Experiences in Teaching of Modeling and Simulation with Emphasize on Equation-based and Acausal Modeling Techniques.

Tomáš Kulhánek*[,1,2], Filip Ježek[2,3], Marek Mateják[2], Jan Šilar[2], and Jiří Kofránek[2]

Abstract— **This work introduces experiences of teaching modeling and simulation for graduate students in the field of biomedical engineering. We emphasize the acausal and object-oriented modeling technique and we have moved from teaching block-oriented tool MATLAB Simulink to acausal and object oriented Modelica language, which can express the structure of the system rather than a process of computation. However, block-oriented approach is allowed in Modelica language too and students have tendency to express the process of computation. Usage of the exemplar acausal domains and approach allows students to understand the modeled problems much deeper. The causality of the computation is derived automatically by the simulation tool.**

I. Introduction

An important aspect of biomedical engineering graduate program is the ability to mathematically formalize the scientific knowledge in biomedicine and utilize such formalization – model – in engineering use cases like simulation, prediction, decision support etc.

There are several approaches, how a mathematical model can be expressed and implemented in an execution code which can be simulated using computers. One approach is to directly incorporate mathematical equations of the model as statements in some programming language code. This includes process of (1) definition of system, (2) decomposition of the system to subsystems, (3) modeling of the subsystems, (4) derivation of the computation causality and (5) implementation in programming language. Another approach is to separate the mathematical model from it's simulator code and allow expressing the causality of the computation model in some higher level programming language and reduce the time of implementation (5). Such tools are usually denoted as block oriented languages. Examples are, e.g., industrial tool MATLAB Simulink, or domain specific languages to model physiology JSIM (NSR Physiome project introduced a JSIM Java based simulation system to support modeling in physiology and introduces a repository of several hundred of models [1]), CellML (IUPS Physiome project introduced XML based standard CellML and FieldML, tools and repository [2]), SBML (Systems Biology Markup Language (SBML) is used for modeling biological system at the level of biochemical reaction and regulatory network [3]).

*corresponding author `tomas.kulhanek@matfyz.cz`

[1]Musical Acoustic Research Center, Music and Dance Faculty, Academy of Performing Arts in Prague, Czech Republic

[2]Institute of Pathological Physiology, First Faculty of Medicine, Charles University in Prague, Czech Republic

[3]Faculty of Electrical Engineering, Czech Technical University in Prague

One of the first complex model of integrative physiology was model of circulatory system with its control regulation published by Guyton et al. [4]. This model was originally implemented in generic programming language FORTRAN and it gradually evolved to the current model HumMod published by Hester et al. [5]. It is not implemented in some programming language directly, they rather use an in-house XML-based domain specific language and tool to interpret and solve this model. Kofránek and Rusz published implementation of the Guyton's original model in MATLAB Simulink [6]. Due to the complexity of further integrative models, it becomes harder to maintain and keep the complex model updated and flexible using the mentioned modeling technology and tools. One of the reason is that the model express the process of computation. Therefore, Kofránek et al. chose acausal and object-oriented modeling language Modelica and implemented the current HumMod model in the standardized Modelica language [7]. Recently we have shown that the block oriented aproach in modeling pulsatile cardiovascular system introduced by Fernandez de Canete et al.[8] may bring problems of further development and understandability. An acausal approach was shown by Kulhánek et al.[9].

Because no other modeling technology was suitable for the complex model Hummod, we started to teach the Modelica language within the classes of modeling and simulation which is executed within the last year of biomedical engineering curriculum with preliminary results promising good acceptance published by Ježek et al.[10]. Additionally, educational text in czech language was published by Kofránek et al.[11] to support the courses of biomedical engineering with focus on patient simulator and modeling methodology with an example of modeling cardiovascular system published originally by Meurs [12] and used in the ®Human Patient Simulator produced by CAE HealthCare[1]. Such models and simulators are used further in teaching of students of medicine. This education methods and tools are shared within MEFANET network, Czech and Slovak Medical Faculties Network [13].

The students of biomedical engineering (of Czech Technical University in Prague, Czech Republic) are familiar with generic programming languages like C++, Java or interpreted Python etc. They are familiar with block-oriented modeling and simulation techniques and capabilities of MATLAB Simulink.

[1]`http://www.caehealthcare.com/` accessed April 2015

978-1-4244-9270-1/15/©2015 IEEE

This work summarizes the main methods of acausal modeling technique in connection with some selected physiological or physical phenomenon. The results is based on the semestral work which is done by students at the end of the course, which shows their ability to cover medium complex phenomenon in integrative way which is essential for further work with complex models incorporating current state-of-the-art knowledge about modeled system.

II. Methods

Modelica language maintained by the Modelica association is object oriented, equation based and acausal modeling language (`http://www.modelica.org`). A Modelica model can be presented by its icon (the icon is used in further model diagrams), defined with a textual notation (source code listing in further text) or defined by diagrams (figures in further text). We recommend textual form to express equations of basic subsystem describing domain specific laws and diagram form to express composition of several basic components. Further reading about Modelica is in published works of Fritzson [14] or free accessible online book of Tiller[15].

Object orientation means that model is defined as a class, which can be instantiated. Each instance share type and differ in parameters and the place where it is used. Inheritance and some sort of polymorphism is possible. *Equation based* means that the model can be expressed using equations instead of assignment statements. Modelica tool will decide which variable is input and output upon compilation. *Acausal* means that the model composed of several submodel do not need explicitly declare what is input and output. Acausal connector is special purpose class to define variables of the model shared with other models or classes. Connecting two or more components via acausal connector will generate analogy of Kirchhoff's law equations, which ensure equality of all "non-flow" variables in connectors $p_1 = p_2 = \ldots = p_n$ (1) and zero sum of all "flow" variables $\sum_{i=1}^{n} q_i = 0$ (2)

The basics of various systems can be explained using analogy between domains. Not only electrical and mechanical domains are analogous [16]. Analogies exist among hydraulic, thermodynamic and chemical domain too [11].

We use the following domains and examples to show the analogies among different domains with focus on acausal and object oriented approach.

A. Hydraulic domain

Hydraulic domain can be used to express cardiovascular system (CVS) using Windkessel approach[17]. CVS can be decomposed into abstract component expressing hydraulic elasticity and hydraulic resistance. Connector *HydraulicPort* with "flow" variable q and non-flow variable pressure p is presented by it's Icon and by the Modelica source code definition:

Icon — Modelica source

```
connector HydraulicPort
  flow Real q;
  Real p;
end HydraulicPort;
```

Model of hydraulic resistor(conductor) with parameter G denoting conductance and two hydraulic ports is expressed by the equations:

$$q_{in}.q = -q_{out}.q \tag{3}$$

$$q_{in}.q = G \times (q_{in}.p - q_{out}.p) \tag{4}$$

Model of hydraulic elastance, with parameters V_0 as unstressed volume, p_0 external pressure and C compliance(reciprocal value of elastance) with state variable V volume, is expressed by these equations:

$$p - p_0 = \begin{cases} 0 & \text{if } V < V_0 \\ \frac{V - V_0}{C} & \text{otherwise} \end{cases} \tag{5}$$

$$\frac{\mathrm{d}V}{\mathrm{d}t} = q \tag{6}$$

Both models can be written in Modelica.

Icon — Modelica source


```
model HydraulicConductor
  parameter Real G;
  HydraulicPort qin;
  HydraulicPort qout;
equation
  qin.q= -qout.q; // eq.(3)
  qin.q = G*(qin.p-qout.p); // eq.(4)
end HydraulicConductor;
```

```
model HydraulicElastance
    Real V;
    parameter Real V0;
    parameter Real p0;
    parameter Real C;
    HydraulicPort qin;
equation
   // eq.(5)
  qin.p-p0 = if (V<V0) then 0 else (V-V0)/C;
  der(V) = qin.q; // eq.(6)
end HydraulicElastance;
```

This can be used to model two ideal baloons with liquid interconnected via a tube characterized by some resistance. The acausal connectors *qin* and *qout* are connected via the *connect()* statement in the following listing:

```
model twoballons
  HydraulicConductor systemicResistance;
  HydraulicElastance arteries;
  HydraulicElastance veins;
equation
  connect(arteries.qin, systemicResistance.qin);
  connect(systemicResistance.qout, veins.qin);
end twoballons;
```

This textual form is equivalent to the diagram form in Figure 1.

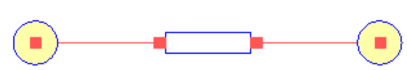


Fig. 1: Diagram of the model of two balloons. It connects two HydraulicElastance via HydraulicResistance represented by its icons via acausal connectors. Each connection (red lines) is equivalent to the Modelica "connect" statement above.

The concrete instances may differ according to what is known about the system, either by external measurement, or by some superior model. The *ballsVolume* is initialized with initial volume of first balloon $V(start) = 5000$ and by

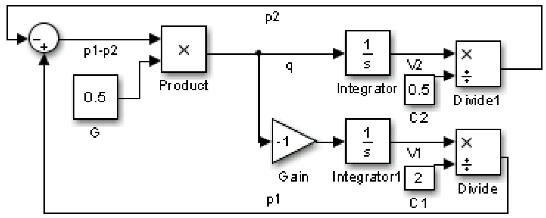


Fig. 2: Causal model of two balloons in MATLAB/Simulink (V_0 and p_0 are not considered). The model presents the process of computation rather than the physical concept. Causality needs to be solved before the model implementation.

setting parameter values of V_0, p_0, C and G as seen in the following Modelica listing:

```
model ballsVolume
  extends twoballons(
    arteries(V(start=5000), V0=529, p0=0, C=1.5),
    systemicResistance(G=1),
    veins(V0=2845, p0=0, C=200));
end ballsVolume;
```

The Modelica tool will decide automatically which variables will be dependent and which independent. Computation flow is solved as seen from the following generated code (note the assignment statement :=).

```
// Translated Modelica model generated by Dymola ...
// Dynamics Section
  systemicResistance.qout.p := veins.p0+
    (if veins.V < veins.V0 then 0
    else (veins.V-veins.V0)/veins.C);
  systemicResistance.qin.p := arteries.p0+
    (if arteries.V < arteries.V0 then 0
    else (arteries.V-arteries.V0)/arteries.C);
  der(arteries.V) := systemicResistance.G*
    (systemicResistance.qout.p-
      systemicResistance.qin.p);
  der(veins.V) :=  -der(arteries.V);
```

When simulated the system goes to some equilibrium, steady state, which illustrates that the liquid has tendency to go from the compartment with higher pressure to compartment with lower pressure until the pressures are ballanced. In physiology, the blood has tendency to flow from arteries (having low compliance) to veins (having high compliance).

B. Spring/Mass System

An example of harmonic oscillator is decomposed into mass, fix, spring and joint subsystems. The mass is characterized by the parameter m and following equations among F–force, a–acceleration: $F = m \times a$ (7) $a = \frac{\mathrm{d}^2y}{\mathrm{d}t^2}$ (8)

The spring is characterized by the parameter k–stiffness and following equations among F–force, dy–displacement: $F_1 = F_2$ (9) $F = -k \times dy$ (10) $F = m \times a$ (11) $dy = y_2 - y_1$ (12)

The fixed point is characterized by the fixed position set to 0: $y = 0$ (13)

The joint is presented as acausal connector similar way as in the hydraulic domain. But contrary to common understanding, F–force is the "flow" variable and

Icon	Modelica source

	`connector MechanicalJoint`
	`  Real y "position";`
	`  flow Real F "force";`
	`end MechanicalJoint;`

	`model MechanicalFix`
	`  MechanicalJoint mechanicalJoint;`
	`equation`
	`  mechanicalJoint.y=0;`
	`end MechanicalFix;`

	`model MechanicalSpring`
	`  Real dy "displacement";`
	`  parameter Real k= 10;`
	`  MechanicalJoint upperJoint;`
	`  MechanicalJoint lowerJoint;`
	`equation`
	`   lowerJoint.F = -k * dy;`
	`   upperJoint.F +lowerJoint.F = 0;`
	`   dy = upperJoint.y - lowerJoint.y;`
	`end MechanicalSpring;`

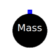
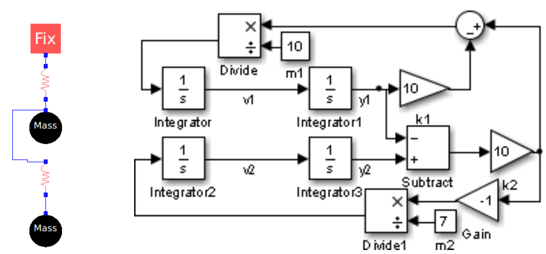
	`model MechanicalMass`
	`  MechanicalJoint mechanicalJoint;`
	`  Real y "position of the mass";`
	`  parameter Real initPos=0 "initial position";`
	`  parameter Real m "mass";`
	`  Real a "acceleration";`
	`  Real v "velocity";`
	`initial equation`
	`  y=initPos;`
	`equation`
	`  mechanicalJoint.y = y;`
	`  mechanicalJoint.F = m * a;`
	`  v = der(y);`
	`  a = der(v);`
	`end MechanicalMass;`

TABLE I: Spring/Mass system components in Modelica.



(a) (b)

Fig. 3: Acausal model of spring-mass system in Modelica (a) vs. causal model in MATLAB/Simulink (b)

y–position is "non-flow" variable. The Modelica implementation of these subsystem is in Modelica listing in Table I. An example of two serially connected spring mass system lead to the following differential equations: $m_1 \frac{\mathrm{d}^2 y_1}{\mathrm{d}t^2} - k_2(y_2 - y_1) + k_1 y_1 = 0$ (14) $m_2 \frac{\mathrm{d}^2 y_2}{\mathrm{d}t^2} + k_2(y_2 - y_1) = 0$ (15)

The causality of this system of equations needs to be solved when modeling with causal tool (Figure 3b). However, this is not needed when using acausal Modelica (Figure 3a) which connects fixed point component with first spring and a first mass, second spring is connected with first spring and second mass component. Causality is solved by the simulation tool as in the previous example of two balloons.

The comparison of the equations 14 and 15 and causality solution in MATLAB/Simulink with acausal Modelica diagram in Figure 3 is used to motivate students in order to decompose system to basic physical components and prefer acausal approach.

There are several tools supporting Modelica, we use the commercial Dymola tool (`www.dynasim.se`), Wolfram System Modeler (`www.wolfram.com/system-modeler`) and open source OpenModelica (`www.openmodelica.org`). Matejak et al. published the Modelica library Physiolibrary containing the components for hydraulic, chemical, osmotic and thermal domain, which are useful for building complex models of human physiology [18], [19]. We encourage to use this library and other numerous Modelica libraries.

We provide one-semestral course, enough to show basic concepts of Modelica only. In the end of the course though, the students are capable of implementing a model from biomedical domain on their own. The objective of semester project is to utilize the taught Modelica language to formalize arbitrary object from biomedical field. Because we emphasize using acausal approach, we force them to use knowledge from neighboring fields (such as physiology, physics, chemistry or biology) and better understand the modeled reality.

III. Results

The models implemented by graduate students varied from model of small water power plant to model of scuba diver and formation of gas bubbles to model of vocal tract, capable of producing understandable vowel waveforms (`modelica.creativeconnections.cz/student-works/2014/`). Some students chose to not only implement, but design the model of their own or combine existing models and enhance them, which shows deep understanding of the issue. We state, that when using traditional causal tools, this might not be possible or at least as common. Some students continue and e.g. Dolezalova published demonstration of extracorporeal membrane oxygenation (ECMO)[20].

Students often use primary literature, where the accompanying models (if present), are most usually either in imperative programming language or block modeling language. Often the first reaction is to hold the block scheme, which is possible in Modelica, but not appropriate for the task, as it does not require deep insight.

IV. Conclusion

Our students are generally used to choose the easier way, which means to mirror the proposed procedure, without profound understanding. Using acausal approach we make them to really understand the process. Our experience proves, that using acausal modeling deepens the understanding of the model instead of imitate the function only. Using acausal approach, by the end of the course students are capable of creating complex systems with tens to hundreds of equations and still being able to comprehend it as a whole.

Acknowledgment

This work was supported by the IP DKR of the Academy of Performing Arts and FR CESNET z.s.p.o. 552/2014.

References

[1] Erik Butterworth, Bartholomew E. Jardine, Gary M. Raymond, Maxwell L. Neal, and James B. Bassingthwaighte. JSim, an Open-source Modeling System for Data Analysis. *F1000Research*, 2, May 2014. `doi:10.12688/f1000research.2-288.v2`.

[2] Tommy Yu, Catherine M Lloyd, David P Nickerson, Michael T Cooling, Andrew K Miller, Alan Garny, Jonna R Terkildsen, James Lawson, Randall D Britten, Peter J Hunter, and Poul M F Nielsen. The Physiome Model Repository 2. *Bioinformatics (Oxford, England)*, 27(5):743–744, 2011. `doi:10.1093/bioinformatics/btq723`.

[3] Michael Hucka, ABBJ Finney, Benjamin J Bornstein, Sarah M Keating, Bruce E Shapiro, Joanne Matthews, Ben L Kovitz, Maria J Schilstra, Akira Funahashi, and John C Doyle. Evolving a Lingua Franca and Associated Software Infrastructure for Computational Systems Biology: The Systems Biology Markup Language (SBML) Project. *Systems biology*, 1(1):41–53, 2004.

[4] A C Guyton, T G Coleman, and H J Granger. Circulation: Overall Regulation. *Annual review of physiology*, 34:13–46, 1972. `doi:10.1146/annurev.ph.34.030172.000305`.

[5] Robert L Hester, Alison J Brown, Leland Husband, Radu Iliescu, Drew Pruett, Richard Summers, and Thomas G Coleman. HumMod: A Modeling Environment for the Simulation of Integrative Human Physiology. *Frontiers in physiology*, 2(April):12, January 2011. `doi:10.3389/fphys.2011.00012`.

[6] Jiří Kofránek and Jan Rusz. Restoration of Guyton's Diagram for Regulation of the Circulation as a Basis For Quantitative Physiological Model Development. *Physiological research / Academia Scientiarum Bohemoslovaca*, 59(6):897–908, January 2010.

[7] Jiří Kofránek, Marek Mateják, and Pavol Privitzer. Hummod – Large Scale Physiological Models in Modelica. In *Proceedings 8th Modelica Conference, Dresden, Germany*, pages 713–724, 2011.

[8] J Fernandez de Canete, J Luque, J Barbancho, and V Munoz. Modelling of Long-term and Short-term Mechanisms of Arterial Pressure Control in the Cardiovascular System: An Object-oriented Approach. *Computers in biology and medicine*, 47:104–112, April 2014. `doi:10.1016/j.compbiomed.2014.01.006`.

[9] Tomáš Kulhánek, Jiří Kofránek, and Marek Mateják. Modeling of Short-term Mechanism of Arterial Pressure Control in the Cardiovascular System: Object-oriented and Acausal Approach. *Computers in Biology and Medicine*, 54:137–144, September 2014. `doi:10.1016/j.compbiomed.2014.08.025`.

[10] Filip Ježek, Tomáš Kročeк, Marek Mateják, and Jiří Kofránek. Zkušenosti z inovace výuky Modelování a Simulace na FEL ČVUT. In *sborník příspěvků MEDSOFT*, pages 139–146, 2012.

[11] Jiří Kofránek and Jiří Hozman. *Pacientské simulátory*, volume 000. Creative Connection s.r.o., 2013. URL: `http://www.creativeconnections.cz/pacientskeSimulatory`.

[12] W van Meurs. *Modeling and Simulation in Biomedical Engineering: Applications in Cardiorespiratory Physiology*. McGraw-Hill Professional, 2011.

[13] Daniel Schwarz and Ladislav Dušek, editors. *Computer Applications, Systems and Networks for Medical Faculties*. Institute of Biostatistics and Analyses, Masaryk University, Brno; Facta Medica, 2014.

[14] Peter Fritzson. *Principles of object-oriented modeling and simulation with Modelica 2.1*. Wiley. com, 2010.

[15] Michael Tiller. Modelica by Example, 2014. URL: `http://book.xogeny.com`.

[16] F A Firestone. A New Analogy Between Mechanical and Electrical Systems. *The Journal of the Acoustical Society of America*, 4(3):249–267, 1933. `doi:10.1121/1.1915605`.

[17] Tomáš Kulhánek, Martin Tribula, Jiří Kofránek, and Marek Mateják. Simple Models of the Cardiovascular System for Educational and Research Purposes. *MEFANET Journal*, 2(2):56–63, 2014.

[18] Marek Mateják, Tomáš Kulhánek, Jan Šilar, Pavol Privitzer, Filip Ježek, and Jiří Kofránek. Physiolibrary - Modelica Library for Physiology. In *Proceedings of the 10th International Modelica Conference, March 10-12, 2014, Lund, Sweden*, pages 499–505. Linköping University Electronic Press, 2014. `doi:10.3384/ecp14096499`.

[19] Marek Mateják. Physiology in modelica. *MEFANET Journal*, 2(1):10–14, 2014. URL: `http://mj.mefanet.cz/mj-03140307`.

[20] Anna Doležalová. *Design of a Model for ECMO Demonstration and Teaching*. PhD thesis, Czech Technical University, Prague, 2014. URL: `http://www.fel.cvut.cz/education/prace/00036.pdf`.