Modeling and Simulation in Process Technology with Modelica

Journal:	*Summer Computer Simulation Conference*
Manuscript ID:	SCSC-05-09-009.R1
Track:	M&S Methodology & Tools
Date Submitted by the Author:	31-May-2005
Complete List of Authors:	Zupanèiè, Borut; University of Ljubljana, Faculty of Electrical Engineering, Laboratory for Modeling, Simulation and Control Zauner, Guenter; Vienna University of Technology, Institute for Analysis and Scientific Computing Breitenecker, Felix; Vienna University of Technology, Institute for Analysis and Scientific Computing
Keywords:	object oriented modeling and simulation, Modelica libraries, standardization, process industry

Modeling and Simulation in Process Technology with Modelica

Borut Zupančič
University of Ljubljana
Faculty of Electrical Engineering
Tržaška 25, 1000 Ljubljana, Slovenia
borut.zupancic@fe.uni-lj.si

Günther Zauner, Felix Breitenecker
Vienna University of Technology
Institut for Analysis and Scientific Computing
Wiedner Hptstr. 8-10
1040 Wien, Austria
guenther_zauner@gmx.at

Keywords: process industry, object oriented modeling and simulation, Modelica libraries, standardization

Abstract
Many modeling and simulation tools have a lack of object orientation. So it is not possible to build true reusable components. Modelica is on the other hand a nice example of OO modeling language which can become a kind of international standard. In Modelica, a library which enables an efficient modeling in process technology, was implemented. All basic components of hydraulic systems such as the tank, pump, valve, … were included. The library was tested on a control systems design problem. Namely a PI controller and a cascade controller were designed with an optimization in Matlab. The example showed that the combination of Dymola- Modelica (for modeling) and Matlab-Simulink environments (for simulation and optimization) can be a very efficient tool for control system design.

INTRODUCTION

Standardization of languages for modeling and simulation was always very important in the history. However the last standard that was really accepted was CSSL standard from 1967 [1]. Nowadays perhaps the most promising activities are in conjunction with the so called Modelica activities [4] (www.modelica.org). After an initiative of the Federation of European Simulation Societies EUROSIM and The Society for Computer Simulation SCS in the middle of nineties a new language Modelica [3,4], which gives a hope to become a kind of international standard for model exchange, was defined.

A lack of object-oriented properties, which disables the reuse of already build models, is another disadvantage of many modern modeling and simulation tools. Due to this reason some special-purpose tools were developed (for mechanical, electrical, chemical systems, …). In modeling however combinations of systems from different areas are frequently needed (e.g. mechanical, electrical, hydraulic as well as control systems in mechatronics, particularly within automotive, aerospace and robotics applications), and Modelica [3,4] (with appropriate working environment e.g. Dymola [1,2]) represents a tool that efficiently solves this problem in an object-oriented manner.

BLOCK ORIENTED CONVENTIONAL SIMULATORS VERSUS OBJECT ORIENTED MODELLING AND SIMULATION ENVIRONMENTS

In order to allow reuse of component models, the equations should be stated in a neutral form without consideration of computational order. This is so called acausal modeling. However 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. This means that the models are expressed as an interconnection of submodels on explicit state-space form (ODE – ordinary differential equation form)

$$\dot{x} = f(x,u,t) \qquad (1)$$
$$y = g(x,u,t)$$

where u is an input, y is an output and x is a state. It is rare that a natural decomposition into subsystems leads to such a model. 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.

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

$$0 = f(\dot{x},x,y,u,t) \qquad (2)$$

where x is the vector of unknowns that appears differentiated in the equation and y is the vector of unknowns that do not appear differentiated. Computer

algebra is utilized to achieve as efficient simulation code as possible, similar as if the model would be converted to ODE form manually.

OBJECT ORIENTATION

We shall not talk about general concepts in OO programming where well known terms as encapsulation, data abstraction, inheritance, dynamic binding, … are used. From a modeler point of view, OO means that one can build a model similar to real system: to take a pump, a pipe, a valve, … and to connects them. For an efficient modeling, modeled systems are decomposed into subsystems (components), which are modeled as submodels and then hierarchically connected into a complete model. Modeling languages enable simple reuse of already build models. To reuse a certain model in other models it should be defined as a class. Model classes can be defined by physical laws (energy and mass balance equations and not necessarily with state space description (Eq. 1**).** This contributes to a better understanding and reusability of models.

PHYSICALLY ORIENTED CONNECTIONS

The appropriate complexity of the implementation of the connections between model building blocks is probably the most important property of OO M&S tools. Connections between submodels are based on variables, which define proper relations and influences between movements, angles, currents, pressures, etc. It is similar as when real systems are built. Fig. 1 shows how three hydraulic subsystems are connected. Three physical variables are presented in connections (connectors in Modelica): q_i (hydraulic flow), p_i (hydraulic pressure), T_i (temperature)

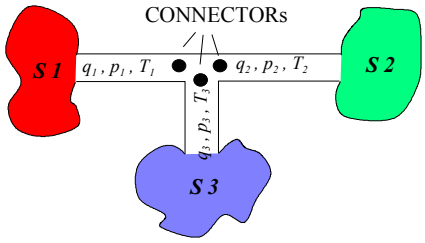


Figure 1. Connection of three hydraulic subsystems

There are two types of variables, which are defined in CONNECTORSs of subsystems: variables that become equal in connection points, in our example temperature and pressure (ACROSS variables e.g. potential, temperature, pressure,…):

$$p_1 = p_2 = p_3 \qquad T_1 = T_2 = T_3 \tag{3}$$

and variables which sum equals zero (THROUGH variables, e.g. current, momentum, force,…prefix FLOW in Modelica):

$$q_1 + q_2 + q_3 = 0 \tag{4}$$

CONNECTOR is a special structure in which all the variables are collected. Each CONNECTOR has a name, which is composed of a submodel name and a name of a particular connector . Connections in traditional block diagram simulation languages can be treated as a subset of connections introduced by conectors. Namely they possess only variables of the type ACROSS, which become equal in junction points.

IMPORTANT FEATURES OF OO M&S ENVIRONMENTS

Some important features of modern OO M&S environments are (e.g. Dymola with Modelica):

- Modeling of various kinds of complex physical systems with object oriented approach.
- General-purpose tools, equivalently usable for modeling of mechanical, electrical, chemical, thermo dynamical and other systems.
- Possibilities to reuse already built models.
- Acausal model building.
- Hierarchical structure of models.
- Description of processes through physical laws (differential equations) irrespective to the type and purpose of a model.
- Easy and efficient way for submodels connections through connectors (more general then input-output connections known in block oriented simulation tools).
- Symbolical and numerical solving of systems of equations - algebraic formula manipulation.

CONTROL DESIGN IN PROCESS TECHNOLOGY

Process systems are dynamical systems dealing with physical quantities like level, flow, temperature, pressure, ph. Appropriate control strategies in such systems are very important. As all modern and sophisticated control methods are model based, the appropriate M&S environment is very important. In the past we used more or less only Matlab-Simulink environment. This package is extremely efficient for the design of control schemes. However due to the lack of object orientation the modelling of the process to be controlled is inefficient. So we started with Dymola-Modelica environment which has all previously described features. We also examined the combination of Dymola-Modelica and Matlab-Simulink. Modelica was used to model a hydraulic process. The whole process was then used in Simulink as a Dymola block . So the complete Matlab environment can be used for control system design (e.g. Control toolbox, Optimization toolbox, …)

PROCESS TECHNOLOGY LIBRARY: AN OO IMPLEMENTATION IN DYMOLA-MODELICA ENVIRONMENT

Dymola 5.3 includes several Modelica libraries: electrical systems, mechanical systems, control systems,…

So our process technology library (PTL) also called hydraulic library supplements the basic configuration.

The building blocks of the PTL can be divided into four groups:

1st group	Components, which define output pressure in their equations	• Reservoir • Pump • Controlled pump • Reference pressure
2nd group	Components, that are based on a pressure difference define a flow	• Valve • Controlled valve • Flow element
3rd group	Sources	• Constant volume flow • Controlled volume flow
4th group	Interfaces between hydraulic and control signals	• Linear signals transducer • Flow measurement

Table 1. Components in the process technology library

Elements of the first and the second group must be alternatively used when building a model. So an element of the second group which is used behind an element of the first group, determines the flow. The reservoir and the pump can be directly connected, but then a flow element should be used in order to define the flow.

Description of the PTL

The library was implemented as a new subpackage HydraulicSystems with eleven components (see Table 1). This package includes also a subpackage HydInterfaces – a subpackage of all needed connectors. All components are defined with icon layer, diagram layer and Modelica text layer. So users can build models graphically but a textual mode is also available in order to introduce more specific model properties.

Connectors

Four different connectors were implemented. The first connector - connector Hyd_Input is used to define a hydraulic input into a component. It is defined by the following Modelica program:

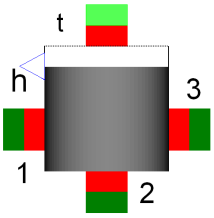
```
connector Hyd_Input
  "Layout of a port where liquid flows into an element"
    Modelica.SIunits.Pressure p "pressure at port";
    flow Modelica.SIunits.VolumeFlowRate q
                              "flow rate through the port";
 end Hyd_Input;
```

The graphical appearance information is omitted in the above listing. The pressure p is defined as an across variable and flow q as a through (flow) variable. Both quantities also inherit the properties defined in the package Modelica.SIunits providing predefined types based on international standard ISO 31-1992. The described connector can be used to model the inflow to the pump, valve, reservoir and some other components. The definition of the connector Hyd_Output, which defines a hydraulic output differs only in the graphical part, as the color is different.

The third connector HeightOutput enables the "measurement" of a signal. In the described library it was used in the reservoir for level measurement, which can be used for control purposes. The last connector InPortControl is used in externally controlled components (flow source, pump, valve). Through this signal external sources (e.g. controller) can influence the activity of appropriate components.

PTL components

model tank



This is the model of a reservoir with the inflow connector at the top and three inflow/outflow connectors (side and bottom). Besides there is also the connector for level "measurement". This is the program in Modelica for the reservoir:

```
model tank "with input and output elements and a height output"
  parameter Real area = 1 "cross-section in m^2";
  parameter Real hr =  0.5 "height of the tank/reservoir in meter";
  parameter Real h1 =  0 "height of the first input/output-element";
  parameter Real h3 =  0 "height of the third input/output-element";
  parameter Real hc1 = 0 "height difference of the connected flow
                          element in meter";
  parameter Real hc2 = 0;
  parameter Real hc3 = 0;
  parameter Real beginninglevel=0 "initial level";
  constant Real ro=1000 "density of the water at 4°C";
  constant Real g=9.81 "earth acceleration";
  Real level(start=beginninglevel);

  HydInterfaces.HeightOutput HeightOutput1;
  HydInterfaces.Hyd_Input Hyd_Input1;
  HydInterfaces.Hyd_Output IO1;
  HydInterfaces.Hyd_Output IO3;
  HydInterfaces.Hyd_Output IO2;
 equation
  der(level)=if level>hr then
  (if  (Hyd_Input1.q + IO1.q + IO2.q + IO3.q)>0 then
  0 else (Hyd_Input1.q+IO1.q+IO2.q+IO3.q)/area)
  else(Hyd_Input1.q+IO1.q +IO2.q+IO3.q)/area;
  HeightOutput1.signal[1]=level;
  Hyd_Input1.p=0;
  IO1.p=if (level-h1)<0 then 0 else ro*g*(level-h1+hc1);
  IO2.p=if level<0 then 0 else ro*g*(level+hc2);
  IO3.p=if (level-h3)<0 then 0 else ro*g*(level-h3+hc3);
end tank;
```

The program consists of three parts: declarations of model class, declarations of connectors and equations.

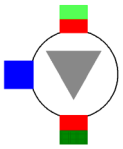
model Pump

The pump introduces the appropriate pressure difference in a pipe. The important parameter is the maximal height

to which the pump can pump. The pump always pumps in the direction of the arrow. It is necessary to use an element, which defines that flow (valve, flowelement) behind the pump. This is the program in Modelica for the pump:

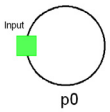
```
model Pump "model of the normal pump"
  parameter Real hp=1 "maximal height that can be pumped";
  parameter Real hc=0  "height difference between the element
                        before and behind the pump";
  parameter Real k=1 "+1 or -1, depending on
                      the pumping direction, up or down";
  constant Real g=9.81 "earth acceleration";
  constant Real ro= 1000 "density of water at 4°C";
  HydInterfaces.Hyd_Input Hyd_Input1;
  HydInterfaces.Hyd_Output Hyd_Output1;
equation
  Hyd_Output1.p=if Hyd_Input1.p>0 then
                ro*g*(hp-k*hc)+Hyd_Input1.p else 0;
  Hyd_Output1.q=-Hyd_Input1.q;
end Pump;
```

model CPump



The only difference in comparison with the pump is the possibility, that the maximal height to which the pump can pump can be settled with a scalar input signal. So it is possible to control a flow or indirectly a level in a tank.

model p0



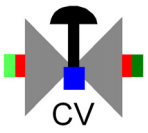
As in electrical circuits when points with zero potential must be defined, there is also a need to define air pressure in open hydraulic systems. This is done with the drain element. For simplicity this pressure is set to zero as a flow is always a result of pressure differences.

model valve

Valve defines the volume flow which depends on the square root of the pressure difference between connecting points. The basic valve parameter is the cross section area. Additional parameter is the opening (value between 0 and 1). The program in Modelica for the valve is the following:

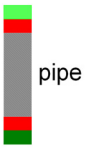
```
model Valve
  parameter Real sv=1 "degree of opening of the valve";
  parameter Real kv=0.000001 "cross-section in m^2";
  constant Real g=9.81 "earth acceleration";
  constant Real ro=1000 "density of water with 4°C";
  Real p;   // local parameter - the pressure difference
  Real qv;  // volume flow parameter
  HydInterfaces.Hyd_Input Hyd_Input1;
  HydInterfaces.Hyd_Output Hyd_Output1;
equation
  p=Hyd_Input1.p-Hyd_Output1.p;
  qv=kv*sv*sqrt(abs(2*p)/ro);
  -Hyd_Output1.q=if p<0 then -qv else qv;
  Hyd_Input1.q=-Hyd_Output1.q;
end Valve;
```

model CValve



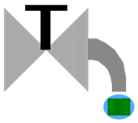
The only difference in comparison with the valve is the possibility, that the opening of the valve can be settled with a scalar input signal in the range between 0 and 1. So it is possible to control a flow or a level in a tank.

model flowelement



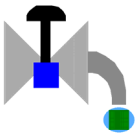
The flowelement is a connecting pipe, which is used to calculate the volume flow between two connected elements with pressure as input/output variable (i.e. connection between two tanks, after the pump, …). The flow depends on the pressure difference between the two connected elements and the cross-section area of the element.

model Source



This component is a model of the source with constant volume flow. This flow is independent of the pressure at the connector. The source can be connected with a tank, a valve, a controlled valve or a flow element.

model CSource

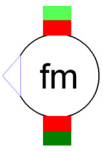


This element is a model of the source with controlled outflow. The external control variable is connected to the appropriate connector and defines the volume flow.

model Conv

This is a model of the linear signals transducer. Its input signal is linearly transformed to the output signal.

model flowmeter



This is a model of the ideal flowmeter. It is used for a pipe flow determination.

EXAMPLE: OPTIMIZATION OF A HYDRAULIC CONTROL SYSTEM

Optimization of control in industrial process systems is very significant. In this sense model based methods are very important. Our example will show how Dymola-Modelica with the developed process technology library will be efficiently used for the modeling part and Matlab-Simulink for the control design part.

The process is a three tank laboratory set-up which is shown in Fig. 2.

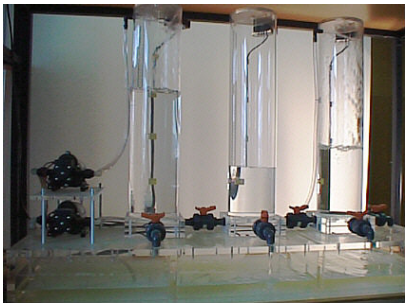


Figure 2. Laboratory three tank system

The process was modeled with the developed Modelica PTL. The appropriate Modelica model is shown in Fig. 3.

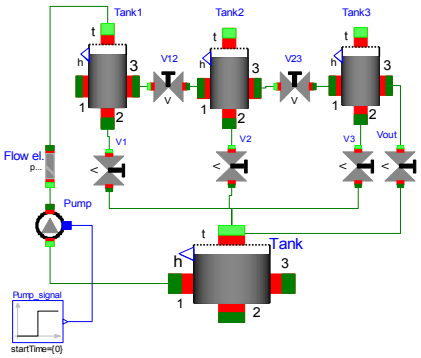


Figure 3. Scheme of the hydraulic system in Modelica

The modeling data were the following: Tank 1, Tank 2 and Tank 3 are the same with the cross section area 0.0154m^2, maximal level is 0.63m and initial levels are 0. The main reservoir Tank has the cross section area 1m^2, initial level is 0.2m. The valves V12, V23 and Vout are fully opened with the valve constants 0.0001m^2. The valves V1 and V3 are shut, the valve V2 has the valve constant 0.0001m^2 with the opening 0.1 (10%). The flow element has the cross section area 0.00002 m^2. The controlled pump has the maximal pump level 10m (control signal 0-10) and it has to pump to the level 1.37m.

In the first open loop experiment the pump was excited with the constant signal 4.5m. Fig. 4 shows the appropriate level signals. The highest level is reached in the Tank 1 and the lowest level in the Tank 3. The responses need app. 500 s to reach steady states.

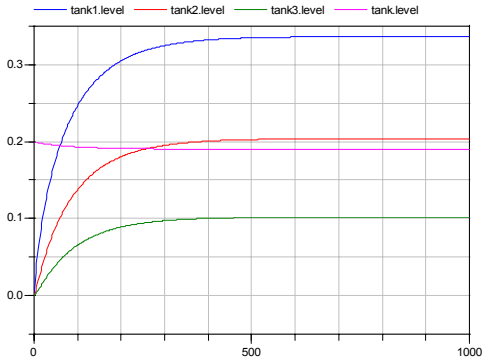


Figure 4. Open loop responses

To confirm the efficiency of the combination of the Matlab-Simulink and Dymola-Modelica environments we developed the following control system: with controlled pump the level in the Tank 3 must be controlled to the level 0.1m. At time t=0 the reference step change 0.1 appeared and at time t=200s there was a disturbance: the valve V1 was opened for 20%.

A PI controller and a cascade controller with Matlab Optimization Toolbox (function fminsearc for multidimensional unconstrained nonlinear minimization (Nelder-Mead)) were developed. The objective function was

$$Of = \int_0^{400} t\,|e(t)|\,dt \qquad (5)$$

where *e(t)* is the difference between the reference (desired) and actual level in Tank 3.

Fig. 5 shows the model, which was prepared in Dymola-Modelica for the use in Matlab-Simulink as a Dymola block. The connectors (input and outputs) between Dymola-Modelica and Matlab-Simulink were defined.

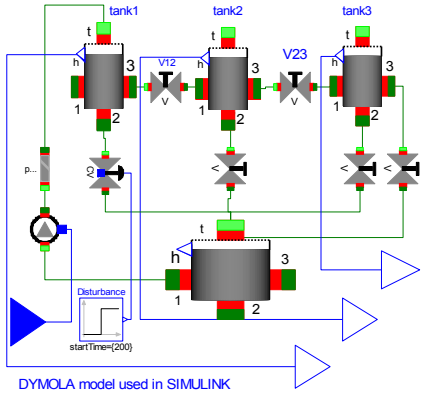


Figure 5. Dymola-Modelica model for Matlab-Simulink

Fig. 6 depicts the Simulink model for the cascade control system which includes also the Dymola model block.

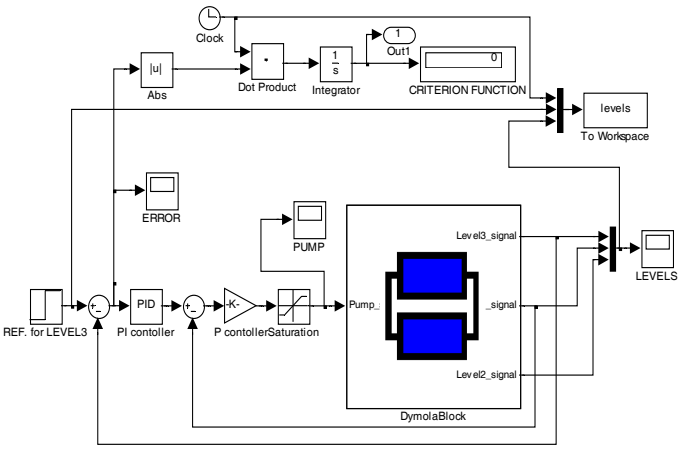


Figure 6. Simulink model which includes Dymola block

The first closed loop study was however made with a single loop PI controller, which was optimized for the reference step change in the interval 0-200s. The optimization calculated the following parameters for the proportional and integral gains: K_p= 105.6 and K_i=1.06. Fig. 7 presents all three levels as results to the reference change and the disturbance.

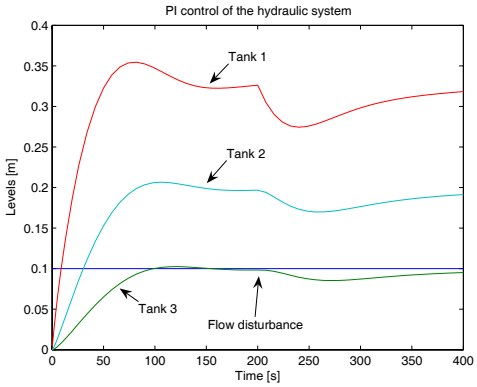


Figure 7. Tank levels using PI controller

We can notice a very good performance for the reference signal (time interval 0-200s) but the disturbance elimination is very slow (time interval 200-400s). Such results were expected as the controller was optimized for the reference step change.

In the next study we optimized the cascade controller with the main (PI) and auxiliary (P) controllers. The optimization was performed on the interval 0-400s with the presence of reference and disturbance signals. The following parameters were calculated: for the main PI controller- K_{p1}=85.6, K_{i1}=0.92, for the auxiliary P controller- K_{p2}=1.24. The simulation results are presented in Fig. 8.

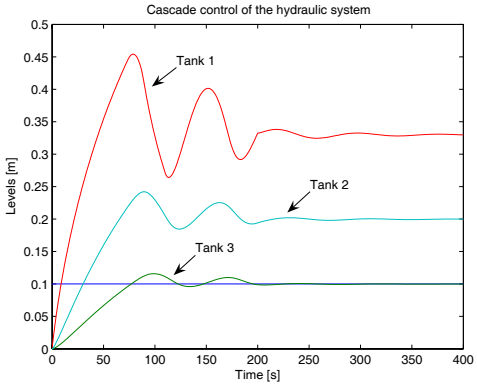


Figure 8. Tank levels using cascade controller

As expected the cascade control is very efficient in the disturbances elimination. The reference response shows a small overshoot but the flow disturbance influence to the level in Tank 3 is so small that it can hardly be observed from Fig. 8.

CONCLUSIONS

Traditional simulation tools (e.g. Matlab-Simulink, ACSL,…) are not object oriented so it is very difficult or even impossible to built fully reusable models or components. Modelica language on the other hand enables true object oriented support and as equations are algebraically preprocessed during model translation, this means also a strong modeling support as there is no need to define a model in a state space form.

Dealing with the design of control systems in process industry a Modelica library for modeling and simulation in process technology was developed. This is a library with eleven components, very suitable for control system design but also for modeling and simulation education. Namely it is possible to develop quite complex models without profound understanding and knowledge from the area of modeling and simulation.

Our experiments also confirm the efficiency of the possibility that Dymola-Modelica models can be included in Matlab-Simulink environment. All advantages of both environments can be used: Dymola-Modelica for efficient object oriented modeling and Matlab-Simulink for complex experimentations. In our example optimization was used to calculate the parameters of the controllers.

REFERENCES

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

[2] *Dymola, Dynamic Modeling Laboratory*. 2004. Users Manual, Ver. 5.3, Dynasim AB, Lund, Sweden.

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

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

[5] Strauss, J.C. 1967. "The SCi continuous system simulation language". *Simulation*, no.9, 281-303.

BIOGRAPHY

B. Zupančič

Ph.D. in electrical engineering from University of Ljubljana, Slovenia, full professor from 2000, major research interests: modelling, simulation and control, the president of the Slovene Society for Modelling and Simulation 1994-2002, member of the EUROSIM Board from 1995, vice Dean at the Faculty of Electrical Engineering 1999-2003, at present head of the Laboratory for Modelling, Simulation and Control, president of EUROSIM (Federation of European Simulation Societies) author of 175 conference papers and 30 papers in scientific journals, co-author of one international book (published by Elsevier) from the area of modelling and simulation.