Dynamical Systems implementation in Siconos.

F. Pérignon

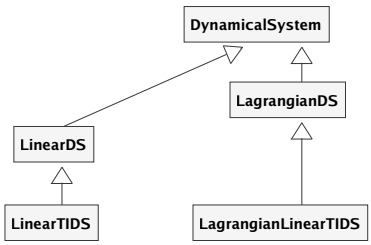
For Kernel version 1.3.0
November 7, 2006

1 Introduction

This document is only a sequel of notes and remarks on the way Dynamical Systems are implemented in Siconos.
It has to be completed, reviewed, reorganized etc etc for a future Developpers'Guide.
See also documentation in Doc/User/DynamicalSystemsInSiconos for a description of various dynamical systems types.

2 Class Diagram

There are four possible formulation for dynamical systems in Siconos, two for first order systems and two for second order Lagrangian systems. The main class is DynamicalSystem, all other derived from this one, as shown in the following diagram:



3 Construction

Each constructor must:

- initialize all the members of the class and of the top-class if it exists
- allocate memory and set value for all required inputs
- allocate memory and set value for optional input if they are given as argument (in xml for example)
- check that given data are coherent and that the system is complete (for example, in the LagrangianDS if the internal forces are given as a plug-in, their Jacobian are also required. If they are not given, this leads to an exception).

No memory allocation is made for unused members $\Rightarrow$ requires if statements in simulation. (if!=NULL ...).

3.1 DynamicalSystem

Required data:
n, x0, f, jacobianXF
Optional:
T,u

Always allocated in constructor:
x, x0, xFree, r, rhs, jacobianXF
Warning: default constructor is always private or protected and apart from the others and previous rules or remarks do not always apply to it. This for DS class and any of the derived ones.

3.2 LagrangianDS

Required data:
ndof, q0, velocity0, mass
Optional:
fInt and its Jacobian, fExt, NNL and its Jacobian.

Always allocated in constructor:
mass, q, q0, qFree, velocity, velocity0, velocityFree, p.
All other pointers to vectors/matrices are set to NULL by default.
Memory vectors are required but allocated during call to initMemory function.
Various rules:

- **fInt (NNL) given as a plug-in $\Rightarrow$ check that JacobianQ/Velocity are present (matrices or plug-in)**
- **any of the four Jacobian present $\Rightarrow$ allocate memory for block-matrix jacobianX (connectToDS function)**
-

check: end of constructor or in initialize?
computeF and JacobianF + corresponding set functions: virtual or not?

4 Specific flags or members

- **isAllocatedIn: to check inside-class memory allocation**
- **isPlugin: to check if operators are computed with plug-in or just directly set as a matrix or vector**
- **workMatrix: used to save some specific matrices in order to avoid recomputation if possible (inverse of mass ...)**

5 plug-in management

DynamicalSystem class has a member named parameterList which is a $map < string, SimpleVector* >$, ie a list of pointers to SimpleVector*, with a string as a key to identified them. For example, $parametersList["mass"]$ is a SimpleVector*, which corresponds to the last argument given in mass plug-in function.
By default, each parameters vectors must be initialized with a SimpleVector of size 1, as soon as the plug-in is declared. Moreover, to each vector corresponds a flag in isAllocatedIn map, to check if the corresponding vector has been allocated inside the class or not.
For example, in DynamicalSystem, if $isPlugin["vectorField"] == true$, then, during call to constructor

or set function, it is necessary to defined the corresponding parameter:
$parametersList[''vectorField''] = newSimpleVector(1)$
and to complete the $isAllocatedIn$ **flag:**
$isAllocatedIn[''parameter_{f}or_{v}ectorField''] = true$**.**