Modeling, Simulation and Control of Inverted Pendulum on a Chart Using Object Oriented Approach with Modelica

Josip Musić, Borut Zupančič
University of Ljubljana, Faculty of Electrical Engineering
Tržaška 25, 1000 Ljubljana, Slovenija
jmusic@fesb.hr

Abstract
The paper briefly presents Modelica [1], an object oriented language for modeling and simulation of complex multi domain systems. The main advantage of object oriented modeling is in reusability of components which in turn is a key for handling complexity. The application of Modelica language is demonstrated through classical control problem of inverted pendulum on a cart [6, 7]. The model was built with aid of Dymola [5], a programming environment that uses Modelica. The same model was built in Simulink and compared to Dymola-Modelica model, and conclusions were drawn.

1. Introduction

Modelica is a kind of new standard for modeling of continuous, discrete and hybrid dynamical systems. It is a result of many years effort by number of experts in the field of modeling and simulation. Modelica can be used in several ways: through textual editor or graphical interface. There are several programming environments that make use of both implementations. The most used ones are Dymola and MathModelica. In this paper Dymola from Dynasim was used [5].

2. Overview of Modelica modeling language

Modelica is general purpose, object oriented language for modeling of multi domain complex systems [1], [3, 4]. It is used for acasual modeling, as opposed to some other widely used programming environments (like Simulink) that use casual modeling. In acasual modeling state equations are written in neutral form without consideration for computational order, while in casual modeling one needs to take into account computational order [3], [5]. The difference between two also results in difference in the way the system equations are written. In casual modeling the model is described by ODE (ordinary differential equations) while in acasual modeling it is possible to write system state equations in more natural form as DAE (differential-algebraic equations). Modelica supports two kinds of modeling: high level modeling through graphical interface and detailed library component modeling by equations. Models of standard components are typically available in model libraries. Modelica comes with Modelica Standard Library which is maintained and constantly upgraded by Modelica Association. Current Modelica version 2.2.1 has the following standard libraries: Blocks, Constants, Electrical, Math, Mechanics, Media, SIunits, StateGraph, Thermal, Utilities. User can also construct new library for his research interest and seamlessly integrate it into Modelica. Modelica supports two basic types of variables: across and through variables [1], [3]. Across variables are those variables that become equal in connection points between two subsystems while through variables are those variables whose sum equals zero at any point in the system and are especially marked in Modelica language with prefix FLOW. The basic structure in Modelica is class which in order to facilitate programming and maintenance is divided into following special cases: model, connector, record, block, function, type and package. The most used one is model class. In order to handle large models and maintain reusability, careful structuring is required. Thus a model class contains: basic component, structured component, component arrays, equation and/or algorithm, connections, functions [1], [3]. For example, let's look at the declaration of model for Ideal Resistor from library Modelica.Electrical.Analog.Basic

```
model Modelica.Electrical.Analog.Basic.Resistor
extends Interfaces.OnePort;
parameter SI.Resistance R=1;
equation
  R*i = v;
end Resistor;
```

In above simple example another very important feature closely related to the reusability can be seen. It is definition and subsequent use of partial model.

```
partial model Modelica.Electrical.Analog.Interfaces.OnePort
SI.Voltage v;
SI.Current i;
PositivePin p;
NegativePin n;
equation
  v = p.v - n.v;
  0 = p.i + n.i;
  i = p.i;
end OnePort;
```

This partial model can be used to define some other electrical component like capacitor. In order to be functional, elements of some complex model need to be interconnected. Connectors are special class that defines interrelation between variables. For a simple resistor example it is done by

connector Modelica.Electrical.Analog.Interfaces.PositivePin
SI.Voltage v;
flow SI.Current i;
end PositivePin;
connector Modelica.Electrical.Analog.Interfaces.NegativePin
SI.Voltage v;
flow SI.Current i;
end NegativePin;

When connector of a model is defined one can connect it to another component.

connect(Resistor.p, Capacitor.p);
connect(Resistor.n, Capacitor.n);

2.1. Dymola from Dynasim

Dymola consists of three main modules: modeling module, simulation module and module for visualization of results. Dymola offers wide range of features some of which include: acasual modeling, reusability, hierarchical structure of models, symbolic and numerical solving of system equations, algebraic loop solving, modeling with Bond graphs, hybrid system modeling, modeling with Petri nets, open interface to other programs like Simulink, real time simulation, 3D animation of modeled system, etc. Models developed in Dymola can be seen on several levels or layers: icon, diagram, documentation and Modelica text layer. This kind of layer separation offers better insight and understanding of a model. Important feature of Dymola is Dymola-Simulink interface. The DymolaBlock in Simulink is shielded around an S-function MEX block i.e. the interface to the C-code generated by Dymola for the Modelica model. Connectors available in Simulink depend on modeling done in Dymola. We believe an effort should be made to combine Dymola and Simulink to provide an optimal solution for modeling and simulation purposes.

3. Example of Modelica and Simulink modeling and simulation: Control of inverted pendulum on a cart

As an example inverted pendulum on a cart was modeled in Dymola-Modelica. Then control scheme was implemented and same procedure was repeated in Simulink and resulting systems were compared.

3.1. Inverted Pendulum on a Cart

Figure 1 presents physical setup of the inverted pendulum on a cart. The pendulum can freely move only in X-Y plane. Once the problem has been described with initial mathematical equations and after some minor mathematical manipulations, nonlinear dynamic equations describing movement of the cart and inverted pendulum are obtained

$$(M+m)\ddot{x} + b\dot{x} + ml\ddot{\Theta}\cos\Theta - ml\dot{\Theta}^2\sin\Theta = F \quad (1)$$

$$(I+ml^2)\ddot{\Theta} + ml\ddot{x}\cos\Theta - mgl\sin\Theta = 0 \quad (2)$$

To obtain linearized model of the system which will be used for control, the system is observed at or near the working point (in our case the upright position $\Theta_0 = 0$). This yields

$$(M+m)\ddot{x} + b\dot{x} - ml\ddot{\Theta} = F \quad (3)$$

$$(I+ml^2)\ddot{\Theta} - ml\ddot{x} - mgl\Theta = 0 \quad (4)$$

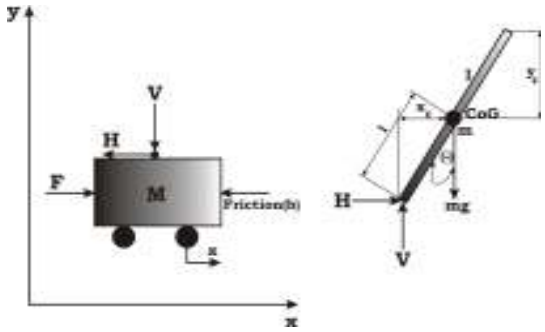

Figure 1: Physical setup of inverted pendulum on a cart

3.2. Control strategy

Swing-up phase

Widely accepted approach is based on energy pumping [6]. Basically, energy is added to the system at those times when it results in increase of kinetic energy of the pendulum. The direction of force which is to be applied on the cart depends on the sign of the product $\dot{\Theta}\cos\Theta$. The detailed explanation of this procedure and associated theoretical background can be found in [6]. In our case we used the simplest strategy: bang-bang control. The absolute value of force which is applied to the cart is constant. The direction of applied force changes depending on the above mentioned sign function. In practice one also needs to take into account the finite length of the rail on which the cart is mounted as well as the DC motor dynamics and limitations in regard to maximal motor moment/force output.

Stabilization phase

When the pendulum is near the upright position the control unit is switched to stabilization mode [7]. Stabilization of the pendulum in the upright position is achieved by means of Linear Quadratic Regulator (LQR). The theoretical background of LQR control is well known in control theory and can be found in literature [8]. The assumption made here was that all of the state variables can be measured and are available e.g. full-state feedback.

3.3. Dymola-Modelica model of inverted pendulum on a cart

Model of inverted pendulum on a cart with associated sensors and elements necessary to achieve desired control strategy is shown in Figure 2a. The model of inverted pendulum on a cart is shown in Figure 2b. Note the icon World in the model. Obviously this is not a part of physical model, but rather the requirement of Modelica that every model built up utilizing MultiBody library must have *World* icon on top level [2].

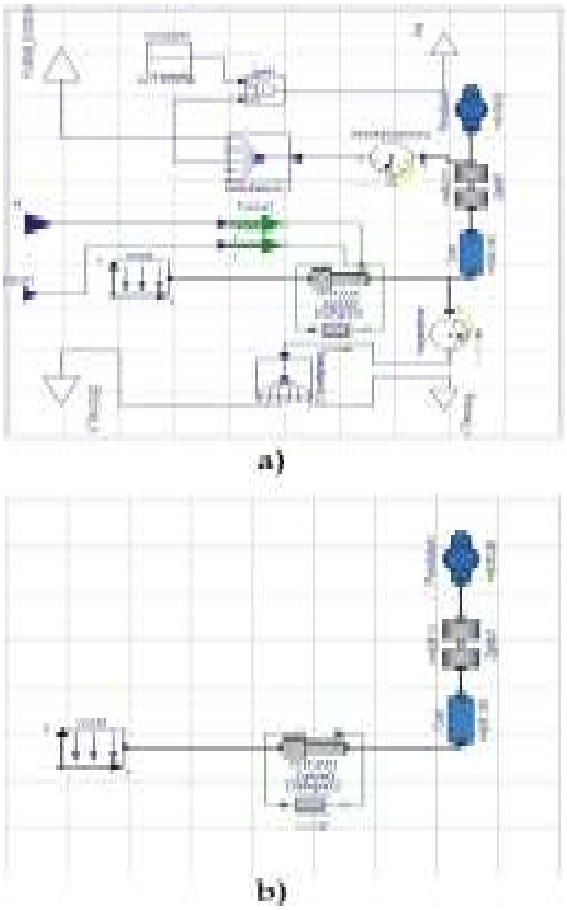

Figure 2: a) complete model of inverted pendulum on a cart; b) inverted pendulum on a cart model

The model is made up from six components

model Modelica.Mechanics.MultiBody.World
model Modelica.Mechanics.MultiBody.Parts.BodyShape
model Modelica.Mechanics.MultiBody.Parts.BodyBox
model Modelica.Mechanics.MultiBody.Joints.ActuatedPrismatic
model Modelica.Mechanics.Translational.Damper
model Modelica.Mechanics.MultiBody.Joints.Revolute

We will now in more detail (as an example) examine Translational Damper. This element was chosen because elementary concepts introduced in the previous sections can be clearly seen here.

```
model Modelica.Mechanics.Translational.Damper
extends Interfaces.Compliant;
parameter Real d(
final unit
final min=0) = 0;
SI.Velocity v_rel ;
equation
  v_rel = der(s_rel);
  f = d*v_rel;
end Damper;
```

Translational Damper extends Interfaces.Compliant partial model with constitute equation (characterized by damper constant *d* whose value can not me smaller then 0). Partial model is defined by

```
partial model
Modelica.Mechanics.Translational.Interfaces.Compliant
Flange_a flange_a;
Flange_b flange_b;
SI.Distance s_rel;
SI.Force f;
equation
  s_rel = flange_b.s - flange_a.s;
  flange_b.f = f;
  flange_a.f = -f;
end Compliant;
```

In this partial model two flanges are defined (flange_a and flange_b). It also contains equations defining relative displacement between two flanges and force and its direction. This partial model can be used for definition of some other element like spring. Now we move on, to define connectors of the damper element. Damper has two points where it is attached to the system and thus two connectors are defined.

```
connector Modelica.Mechanics.Translational.Interfaces.Flange_a
SI.Position s;
  flow SI.Force f;
end Flange_a;
connector Modelica.Mechanics.Translational.Interfaces.Flange_b
SI.Position s;
  flow SI.Force f;
end Flange_b;
```

When all of the elements have been properly defined they need to be interconnected.

```
connect(world.frame_b, Joint0.frame_a);
connect(Zglob0.frame_b, Cart.frame_a);
connect(Damper2.flange_a, Joint0.bearing);
connect(Damper2.flange_b, Zglob0.axis);
connect(Joint1.frame_b, Pendulum.frame_a);
connect(Joint11.frame_a, Cart.frame_b);
```

The next step is to implement control strategy. Detailed description of the elements and procedures used is beyond the scope of this article. We will just present the final scheme (Figure 3) where hierarchical structuring was used.

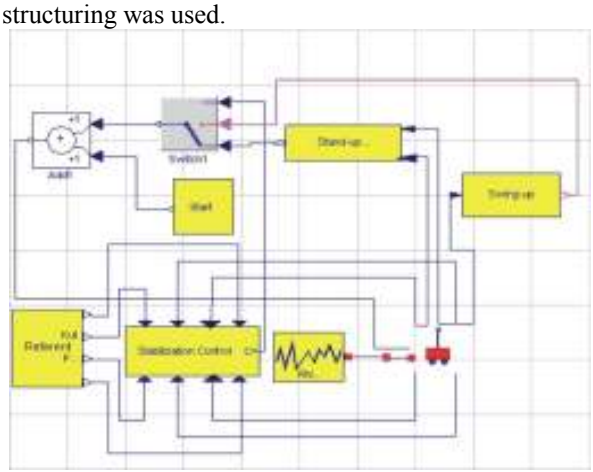
Figure 3: Complete system overview

3.4 Matlab-Simulink model of inverted pendulum on a cart

For reasons of comparison, equivalent model of inverted pendulum on a cart was developed in Simulink In order to achieve this, we used Equations (1) and (2), yielding model presented in Figure 4. One can see that Simulink model is more complex compared to Dymola-Modelica model in Figure 2b and offers no obvious connection to the physical setup. But Simulink has its

advantages in the use of Matlab and its toolboxes

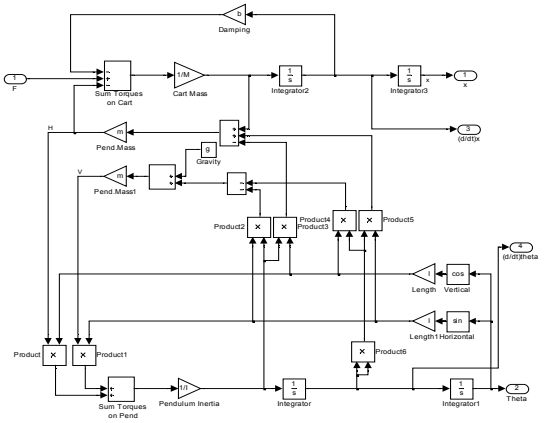

Figure 4: Simulink model

which offer great support for variety of applications. In the part of the system used to implement control for erection and stabilization of the pendulum, there is no notable difference between Dymola-Modelica model and Simulink-Matlab model.

3.6 Simulation results

Simulation results obtained with Dymola-Modelica are presented in Figure 5. Simulation results obtained by Modelica and Simulink are the same. Only difference can be observed in the angle of inverted pendulum. This difference arises due to the way Dymola-Modelica sensor used measures angle. The difference is not in actual angle itself but rather in "measured" value. This is easily taken into account and final results don't suffer because of it, which can be seen from control signal applied to the moving cart.

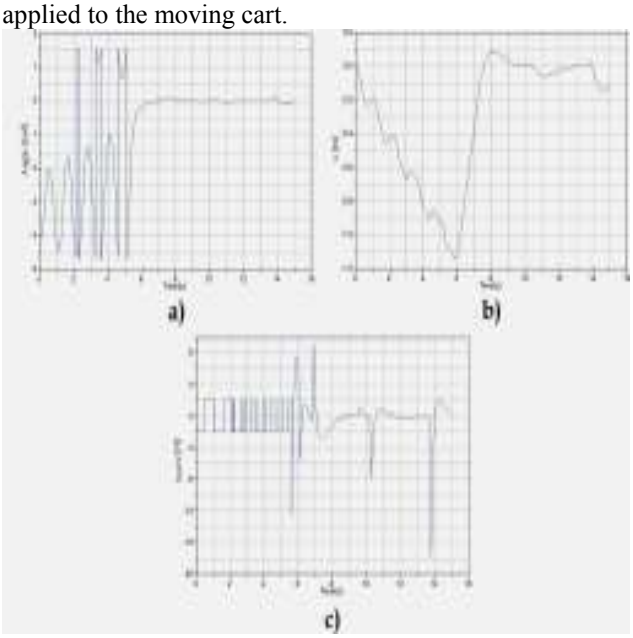

Figure 5: Dymola-Modelica simulation results
a) Inverted pendulum angle; b) Cart position (x);
c) Control Signal

4. Conclusion

The paper examines two approaches to modeling and simulation of inverted pendulum on a cart and associated control strategies. Modeling in Dymola-Modelica has proved to be easier and similar to building a real world system. Also, animation module of Dymola offers great insight into system behavior. When simulation is done all possible quantities are readily available for plotting. It is worth noting that for building actual inverted pendulum on a cart associated mathematical background was not required. Also it must be emphasized that Dymola-Modelica model was built very fast, once the user is familiar with the Dymola-Modelica programming environment. On the other hand model built in the Simulink environment is complex and cumbersome and takes much more time to develop and implement (about 4 times more) and it can be used only for that system configuration. If one wants to introduce some changes, whole new model needs to be developed. This is not the case in Dymola-Modelica due to reusable components. Comparing simulation results obtained from Dymola and Simulink it can be seen that they are the same, which implies that final result is not affected by environment used. In the end it was decided that Modelica is better choice for modeling and simulation, but the best solution is to take best of both programs. By combining Dymola-Modelica (for modeling) and Simulink-Matlab (for simulation and subsequent analysis) one gets very powerful tool that can model and simulate both complex and simple systems.

5. Literature

[1] Modelica – A Unified Object-Oriented Language For Physical system Modeling, Modelica Association, 2000

[2] M. Otter, H. Elmqvist, S.E. Mattsson, The New Modelica MultiBody Library, Modelica Association, 2003

[3] P. Fritzons, Principles of Object-Orientred Modeling and Simulation with Modelica, Wiley-IEEE Press, 2003

[4] B. Zupančič, G. Zauner, F. Breitenecker, Modeling and Simulation in Process Technology with Modelica, Proc. of the 2005 Summer Computer Simulation Conference, pp. 207-212, New Jersey, USA, 2005

[5] B. Zupančič, Dymola – an example of object orianted modeling and simulation enviroment, Lecture notes, Vienna, 2004

[6] K.J. Astrom, K. Furuta, Swinging up a pendulum by energy control, Automatica 36, pp. 287-295, 2000

[7] S. McGilvray, Self-Erecting Inverted Pendulum: Swing up and Stabiliyation control, IEEE Student paper contest, 2002

[8] Edt. W.S. Levine, The Control Handbook, CRC Press, Boca Ration, USA, 1996