sample_model;**
- Loading an XML file : **sample_model.createModel("xml_input_file");**

6.2 Loading in a C++ program

We understand here that no file exists to load input data. The user must do these operations manually. The following example (in C++) show the way to create the platform :

- Declaration of a Model : **Model sample_model;**
- Creation of the Model : **sample_model.createModel(NULL**/*the XML file*/**, 0**/*current time t*/**, 0**/*initial time t0*/**, 10**/*final time T*/**);**
- Creation of the NSDS : **NSDS* nsds = m.createNSDS(false /*determines if the NSDS is BVP or not*/);**
- Add of a new dynamical system to the NSDS : **nsds->addNonLinearSystemDS(1**/*the number of the dynamical system*/**, 2**/*the number of dimension of the system*/**, x0**/*the x0 vector*/**, "BasicPlugin:vectorField"**/*the vector field plugin*/ **);**
- ...

6.3 Loading using the two previous methods

It consists in using an XML file to load some of the data and to give more data in a program. For example, tha XML file can bring the data of the Model and all the model

formalization informations (the data of the NSDS, the different dynamical systems, the interactions, ...), whereas strategy informations are given at the continuation of the program (the TimeDiscretisation, the integrators, the OneStepNSProblem).

Chapter 7

How to save data of the platform

This chapter will explain the various way to save the data contained in the platform. Two main storage methods are possible.

7.1 Complete save with XML! files

In this case, the users use the **api!**'s methods.
The user has a dedicated method of the **API!** to manage the **XML!** output file writing. The method "saveToXMLFile" saves the data of the platform in the **dom!** tree, checks if the data are coherent and then write the **XML!** file.

7.1.1 Redaction note

`The method "saveToXMLFile" is not yet compatible with the save of the data at each time step, we must give it a name for the file to save...`

The user has only to call this method when he wants, for example, in the simulation loop, he can save the data each n steps.

7.2 Partial results

In this case, the users manage the data save as he wants. He selects some information, can make computation with them, and then, he can save them in an ascii file. So the users have to use basic C++ functions.
The user has to work to create these output files. But he can do what he wants, he can use all the data of the platform, make computations with them and save his calculation in a file.

7.2.1 Redaction note

`To Be Continued...`

Chapter 8

Execution exception

This part will explain error messages which may appears during the execution of the platform. There are 4 differents kind of exception : XML Exception, Siconos Matrix / Vector Exception, Shared Library Exception and Runtime Exception. Each exception is accompanied by a message which explain the cause of the error and where it appears.
Remark : It is possible that an error occurred just after the launch of the platform. Its appears when you try to run the platform without the script file "siconos", and your SICONOSPATH environment variable is not set. Just set this variable to fix this problem using "source siconos.csh"

8.1 XML Exception

This exception is thrown when an error occurred in the **XML!** part of the platform. The error message is :

XML Exception : ...

followed by a description of this error. In general, the cause is a malformed **XML!** file or **XML!** schema. Verify if these files exist and if they are well formed.

8.2 Siconos Matrix / Vector Exception

This exception is thrown when an error occurred in a Siconos Vector or a Siconos Matrix. The error message is :

Siconos Vector Exception : ...

or :

Siconos Matrix Exception : ...

followed by a description of this error. The probable source of exception is a mathematic error (division by 0, ...), an index out of range or a Matrix / Vector file not found.

8.3 Siconos Memory Exception

This exception appears when an error occured in a SiconosMemory of the platform. The error message is :

SiconosMemory Exception : ...

followed by a description of this error. This exception is throw when you try to add a memory vector whereas the memory is already full, when you try to add a vector which size is greater than the defined size for the memories, ...

8.4 Shared Library Exception

This exception is thrown when an error occurred when trying to load a plugin. The error message is :

Shared Library Exception : ...

followed by a description of this error. An error like this may occurred when a plugin is not found (pay attention to the `LD_LIBRARY_PATH` variable), or a function doesn't exist in a plugin.

8.5 Runtime Exception

This exception appears when an error occured during execution of the platform. The error message is :

Runtime Exception : ...

followed by a description of this error. This exception is throw when you try to access to a no allocated pointer, to construct an object with bad parameters, ...

Chapter 9

Platform benchmarks

9.1 BouncingBall

9.1.1 Purpose

This sample was the first non smooth dynamical system simulated with SICONOS platform software. There's a ball (assimilated to a point) moving on 1 axis. The ball is falling because of gravity force, and rebounds on a solid floor.

9.1.2 results

We can see on the figure (9.1) the position of the moving ball and the position of the floor. Moreover we can see the speed of the ball and the strength of the reaction force when the ball touchs the floor.
The bottom axis corresponds to the time, whereas the left axis corresponds to the heigth of the ball.

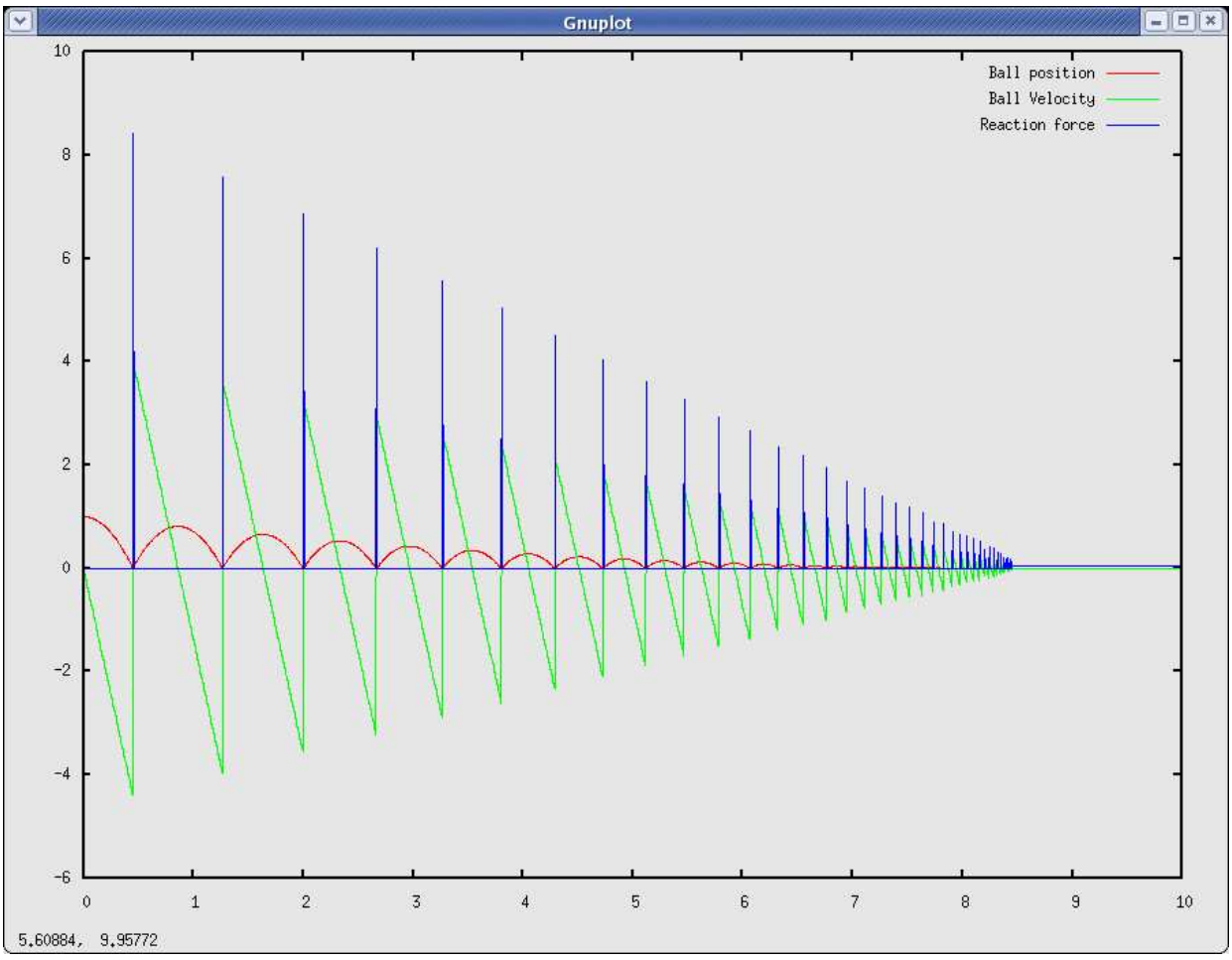

Figure 9.1: Evolution of the BouncingBall in relation to the time

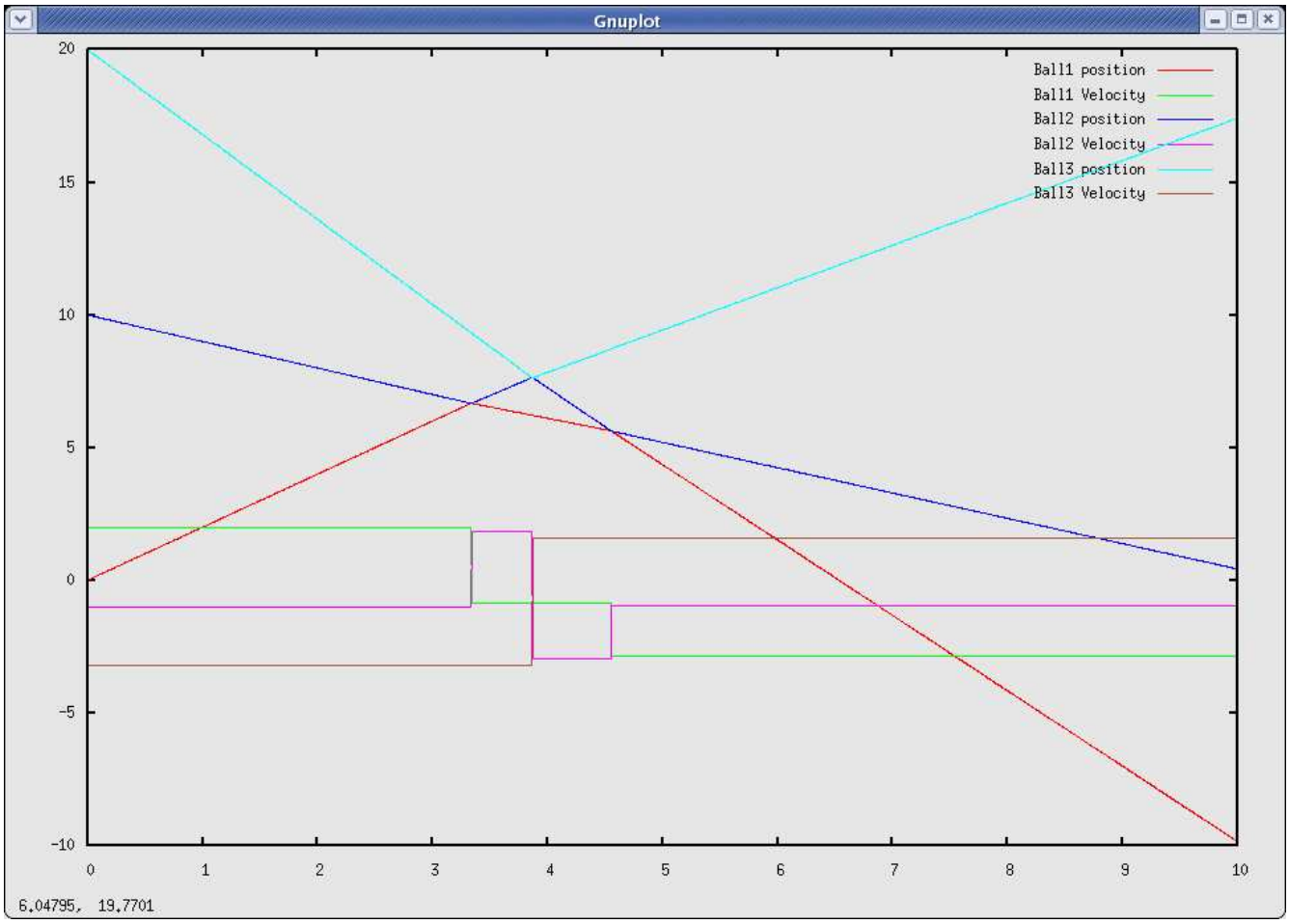

Figure 9.2: Evolution of the RollingBalls in relation to the time

9.2 RollingBalls

9.2.1 Purpose

This sample has been made to test a non smooth dynamical system with several dynamical systems.
There are 3 moving points.
All the movements are only on 1 axis!
The 3 points are aligned. Each point is moving with different speeds, and will enter in contact with the other points during the simulation.

9.2.2 Results

We can see on the figure (9.2) the position of the points and the speed of these points.

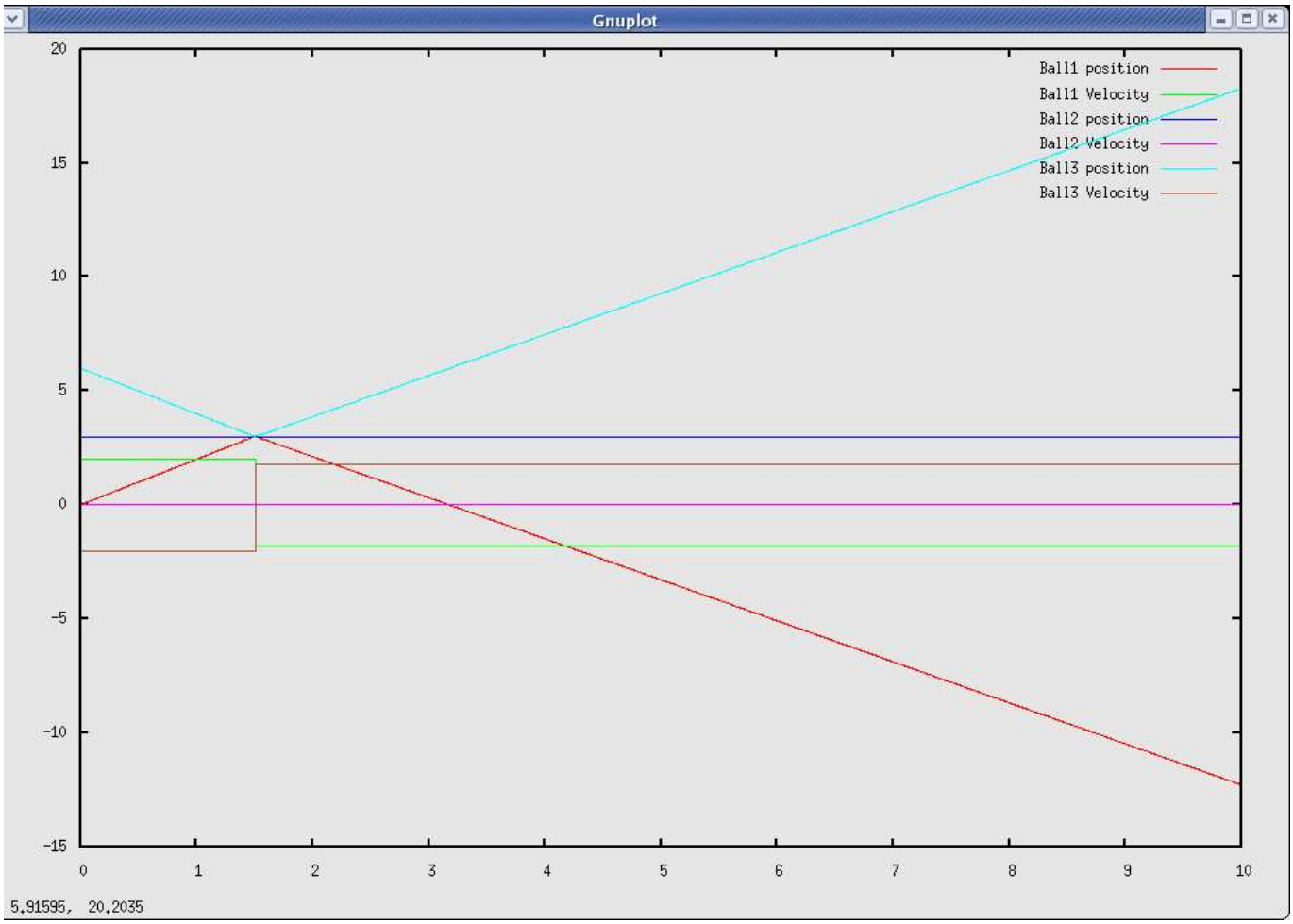

Figure 9.3: Evolution of the DoubleContact balls in relation to the time

9.3 DoubleContact

9.3.1 Purpose

This sample has been made to test multiple contacts between dynamical systems.
There are 3 points of which 1 is static, and the 2 others are moving.
All the movement are only on 1 axis!
The 3 points are aligned. The 2 points that are moving are converging, and the convergence point correponds to the place of the third point.
Interactions are defined between points 1 and 2, 2 and 3, 1 and 3. So we have one contact point where the 3 interactions are activated.

9.3.2 Results

We can see on the figure (9.3) the position of the points and the speed of these points.
When the contact occurs, the moving points are bouncing, changing there direction and losing some speed because of the Newton impact law coefficient (0.9). The middle point don't move because the 2 moving balls hit it with the same speed at the same time.

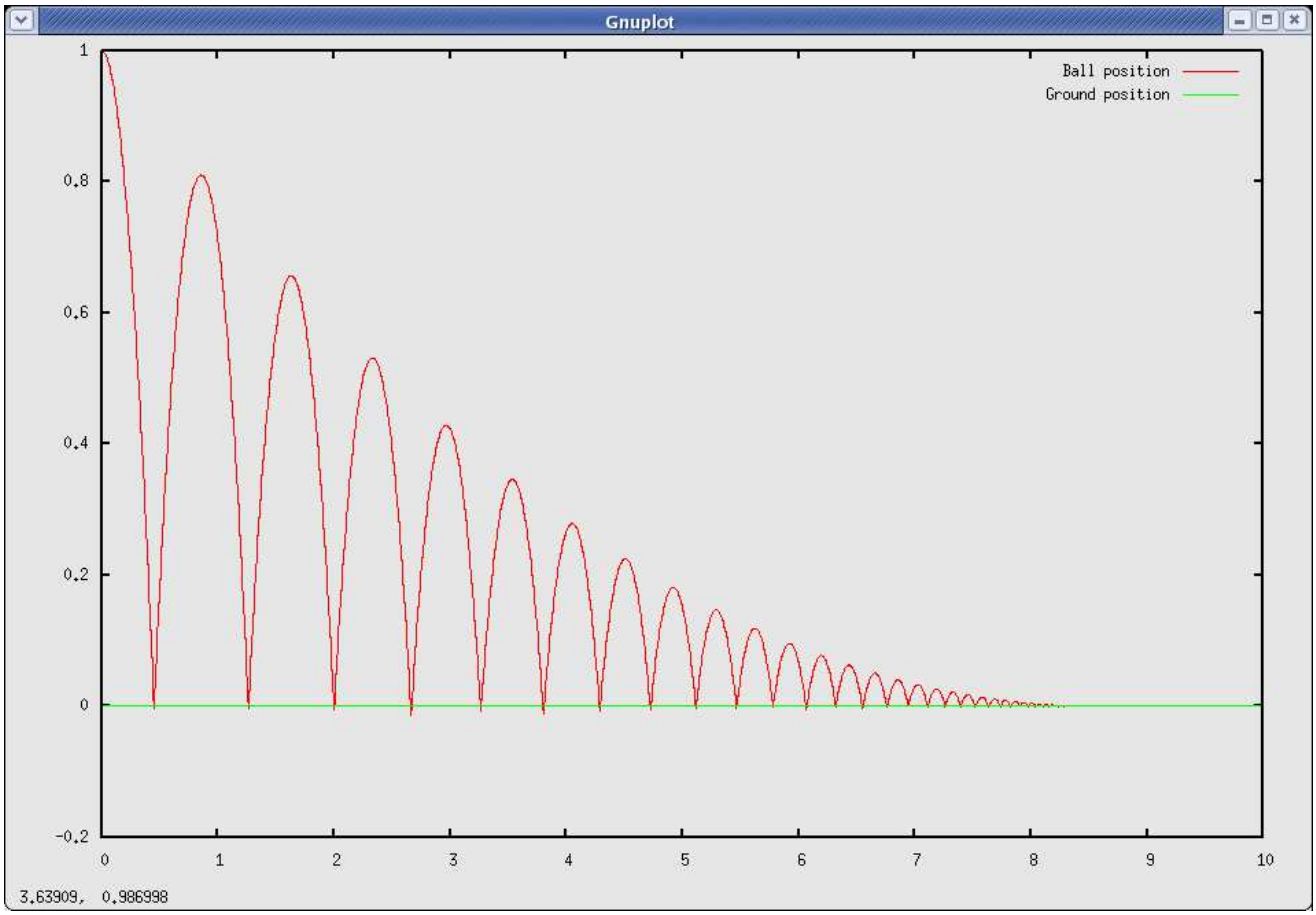

Figure 9.4: Evolution of the position of a BouncingBall in 2D

9.4 Ball2D

9.4.1 Purpose

This sample is the same as the BouncingBall, but this time, the point is moving in 2 dimensions.
The point is falling because of gravity force, has an advance velocity, and rebounds on a solid floor.

9.4.2 Results

We can see on the figure (9.4) the height of the point (Y axis) according to his position on the X axis.

Appendix A

XML file sample

```
<SiconosModel Author="jbarbier">

<Time>
<t0>0</t0>
<T>1000</T>
</Time>

<NSDS bvp='true'>
<!-- DSs defined in the problem -->
<DS_Definition>
<LagrangianNLDS number='1'>
<Id> toto </Id>
<x vectorSize='6'>
4.5454 454.5 5 5 5
4.95222
</x>

<xDot vectorSize='6'>
4.5454 454.5 8 8 8
4.95222
</xDot>

<xMemory sizeMax='5'>
  <Memory vectorFile='./sample/vector.dat'/>
  <Memory vectorSize='2'>
   54.54 5454212.5445456456
  </Memory>
  <Memory vectorFile='./sample/vector.dat'/>
</xMemory>

<xDotMemory sizeMax='4'>
  <Memory vectorFile='./sample/vector.dat'/>
  <Memory vectorSize='2'>
   54.54 5454212.5445456456
  </Memory>
```

```
  <Memory vectorFile='./sample/vector.dat'/>
</xDotMemory>

<x0 vectorFile='./sample/vector2.dat'/>
<StepsInMemory>8</StepsInMemory>

<QNLInertia vectorPlugin="BasicPlugin:computeQNLInertia"/>
<q vectorFile='./sample/vector.dat'/>
<q0 vectorFile='./sample/vector.dat'/>

<qMemory sizeMax='4'>
  <Memory vectorFile='./sample/vector.dat'/>
  <Memory vectorSize='2'>
   54.54 5454212.5445456456
  </Memory>
  <Memory vectorFile='./sample/vector.dat'/>
</qMemory>

<Velocity vectorFile='./sample/vector.dat'/>
<Velocity0 vectorFile='./sample/vector.dat'/>
<VelocityMemory sizeMax='1'>
<Memory vectorFile='./sample/vector.dat'/>
</VelocityMemory>

<Fint vectorPlugin="BasicPlugin:computeFInt"/>
<Fext vectorPlugin="BasicPlugin:computeFExt"/>

<JacobianQFint matrixPlugin="BasicPlugin:computeJacobianQFInt"/>
<JacobianVelocityFint matrixPlugin="BasicPlugin:computeJacobianVelocityFInt"/>
  <JacobianQQNLInertia matrixPlugin="BasicPlugin:computeJacobianQQNLInertia"/>
  <JacobianVelocityQNLInertia matrixPlugin="BasicPlugin:computeJacobianVelocityQNLIne

<ndof>3</ndof>

<!-- Optional : if NSDS is bvp -->
<BoundaryCondition type="NLinear">
</BoundaryCondition>
</LagrangianNLDS>

<LagrangianTIDS number='2'>
<!-- Commun at all DS -->
<Id> toto </Id>
<xDot vectorFile='./sample/vector2.dat'/>

<xMemory sizeMax='4'>
  <Memory vectorFile='./sample/vector2.dat'/>
  <Memory vectorSize='2'>
   54.54 5454212.5445456456
  </Memory>
```

```
  <Memory vectorFile='./sample/vector2.dat'/>
</xMemory>

<xDotMemory sizeMax='4'>
  <Memory vectorFile='./sample/vector2.dat'/>
  <Memory vectorSize="2">
   54.54 5454212.5445456456
  </Memory>
  <Memory vectorFile='./sample/vector2.dat'/>
</xDotMemory>

<x0 vectorFile='./sample/vector2.dat'/>
<StepsInMemory>8</StepsInMemory>

<!-- Specific to LagrangianTIDS-->

<q vectorFile='./sample/vector.dat'/>
<q0 vectorFile='./sample/vector.dat'/>

<qMemory sizeMax='4'>
  <Memory vectorFile='./sample/vector.dat'/>
  <Memory vectorSize='2'>
   54.54 5454212.5445456456
  </Memory>
  <Memory vectorFile='./sample/vector.dat'/>
</qMemory>

<Velocity vectorFile='./sample/vector.dat'/>
<Velocity0 vectorFile='./sample/vector.dat'/>
<VelocityMemory sizeMax='1'>
<Memory vectorFile='./sample/vector.dat'/>
</VelocityMemory>

<M matrixFile='./sample/matrix.dat'/>

<ndof>3</ndof>

<K matrixFile='./sample/matrix.dat'/>
<C matrixFile='./sample/matrix.dat'/>

<!-- Optional : if NSDS is bvp -->
<BoundaryCondition type="Linear">
<Omega vectorFile='./sample/vector.dat'/>
<Omega0 matrixFile='./sample/matrix.dat'/>
<OmegaT matrixFile='./sample/matrix.dat'/>
</BoundaryCondition>
</LagrangianTIDS>

<LinearSystemDS number='3'>
```

```
<!-- Commun at all DS -->
<Id> tutu </Id>
<n> 3 </n>
<x vectorFile='./sample/vector.dat'/>

<xDot vectorFile='./sample/vector.dat'/>

<xMemory sizeMax='4'>
  <Memory vectorFile='./sample/vector.dat'/>
  <Memory vectorSize="2">
   54.54 5454212.5445456456
  </Memory>
  <Memory vectorFile='./sample/vector.dat'/>
</xMemory>

<xDotMemory sizeMax='4'>
  <Memory vectorFile='./sample/vector.dat'/>
  <Memory vectorSize="2">
   54.54 5454212.5445456456
  </Memory>
  <Memory vectorFile='./sample/vector.dat'/>
</xDotMemory>

<x0 vectorFile='./sample/vector.dat'/>

<StepsInMemory>8</StepsInMemory>

<!-- Specific to LinearSystemDS-->
<A matrixFile='./sample/matrix.dat'/>
<B matrixFile='./sample/matrix.dat'/>

<u vectorFile='./sample/vector.dat'/>
<f vectorPlugin="BasicPlugin:computeF"/>

<!-- Optional : if NSDS is bvp -->
<BoundaryCondition type="Periodic">
</BoundaryCondition>
</LinearSystemDS>


</DS_Definition>

<!-- Interactions defined in the problem -->
<Interaction_Definition>
<Interaction number='1'>

<Status>1</Status>
<Id>totu</Id>
<nInter>1</nInter>
```

```
<!-- List of couple of DS concerned by this interaction -->

<DS_Concerned size='2'>
<DS number='1' interactsWithDS_Number='2'/>
<DS number='1' interactsWithDS_Number='3'/>
</DS_Concerned>

<!-- Relation of this interaction -->
<Interaction_Content>
<LTI>
<C matrixFile='./sample/matrix.dat'/>
<D matrixFile='./sample/matrix.dat'/>
<E matrixFile='./sample/matrix.dat'/>
<a vectorFile='./sample/vector.dat'/>
</LTI>

<!-- NS Law of this interaction -->
<ComplementarityCondition>
</ComplementarityCondition>
</Interaction_Content>

</Interaction>

<!-- A definition of a DS interaction, and list of couple of DSs who are concerned by
<Interaction number='2'>

<Status>1</Status>
<Id>twtu</Id>
<nInter>1</nInter>
<!-- List of couple of DS concerned by this interaction -->
<DS_Concerned size='2'>
<DS number='1' interactsWithDS_Number='2'/>
<DS number='1' interactsWithDS_Number='3'/>
</DS_Concerned>

<Interaction_Content>
<!-- RELATION of this interaction -->
<LL>
<H matrixFile='./sample/matrix.dat'/>
<b vectorFile='./sample/vector.dat'/>
</LL>

<!-- NS Law of this interaction -->
<Relay>
<c>5.021</c>
<d>5456.65</d>
</Relay>
</Interaction_Content>
```

```
</Interaction>


<!-- A definition of a DS interaction, and list of couple of DSs who are concerned by
<Interaction number='3'>

<Status>5 8</Status>
<Id>tuto</Id>
<nInter>1</nInter>
<!-- List of couple of DS concerned by this interaction -->
<DS_Concerned size='2'>
<DS number='1' interactsWithDS_Number='2'/>
<DS number='1' interactsWithDS_Number='3'/>
</DS_Concerned>

<Interaction_Content>
<!-- Relation of this interaction -->
<LNL>
<computeInput plugin="BasicPlugin:computeInput"/>
<computeOutput plugin="BasicPlugin:computeOutput"/>
</LNL>

<!-- NS Law of this interaction -->
<NewtonImpactLaw>
<e>0.95</e>
</NewtonImpactLaw>
</Interaction_Content>


</Interaction>
</Interaction_Definition>
</NSDS>

<!-- Strategy to use in order to solve the problem -->
<Strategy type='TimeStepping'>
<TimeDiscretisation isConstant='true'>
<h>0.65</h>
<N>52</N>
<tk vectorSize='2'>5.545 548.2</tk>
<hMin>54.534</hMin>
<hMax>105.54</hMax>
</TimeDiscretisation>

<!-- One Step Integrators -->
<OneStepIntegrator_Definition>
<Moreau>
<DS_Concerned size='1'>
<DS number='2'/>
</DS_Concerned>
```

```
<r>4</r>
<Theta> 0.5</Theta>
<W matrixColSize="0" matrixRowSize="0"/>
</Moreau>

<!-- A definition of a OneStepIntegrator, and list of couple of DSs who are concerned
<Adams>
<DS_Concerned size='1'>
<DS number='1'/>
</DS_Concerned>
<r>4</r>
</Adams>

<!-- A definition of a OneStepIntegrator, and list of couple of DSs who are concerned
<LSODAR>
<DS_Concerned size='1'>
<DS number='3'/>
</DS_Concerned>
<r>4</r>
</LSODAR>
</OneStepIntegrator_Definition>

  <LCP>
<!-- General for one step NS problem -->
<n>8</n>

<!-- Specific at LCP -->
<M matrixFile='./sample/matrix.dat'/>
<q vectorFile='./sample/vector.dat'/>

<Interaction_Concerned size='3'>
<Interaction number='1'/>
<Interaction number='2'/>
<Interaction number='3'/>
</Interaction_Concerned>

  </LCP>
</Strategy>
</SiconosModel>
```

Appendix B

High level commands

These functions are the highest level of granularity of command to drive a simulation with the platform :

- **LoadXMLFile([file name])** Allows to load a SICONOS XML file. [file name] is an absolute path (tbc).
- **InitStrategy()** Initializes the strategy of simulation.
- **GetStartTime()** Gets the starting time of simulation.
- **GetCurrentTime()** Gets the current time of simulation.
- **GetEndTime()** Gets the final time of simulation.
- **NextStep()** Passes to next step of simulation, and saves values in memory if needed.
- **ComputeFreeState()** Computes new state of the system without applying constraints
- **FormaliseOneStepNSProblem()** Formalizes non-smooth problem.
- **ComputeOneStepNSProblem()** Computes one step of the problem.
- **UpdateState()** Updates state of the system regards to result of computing problem (active relations, etc.).
- **SaveXMLFile([file name])** Saves the state of the system in a SICONOS XML file.

Acronyms

List of Figures

List of Tables

Bibliography