# Continuous Systems Modelling Education – Causal or Acausal Approach?

**Article** · June 2008

DOI: 10.1109/ITI.2008.4588514

CITATIONS
2

READS
6,807

**3 authors:**

Borut Zupančič
University of Ljubljana
**121** PUBLICATIONS **1,048** CITATIONS
SEE PROFILE

Rihard Karba
University of Ljubljana
**155** PUBLICATIONS **921** CITATIONS
SEE PROFILE

Maja Atanasijevic-Kunc
University of Ljubljana
**71** PUBLICATIONS **230** CITATIONS
SEE PROFILE

# Continuous Systems Modelling Education – Causal or Acausal Approach?

Borut Zupančič, Rihard Karba, Maja Atanasijević-Kunc, Josip Musić
*University of Ljubljana, Faculty of Electrical Engineering, Tržaška 25, 1000 Ljubljana, SLOVENIA*
*borut.zupancic@fe.uni-lj.si*

**Abstract**. The paper deals with some important aspects of continuous systems modelling education. Namely the traditional approach is based on block oriented schemes in which causal relations play an important role. However this causality is artificially generated in order to fulfil appropriate conditions for simulation on conventional sequential computers. Fortunately new concepts which are based on object oriented approaches, physically oriented connections and algebraic manipulation enable so called acausal modelling. Both approaches have to be educated and are presented on an example. Advantages and disadvantages are discussed.

**Keywords.** Object oriented modelling, simulation, control system, education

## 1. Introduction

Each modelling and simulation course usually starts with the following traditional approach: after mass and energy balance equations the block diagram scheme where derivatives are explicitly expressed [4] is developed. However traditional general-purpose simulation tools have a lack of object-oriented properties, which disables the reuse of already build models. Due to this reason some special-purpose tools were developed (for mechanical, electrical, chemical systems, …). In modelling however combinations of systems from different areas are frequently needed particularly within automotive, aerospace and robotics applications. For modelling such systems some new concepts are needed.

When analysing the advantages and disadvantages of traditional and more advanced modelling and simulation tools the basic distinction appears from the term- causality. This term can explain the evolution which was in the past declared as the evolution from block oriented tools into object oriented tools.

## 2. Causal versus acausal modelling and simulation

Most of the general-purpose simulation software on the market such as ACSL, SIMULINK,… assume that a system can be decomposed into block diagram structures with causal interactions. Often a significant effort in terms of analysis and analytical transformations is needed to obtain a problem in this form. It requires a lot of engineering skills and manpower and it is error-prone. However in order to allow the reuse of component models, the equations should be stated in a neutral form without consideration of computational order. This is so called acausal modelling. Namely in nature real systems are acausal. We never know whether in resistor current causes voltage or voltage causes current. Causality is artificially made because physical laws must be transformed in convenient computational descriptions. It is much easier, more convenient and more natural then to use acausal modelling tools such as Dymola with Modelica [3,6]. We write balance and other equations in their natural form as a system of differential-algebraic equations. Computer algebra is then utilized to achieve an efficient simulation code similar as the equations would be converted to ODE form manually - what we actually do when using traditional tools.

## 3. Basic processing tasks in acausal oriented approaches

There are some important processing tasks in acausal oriented modelling tools. Computer algebra which transforms all model equations into a set of vertically (appropriate variable is derived from each equation) and horizontally sorted equations (well known sorting algorithm: all variables on the right hand side should be evaluated in previous rows-equations) is probably the first one [2]. The most important difference with regard to the traditional block oriented tools is also in a different way of components connections.

## 3.1. Physically oriented connections

Connections between submodels are based on variables, which define proper relations and influence between i.e. movements, angles, currents, pressures, etc. Fig. 1 shows how three fluid subsystems are connected. Three physical variables are presented in connectors:

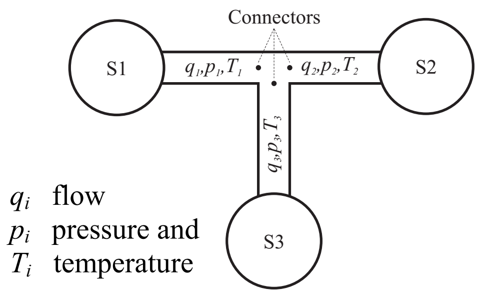

**Figure 1: Connection of three fluid subsystems**

In all systems there are two types of variables, which are defined in connectors of subsystems: variables that become equal in connection points, in our example temperature and pressure (across variables e.g. potential, temperature, pressure,…): $p_1 = p_2 = p_3$ $T_1 = T_2 = T_3$ and the variables which sum equals zero (through variables, e.g. current, momentum, force,…prefix flow in Modelica): $q_1 + q_2 + q_3 = 0.$

This concept is similar to the concept when using Bond graphs, when slightly different terminology is used (effort and flow node).

Connector is a special structure in which all variables are collected. Each connector has (after processing) a name which is composed of a submodel name and a name of a particular connector.

Example of a connector definition in Modelica

```
connector outflow
   Real p;            // Type across
   flow Real q;       // Type through
 end outflow;
```

By joining connectors the submodels are connected. During processing the modelling tool generates appropriate equations from submodel connector definitions.

## 3.2. Object- orientation

As in each OO programming principles "encapsulation", "information hiding" and "inheritance" are very important. The former means that one can use a model by composition without knowing the implementation details. Information hiding means that a model can be written with reference to connector variables, without any further assumptions about how it will actually be connected later. Inheritance is a way to form new classes (instances of which are called objects) using classes that have already been defined. These new classes take over (or inherit) attributes and behavior of the pre-existing classes, which are referred to as base classes (or ancestor classes). So an existing code can be used with little or no modification.

However we do not intend to discuss about general concepts in OO programming. From a modeller point of view, OO means that he builds a model similar to real system: he takes a pump, a pipe, a valve, … and connect them. For an efficient modelling, systems are decomposed into subsystems (components), which are modelled as submodels and then hierarchically connected into a complete model. Modelling languages enable simple reuse of already built models. It is important, that model classes can be defined directly by physical laws (energy and mass balance equations) and not necessarily with state space description $dx/dt=f(x,t)$ (as is the case in conventional simulation tools). Such concept contributes significantly to the better understanding and reusability of models.

## 3.3. Modelica – OO modelling standard

Modelica [3,6] is a modelling language which supports both high level modelling using pre-prepared complex model components and detailed modelling by equations. Graphical diagram layer and textual layer can be efficiently combined. The basic construct in Modelica is class. There are several types of classes with several restrictions: class, model, block, function, package, type etc.

## 4. Example: Inverted pendulum on a chart

The described concepts will be illustrated and some conclusions will be given with a very known and suitable laboratory set up, which is extremely efficient for modelling and control systems design courses – inverted pendulum on a chart. As it is a mechanical system, this is a nice modelling example. As it is also unstable, it is a good problem for studying more advanced control algorithms.

## 4.1. Mathematical model

Fig. 2 shows the physical setup of the inverted pendulum on a cart with parameters: mass of the chart $M$=2$kg$, mass of the pendulum $m$=0.2$kg$, half of the length of the pendulum $l$=0.25$m$, inertia of the pendulum when the rotational axis is the centre of gravity $I$=0.0017$kgm^2$ ($I$=$m$(2$l$)$^2$/12), damping coefficient of the chart $b$=0.1$Ns/m$.

The pendulum can freely move only in x-y plane. The equations for balance of forces acting on inverted pendulum in vertical and horizontal directions are as follows [5]:

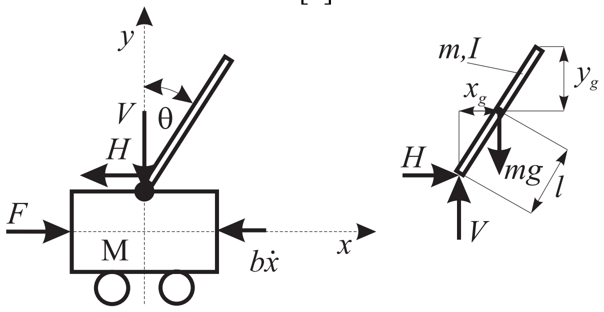

**Figure 2: Inverted pendulum on a cart**

$$m\ddot{y}_g = V - mg \tag{1}$$

$$m\ddot{x}_g = H \tag{2}$$

where $y_g = l\cos\Theta$ and $x_g = x + l\sin\Theta$. The balance of rotational motion of the pendulum around its centre of gravity is defined by

$$I\ddot{\Theta} = Vl\sin\Theta - Hl\cos\Theta \tag{3}$$

The balance of the forces acting on the cart is expressed by

$$M\ddot{x} + b\dot{x} = F - H \tag{4}$$

Combining Eqs. (1), (2), (3) and (4) nonlinear dynamic equations of the system are obtained

$$(M+m)\ddot{x} + b\dot{x} + ml\ddot{\Theta}\cos\Theta - ml\dot{\Theta}^2\sin\Theta = F \tag{5}$$

$$(I+ml^2)\ddot{\Theta} + ml\ddot{x}\cos\Theta - mgl\sin\Theta = 0 \tag{6}$$

## 4.2. Control strategy

There are two different control problems: swing-up phase and the control near upright position. So two different and coordinated strategies are needed.

The overall block scheme of the control problem is shown in Fig. 3. This is the scheme in Modelica. However there is no conceptual difference to the Matlab Simulink scheme in this highest model diagram layer.

There are many approaches for swing up phase. Widely accepted approach is based on energy pumping [1]. In our case we used a simplified version using the control force signal ±2N. The sign depends on the sign of the expression $\dot{\Theta}\cos\Theta$ (implemented in Swing-up Control block). When the pendulum is near the upright position (|$\Theta(t)$|<25°), the control unit is switched to stabilization mode [5] (block Stabilization Control). Stabilization of the pendulum in the upright position is achieved by means of Linear Quadratic Regulator (LQR, [1,5]) . In our case all state variables were measurable so we did not need the state observer which is convenient and commonly used approach. The design procedure resulted in the following state controller gains: $K = [-10 \quad -10.3 \quad 75.6 \quad 11.3]$ with which the state vector $[\, x \;\dot{x}\; \Theta\; \dot{\Theta}\,]$ is processed.

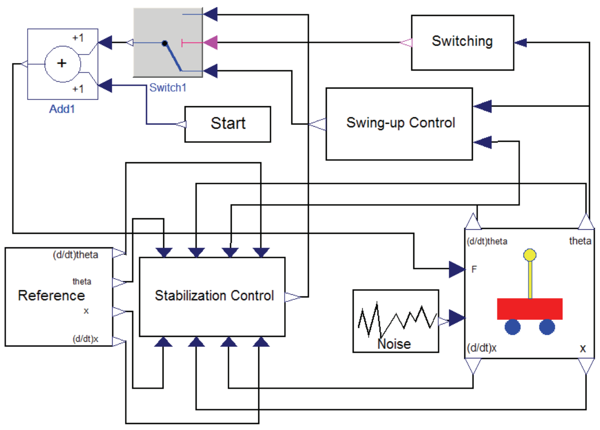

**Figure 3: Modelica diagram layer scheme**

The switching between swing-up and stabilization control is implemented in Switching block. Start block is used for an initial force impulse which drives the system from stable mode (Θ(0)=-π) which initiates the swing up procedure. There is also a block which generates a noise to the force control signal.

The overall scheme can be divided into two major parts: control system and the model of the physical system. The implementation of the control system is very similar in causal and acausal tools. However there are important differences when modelling the physical system.

## 4.3. Matlab-Simulink model

Using Simulink we assume that a model can be decomposed into block diagram structures with causal interactions. This means that the models are expressed as an interconnection of submodels on explicit state-space form (ODE – ordinary differential equation form)

$$\dot{x} = f(x,u,t)$$
$$y = g(x,u,t) \tag{7}$$

Using equations (5) and (6) and well known indirect approach in which we express derivatives which have to be solved with integration, a Simulink scheme presented in Fig. 4 is obtained.

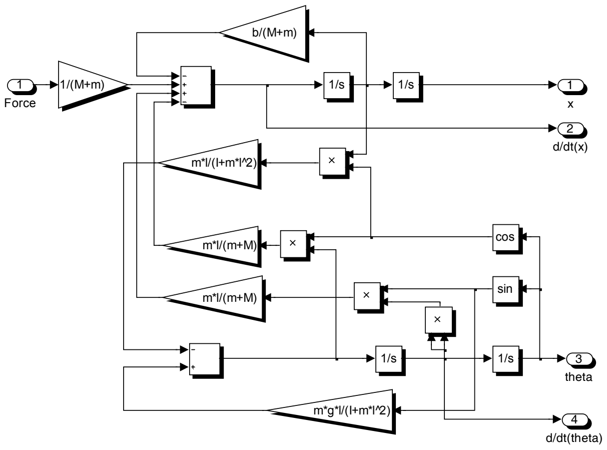

**Figure 4: Model of the inverted pendulum on a chart in Matlab Simulink**

## 4.4. Modelica model

In MODELICA it is possible to write balance and other equations in their natural form as a system of differential-algebraic equations

$$0 = f(\dot{x}, x, y, u, t) \tag{8}$$

Fig. 5 is the graphical presentation of the inverted pendulum on a chart, so called diagram layer in Modelica.

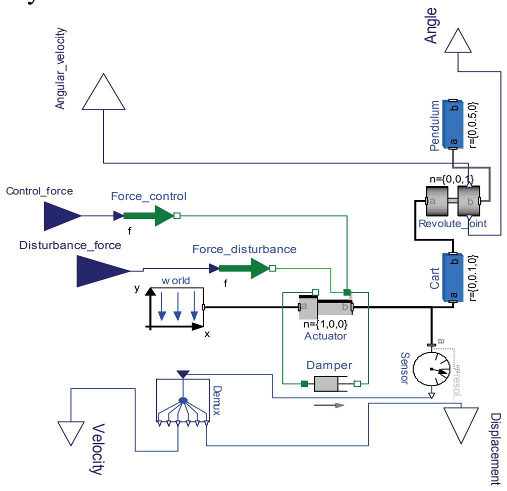

**Figure 5: Inverted pendulum on a cart model**

The important advantage of Modelica is that behind the diagram layer there are always very transparent textual models – Modelica text layer. Physical (mechanical) parts can be modelled very quickly and efficiently using mostly appropriate objects from Modelica Mechanics MultyBody library and Modelica Mechanics Translation library – see Fig. 5. The model is composed of six components. As the standard Modelica library is hierarchically built, the nomenclature of the basic model components is the following:

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

- World –represents a global coordinate system fixed in ground.
- Joints.ActuatedPrismatic Actuated prismatic joint (one translational degree-of-freedom) – used for the generation of control force.
- Translational.Damper Linear one dimensional translational damper.
- Parts.BodyBox- Rigid body with box shape. Mass and animation properties are computed from box data and density – used for the chart and the pendulum.
- Joints.Revolute Revolute joint (one rotational degree-of-freedom). Actually this component was slightly changed for our use. Namely we added two connectors which directly generate angle and angular velocity variables. Otherwise we would have to use a complicated library component which measures absolute kinematic quantities of appropriate frame connector. We even had some problems with this component as trigonometric functions did not work correctly.

The chart was defined as a rigid body with box shape with edge dimensions 0.125*m*, 0.1m, 0.1m and with density 1.6*g/cm*$^3$. The same component was used for the pendulum. The dimensions were 0.02*m*, 0.02*m,* 0.5*m,* and the density 1*g/cm*$^3$*.*

Elementary concepts of OO modelling in Modelica can be illustrated with the textual Modelica code for Translational damper:

```
model Damper "Linear 1D translational damper"
  extends Interfaces.Compliant;
  parameter Real d(
    final unit="N/ (m/s)",
    final min=0) = 0 "damping constant [N/ (m/s)]";
  SI.Velocity v_rel "relative velocity between flange_a and
flange_b";
equation
  v_rel = der(s_rel);
  f = d*v_rel;
end Damper;
```

The model consists of declaration and equation parts. Model Damper extends Interfaces.Compliant partial model (which declares some basic equations common to several components of this type) with constitute equation which defines the relative velocity between two damper flanges equal to the derivative of relative distance between same flanges and force which is equal to the product of relative velocity and user defined damper constant d (whose value can not be smaller then 0). The code also defines unit for the relative velocity to be from SI system (i.e. m/s).

Similar to the physical system the model Damper has two points where it is attached (interfaced) to the system and thus two connectors are defined entitled Flange_a and Flange_b:

```
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;
```

Two variables are defined in each connector: displacement s and force f. The first one is of the type across (automatic generation of the equation Flange_a.s=Flange_b.s) and the second one is of the type through (automatic generation of equation Flange_a.f+Flange_b.f=0).

When all the elements are properly defined they need to be interconnected. If the components are graphically connected these connect statements are automatically generated. Some connect statements are:

```
  connect(World.frame_b, Prismatic.frame_a);
  connect(Prismatic.frame_b, Cart.frame_a);
  connect(Cart.frame_b, Revolute.frame_a);
  connect(Revolute.frame_b, Pendulum.frame_a);
```

The complete model of the described control problem is shown as the diagram layer in Fig. 3.

## 4.5. Simulation results

The paper deals mainly with the evaluation of modelling and simulation tools in education and not with the control system design. However due to the completeness we present also some basic simulation results – see Fig. 7. Simulation results obtained with Modelica and Simulink were almost the same. This is not self understanding because the simulation of the overall scheme is numerically problematic, especially due to many discontinuities and an algebraic loop. So we have to select carefully the appropriate integration method.

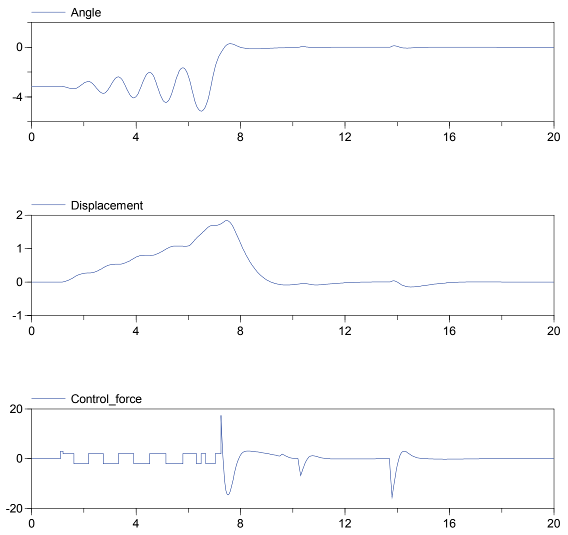

**Figure 7: Simulation results: pendulum angle, chart displacement and control force**

We can notice that the pendulum is in the stable position $\Theta(0)=-180^0$ at the beginning. After a small disturbance signal, the swing-up procedure is started and the amplitude of oscillations increases. At $t$=7.25 the stabilization phase is started and after that the state controller forces all states to zero reference values. The robustness of the controller is tested with two disturbances appearances: at 10.1s 7N for 0.1s and at 13.7s 16N for 0.1s. Therefore the control signal is changed however the angle remains almost unchanged.

Mechanical system can be animated in Dymola. Animation is important because it increases the motivation of students.

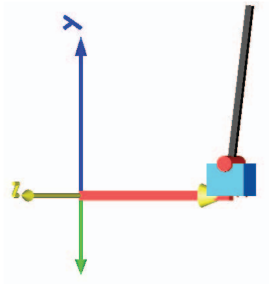

**Figure 8: Animation of the inverted pendulum**

## 5. Advantages and disadvantages of both solutions

There is no important difference when modelling control schemes (see Fig. 3). However there is a big difference when modelling physical systems. The structure of the block diagram model in SIMULINK does not reflect the topology of the physical system. In the SIMULINK model it is not possible to observe the chart, damper, revolute joint or pendulum itself. There is a fundamental limitation of block diagram modelling. The blocks have an unidirectional data flow from inputs to outputs. This is the reason why objects like chart, damper, revolute joint or pendulum cannot be dealt with directly. It is also the reason why some parameters of the mentioned components appear in mixed expressions in the SIMULINK model. And the developed structure can be more or less used only for the configuration it was developed – with force control signal as the input and four state variables which are also outputs. The effort to produce the Simulink simulation model is incomparable with the Modelica model. One needs much more time and much more modelling knowledge for Simulink based approach. And finally there are problems which are observed only by skilled modellers. Fig. 4 shows that the used modelling has led to an algebraic loop. This usually means not a very constructive modelling. We have two choices: to leave Simulink to deal with the algebraic loop, what is numerically sometimes very risky or to algebraically eliminate the loop (what Dymola does automatically). First choice means more transparent model but with bad numerical features. The second solution leads to much more efficient numerical simulation; however the model becomes less transparent.

There is no useful textual layer behind the diagram layer which is the important disadvantage of the Simulink model. So it is problematic to deal with complex and sophisticated models. The documentation is very difficult and inappropriate.

In Modelica models (Figs. 3, 5), the connections between physical system and computer model are very transparent. All the components are fully reusable in other configurations. The combination of textual and diagram programming is efficient. So Modelica can be used for very complex problems and is also superior for model documentation.

We can conclude that MATLAB-Simulink is more appropriate for the design and implementation of control schemes as it has more facilities especially in conjunction with some toolboxes. – e.g. Control System Toolbox, Optimization Toolbox. In our example the state controller was developed. We studied two proposed approaches - pole placement approach with Ackermann formula Matlab support but also LQR optimization approach with results in our example. Modelica is superior when modelling physical systems when the concept of algebraic manipulation and specially defined connectors bring many advantages. This approach is also very useful in education. We propose to start modelling courses with OO acausal approach especially when one can deal with implemented libraries which do not demand a deep theoretical background. And it can motivate students much more than the low level Simulink approach.

## 6. Conclusions

Modelling and simulation is extremely important subject in all control engineering courses. Namely it is much easier, cheaper and safe to experiment in simulation environment as on real processes. However traditional block oriented causal approaches should be extended by modern object oriented acausal approaches and tools. In this paper advantages and disadvantages were discussed and shown on an example. We can conclude that both approaches are needed especially in education.

## 7. References

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

[2] Cellier F.E. Continuous System Modeling, Springer - Verlag, New York, 1991.

[3] Fritzson P. Principles of Object Oriented Modeling and Simulation with Modelica 2.1. IEEE Press, John Wiley&Sons, Inc., Publication, USA, 2004.

[4] Matko D., Karba R., Zupančič B. Simulation and Modelling of Continuous Systems: A Case Study Approach, Prentice Hall Int., New York, 1992.

[5] McGilvray S. Self-Erecting Inverted Pendulum: Swing up and Stabilization control. 2002; IEEE Student paper contest

[6] Modelica Association, "Modelica- A Unified Object-Oriented Language for Physical Systems Modeling": Language Specification Version 2.1, 2003.

[7] Zupančič B., Karba R. Control engineering : new trends with OO modelling approach. Proceedings of the first Asia International Conference on Modelling & Simulation, AMS 2007, 2007, pp. 595-600.