Siconos Software Overview

Franck Pérignon

INRIA Rhône-Alpes

September 14, 2006

Overview of the Siconos Platform

Functionalities:

modeling, simulation, (analysis and control) of Non Smooth Dynamical Systems.

Overview of the Siconos Platform

Functionalities:

modeling, simulation, (analysis and control) of Non Smooth Dynamical Systems.

Constraints and Requirements:

- various applications fields (Mechanics, Electronics . . .) and corresponding modeling habits and formulations
- various mathematical and numerical tools
- various skills in computer science (from the high perfomance computing to the Matlab users)

Overview of the Siconos Platform

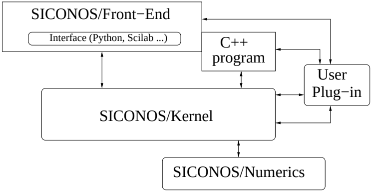
Functionalities:

modeling, simulation, (analysis and control) of Non Smooth Dynamical Systems.

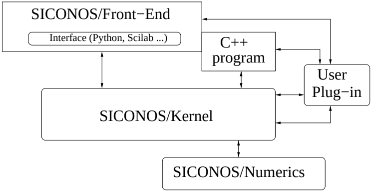
Constraints and Requirements:

- various applications fields (Mechanics, Electronics . . .) and corresponding modeling habits and formulations
- various mathematical and numerical tools
- various skills in computer science (from the high perfomance computing to the Matlab users)
- links and interfaces with existing softwares:
  - low-level numerical libraries (BLAS, LAPACK, ODEPACK, . . .)
  - Matlab or Scilab dedicated user toolbox
  - simulation tools for an application field: Scicos, Simulink, FEM and DEM Sofware (LMGC90, . . .), Hybrid Modeling Language (Modelica, . . .)

Siconos components diagram

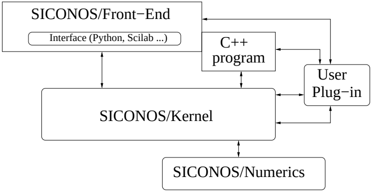


Siconos components diagram



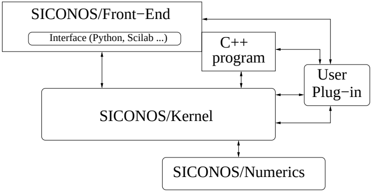
- *SICONOS/Numerics API C:*
  shared dynamic library that provides low-level solvers and algorithms in C and fortran.
  Sources: NSSpack (LCP, Friction ...), odepack (Lsodar ...).

Siconos components diagram



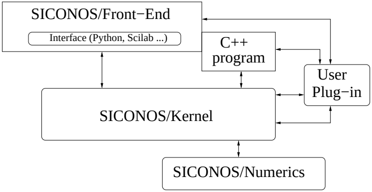
- *SICONOS/Numerics API C:*
  shared dynamic library that provides low-level solvers and algorithms in C and fortran.
  Sources: NSSpack (LCP, Friction ...), odepack (Lsodar ...).
- *SICONOS/Kernel:* API C++: compiled command files with high level methods (C++ Constructors and/or XML file data loading.)
  $\Rightarrow$ from $simulation \rightarrow run()$ to $DynamicalSystem \rightarrow computeFext(t)$ ......

Siconos components diagram



- *SICONOS/Numerics API C:*
  shared dynamic library that provides low-level solvers and algorithms in C and fortran.
  Sources: NSSpack (LCP, Friction ...), odepack (Lsodar ...).
- *SICONOS/Kernel:* API C++: compiled command files with high level methods (C++ Constructors and/or XML file data loading.)
  $\Rightarrow$ from *simulation* $\rightarrow$ *run*() to *DynamicalSystem* $\rightarrow$ *computeFext*(t) ......
- *SICONOS/Frond-End*: "user-friendly" interface providing a more interactive way of using the platform.
  - API C++ with interactive environment Python scripting (Swig wrapper).
  - API C: Scilab and Matlab interfaces.

Siconos components diagram

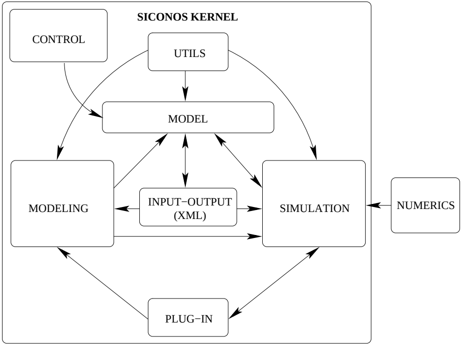


- *SICONOS/Numerics API C:*
  shared dynamic library that provides low-level solvers and algorithms in C and fortran.
  Sources: NSSpack (LCP, Friction ...), odepack (Lsodar ...).
- *SICONOS/Kernel:* API C++: compiled command files with high level methods (C++ Constructors and/or XML file data loading.)
  $\Rightarrow$ from *simulation* $\rightarrow$ *run*() to *DynamicalSystem* $\rightarrow$ *computeFext*(t) ......
- *SICONOS/Frond-End*: "user-friendly" interface providing a more interactive way of using the platform.
  - API C++ with interactive environment Python scripting (Swig wrapper).
  - API C: Scilab and Matlab interfaces.
- *User Plug-In*: to allow user to add specific dedicated functions or toolboxes.

Kernel Components

Kernel

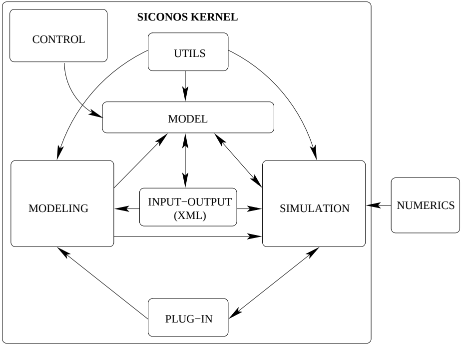
C++ stand-alone dynamic library, based on Numerics (simulation part)



Kernel Components

Kernel

C++ stand-alone dynamic library, based on Numerics (simulation part)



- *Modeling* and *Simulation* clearly separated and independent (communicate through object *Model*) ⇒ easiest handling for user
- data I/O: xml management, independent package (possibly removed in a "light" version ...)
- User plug-in
- Utils: matrices, vectors, exceptions, handling.

Modeling Principle:

Non Smooth Dynamical system

Dynamical system

$$\dot{x} = f(x,t) + r$$

Modeling Principle:

Non Smooth Dynamical system

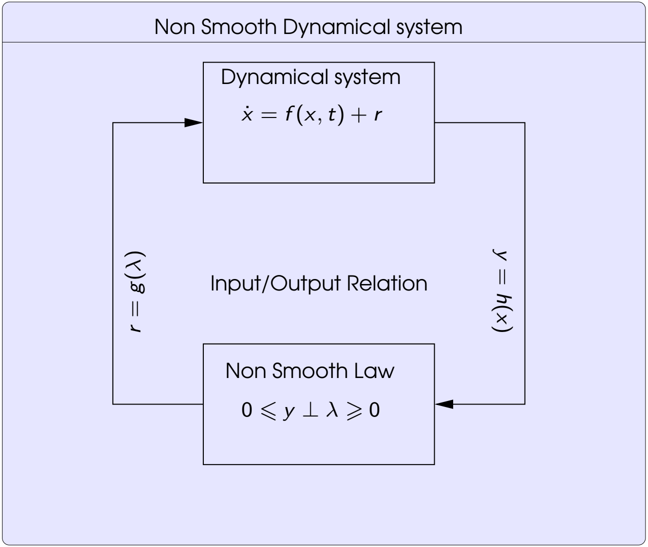
Dynamical system

$$\dot{x} = f(x, t) + r$$

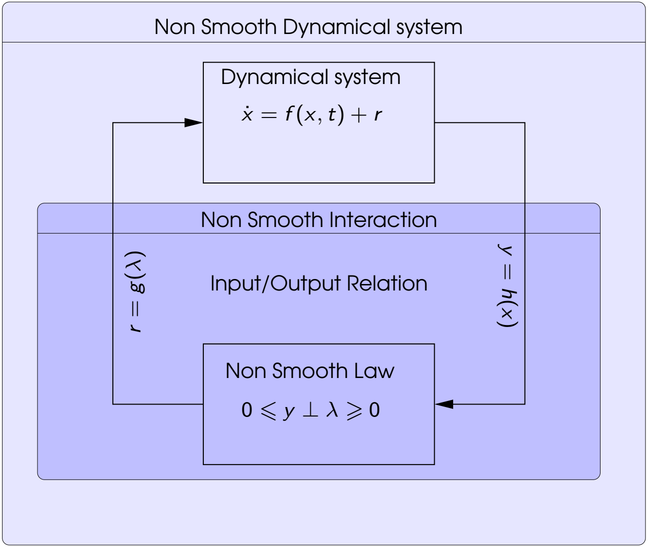
Non Smooth Law

$$0 \leqslant y \perp \lambda \geqslant 0$$

Modeling Principle:

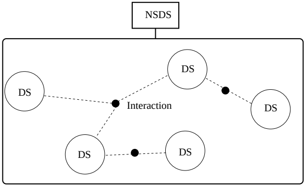


Modeling Principle:



Kernel Modeling Part

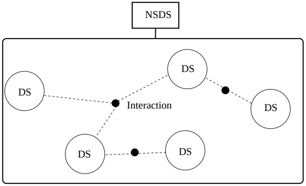
Siconos Non Smooth Dynamical System:



- *Dynamical System*: a set of ODEs
- *Interaction*: a set of relations (ie constraints) and a non-smooth law

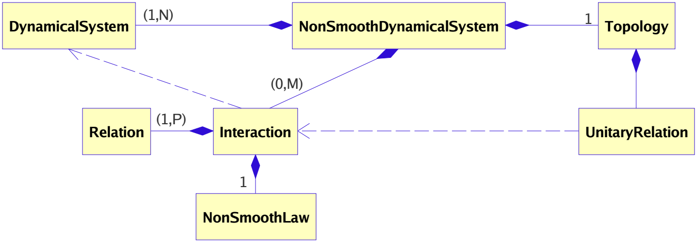
Kernel Modeling Part

Siconos Non Smooth Dynamical System:

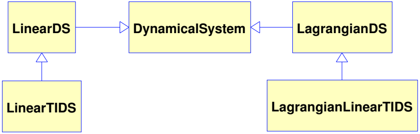


- *Dynamical System*: a set of ODEs
- *Interaction*: a set of relations (ie constraints) and a non-smooth law
- *Topology*: link with the simulation, handles relative degrees, index sets ...

Simplified Modeling Tools class diagram:



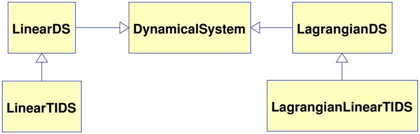
Dynamical Systems in Siconos/Kernel



- Parent Class **DynamicalSystem**

$$\dot{x} = f(x, \dot{x}, t) + T(x)u(x, t) + r$$

Dynamical Systems in Siconos/Kernel



- Parent Class **DynamicalSystem**

$$\dot{x} = f(x, \dot{x}, t) + T(x)u(x, t) + r$$

- Derived Classes
  - **LinearDS** Linear Dynamical Systems

$$\dot{x} = A(t)x + Tu(t) + b(t) + r$$

  - **LagrangianDS** Lagrangian Dynamical Systems

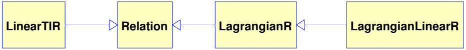
$$M(q)\ddot{q} + NNL(q, \dot{q}) + F_{int}(\dot{q}, q, t) = F_{ext}(t) + T(q)u(q, t) + p$$

  - **LagrangianLinearTIDS** Lagrangian Linear Time Invariant Systems

$$M\ddot{q} + C\dot{q} + Kq = F_{ext}(t) + Tu(t) + p$$

Note: all operators ($f(x, t)$, $M(q)$, ...) can be set either as matrices (when constant) or with a user-defined external function (plug-in).

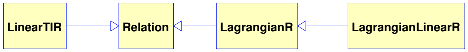
Relations



- Parent Class **Relation**

$$y = h(x, t, \ldots) \quad , \quad r = g(\lambda, t, \ldots)$$

Relations



- Parent Class **Relation**

$$y = h(x, t, ...) \quad , \quad r = g(\lambda, t, ...)$$

- Derived Classes:
  - **LinearTIR** Linear Time Invariant Relation

  $$y = Cx + Fu + D\lambda + e, \quad r = B\lambda$$

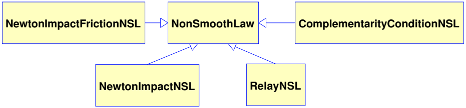
  - **LagrangianR** Lagrangian Relation

  $$\dot{y} = H(q, t, \ldots)\dot{q}, \quad p = H^t(q, t, \ldots)\lambda$$

  - **LagrangianLinearR** Lagrangian Linear Relation

  $$\dot{y} = H\dot{q} + b, \quad p = H^t\lambda$$

Non Smooth laws



- Parent Class **NonSmoothLaw**
- Derived Classes
  - **ComplementarityConditionNSL** Complementarity condition or unilateral contact

$$0 \leqslant y \perp \lambda \geqslant 0$$

  - **Relay** condition.

$$\begin{cases} \dot{y} = 0, |\lambda| \leqslant 1 \\ \dot{y} \neq 0, \lambda = \text{sign}(y) \end{cases}$$

  - **NewtonImpactLawNSL** Newton impact Law.

$$\text{if } y(t) = 0, \quad 0 \leqslant \dot{y}(t^+) + e\dot{y}(t^-) \perp \lambda \geqslant 0$$

  - **NewtonImpactFrictionNSL** Newton impact and Friction (Coulomb) Law.

C++ description of a Model

- Dynamical Systems definition:

```
DynamicalSystem * DS1 = new LagrangianLinearTIDS(nDof,q0,v0,Mass);
DS1→setComputeFExtFunction("BallPlugin.so", "ballFExt");
```

C++ description of a Model

- Dynamical Systems definition:

  DynamicalSystem * DS1 = new LagrangianLinearTIDS(nDof,q0,v0,Mass);
  DS1→setComputeFExtFunction("BallPlugin.so", "ballFExt");

- Interactions definition: non smooth law and relation:

  NonSmoothLaw * nslaw = new NewtonImpactNSL(e);
  Relation * relation = new LagrangianLinearR(H,b);
  Interaction * inter = new Interaction(name, listOfDS,dim, nslaw, relation);

C++ description of a Model

- Dynamical Systems definition:

```
DynamicalSystem * DS1 = new LagrangianLinearTIDS(nDof,q0,v0,Mass);
DS1→setComputeFExtFunction("BallPlugin.so", "ballFExt");
```

- Interactions definition: non smooth law and relation:

```
NonSmoothLaw * nslaw = new NewtonImpactNSL(e);
Relation * relation = new LagrangianLinearR(H,b);
Interaction * inter = new Interaction(name, listOfDS,dim, nslaw, relation);
```

- Non Smooth Dynamical System and Model

```
NonSmoothDynamicalSystem * nsds = new NonSmoothDynamicalSystem(allDS,
allInteractions);
Model * theModel = new Model(t0,T);
theModel→setNonSmoothDynamicalSystemPtr(nsds);
```

C++ description of a Model

- Dynamical Systems definition:

```
DynamicalSystem * DS1 = new LagrangianLinearTIDS(nDof,q0,v0,Mass);
DS1→setComputeFExtFunction("BallPlugin.so", "ballFExt");
```

- Interactions definition: non smooth law and relation:

```
NonSmoothLaw * nslaw = new NewtonImpactNSL(e);
Relation * relation = new LagrangianLinearR(H,b);
Interaction * inter = new Interaction(name, listOfDS,dim, nslaw, relation);
```

- Non Smooth Dynamical System and Model

```
NonSmoothDynamicalSystem * nsds = new NonSmoothDynamicalSystem(allDS,
allInteractions);
Model * theModel = new Model(t0,T);
theModel→setNonSmoothDynamicalSystemPtr(nsds);
```

or in a simpler way, xml loading:

Model * = new Model(nameOfXMLFile);

$< SiconosModel >$

...

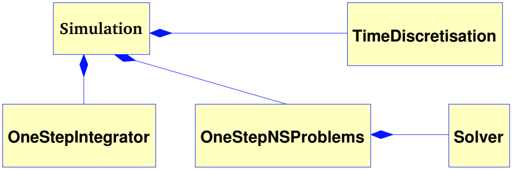
$< DS\_Definition >< LagrangianLinearTIDSnumber = 1 >$

$< ndof > 3 < / \ ndof >$

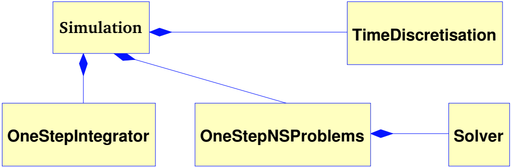
$< q0vectorSize = 3 > 1.0\ 0.0\ 0.0 < / \ q0 >$

...

Simulation tools in Siconos/Kernel



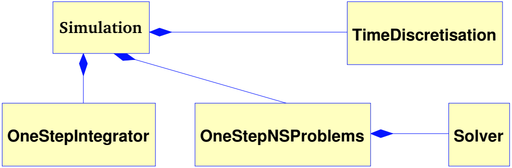
Simulation tools in Siconos/Kernel



Simulation description in C++ input file:

```
Simulation* s = new TimeStepping(theModel);
TimeDiscretisation * t = new TimeDiscretisation(timeStep,s);
OneStepIntegrator * OSI = new Moreau(listOfDS,theta,s);
OneStepNSProblem * osnspb = new LCP(s, "LCP",Lemke,parameters);
```

Simulation tools in Siconos/Kernel



Simulation description in C++ input file:

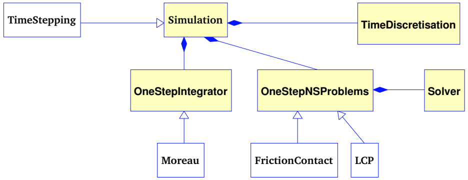
```
Simulation* s = new TimeStepping(theModel);
TimeDiscretisation * t = new TimeDiscretisation(timeStep,s);
OneStepIntegrator * OSI = new Moreau(listOfDS,theta,s);
OneStepNSProblem * osnspb = new LCP(s, "LCP",Lemke,parameters);
```

Unitary Relation and Index Sets

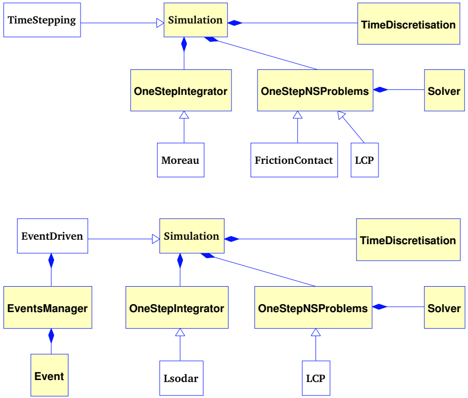
UR: $y^i = h(q, \dots)$.
Index Sets: set of Unitary Relations (UR).

- $I_0 = \{UR_\alpha\}$ all unilateral constraints in the system, ie all the potential interactions/relations of the systems.
- $I_i = \{UR_\alpha, \alpha \in I_{i-1}, y^{(i-1)} = 0\} \subset I_{i-1}$

Simulation tools in Siconos/Kernel



Simulation tools in Siconos/Kernel



OneStepIntegrator:

- **Moreau**: Moreau's Time-stepping integrator
- **Lsodar**: Numerical integration scheme based on the Livermore Solver for Ordinary Differential Equations with root finding.

OnestepNSproblem: Numerical one step non smooth problem formulation and solver.

- **LCP** Linear Complementarity Problem

$$\begin{cases} w = Mz + q \\ 0 \leqslant w \perp z \geqslant 0 \end{cases}$$

- **FrictionContact2D(3D)** Two(three)-dimensional contact friction problem
- **QP** Quadratic programming problem

$$\begin{cases} \min \frac{1}{2} z^T Q z + z^T p \\ z \geqslant 0 \end{cases}$$

- **Relay**

Moreau Time-Stepping

One Step of Integration: (Start from state at time t_i, q_i, v_i, y_i... known).

- compute free state (ie without non smooth part) $\Rightarrow q_{free}, v_{free}$
- update index sets: compute $y_p = y_i + 0.5 * \dot{y}_i$, if $y_p < 0$, add the corresponding UR in I_1
- build and solve LCP "impact" (ie at velocity level) for Unitary Relations in $I_1 \Rightarrow (\lambda, y)_{i+1}$
- compute non smooth part $p_{i+1} = f(\lambda_{i+1}, \ldots)$
- update state of the Dynamical Systems: $(q, v)_{i+1} = function(q_{free}, v_{free}, p_{i+1}, \ldots)$
- update output $y_{i+1} = h(q_{i+1}, \ldots)$

Moreau Time-Stepping

One Step of Integration: (Start from state at time t_i, q_i, v_i, y_i... known).

- compute free state (ie without non smooth part) $\Rightarrow q_{free}, v_{free}$
- update index sets: compute $y_p = y_i + 0.5 * \dot{y}_i$, if $y_p < 0$, add the corresponding UR in I_1
- build and solve LCP "impact" (ie at velocity level) for Unitary Relations in $I_1 \Rightarrow (\lambda, y)_{i+1}$
- compute non smooth part $p_{i+1} = f(\lambda_{i+1}, \ldots)$
- update state of the Dynamical Systems: $(q, v)_{i+1} = function(q_{free}, v_{free}, p_{i+1}, \ldots)$
- update output $y_{i+1} = h(q_{i+1}, \ldots)$

In Siconos C++ input file:

```
while(currentTimeStep < max)
{
s→computeFreeStep();
s→updateIndexSets();
s→computeOneStepNSProblem();
s→update
}
```

Event-Driven

EventsManager: member of EventDriven simulation class, a list of all possible Events.
Events: Time Discretisation or Non Smooth.

Start = current event, known.

- Computation of the temporary values of $(y_{k+1}, \dot{y}_{k+1})$ by performing the time-integration of the smooth dynamics up to an event (lsodar with roots finding).
- Compute the temporary index-sets
- if $I_1 - I_2 \neq \emptyset$ (*impacts occur*) then
  build, solve the LCP impact and update the index-sets
- if $I_2 \neq \emptyset$ then build and solve the LCP at acceleration level, and update index sets.

Event-Driven

EventsManager: member of EventDriven simulation class, a list of all possible Events.
Events: Time Discretisation or Non Smooth.

Start = current event, known.

- Computation of the temporary values of $(y_{k+1}, \dot{y}_{k+1})$ by performing the time-integration of the smooth dynamics up to an event (lsodar with roots finding).
- Compute the temporary index-sets
- if $I_1 - I_2 \neq \emptyset$ (*impacts occur*) then
  build, solve the LCP impact and update the index-sets
- if $I_2 \neq \emptyset$ then build and solve the LCP at acceleration level, and update index sets.

In Siconos C++ input file:

```
while(eventsManager→hasNextEvent())
{
s→advanceToEvent();
eventsManager→processEvents();
}
```

Other usefull tools ...

Siconos Algebra

Matrices and vectors handling. Based on lapack++.
Next release: migration to Boost C++ library ⇒ sparse, band, block . . . matrices.

Other usefull tools ...

Siconos Algebra

Matrices and vectors handling. Based on lapack++.
Next release: migration to Boost C++ library ⇒ sparse, band, block . . . matrices.

Data input/output

use xml schemes and libxml2 library. Next release: introduce Factory methods to make xml management optional.

Other usefull tools ...

Siconos Algebra

Matrices and vectors handling. Based on lapack++.
Next release: migration to Boost C++ library ⇒ sparse, band, block . . . matrices.

Data input/output

use xml schemes and libxml2 library. Next release: introduce Factory methods to make xml management optional.

Install

- Autoconf, automake and libtool utilities for each package (Numerics, Kernel and Front-End): configure.ac and Makefile.am files distributed with the software.
  ⇒ platform independent. Future: MacOS and Windows (...)
- One "simple" way to install the platform: configure ; make ; make install.
- Required or optional external libraries: lapack++ (boost++), libxml2, cppunit ...

Other usefull tools ...

Siconos Algebra

Matrices and vectors handling. Based on lapack++.
Next release: migration to Boost C++ library ⇒ sparse, band, block . . . matrices.

Data input/output

use xml schemes and libxml2 library. Next release: introduce Factory methods to make xml management optional.

Install

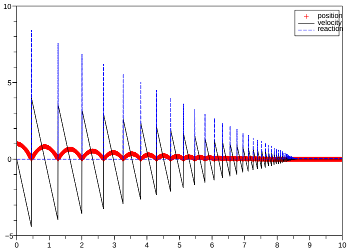
- Autoconf, automake and libtool utilities for each package (Numerics, Kernel and Front-End): configure.ac and Makefile.am files distributed with the software.
  ⇒ platform independent. Future: MacOS and Windows (...)
- One "simple" way to install the platform: configure ; make ; make install.
- Required or optional external libraries: lapack++ (boost++), libxml2, cppunit ...

Running a simulation

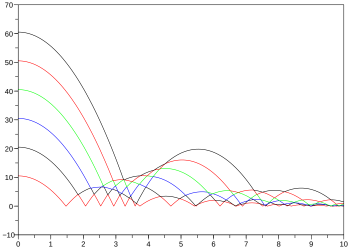
- Direct c++ program writing.
- Python (Swig → Boost), Scilab or Matlab (API C) interfaces.

Model: Lagrangian Linear Time Invariant Dynamical Systems with Lagrangian Linear Relations, Newton Impact Law.
Simulation: Moreau's Time Stepping or Event Driven.

Bouncing Ball

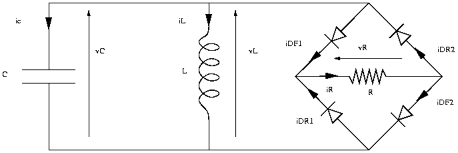


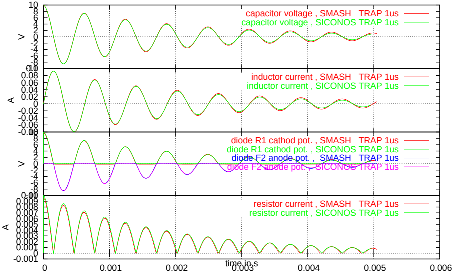
Beads column



A 4 diodes bridge wave rectifier.

Model: Linear Dynamical System with Linear Relations, Complementarity Condition Non Smooth Law.
Simulation: Moreau's Time Stepping

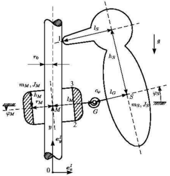


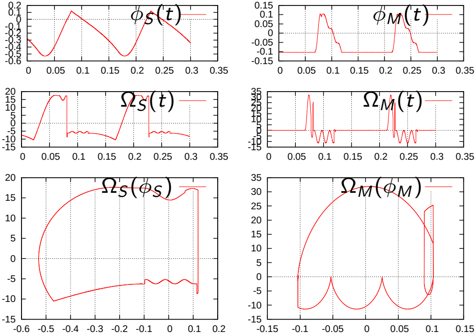


Comparison between the SICONOS Platform (Non Smooth LCS model) and SPICE simulator (Smooth Diode model).

Woodpecker toy (sample from Michael Moeller (CR10))

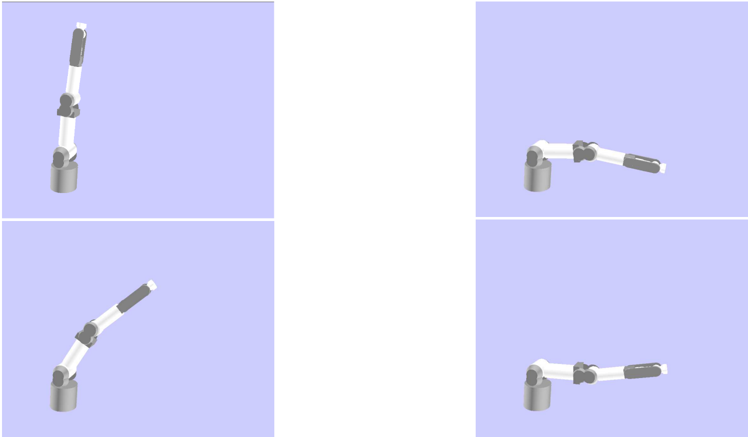
Model: Lagrangian Linear Dynamical System, Lagrangian Linear Relations, Newton impact-friction law.
Simulation: Moreau's Time Stepping





A Robotic Arm (Pa10)

Model: Lagrangian Non Linear Dynamical System with Lagrangian Non Linear Relations, Newton impact.
Simulation: Moreau's Time Stepping



Help and Documentation

- Doxygen tools for automatic documentation in Numerics and Kernel
- Users, developers and theoretical manuals (in progress ...)
- Web pages, Bug tracker, forum ... on Gforge.
- Samples library as templates.

Help and Documentation

- Doxygen tools for automatic documentation in Numerics and Kernel
- Users, developers and theoretical manuals (in progress ...)
- Web pages, Bug tracker, forum ... on Gforge.
- Samples library as templates.

Diffusion

- The SICONOS platform is distributed under GPL licence.
- Visit the Gforge Web site for
  - Documentations
  - Mailing lists
  - Downloads
  - Bug tracker
  - Contributing, . . .

`http://gforge.inria.fr/projects/siconos/`

Conclusion

Siconos software provides:

- a stand-alone open platform, operational and able to solve various non smooth problems. Specifities: plug-in system, multiple interfaces, "macro language" (from *simulation* $\longrightarrow$ *run*() to *computeRHS*())

Conclusion

Siconos software provides:

- a stand-alone open platform, operational and able to solve various non smooth problems. Specifities: plug-in system, multiple interfaces, "macro language" (from *simulation* $\rightarrow$ *run*() to *computeRHS*())
- or a plugged external library that provides a computational tools

Conclusion

Siconos software provides:

- a stand-alone open platform, operational and able to solve various non smooth problems. Specifities: plug-in system, multiple interfaces, "macro language" (from *simulation* $\longrightarrow$ *run*() to *computeRHS*())
- or a plugged external library that provides a computational tools
- not dedicated to a specific domain, possibly user add-on (toolboxes).

Conclusion

Siconos software provides:

- a stand-alone open platform, operational and able to solve various non smooth problems. Specifities: plug-in system, multiple interfaces, "macro language" (from *simulation* $\rightarrow$ *run*() to *computeRHS*())
- or a plugged external library that provides a computational tools
- not dedicated to a specific domain, possibly user add-on (toolboxes).
- more and more users (at least interested in ...): American Airlines, University of New Delhi, of Hong-Kong, JRL in Japan (haptic-robotic applications) ...

Conclusion

Siconos software provides:

- a stand-alone open platform, operational and able to solve various non smooth problems. Specifities: plug-in system, multiple interfaces, "macro language" (from *simulation* $\rightarrow$ *run*() to *computeRHS*())
- or a plugged external library that provides a computational tools
- not dedicated to a specific domain, possibly user add-on (toolboxes).
- more and more users (at least interested in ...): American Airlines, University of New Delhi, of Hong-Kong, JRL in Japan (haptic-robotic applications) ...

To do ...

- Documentation
- Complete Event Driven (for first order systems, with friction ...)
- Test on large systems + Optimisation
- Use feedback from users to improve reliability
- User-friendly post-treatment (VRML ...)
- . . .