SIMULATION MODELLING PRACTICE AND THEORY
Volume 33, April 2013
ISSN 1569-190X
International Journal of the Federation of European Simulation Societies
ELSEVIER
SIMULATION MODELLING PRACTICE AND THEORY
Special Issue
EUROSIM 2010
Guest Editors: Mikuláš Alexík, Miroslav Šnorek and Miroslav Čepek
Available online at www.sciencedirect.com
SciVerse ScienceDirect

Simulation Modelling Practice and Theory 33 (2013) 45–67

Contents lists available at SciVerse ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Computer-aided physical multi-domain modelling: Some experiences from education and industrial applications

Borut Zupančič *, Anton Sodja

University of Ljubljana, Faculty of Electrical Engineering, 1000 Ljubljana, Tržaška 25, Slovenia

A R T I C L E   I N F O

Article history:
Available online 17 April 2012

Keywords:
Education
Object oriented modelling
Acausal modelling
Physical modelling
Multi-domain modelling
Thermal processes

A B S T R A C T

For the purposes of this paper, computer-aided physical modelling means a type of modelling in which a computer-aided approach is used, with the basic aim being to maintain the physical structure of a real system or its topology as much as possible in the model. Bond graphs represent a very efficient and traditional approach. However, new, object-oriented and multi-domain tools based on the Modelica language are more appropriate for industrial staff or for the people who do not have a deep insight into modelling and simulation. In this paper we describe several educational and industrial application projects in the Dymola–Modelica environment: a process-systems library, two mechanical systems (an inverted pendulum and a laboratory helicopter), a model of thermal and radiation flows in buildings and two models of processes in mineral-wool production, i.e., a pendulum system and a recuperator system. We describe some experiences from these projects, but also from a more general use of the Matlab–Simulink and Dymola–Modelica environments over many years. One simple conclusion is that we need to educate with two approaches: a more physical and advanced acausal Modelica-like approach, but also a more traditional causal or block-oriented approach according to the historical CSSL standard. The important advantages and disadvantages of both approaches are described. The Modelica-based approach enables true 'physical' modelling with fully reusable components. However, there is a particular danger, i.e., users occasionally forget some basic modelling principles when using sophisticated libraries. The result is a very complex modelling structure that is relatively inefficient for the simulation and sometimes has many numerical problems. It is usually very difficult to detect the real reasons for that.

1. Introduction

Good modelling means a good understanding of the real physical system. In order to truly understand how a physical system works, we need to collect knowledge about the system being studied and organise that knowledge in a meaningful way. If we study the traditional literature about the modelling of predominantly continuous dynamical systems, the usual term applied is mathematical modelling and the approaches are usually divided into theoretical, experimental and combined modelling. It is well known that theoretical modelling is based on basic physical laws or, better, on mass- and energy-balance equations. Experimental modelling is based on measurements and the whole system is regarded as a black box with inputs and outputs, and therefore only input–output relations are modelled or, more realistically, identified. Finally, combined modelling is a combination of both approaches. But, what then is computer-aided physical modelling – usually referred to with the shortened expression "physical modelling" – a term that has become very popular recently? It is clear

* Corresponding author. Tel.: +386 14768306.
E-mail address: borut.zupancic@fe.uni-lj.si (B. Zupančič).

1569-190X/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.simpat.2012.03.009

that all modelling is based on physics. In addition, it is well known that one can model real systems with real physical prototypes. However, it is also clear that the known, traditional definition of the term physical modelling does not cover completely its present usages. Therefore, the following definition can be given: *for the purpose of this paper, computer-aided physical modelling introduces a computer-support modelling process, when users need not work, at least at higher hierarchical levels, with differential equations, transfer functions or state space, but with more practical possibly graphical and less mathematical building blocks or library components. So, models clearly reflect the topology or the physics of systems and are therefore good media for understanding and discussions in interdisciplinary working groups. This approach also gives chances to those who are not specialists in modelling and simulation, to work with modelling and simulation.* Another very popular term in conjunction with physical modelling and sophisticated modelling tools is multi-domain modelling. It means the need to build a model with components from different domains, which is often needed when designing hi-tech products: mechanical, electrical, and process control systems, especially within mechatronics, automotive, aerospace, robotics and process-control applications.

Which then are the approaches that can be characteristic for computer-aided physical modelling? Among several approaches we should mention two: (1) Bond graphs and (2) object-oriented modelling approaches originating in the Dymola and, later, the Modelica developments.

Both approaches have many common features and allow us to capture and organise our knowledge about physical systems in a systematic way, enable so-called acausal modelling, enable multi-domain modelling and connect the components in a more complex way: not like in the traditional CSSL or Simulink models, when one variable is present in each connection point, but there are two types of variables with different features. These two types of model-junction variables are not dislocated from each other as in traditional block or signal flow diagrams. These special types of connections actually enables the appropriate working of such models with basic feature so that they preserve the topological structure of real systems.

When the first author started as an assistant in the 1980s, everything about computer-aided modelling (of predominantly continuous systems) was in conjunction with the CSSL-type languages (and earlier analogue computers) and then ten years later with Matlab–Simulink. A group at the faculty modelled a thermal process with some approaches using the analogy of an electric circuit. However, they asked him whether there were some better approaches. He then proposed using CSSL because at that time he did not have today's understanding of physical modelling. But then he read a book by François Cellier [7] and some papers in which he found his experiences and suggestions about how it is possible to dramatically increase the motivation of students when using the acausal OO approach instead of CSSLs. As a result, he was motivated to change his simulation lectures. From then on we used Dymola and later Modelica in parallel with the Matlab–Simulink environment, and we can confirm that this approach is really effective for improving students' motivation.

In this paper we will describe the basic concepts of the introduced approaches for computer aided physical modelling. Then we shall describe some educational and industrial applications. On the basis of our experiences with these applications, some conclusions for a particular application and some general conclusions will be given. There are actually no novel approaches in this paper; however, hopefully, many suggestions, especially for younger modellers (who sometimes have, on the basis of my many years of experience, very biased thinking about methodologies and tools: some want to use the Matlab–Simulink environment for everything, which is a rather low level approach, or they use sophisticated libraries of advanced OO tools without being aware of the significant dangers.

2. Bond graphs

While block diagrams and signal-flow graphs only preserve the computational structure of a system, circuit diagrams (e.g., in SPICE) reflect the topological structure exclusively and are restricted to electric circuits. For these reasons, H.M. Paynter, a professor at MIT, recognised the need for yet another graphical representation of systems, which could show simultaneously the topological and computational structure, and which would be general, i.e., applied to all kinds of systems. The bond-graph form of model representation [1–4] is now routinely used in a number of different disciplines. This is a graphical technique in which the movements of materials or energy are central to bond-graph representations. Most of the early applications involved engineering systems. More recently, however, applications have been reported in entirely different areas, such as biology and economics. The notations of causality provide a tool that is not only for the formulation of system equations, but also for the intuition-based discussion of system behaviour.

There are many computer-modelling tools supporting bond graphs. For example, 20-sim from Twente university [5] already has a long tradition. In addition, there are several public-domain libraries in Modelica [6].

A bond graph is perhaps the most 'physical' modelling technique. Skilled bond-graph modellers say that there are many situations in the modelling of complex systems that can be properly understood and analysed only with this technique. As a bond graph preserves the topological and computational structure (if we use a causal bond graph), it is very convenient for demonstrations of how can modern modelling and simulation tools, which are based on preserving the topology, automatically perform the computational structure during the model compilation. The drawback is probably that it is rather difficult to motivate modelling beginners, i.e., people from industry and other people who do not have such an in-depth modelling background for the bond-graph methodology.

2.1. Drawing

To illustrate the bond graph's usability in a simple way, let us briefly describe the basic features [7]. A bond, which is represented by a harpoon, is an element (connector) that includes two variables, in bond-graph terminology *e* is the effort (e.g., potential) and *f* is the flow (e.g., current). Bond graphs originate from effort or flow sources (SE, SF) and terminate in model components (resistors *R*, capacitances *C*, inductances *I*, etc.). So the flow of energy is very evident. The energy is coming from sources and has sinks in the passive components of the model. The product of an effort and flow variable at a particular bond is actually the energy transmitted over it. Additionally, there are two important junctions in each diagram: (1) the zero junction, in which all the effort variables are equal while all flow variables add to zero, and (2) the one junction, with equal flow variables while all the effort variables add up to zero.

Let us show a simple electric circuit example [7] – see Fig. 1. It is important at the beginning that we separate all the ground nodes.

Based on some principles, we can easily draw the bond graph – see Fig. 2.

As the potential in the ground node can be defined as zero, we can conclude that there are no energy flows into these nodes and therefore we can eliminate such bonds. Furthermore, junctions with only two bonds and with harpoons oriented in the same direction can be eliminated and simplified with only one bond. So the final bond graph shown in Fig. 3 is obtained.

2.2. Bond-graph causality

As mentioned previously, bond graphs can be efficiently used for defining a computational structure, i.e., to define the causality. With some principles we add causality bars to each bond – see Fig. 4.

The basic principles are as follows: for effort and flow sources the causality is physically determined: for passive elements, e.g., resistors, both causalities (current causes voltage or voltage causes current) are computationally meaningful. However, when we have storage elements, it is desirable to derive the variables that are integrals of other variables or, in other words, we have to derive the derivatives of model states.

From the causal bond graph a unique solution with equations can be derived.

$$\begin{array}{llll} U_0 = 10V & i_0 = i_L + i_{R1} & \frac{di_L}{dt} = \frac{1}{L_1} U_0 u_{R1} = U_0 - u_c \\ i_{R1} = \frac{u_{R1}}{R_1} & i_C = i_{R1} - i_{R2} & \frac{du_C}{dt} = \frac{1}{C_1} i_C i_{R2} = \frac{U_{R2}}{R_2} \end{array}$$

However, in more complicated situations we are usually confronted with degenerate systems, structural singularities, algebraic loops, etc. This reality complicates all the procedures for the automatic generation of an efficient computational structure of the models.

In the paragraphs above we only describe some of the main ideas in order to find better relations with modern OO modelling trends, particularly with the Modelica developments. Therefore, there was no reason to present the bond graphs in a more detailed way.

3. Multi-domain and OO modelling with Modelica

Modelica is another approach that solves the important disadvantage of traditional general-purpose simulation tools, i.e., a lack of object-oriented properties, which does not make it possible to develop truly reusable components. For this reason some special-purpose tools were developed (for mechanical, electrical, and chemical systems, etc.). However, such systems do not support multi-domain modelling – modelling of systems from different areas – which is frequently needed, particularly within automotive, aerospace and robotics applications.

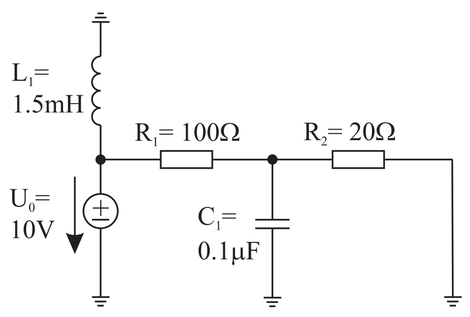

Fig. 1. Electric circuit.

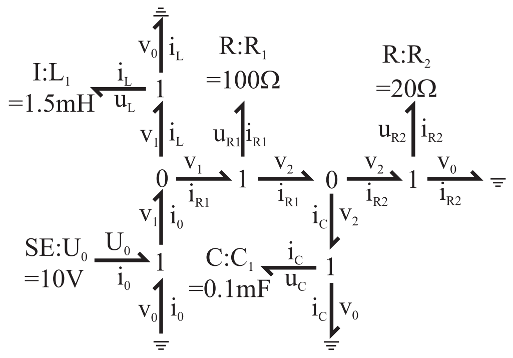

Fig. 2. Bond graph of an electric circuit.

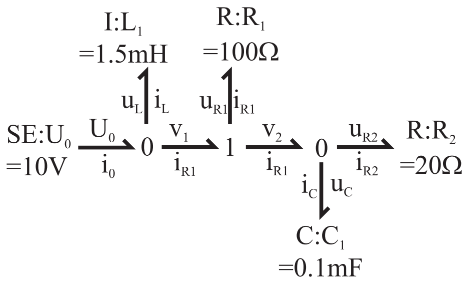

Fig. 3. Simplified bond graph of an electric circuit.

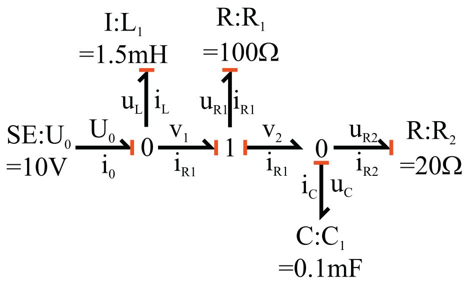

Fig. 4. Causal bond graph.

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 that was, in the past, declared as the evolution from block-oriented tools into object-oriented tools.

3.1. Acausal modelling

When using general-purpose simulation software for predominantly continuous systems (e.g., Simulink) we 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. This procedure requires a lot of engineering skills and manpower as well as being error-prone. However, in order to allow the reuse of component models, the equations should be stated in a neutral form without any consideration of the computational order. This is the so-called acausal modelling approach. Systems in Nature can be thought of as acausal. We never know whether a flow causes a pressure difference in a tube or vice versa. Causality is artificially made because physical laws have to be transformed into a convenient computational description. It is much easier, more convenient and more natural then to use acausal modelling tools such as Dymola with Modelica [7–10]. We write balance and other equations in their natural form as a system of differential algebraic equations. Computer algebra is then utilised to achieve an efficient simulation code.

There are some important processing tasks and features in acausal-oriented modelling tools, which will be briefly described in what follows.

3.1.1. Object orientation

As with every object oriented programming, the principles *encapsulation*, *information hiding* and *inheritance* are very important. 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) the attributes and behaviour of the pre-existing classes, which are referred to as base (ancestor) classes. So an existing code can be used with little or no modification.

However, we do not intend to discuss the well-known general concepts of OO programming. From a modeller's point of view, object orientation means that he/she builds a model similar to a real system, using the libraries' components and then putting them together.

3.1.2. Component coupling

The connections between sub-models are based on variables that define the proper relations and influences between the movements, flows, temperatures, etc. Fig. 5 shows how three thermal systems are connected. Three physical variables are presented in connectors: q_i – thermal flow, j_i – radiant flow and T_i –temperature. Generally, there are two types of variables in the connectors of subsystems: *across* (*potential*) *variables* that become equal in connection points (potentials, temperatures, pressures) and *through (flow) variables* the sum of which equals zero (flows, currents, momentum, forces). A connector is a structure in which all the mentioned variables are collected. By joining connectors the sub-models are connected. During processing the tool generates the appropriate equations from *connect* statements and *connector* definitions and this is a fundamental concept that makes acausal modelling work.

3.1.3. Hybrid features

This means that continuous-time, discrete-time and discontinuous modelling are supported. Such an approach demands many original solutions in symbolic processing and in the run time during the simulation. Events can be periodically or relation (state) triggered. The variable structure models which can be studied in such environments are very useful, especially in control system studies.

3.1.4. Symbolic processing

Multi-domain OO modelling tools have a modelling pre-processor with some basic functions. Each variable (unknown) must be calculated from a single equation. The equations should be sorted in a way that all the variables on the right-hand side are calculated before the equation is executed (in each step size). The size and complexity are also reduced at this stage (the elimination of trivial equations v1 = v2, the symbolic elimination of algebraic loops, the reduction of the problem size (number of unknowns) with tearing, index reduction, etc.).

3.2. OO modelling developments – from Mimic to Modelica

The ideas about OO modelling environments originate in the 1960s with probably the first OO modelling tool, Mimic. Later, the OO idea dramatically influenced general-purpose programming and came back to the modelling area with many activities at Lund University within the group of Prof. Åström and especially Hilding Elmqvist and his PhD thesis about Dymola in 1978. Later, in the early 1990s Hilding Elmqvist founded a company called Dynasim with Dymola development. At that time Dymola was the environment but also the modelling language. Later the Dymola language was replaced with a new OO language called Modelica. Another important milestone was the foundation of the Modelica association, in 2000, which took care of the Modelica language's standardization. Modelica [8,10] is a multi-domain, object-oriented modelling language that supports both high-level modelling using pre-prepared complex model components and detailed modelling using equations. The graphical diagram layer and the textual layer can be efficiently combined. Modelica is currently the most promising result in the sense of standardization efforts for the modelling of hybrid dynamical systems. The Modelica Association also organises the Modelica conferences – there have been eight conferences so far.

Dymola [9] is the first Modelica environment and probably the best. The code is being developed furthur and professionally maintained. Another very popular environment is Open Modelica [11]; it is especially appropriate for academic users.

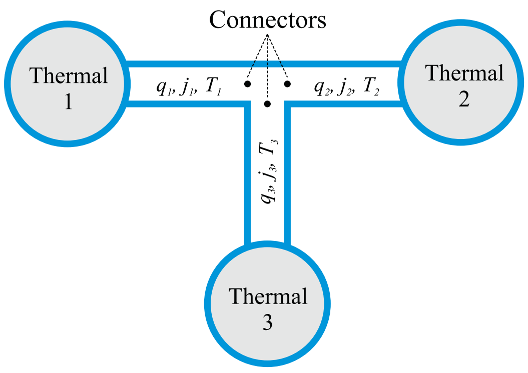

Fig. 5. Connection of three thermal subsystems.

The code is free and open source, which allows additional tools and agorithm developments. Other environments include Simulation X and MapleSim.

Mathworks also recognised that the Matlab environment needs some Modelica-like features for physical modelling. Unfortunately, they did not decide for the Modelica implementation but developed only a Modelica-like environment, Simscape, in 2008.

4. Education applications

Some of the more recent education activities in our control laboratory at the University of Ljubljana, Faculty of Electrical Engineering, will be presented in this section. A process control system, an inverted pendulum and a laboratory helicopter were modelled in the Dymola–Modelica environment and in some areas Matlab–Simulink was also used. The examples themselves are not in the focus here. The main intention is the computer-aided modelling approach, illustrating the thesis from its definition.

4.1. Optimisation of a process control system

Process systems are dynamical systems dealing with physical quantities like level, flow, temperature, pressure, and pH. In such systems the appropriate control strategies 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 generally used the Matlab–Simulink environment. It is extremely efficient for the design of control schemes. However, due to the lack of object orientation the modelling of processes was usually inefficient.

The process used for the demonstration was a three-tank laboratory set-up, which is shown in Fig. 6.

The Dymola–Modelica environment was used for the modelling. The combination of Dymola–Modelica and Matlab–Simulink was also examined. Modelica was used to model the physical part, i.e., the hydraulic process. The process model was then used in Simulink as the 'Dymola' block. So the complete Matlab environment can be used for the design of the control system (e.g., the use of Control toolbox, Optimisation toolbox, etc.).

A new library was developed [12]. The following additional components were implemented: reservoir, pump, valve, flow element, flow generators, interface components between hydraulic and control signals and the appropriate connectors.

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

A cascade controller with the Matlab Optimisation Toolbox (function fminsearc for multidimensional unconstrained nonlinear minimisation (Nelder-Mead)) was developed. The objective function was

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

where $e(t)$ is the difference between the desired and actual levels in Tank 3.

Fig. 7 shows the model that was prepared in Dymola–Modelica for use in Matlab–Simulink as a Dymola block. The connectors (input and outputs) between Dymola–Modelica and Matlab–Simulink were defined.

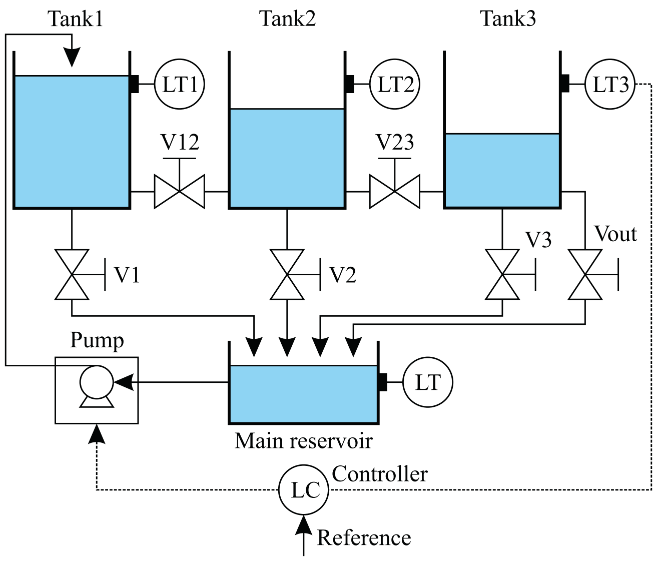

Fig. 6. Process scheme of a three-tank laboratory set up.

Fig. 8 depicts the Simulink model for the cascade control system which also includes the Dymola (or better Modelica) block as the physical part.

We optimised the cascade controller with the main (PI) and auxiliary (P) controllers. The optimisation was performed with the presence of reference and disturbance signals. Fig. 9 shows the optimal responses – all three levels after the reference step change in Tank 3. There was also a disturbance in the middle of the observed period; however, the control system eliminates it so it has almost no influence on the controlled variable (level in Tank3).

4.2. Modelling and control of inverted pendulum on a chart

An inverted pendulum is an extremely efficient system in design courses for modelling and control systems. As it is a mechanical system, it is a nice example for theoretical modelling. And as it is also unstable, it is a good problem for studying more advanced control algorithms.

Fig. 10 shows the physical setup of the inverted pendulum on a cart. The pendulum can only move freely in the x–y plane. The equations for the balance of forces acting on the inverted pendulum in the vertical and horizontal directions can be found in [13].

There are two different control problems: the swing-up phase and control near the upright position. So, two different, but coordinated, strategies are needed. There are many approaches for the swing-up phase. One widely accepted approach is based on energy pumping [14]. In our case we used a simplified version employing the control force signal ±2 N. The sign depends on the sign of the expression $\dot{\Theta}\cos\Theta$ (implemented in the Swing-up Control block). When the pendulum is near the upright position ($|\Theta(t)| < 25^\circ$), the control unit is switched to the stabilization mode [13] (block *Stabilization Control*). The stabilization of the pendulum in the upright position is achieved by means of a Linear Quadratic Regulator (LQR [13,14]). In our case all the state variables were measurable, so we did not need the state observer, which is a convenient and commonly used approach. The overall Modelica scheme of the control problem is shown in Fig. 11.

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

Fig. 12 depicts the diagram layer of the inverted pendulum on the chart in Modelica.

Mechanical parts can be modelled very quickly and efficiently, mostly by using the appropriate objects from the *Modelica Mechanics MultyBody* library and the *Modelica Mechanics Translation* library.

As the paper deals mainly with the evaluation of modelling and simulation tools and not with the control system design, we present only some basic simulation results – see Fig. 13.

We can see that the pendulum is in the stable position $\Theta(0) = -180^\circ$ ($-\pi$ rd) at the beginning. After a small disturbance signal, the swing-up procedure is started and the amplitude of the oscillations increases. At $t = 7.25$ s the stabilization phase

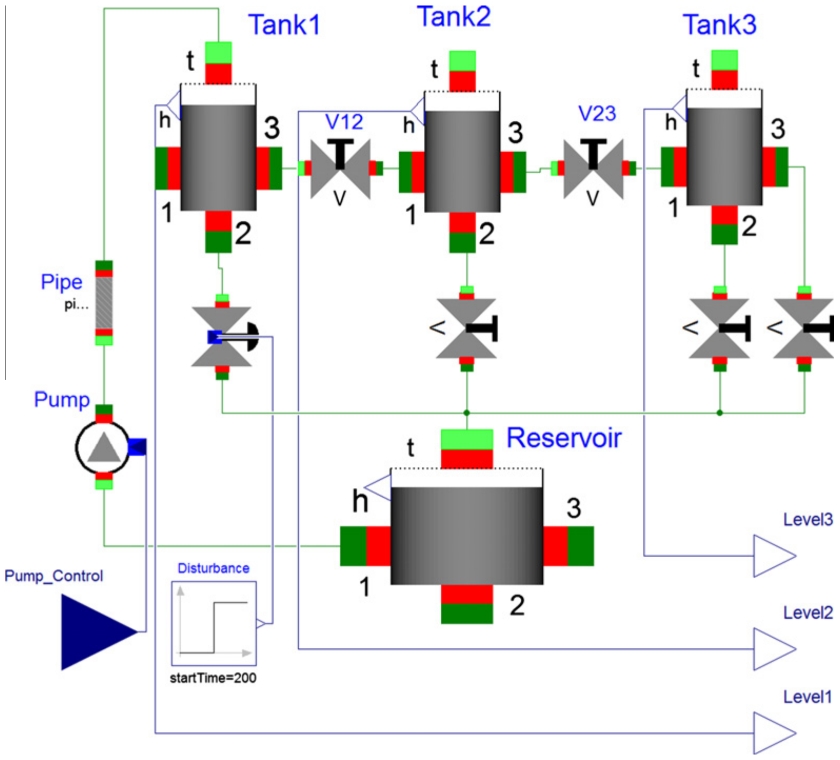

Fig. 7. Dymola–Modelica model prepared for Matlab–Simulink.

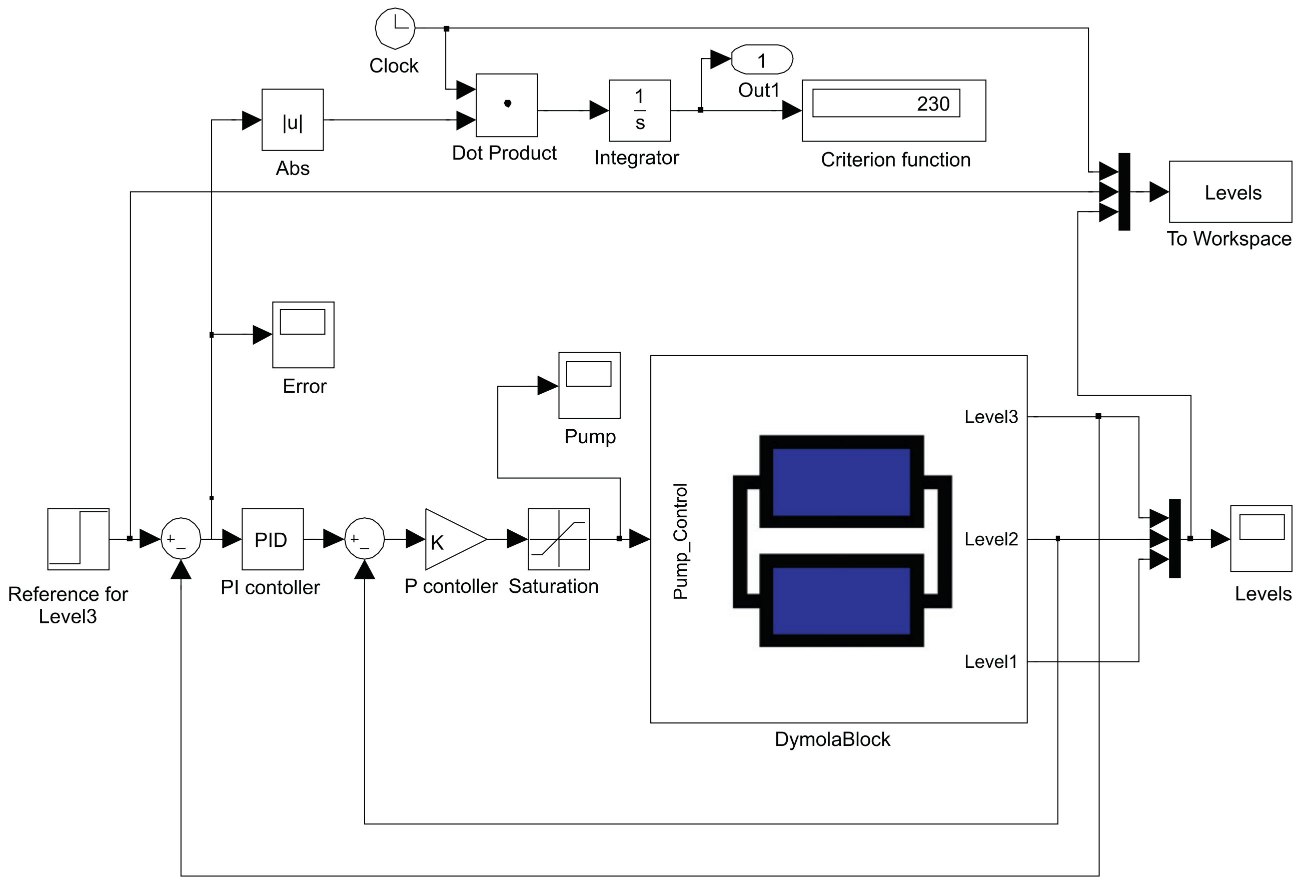

Fig. 8. Simulink model which includes Dymola block.

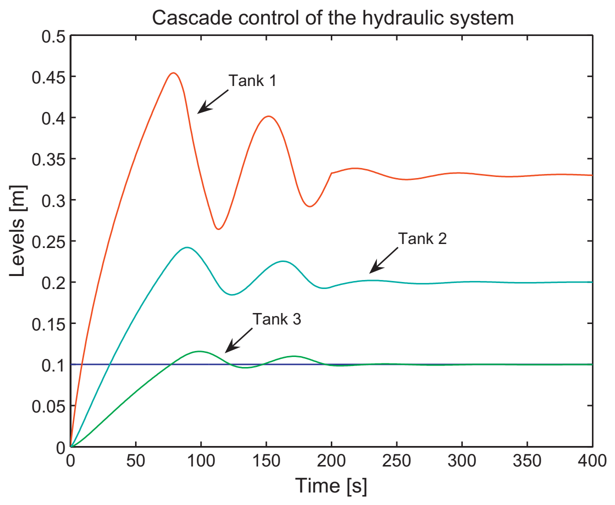

Fig. 9. Tank levels using the cascade controller.

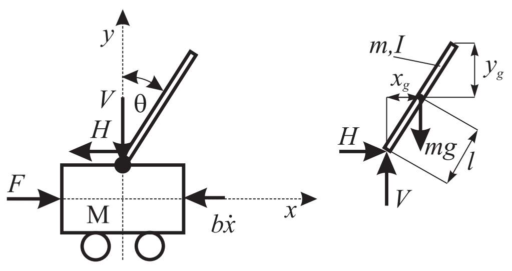

Fig. 10. Inverted pendulum on a cart.

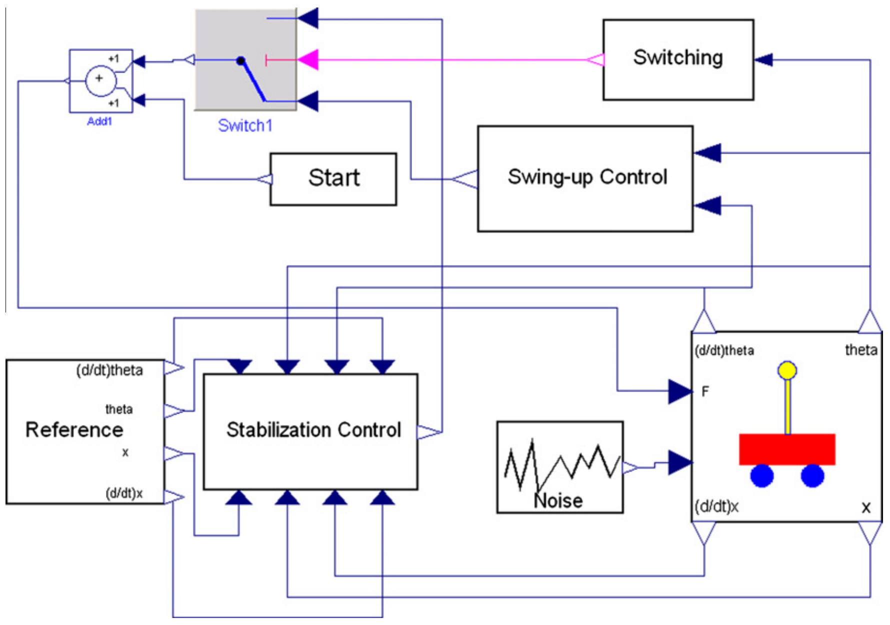

Fig. 11. Modelica scheme for the inverted pendulum control top level.

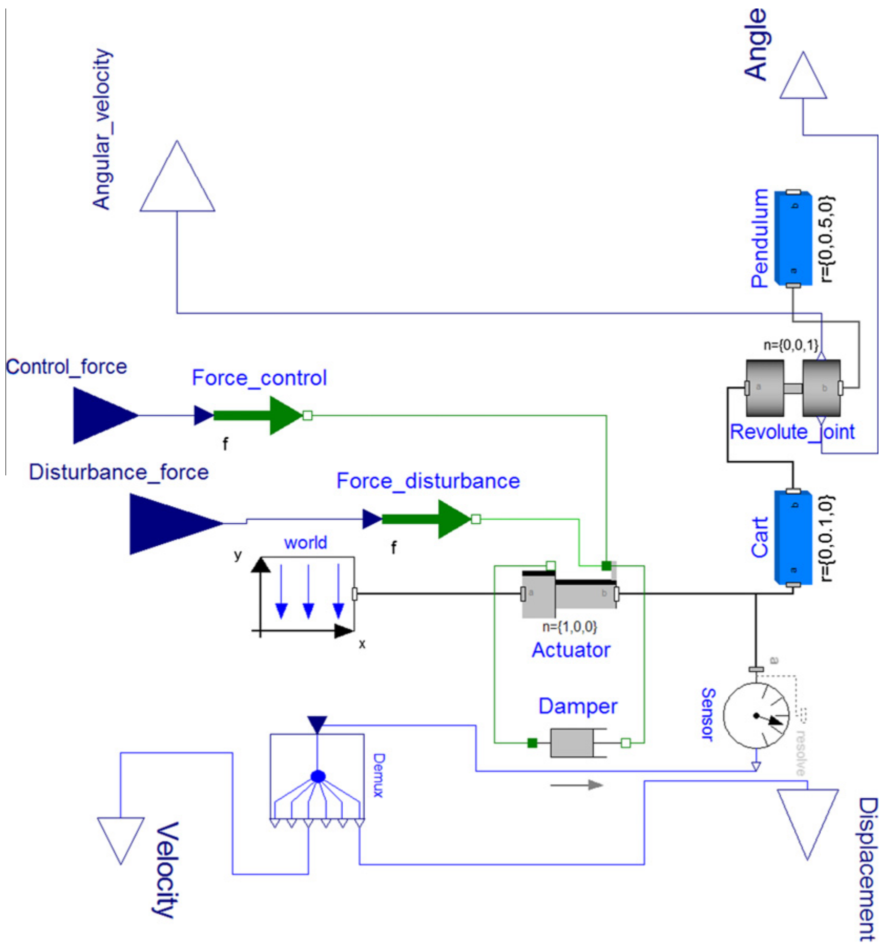

Fig. 12. Inverted pendulum on the cart model.

is started and after that the state controller forces all the states to zero reference values. The robustness of the controller is tested with two disturbance appearances at 10.1 s 7 N for 0.1 s and at 13.7 s 16 N for 0.1 s. Therefore, the control signal is changed, although the angle remains almost unchanged.

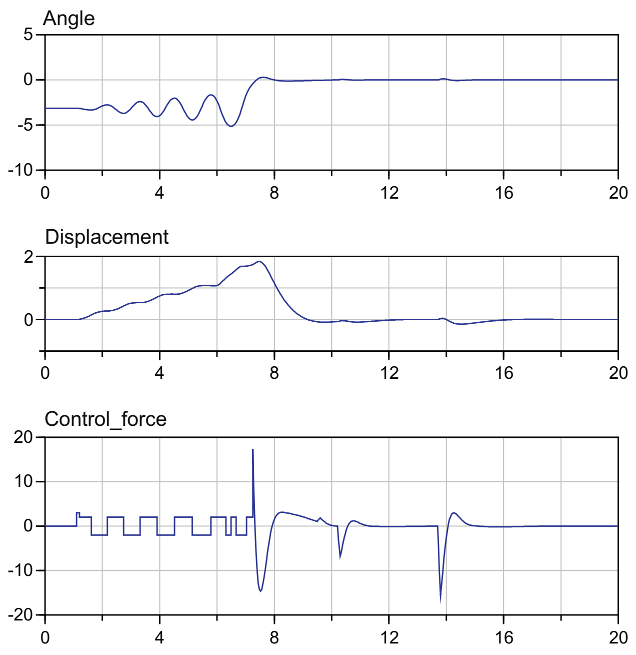

Fig. 13. Simulation results: pendulum angle, chart displacement and control force.

The described inverted pendulum can be animated in Dymola – see animation scheme Fig. 14. The described mechanical components already possess animation capabilities, so no additional effort is needed. We have to choose only the appropriate values for some of the parameters (shape, colour, etc.).

4.3. Modelling and control of a laboratory helicopter

Laboratory set-ups that model real processes and mathematical models have a significant role in efficient control design and education. The CE150 (Fig. 15) is a laboratory helicopter made by Humusoft [15]. It is used for studying system dynamics and control engineering principles and enables a wide range of practical experiments. The goal of modelling and identification is to prepare the basis for the students' laboratory assignments, such as designing a multivariable controller that ensures satisfactory control over a wide operating range. In helicopter modelling, both approaches, i.e., theoretical and experimental, are combined. The mathematical model is described in detail in [16].

The laboratory-helicopter set-up comprises a helicopter body carrying two motors, which drive the main and the tail rotors, and a servomechanism, which shifts the centre of gravity by moving a weight along the helicopter's horizontal axis. The helicopter body is mounted on a stand so that two degrees of freedom are enabled: the rotation around the horizontal axis (pitch angle – y_ψ) and the rotation around the vertical axis (rotation angle or azimuth – y_φ).

Fig. 15 depicts the helicopter as it finally appears during the animation in the Dymola/Modelica environment. The model can be described as a nonlinear multivariable system with three inputs: the voltage driving the main rotor motor (u_1), the voltage driving the tail rotor motor (u_2) and the voltage that drives the servomotor for positioning the weight. In our case

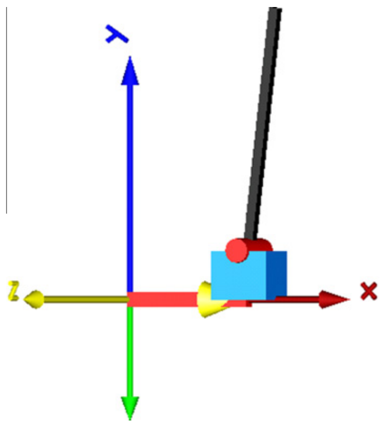

Fig. 14. Animation of the inverted pendulum.

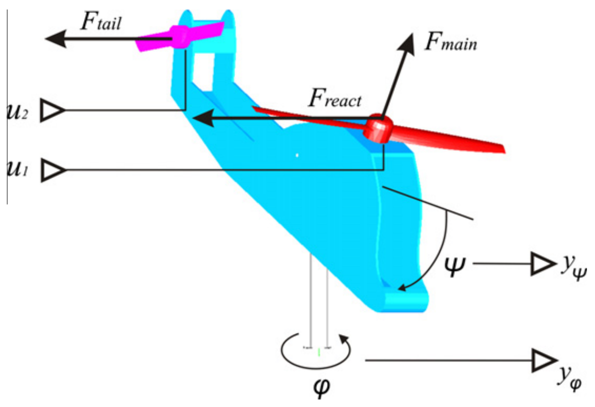

Fig. 15. Laboratory helicopter with the forces affecting the body and control input/output signals.

only the first two inputs were active. The weight was positioned in a fixed place, defining the appropriate mechanical characteristic of the helicopter. The system has two outputs: pitch angle (y_ψ) and rotation angle (y_φ).

A multi-domain approach is useful because we combine two different areas: mechanical systems and control systems, which are usually modelled with block diagrams. We chose *Dymola–Modelica* because the mechanical systems and block diagrams are included in the Standard Modelica library. For the animation the body of the helicopter and both rotors were drawn with the program *SolidEdge* and with *DxfExport* add-in exported into the *dxf* file format. *Dymola* uses *dxf* files to create an animation of the model components. The helicopter body and the stand were modelled with the standard *Modelica* libraries *Mechanics/MultyBody and Mechanics/Rotational.*

The overall scheme of the top-level model in Dymola–Modelica is shown in Fig. 16. The scheme is very clear. It consists of the coordinate system definition (World), the stand model, the helicopter body, the tail and main rotor model (rotor means motor and propeller) and the controller with two reference signals – for the pitch and the rotation angle.

The control system was designed and optimised using the Matlab–Simulink environment. In Simulink the overall mechanical model was presented with the Dymola block. The input u_1 into the helicopter model is the voltage driving the front rotor and the input u_2 is the voltage driving the tail rotor. These are control signals. The output y_1 is the pitch angle and y_2 is the rotation angle. These are the controlled signals.

Both control loops are combinations of the feed-forward control and the PID feedback control with windup protection. Feed-forward signals are obtained from the reference signals r_1 and r_2, which are processed with two nonlinear functions realised with two lookup tables. The feed-forward signals bring the controlled signals into the vicinity of the reference signals. Fine corrections and steady-state error eliminations are achieved with appropriately tuned PID controllers.

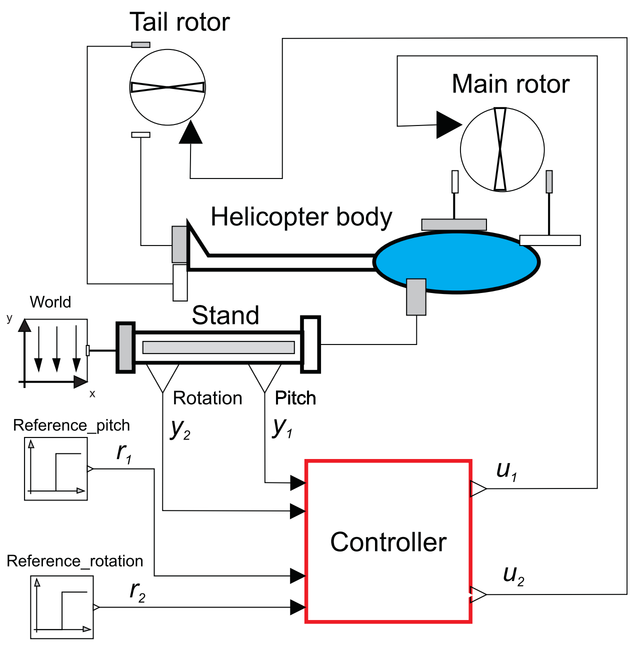

Fig. 16. Top-level model of the control system.

5. Industrial applications

In this section some recent industrial applications will be described: modelling and control of the thermal and radiation flows in buildings and two applications from mineral-wool production: modelling of a mechanical pendulum system and a recuperator. Again, the focus is not on the conventional modelling problems, but more on the computer-aided modelling approach for which, it appears, is much closer to the thinking of industrial people or of the people from different areas than the conventional one.

5.1. Modelling and control of the thermal and radiation flows in buildings

Bioclimatic conditions are extremely important for pleasant and healthy living conditions. As such, they represent a process with inexhaustible possibilities for studying the modelling, simulation and control concepts [17–21]. The described work is based on many years of cooperation between our faculty and the Faculty of Civil and Geodetic Engineering, University of Ljubljana.

There are many aspects when talking about building automation. Our activities were focused on one important aspect: how to ensure good living conditions, but also energy savings, with the appropriate harmonisation of thermal and daylight flows. Our approach was based not only on real experiments, but also on mathematical modelling.

5.1.1. Modelling

The main modelling goal was to efficiently use the model for control systems design. The theoretical modelling of the heat dynamics of a room was based on energy-balance equations that take into account thermal conduction, thermal convection and solar radiation [19,20]. The model was basically developed for a cube-shaped room with several layer walls. Each wall can contain a double-glazed window with a gas between the two panes. The model for the thermal conduction and solar radiation was derived with the aid of Fig. 17.

The incoming thermal flow q_e is partly transmitted (q_{TRg}) and partly absorbed in the glass (q_{ABg}) and in the air inside the room (q_{ABc}). In the isolated wall only the absorption takes place (q_{ABw}). q_{heat} represents the artificial heating (or cooling). A part of the global solar radiation (j_e) is reflected at the surface of the glass and a part is absorbed in the glass (j_{ABg}). The part that comes into the room (j_{TRg}) is absorbed in the isolated wall (j_{ABw}) because of the assumption that the wall is black in the sense of infrared radiation. The effect of the infrared radiation is modelled with flows between the walls and the glass (j_{wg}) and between the glass and the walls (j_{gw}). Many other phenomena were included, many suppositions were used and many simplifications were considered (see details in [19]). The basic inputs (influence variables) of the simulation model are the outside conditions as well as the changeable properties of the envelope: the outdoor air temperature, the temperature of the terrain, the global solar radiation, the properties of opaque elements (thermal capacity and resistance), the properties of transparent elements (geometry of openings, optical characteristics of glass and resistance of fill between panes are variable), the interior properties (the absorption and emission coefficients of the walls and the thermal capacity of the

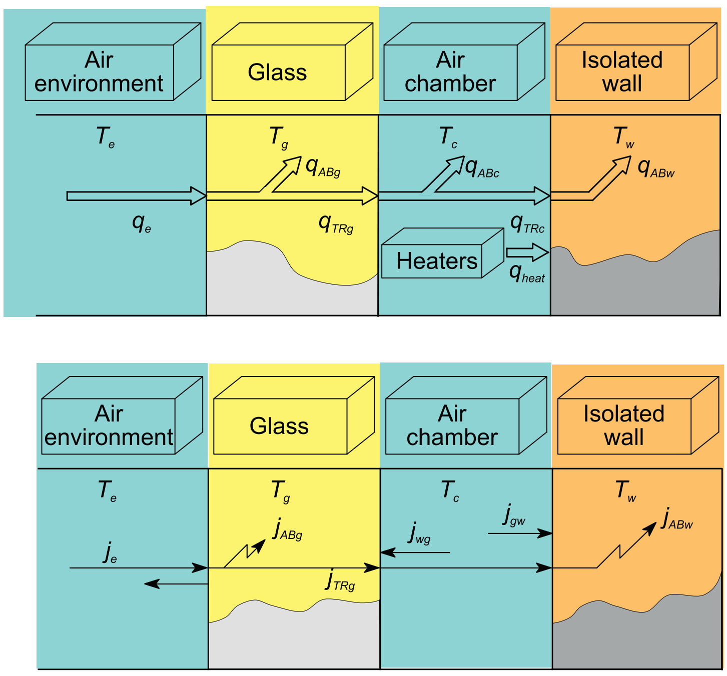

Fig. 17. Thermal and radiant flows.

furnishings are changeable), some other characteristics (changeable orientation, additional heating and cooling – the power of heater, cooler and ventilator). The outputs of the simulation model are the different thermal and radiation flows, the indoor temperature, as well as the temperatures of the walls, windows and surfaces.

5.1.2. Implementation of the simulator in Dymola–Modelica

The simulator was originally developed in the Matlab–Simulink environment [19]. However, the main drawback of this conventional approach became very evident – the simulator components could only be used for the developed configuration. Almost every structural change demanded a new development from scratch. We were not able to establish a library of reusable components for the walls, windows, etc.

So we decided to implement the model in Modelica within the Dymola environment. The basic idea of the implementation in Modelica is to decompose the described system into components that are as simple as possible and then to start from the bottom up, connecting basic components (classes) into more complicated classes, until the top-level model is achieved.

The components provided by the *Modelica Standard Library* were sufficient to start with the implementation of the Modelica model in a graphical way – by connecting the appropriate components (icons). The *HeatTransfer* library contains the components for 1-dimensional heat-transfer modelling with lumped parameters, e.g., *ThermalConductor*, *BodyRadiation, Convection*, etc. All these components include a single connector (port) that contains two variable declarations: the temperature *T* and the heat-flow rate *Q_flow*.

5.1.3. Implementation of the wall

A wall normally consists of several layers. The resulting Modelica scheme for a one-layer implementation is shown in Fig. 18.

The block called the *LayerCapacity* is a model of a heat capacitor, while the blocks *InnerSide* and *OuterSide* are models of the thermal conduction through the layer, and are connected on one side with the *LayerCapacity* and on the other side with the stand-alone connectors *inside* and *outside*. The described structure is defined as a *Layer* model class. There are three connecting points with three different temperatures: the average temperature in the middle of the layer, and two boundary-layer temperatures on both sides. The model of the wall is obtained by simply connecting several layer sub-models in series. The structure of the wall is further connected to the other connectors according to the wall's boundary conditions.

5.1.4. Implementation of the window

A similar procedure was used to implement the model class of a window. The scheme is shown in Fig. 19.

The heat capacities of the outer and inner panes are modelled with two *HeatCapacitor* model classes, *OuterPane* and *InnerPane*, the connectors of which also contain the panes' average temperatures. Both panes interact with each other via thermal radiation and thermal conduction through the air in the gap between the panes. Therefore, *OuterPane* and *InnerPane* are connected with the model classes *AirInside* and *PaneRadiation*. There are also model classes named *OPAbsorbedLight* and *IPAbsorbedLight* in Fig. 19. These are conversion blocks and transform the absorbed solar-radiation flows into connections of the panes' heat-capacity blocks. They are needed to convert the absorbed radiation flows, calculated as a real variable, into the *HeatPort* connector type.

A more detailed description of the solar-radiation flows and the appropriate *Modelica* implementation can be found in [17,18]. All the other blocks, which model other thermal flows coming from the window's surroundings, are connected to the stand-alone connectors *Outside* and *Inside*, respectively. It is clear that the connector *Outside* is not connected directly to the heat-capacity model class *OuterPane* of the pane in Fig. 19, but through the *NightIsolation* model class that models the influence of a (partially) shaded window, which influences the thermal conductance to the outside air (prepared for control).

5.1.5. Implementation of the room

Finally, a model of a room can be built from the prepared model classes. The overall scheme consists of classes that model the room's envelope and those from the interior model class. The appropriate model scheme is shown in Fig. 20. The class *Interior* in the middle is surrounded with the classes of the room envelope. The inner surfaces of the envelope (represented by the connector facing towards *Interior)* are connected to the *RadiationBox* class, which models the thermal radiation exchange between the surfaces (and is beyond the scope of this paper - see [18]), *Interior* class (model of the air mass and the furniture inside) and to the lower-right connector of the *Window* class, which is an array of solar-radiation heat flows received by each surface. The external surfaces of the envelope are connected to connectors that are visible from the outside

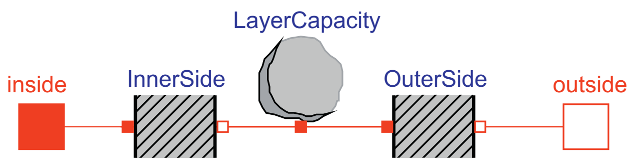

Fig. 18. Scheme of a wall layer in Modelica.

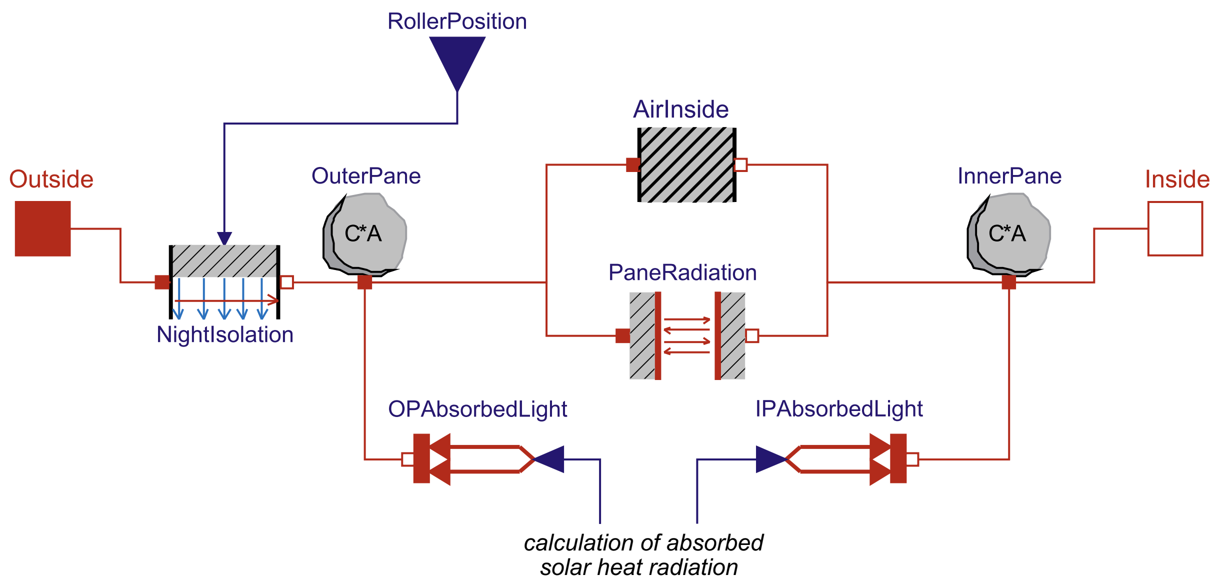

Fig. 19. Scheme of the model describing the thermal processes in a window in Modelica.

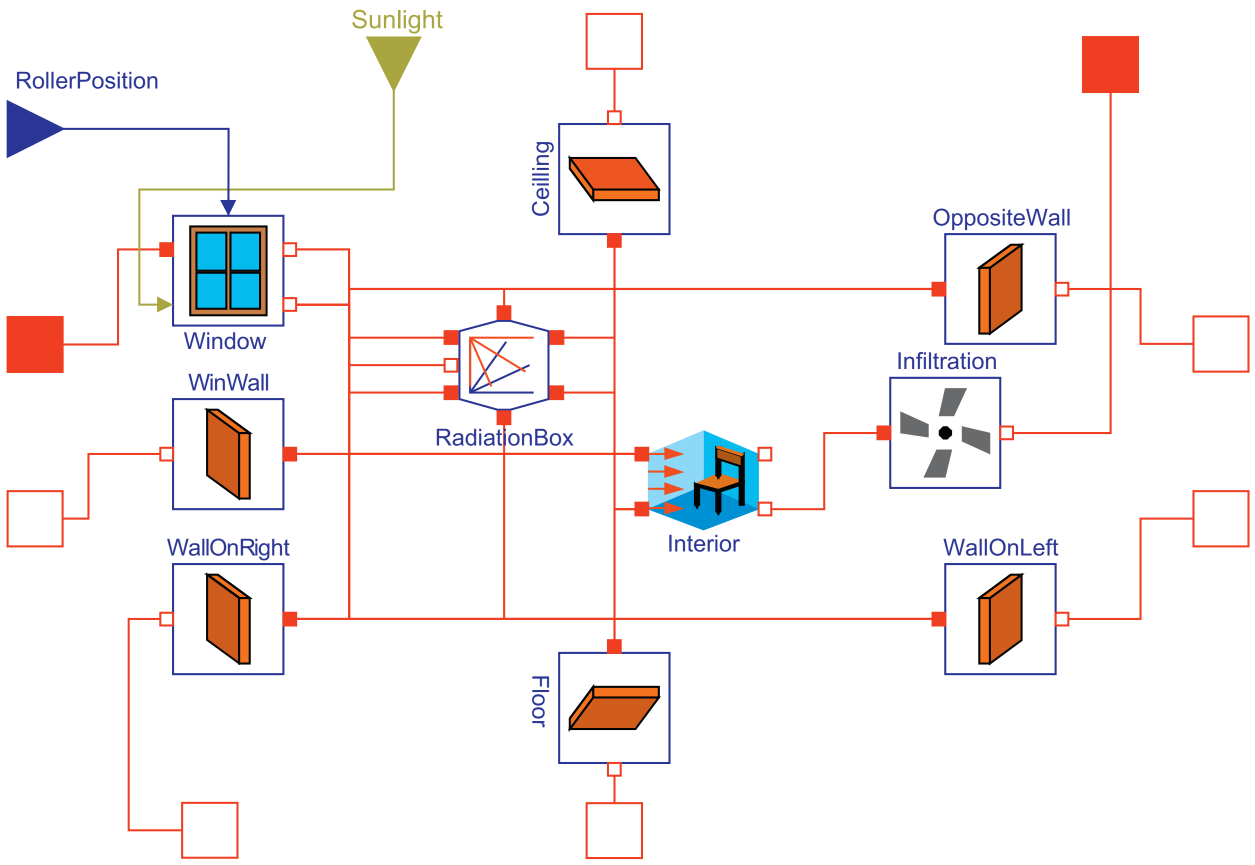

Fig. 20. Modelica model of the room.

of the model of the room, and were used in the top-level model. The blocks that model the convection between the outdoor air and the walls of the building (ceiling, north, south, east and west walls) are therefore connected to those connectors. The *Floor* connector is connected to a constant ground temperature. The intensity of the solar radiation is routed to a class named *Sun* in the top-level model, where the direction vector of the solar rays is also calculated from a specified start date and simulation time and packed together with the solar-radiation component intensity into one connector (*sunlight* in Fig. 20).

A small cube-shaped 'test chamber' with the appropriate sensors for temperature, solar radiation and illumination measurements was built on the roof platform of the Faculty of Civil and Geodetic Engineering, University of Ljubljana, with the main goal being to validate the developed models and control systems. One set of measurements was used for the appropriate final-parameters tuning of the mathematical model of the test chamber. Another set of measurements was used for the simulator validation. Simulations were performed with the measured outdoor temperature and the global solar radiation as the input variables, taken from experiments, as well as with the variable signal for the roller-blind moving regime.

5.1.6. Control system

The simulator was used for different control systems' implementations. The solutions designed with the help of the simulator were then tested on the test chamber. In addition to the conventional approaches with control actions on the heater/cooler and the ventilator, the emphasis was put on the envelope's dynamical adaptations. Therefore, the position of the roller blind was controlled in order to achieve the appropriate harmonisation of the thermal and daylight flows that influence the indoor temperature T_c and the illumination j'_c. The external temperature T_e and the global solar radiation j_e were treated as external disturbances. The skeleton of the control strategy is depicted in Fig. 21.

5.2. Pendulum system in mineral-wool production

This was a modelling project for a Slovenian company that develops production lines for mineral wool. A pendulum system for creating a secondary mineral-wool layer from a primary mineral-wool layer is one of the most important parts of the mineral-wool production technology. To ensure good quality the bottom edge of the pendulum system must move as horizontally as possible and with as constant a velocity as possible. In the current solution the horizontal movement of the bottom edge of the pendulum system is achieved by the movement of the upper rotational axis of the pendulum pair of conveyors – see Fig. 22. The air pockets that are created at the edge of the secondary line due to the rising of the bottom edge of the pendulum represent an important problem.

The linear pendulum system has two motor drives: a pendulum drive, effecting B1 (see Fig. 22), and an eccentric drive, effecting B9. Both drives are electrically coupled: the eccentric drive follows the pendulum drive with a gear ratio of two. A well-known problem is the distribution of the density of the material on the secondary line due to different speed profiles across the secondary line. In the current solution, the speed of the pendulum drive is varying across the line in order to achieve a better distribution. In Fig. 22, SSP represents the direction of the movement of the secondary mineral-wool layer.

5.2.1. Modelling and simulation of the existing solution

The first objective of our work was to build a model of the linear pendulum system in its current form. The Multibody library which includes all the required mechanical components is very convenient. With these elements one can handle the position, speed, acceleration, force and torque of the mechanical components. In Fig. 23 we can see a model of a rigid body with a shape for animation purposes.

The system properties were gathered inside nine model bodies. All the cinematic and dynamic properties of the bodies were calculated with the construction tool called ProEngineer.

Fig. 24 depicts the Dymola animation scheme of the linear pendulum system in the existing construction.

At the beginning all the fixed points T0–T4 were determined. Afterwards, revolute joints were connected to the fixed points. Then the bodies that revolute around the fixed points were attached. In the next steps we built a tree configuration of revolute joints and bodies until all the bodies were used. At that point we faced the tree configuration with six open branches – free frame_B connectors. We closed the open loops with revolute planar cut joints elements. The drives for the pendulum and the eccentric cam in the form of torque generators were added from the Rotational library. They were connected through flanges on the revolute joints from the *MultiBody* library. The pendulum and eccentric drives are directly connected through the gear with the ratio of two. The torque is controlled with a PID controller from the Blocks library. The model components for the position measurements and visualisation were also used. Between the points T4 and V there is a spring–damper pair (not shown in Fig. 24), which ensures an appropriate movement of the point V. We have to stress that

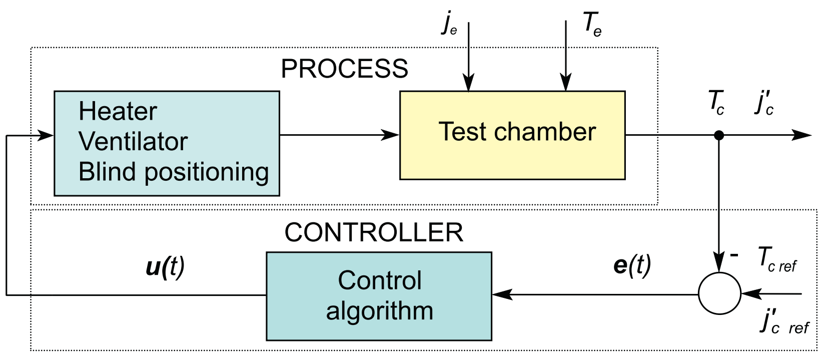

Fig. 21. The skeleton of the chamber control system.

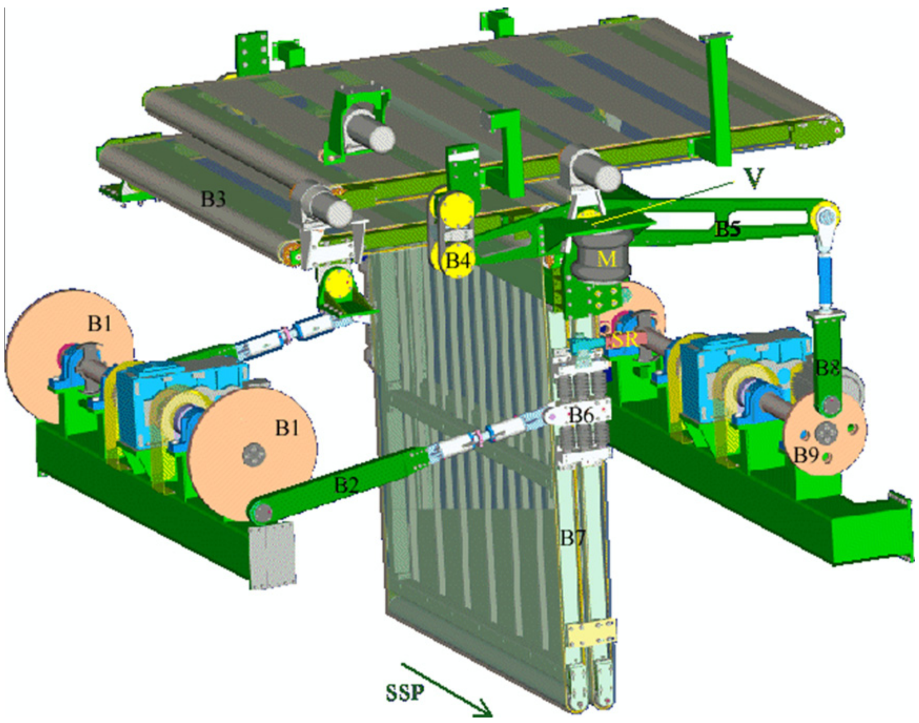

Fig. 22. The construction of a linear pendulum system.

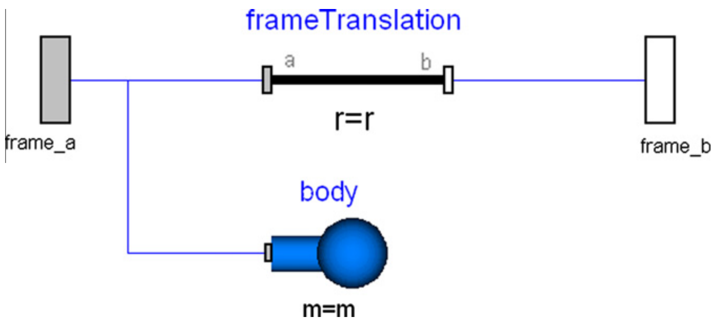

Fig. 23. Rigid-body model with a shape for animation.

the present drives of the described system are rotational. Two asynchronous motors connected in close vector mode are used.

At this stage we should point out some of the conclusions we found when simulating the existing system and some ideas that would lead to the proposal of changing the existing rotational drives with linear drives. With the existing rotational drives it is impossible to achieve an ideal horizontal movement of the bottom edge of the linear pendulum system because the eccentric cam radius is fixed. When changing the amplitude of the pendulum movement, the radius of the eccentric cam also has to be changed.

A better horizontal movement with a more constant speed over the secondary line can be easily achieved if both rotational drives are replaced with linear drives.

5.2.2. Modelling and simulation of the solution with linear drives

So a new model with linear drives was developed and analysed. The rotational drives and arms of the pendulum (B1 and B2) and the eccentric cam (B8 and B9) were replaced with linear drives. The torques needed for the rotational drives were replaced with forces on the flanges of translational joints from the *MultiBody* library. The pendulum and the eccentric drives were separated and controlled with two PID controllers from the Blocks library. The controller reference for the pendulum drive is an ideal movement across the secondary line. The goal is to achieve the horizontal movement of the bottom edge with, as far as possible, a constant speed. So this reference for the horizontal movement is modelled in a way that the maximum acceleration/deceleration at the edges, the width of edges and the amplitude of the movement can be achieved. The position reference for the vertical movement for the eccentric drive is a prescribed constant value. The animation scheme of the pendulum with linear drives in Dymola–Modelica is presented in Fig. 25.

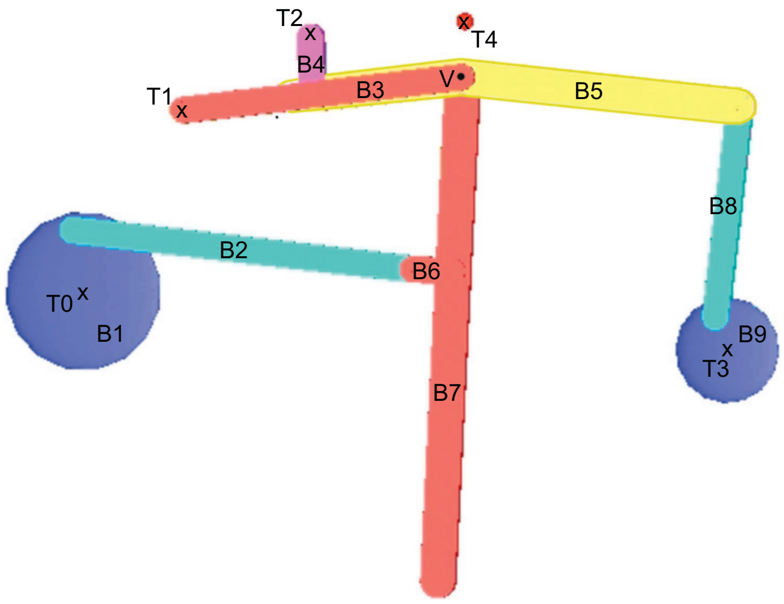

Fig. 24. Animation scheme of the pendulum.

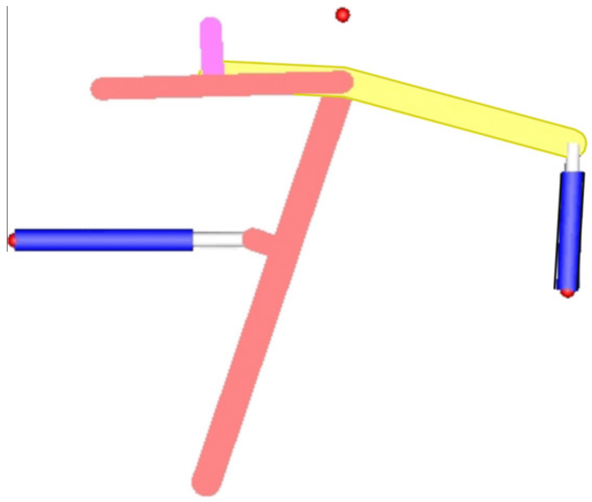

Fig. 25. Animation of the pendulum with linear drives.

The simulation showed that the linear-drives solution is much better because the total vertical movement across the secondary line lies inside 1 cm. With the existing approach the movement lies inside the 4 cm band. For the existing realisation the dynamics is more or less independent of the controller parameters as both drives are coupled through the gear ratio of two. The velocity has a sinusoidal shape, which results in the already mentioned problems. Using linear drives the speed over the secondary line is approximately constant over 85% of the whole area. As the secondary layer is hardened inside the polymerisation oven, where about 5% of the layer is cut off on each side of the layer, the achieved results are satisfactory.

5.3. Recuperator in a mineral wool production

This was another modelling project for a Slovenian company. There are many thermal processes in a mineral-wool production cycle. In cupola furnaces the exhaust gasses must be properly cleaned and this cleaning procedure represents an ideal way to integrate exhaust-air cleaning into a production process. Namely, the exothermic energy from the combustion can be used to preheat the process air, which is used as a blast air in the cupola furnace. This also ensures significant energy savings. The central part of this sub-process consists of four cross-flow and shell and tube heat exchangers – see Fig. 26.

The modelling aim is twofold: it has to provide a better understanding of the thermal processes to technologists and mechanical engineers, who are primarily involved into the design of such systems, and secondly it has to be efficiently used for the design of control strategies that mainly influence the flaps and ventilators. Our goal was to find components from existing Modelica libraries or to develop new components for pipe, heat-exchanger, flap, ventilator, flow source, etc., with the intention to use them for building transparent and user-friendly models that are also understandable to the industrial staff.

5.3.1. Mathematical model of the heat exchanger

A heat exchanger is a device in which energy in the form of heat is transferred. This is usually realised by the confinement of two fluids in some geometry in which they are separated by a conductive material. The properties of a heat exchanger are strongly dependent on the geometry and material as well as on the properties of both fluids. It is well known that such devices usually have nonlinear behaviour [22]. The mathematical modelling is described in more detail in [23]. The observed recuperator process is comprised of shell-and-tube cross-flow heat exchangers.

While the shell's length is relatively small in comparison to its cross-section and the flow through it is highly turbulent (the Reynolds number is of the order 10^5), the temperature differences across the shell are negligible and the shell can be modelled sufficiently accurately as a lumped model with three different temperatures: input temperature, output temperature and average shell temperature. As Modelica does not support the solving of partial differential equations implicitly, the tube section was discretised with the finite-volume method. Some additional equations were added in order to properly model the walls. The heat capacity of the wall is significant and was taken into account. On the other hand, we neglected the thermal conductivity. Then the media changeable properties (density, heat capacity) were considered. The pressure drop was modelled with a quadratic function of the velocity. Finally, we included nonlinear empirical expressions (with Reynolds and Prandtl numbers) for the convective heat-transfer coefficient of the gas.

5.3.2. Implementation in Modelica

5.3.2.1. Heat exchanger. The model of the heat exchanger, which is the most complicated part of the recuperator, basically consists of two thermally coupled pipes. So it should be built up with two pipe classes and an intermediate heat-transfer class. Many publically available Modelica libraries for modelling thermodynamical systems exist. In our case the basic components from the Modelica *Fluid* library are used and adopted. The tube in the upper branch is modelled with two pipes, each with four finite volumes and with the model class for the nonlinear pressure drop because of the tubes' turns in between. The shell in the lower branch consists of two pipes, each with one finite volume and with the model class for the nonlinear pressure drop due to the bundle of tubes in between. Both subsystems are coupled through the model class of the wall (thermal capacitance), which is a custom-made component. Appropriate connectors are used at the outermost edges, i.e., the inlet and outlet. They represent the connecting interface of the heat-exchanger component. The thermo-fluid governing equations and the properties of the media are realised with a nested component of the Modelica *Media* library. The resulting scheme is shown in Fig. 27.

5.3.2.2. Recuperator. In Fig. 28 a model of two connected heat exchangers is shown. It represents a part of the recuperator. The tubes of both exchangers are connected and the temperature at the outlet of the exchanger_1 is controlled by two coupled flaps that define the portion of the flow passing through the heat exchanger_2.

Up to now we validated only the model of the heat exchanger as the validation of the whole recuperator was not possible due to a lack of data. The available measurements were the temperatures of the exhaust gases at the input and output of the shell, the input and output temperatures of the tube gases as well as the volume flow through the tube.

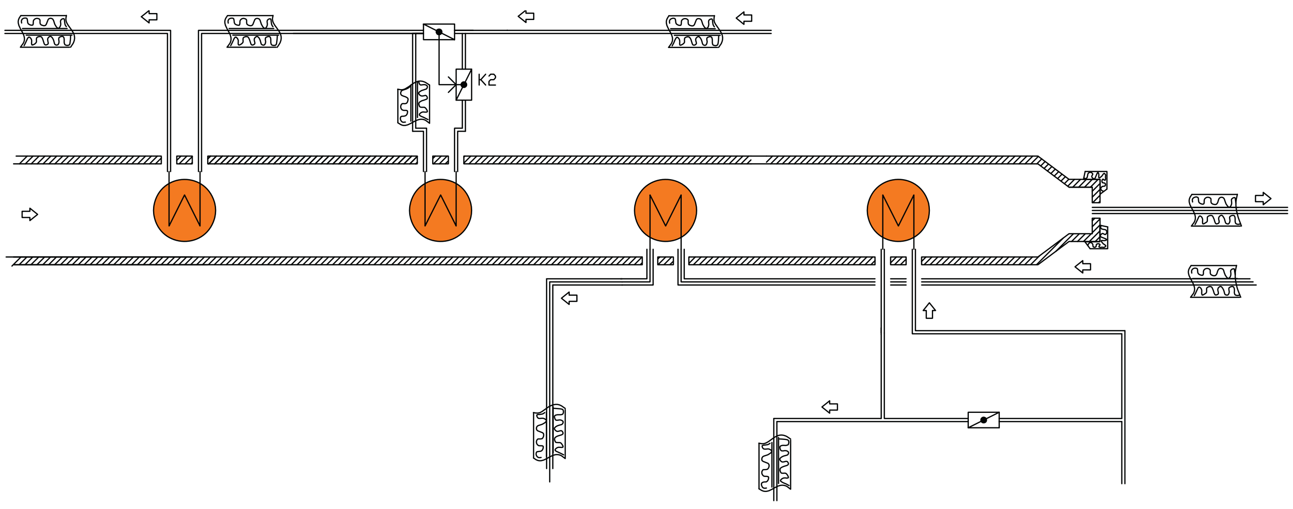

Fig. 26. Recuperator process.

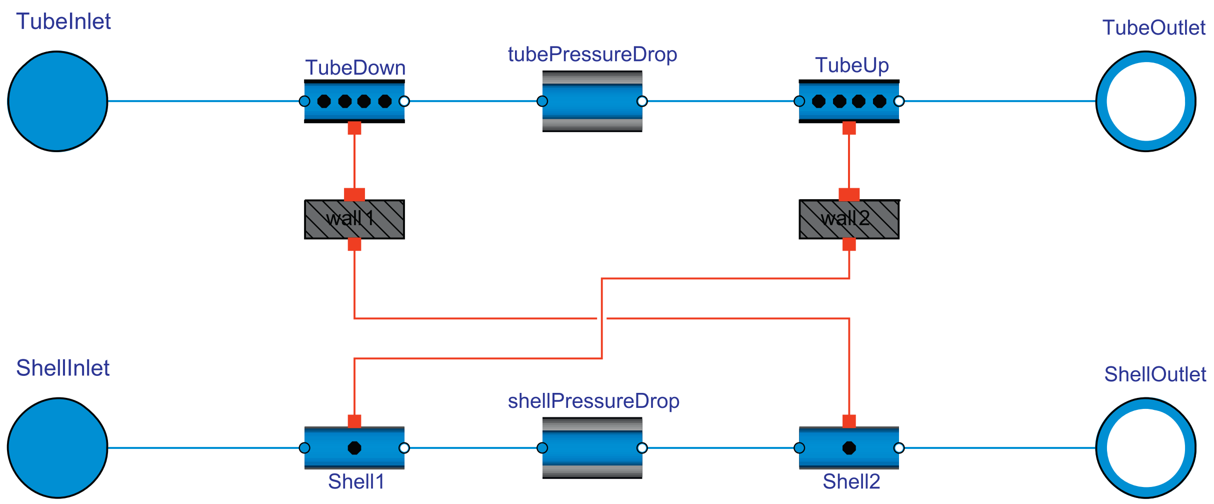

Fig. 27. Modelica scheme of the heat exchanger.

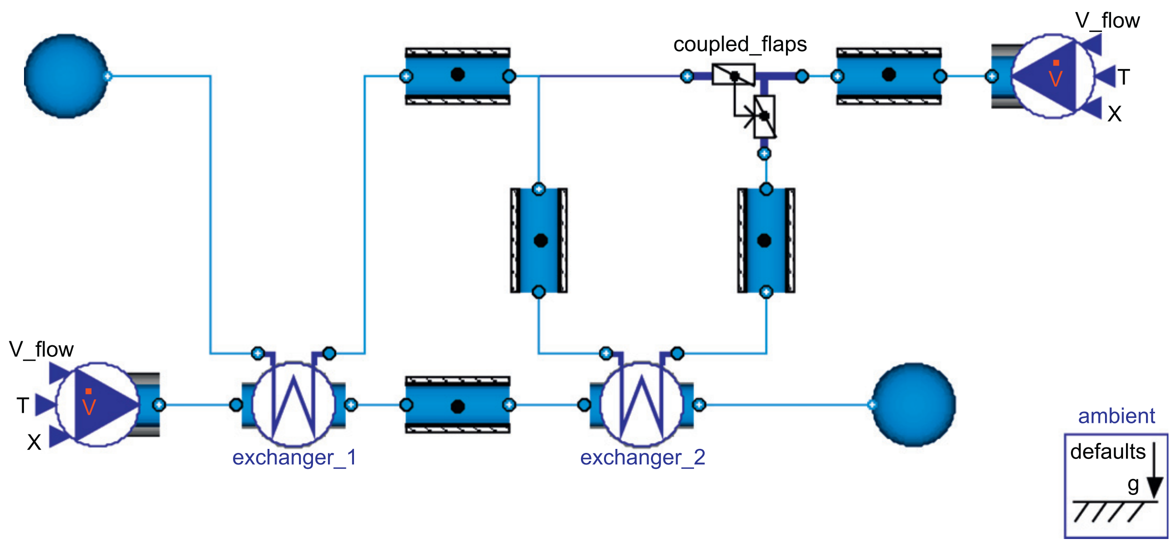

Fig. 28. Part of the recuperator with two heat exchangers.

6. Conclusions

On the basis of our experiences (also from the described projects) we shall formulate some conclusions.

6.1. Some general conclusions

We have a long modelling and simulation education tradition based initially on analogue computers and later on the CSSL languages and the Matlab–Simulink environment. A really big change occurred when we introduced Dymola in the mid 1990s. Students became much more motivated as they had strong modelling support and not only a pure simulation tool. The problem was that we started in the 3rd year with the Matlab–Simulink approach and later with Dymola. Recently, we changed this, starting with Dymola–Modelica in the 2nd year. We begin with basic modelling exercises in which students do not need any additional modelling knowledge as they use standard Modelica libraries. This is so we can start with interesting exercises from the very beginning. Later we continue with other modelling techniques and introduce Matlab–Simulink, which can be treated as a much lower-level approach, but it is important to understand how modelling and simulation actually work. So we would like to emphasise that for us it is not a question of whether to use a causal block-oriented Matlab–Simulink approach or the OO multi-domain Dymola–Modelica approach; it is simply better to use both. We also do not see the necessity of introducing other approaches for the analysis of models in Dymola Modelica, e.g., linearisation, steady-state solver, optimisation, identification, etc. For these, Matlab is simply better. We would propose using Dymola–Modelica for the tasks for which it was designed, i.e. for the physical modelling. And this physical model can be efficiently used in Matlab. The reality is, unfortunately, different. All software producers want to be more or less independent. So we can expect further model analysis extensions in Dymola–Modelica and further physical modelling developments in Matlab with Simscape.

At this stage we would like to mention the old Dymola with the Dymola language in the 1990s. The syntax of this language was more clear and better for education as it was not so much influenced by the later OO programming developments. There were nice connect features for serial and/or parallel components. And it was also much more transparent, to show how equations were solved and sorted. So it was easier to show how Dymola worked [7]. Today the syntax of the compiled models is much more complicated and less transparent. It is much more difficult to show what happens with equations during the translation of a model.

6.2. Process control system

Why did we develop the hydraulic library? In the period when we started with these activities there were no libraries for the modelling of process control systems. Later, many other possibilities became available. However, from the education point of view the professional Modelica libraries sometimes have components that are too complex with an enormous number of parameters for basic education. In addition, the textual parts are not so readable. Our library is the implementation of the elementary equations for tanks, valves, pumps, etc. – just how we teach them. Therefore, the models are understandable and the transparency between graphical and textual modelling is very clear. Students can model the system with library components in a graphical or textual way or they can model the system with elementary Modelica textual modelling features using equations from books. Finally, the example demonstrates possibilities for model validation and efficient optimisation and control system implementation with Matlab. So it is a good example for educational purposes.

6.3. Inverted pendulum, laboratory helicopter

These were two small projects carried out by two PhD. students. In both cases they started with Simulink and later they developed Modelica models. The effort to produce the Simulink simulation model was much greater than for the Modelica. One needs much more time and much more modelling knowledge for a Simulink-based approach. And finally there are problems that are only observed by skilled modellers. The modelling of the inverted pendulum led to an algebraic loop. This usually means not very constructive modelling. There are actually two choices: to leave Simulink to deal with algebraic loops, which is numerically sometimes very risky and extends the simulation time significantly, or to algebraically eliminate loops (which Dymola does automatically). The first choice means a more transparent model, but with bad numerical features. The second solution leads to a much more efficient numerical simulation; however, the model becomes less transparent.

But again we noticed that Matlab–Simulink is more suitable for the design and implementation of control schemes as it has more facilities, especially in conjunction with some toolboxes, e.g., the Control System Toolbox and the Optimisation Toolbox. In our example the state controller was developed for the inverted pendulum, and two PID controllers were used in the helicopter study. Modelica is much superior when modelling physical systems when the concept of algebraic manipulations and specially defined connectors bring many advantages.

There is also one comment that is valid for both examples. Is it better that students work on real (laboratory) systems or on models? For beginners it is difficult to start by working on real systems. First, laboratories do not usually have sufficient set-ups to form small groups. Second, the control on real systems brings so many other application problems that the real modelling and control problematics is completely fuzzyfied. We noticed it many years ago and that was the reason why we introduced more exercises in modelling and simulation environments. But when we used CSSL or Simulink, it was somehow far from reality, the level was too abstract in comparison with real experiments. However, when we started with Dymola–Modelica, this was a completely new story. The connection with the physical system and the OO model became very strong, especially when using diagram layer modelling, e.g., when using icons from pre-prepared model libraries. And if we use animation possibilities then the relation with real systems is even better. So we can conclude that such computer-aided physical modelling can replace, to some extent, the real laboratory set-ups.

Although both examples are rather simple we obtained some numerical problems. Namely, numerical problems are much bigger in OO environments than in CSSL-type simulations. After the model translation we obtain a flat structure of the simulation programme, which becomes rather complex even for relatively simple models.

6.4. Modelling and control of thermal and radiation flows in buildings

As in some other examples we started with Simulink modelling. We developed the models in close cooperation with people from the Faculty of Civil Engineering. We spent many years educating some of those people with modelling and control engineering subjects; however, the Simulink-based environment was definitely not an approach that they could accept. This was the first reason why we started with the Modelica-based platform. They later fully accepted Dymola–Modelica models and our multidisciplinary cooperation became more efficient and easier. The second reason was the clear limitation of the block-oriented causal simulation: the Simulink components were not reusable. We were not able to expand a one-room model into more complex structures. With the Modelica model it was rather easy to model a building with several rooms. The *HeatTransfer* Standard Modelica library was efficiently used. Nevertheless, such an expansion introduces other, above all numerical problems that are usually difficult to solve.

Finally, what did we expect from these models? Hopefully, we would gain a better understanding of the influences of thermal and radiation flows on comfortable living conditions, which will enable the harmonisation of active and passive energy resources, and so important energy savings. Furthermore, the Modelica model is a very suitable environment for education. We plan to use it in conjunction with several subjects at the Faculty of Electrical Engineering and at the Faculty of Civil Engineering.

The model validation is, however, equally important in the Simulink- or Modelica-based approach. Up to now we were able to validate models only with measurements from a cubic-shaped chamber built for experimental purposes. Unfortunately, most experiments were performed with some already implemented control strategies. However, open-loop measurements are much more convenient for modelling purposes.

6.5. Mineral-wool production: pendulum system and recuperator

Modelling and simulation as a modern approach is rarely used in Slovenian industry. People from industry understand that investments are urgent for automation but they do not usually understand that modelling and simulation can be an important part of control system design. As we had one candidate from a mineral-wool production company present at our specialisation courses we succeeded in motivating him to get involved in the described modelling projects. As we had enough experiences with the fact that traditional causal models based, e.g., on the state space approach, are more or less un-understandable for industrial staff, technologists, and maintenance engineers, but also for many other highly educated people who did not have more profound modelling courses in their education, we started from the very beginning with OO Modelica modelling.

However, both projects were quite different. In particular, it was clearly shown how the modelling of mechanical systems is unproblematic in comparison with thermal systems in many aspects. The model of the mechanical system, i.e., the pendulum, was the student's specialisation work. Although he was not a modelling expert and possessed only a basic modelling and simulation knowledge, he was able to develop the model using mainly standard Modelica libraries in just one month. Although the basic aim of this project was that the student becomes familiar with Modelica-based modelling concepts for future work, the results gave us a clear result about how to reconstruct the pendulum drives with linear motors.

We were lucky as we started with the mechanical system, as quick and promising results also encouraged other people and some managers from the company to try such modelling approaches in some other processes. This led us to the recuperator project, which was quite another story. We started with the idea of a half-year project. Unfortunately, we did not finish the planned work after two years. Namely, in this case the models are really complicated, described also with partial differential equations. Although the Modelica *Fluid* library was used we had to add or to adopt some components. Additional problems meant that the library was not finalised, some parts were changing, and there were some bugs. So the implementation in this case became a very research-oriented project, which also had some good consequences. Namely, this model developed my PhD student and some ideas for his PhD arose from this application. In particular, we are facing numerical problems. These problems are related to the gas features, because strong and nonlinear dependencies among the pressures, temperatures, densities and internal energies exist. So the basic equations obtained from the mass and momentum balance equations are extended with nonlinear algebraic equations, which also increases the DAE (differential algebraic equation) index of the system. Under the 'surface' of transparent object-oriented models, very complex structures are generated, which often leads to severe numerical problems. So the Modelica modelling is definitely not the composition of as much as the needed library components, because the models regularly become too complex. If one heat exchanger works, it does not mean that a scheme with two or more exchangers will also work.

The aim of the modelling was to obtain a better insight into the dynamic behaviour of the plant and to use this knowledge for some improvements in the design of new production lines, but also to establish an efficient environment for the appropriate optimisation of processes with advanced model-based control strategies. In this project we cooperate with several mechanical engineers. They are rather skilled in dimensioning components for new lines. But their procedures are based only on steady-state or static understandings. Although they use very complex and sophisticated computer-aided design tools, they never use the possibilities for dynamic simulation. So a simulation with Modelica models gave them a new way of understanding and hopefully we can expect from this cooperation many benefits in the future.

6.6. Some other experiences in relation Simulink-Modelica

Although we have been using Dymola–Modelica for many years, most people in our laboratory still use the Matlab–Simulink environment. What are the reasons?

- There is no need for most people to switch to Modelica. Mainly they work on control system design for which Simulink was a right and sufficient choice. In this connection the process model as a black box input–output relation is usually sufficient.
- It is much easier to detect and solve numerical problems in Simulink.
- Matlab–Simulink is more efficient for the design and implementation of control schemes because of the available toolboxes: Control Systems Toolbox, Optimisation Toolbox, etc.

- Some people dealing with modelling were oriented into alternative AI approaches: neural nets, fuzzy logic. For these approaches Modelica cannot be an advantage.
- Many want to compare Simulink and Modelica; however, there is almost nothing to compare. There are only the well-known advantages and disadvantages of both approaches.
- The main reason is probably that we rarely deal with more complex modelling applications in which the modelling aim would be to get a better insight into a real system.
- Even with many activities in mobile systems (mechanical systems, many multi-domain problems) people continue using Simulink, although Modelica should be a very appropriate tool, which we find difficult to understand.
- Here there is also an objective reason: former versions of Dymola had many bugs, so people were sometimes a bit frustrated. But recent versions are very stable.

However, there are also more optimistic experiences for Modelica arising mainly from the described projects:

- A nice experience with the person from the industrial company, who was not a modelling expert, but was able to develop a rather complex model in reasonable time.
- Extremely good experiences from the education process – much better motivation.
- In contrast to the Simulink approach, the OO Modelica approach enables the development of fully reusable model components.
- In Simulink there is no meaningful textual layer behind the diagram layer, so it is difficult to maintain and document complex models. In Modelica, textual and graphical modelling can be efficiently combined, so we always obtain transparent textual models that are appropriate also for documentation purposes.
- Many public domain Modelica libraries are available.
- The important question is also if we should use Matlab everywhere? When we started with Matlab in the early 1990s, it was very exciting, as we were the first in our institution. We introduced Matlab with Toolboxes instead of CACSD packages (CSSL, Simcos, MatrixX, Model-C, Easy5, etc.). But today students work with Matlab at many subjects and they are sometimes sick of it.
- We feel that it is an advantage if they can use also some other tools, of course not too many. And they must be also exercised with real experiments, which were reduced mainly because of the cheap experimenting possibilities associated with Matlab.
- If Simulink is just a tool, Modelica with an environment (Dymola, MathModelica, Maple, etc.) is more. Much more modelling and simulation concepts can be demonstrated.

6.7. Modelica for the design of control systems

Most approaches for the design of control systems are based on linear or linearised models, usually obtained from experimental modelling. For such a procedure Matlab–Simulink is a very usable environment. For Modelica, however, approaches that are based on 'nonlinear physical' models are of special interest. Methods with inverse models are very popular, especially in the control of mechanical systems. Modelica-based environments can be very usable for it as inverse models can be obtained in many cases with the exchange of inputs and outputs. It is also well known that control-design approaches are usually based on setting the desired poles, damping coefficients, natural frequencies, etc. With Modelica we can change this traditional mathematical way of design into a more physical way. Physical parameters actually primarily cause the behaviour to be inappropriate. Decades ago we influenced these physical parameters with hydraulic, pneumatic or electric controllers, which was really a physical approach. Perhaps this way can serve as a guide how to use Modelica-like tools, because it is easy to implement such a control structure in Modelica in a very physical way. After processing the final structure it is of course causal and the implementation of the control part is then similar to the present approaches. An example: in mechanical systems the damping is usually important for the desired performance. If the damping is too small (e.g., the overshoot in a transient response is too big) we know that the damping should be increased. For several reasons it is of course unreasonable to increase the mechanical damping. Control designers know that it can also be done with the differential component of a PID controller. Our provisional idea in conjunction with Modelica is that a designer in the scheme somehow changes the damping more 'physically', but the translator then automatically establishes the appropriate D component of the controller.

6.8. Particular danger

In conclusion we must emphasise the particular danger when using sophisticated OO multi-domain modelling environments with powerful libraries. Namely, the terms modelling aims, simplifications and model validation are sometimes neglected but remain as the most important tasks in the modelling cycle. So our proposal is not to start with modelling without clear aims, available data for validation, etc. In particular, we need efficient approaches for model simplifications, which are not only mathematically supported but also more physically supported. In this sense there are some efforts, especially those using bond graphs, where one can clearly detect parts with small energy flows, which are then the basis for a model simplification. In our current research activities we are trying to develop some similar approaches for Modelica environments [24].

6.9. Final conclusion

Finally, it can be concluded that using our knowledge, the proposed approach using computer-aided physical modelling is not yet massively and frequently used, especially in industrial applications. However, it becomes clear that such an approach needs the corresponding education, which would, consequently, cause its use in concrete control technology projects, which would undoubtedly mitigate their transfer in industrial practice.

References

[1] H.M. Paynter, Analysis and design of engineering systems, The MIT Press, Cambridge, MA, 1961.
[2] J. Thoma, Introduction to Bond Graphs and their Applications, Pergamon Press, Oxford, 1975.
[3] P.J. Gawthrop, L.S. Smith, Metamodelling: Bond Graphs and Dynamic Systems, Prentice Hall, Englewood Cliffs, NJ, 1996.
[4] W. Borutsky, Bond graph methodology, development and analysis of multidisciplinary dynamic system models, Springer, 2010.
[5] http://www.20sim.com/.
[6] F.E. Cellier, A. Nebot, The Modelica Bond Graph Library. in: Proceedings of the 4th International Modelica Conference, Hamburg, Germany, vol. 1, 2005, pp. 57–65.
[7] F.E. Cellier, Continuous System Modeling, Springer Verlag, 1991.
[8] P. Fritzson, Principles of object oriented modelling and simulation with Modelica 2.1, IEEE Press, John Wiley&Sons Inc., Publication, USA, 2004.
[9] Dymola, Multi-engineering modelling and simulation, Users manual, Ver. 7.3. Dessault System, Dynasim AB, Sweden, Lund, 2010.
[10] Modelica Association. Modelica specification, version 3.1. <http://www.modelica.org/documents/ModelicaSpec31.pdf>, 2009.
[11] A. Pop, P. Fritzson, A. Remar, E. Jagudin, D. Akhvlediani, OpenModelica development environment with Eclipse integration for browsing, modeling, and debugging, in: Proceedings of the 5th International. Modelica Conference, Vienna, Austria, vol. 2, 2006, pp. 459–465.
[12] B. Zupančič, G. Zauner, F. Breitenecker, Modeling and simulation in process technology with Modelica, in: Proceedings of the 2005 Summer Computer Simulation Conference, Cherry Hill, NJ, 2005, pp. 207–212.
[13] S. McGilvray, Self-erecting inverted pendulum: swing up and stabilization control, IEEE Student Paper Contest, 2002.
[14] K.J. Åström, K. Furuta, Swinging up a pendulum by energy control, Automatica 36 (2000) 287–295.
[15] Humusoft, CE 150 Helicopter model. User's manual, Humusoft, Prague, 2002.
[16] G. Karer, B. Zupančič, Modelling and identification of a laboratory helicopter, in: Proceedings of the 5th MATHMOD Conference. Vienna, vol. 2, 2006, p. 9.
[17] A. Sodja, B. Zupančič, Some aspects of thermal and radiation flows modelling in buildings using Modelica, in: Proceedings of 10th International Conference on Computer Modelling and Simulation UKSIM/EUROSIM, Cambridge, UK, 2008, pp. 637–642.
[18] A. Sodja, B. Zupančič, Modelling thermal processes in buildings using an object-oriented approach and Modelica, Simulation Modelling Practice and Theory 17 (6) (2009) 1143–1159.
[19] I. Škrjanc, B. Zupančič, B. Furlan, A. Krainer, Theoretical and experimental fuzzy modelling of building thermal dynamic response, Building and Environment 36 (9) (2001) 1023–1038.
[20] K. Lee, J.E. Braun, Model-based demand-limiting control of building thermal mass, Building and Environment 43 (2008) 1633–1646.
[21] B. Zupančič, A. Sodja, Object oriented modelling of variable envelope properties in buildings, WSEAS Transactions on Systems and Control 3 (12) (2008) 1046–1056.
[22] T. Kupran, Heat Exchanger design handbook, Taylor & Francis, 2000.
[23] A. Sodja, B. Zupančič, J. Šink, Some aspects of the modelling of tube-and-shell heat-exchangers, in: Proceedings of the 7th International Modelica Conference, Como, Italy, 2009, pp. 716–721.
[24] A. Sodja, B. Zupančič, Model verification and debugging of EOO models aided by model reduction techniques, in: Proceedings of the 3rd International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools, Electronic Press Oslo, Linköping, Norway, 2010, pp. 117–120.