C12 Spheres' Collision – Dymola

Numerical simulation / Time-oriented Model

Simulator: Dymola (Dynamic Modeling Laboratory, www.dynasim.se) is a simulation environment suitable for modelling various kinds of physical objects. It uses an object-oriented approach for modelling of large, complex and heterogeneous physical systems. A graphical editor and a specific language (Modelica) allow the user to construct models composed of mechanical, electrical and hydraulic subsystems out of predefined models.

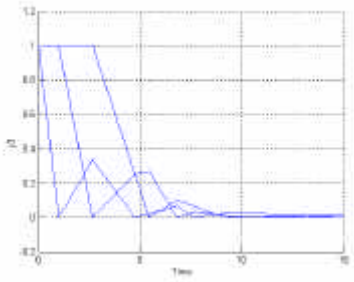
Model: With the following discussed hybrid model approach, the capability of Dymola to handle hybrid systems is tested. In this approach the implementation of the model is made in Dymola because of the simplicity of the Model, no need for a graphical based modelling is given. A hybrid model in Dymola consists of differential, algebraic and discrete equations. Furthermore it offers three variable step size algorithms for the numerical treatment of such Differential-Algebraic-Equations (DAEs).

The Dymola user manual suggests using the statement `when (cond) then` to handle events in continuous systems. To reinitialise a variable in case of an event Dymola provides the `reinit` statement. The implementation is then straightforward:

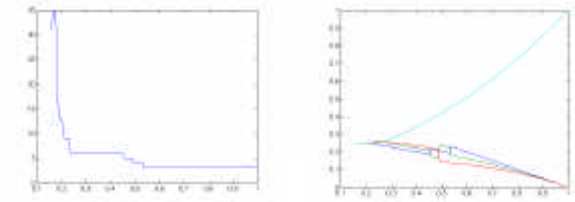
```
class Comparison12
 .. declaration of variables…

algorithm
when y1<=0
  reinit(x1p, x1p + (1+e)*m2/(m1+m2)*y1p);
  reinit(y2p, y2p + (1+e)*m1/(m1+m2)*y1p);
  …..
end
….other events …
equation
der(x1)  = x1p;
der(x1p) = 0;
… other equations…
end Comparison12
```

In Dymola, all differential, algebraic and discrete equations are treated as synchronous in time. Therefore in Dymola/Modelica the assumption is used that all equations in a model may potentially be active at the same time instant. It is easy then to construct conflicting equations. To handle this problem multiply events have to be implemented in the so-called "algorithm" section.

Task a: Simulation of the System. a1: The following figure shows the distance-time function for the parameter value e = 0.2. The DASSL - algorithm (DAE solver of Petzold) with relative accuracy 10^{-6} was used.



Task a2. For e = 1 the final velocities are v_1=v_2= v_3 = 0, v_4 = 1. The quasi-plastic case is reached for e = 0.154303. The following figures show the number of collisions and the final velocities as a function of the restitution coefficient e, which is varied from 1 to the quasi-plastic case.



Dymola itself offers a relatively poor script language, almost incapable of handling the tasks required (parameter loops, interpolation or optimisation, stochastics). Consequently all experiments were performed with help of the Dymola – MATLAB interface (for task b loops for the restitution coefficient e).

When a Dymola model is translated, an exe-File (the Dymola model) is created. Combined with the Dymola-MATLAB interface (various m-files) one can easily provide a MATLAB m-file for the experiments of all necessary tasks. For example, a simulation run of the Dymola model in MATLAB looks like

```
[s,n]=dymosim(simparams, inits, params)
```

There `simparams` is a vector of simulation parameters such as model name, starttime, endtime, algorithm, `inits` is the vector of initial values, `params` a vector of model parameters. Results are stored in a matrix s; the vector n includes all variables as strings.

Task c1: Boundary value problem -: Stochastic deviation of restitution. Instead of an iteration for e, the solution for e, e=0.587401 with end velocity v4 = v0/2, is interpolated from results from task b by means of splines (standard MATLAB feature).

Statistical deviations for e and statistical summaries for the end velocity are also done in MATLAB. For a normally distributed e the end velocity of the fourth ball is normally distributed with mean= 0.4243, dev= 0.0421. The confidence intervals are [0.4216<mean< 0.4269] and [0.0403<dev< 0.0440].

Michael Wibmer
Dept. Simulation / ARGESIM, TU Vienna
mwibmer@osiris.tuwien.ac.at