Mixed Quantitative and Qualitative Simulation in Modelica

François E. Cellier
ETH Zürich
Switzerland
FCellier@Inf.ETHZ.CH

Victorino Sanz
UNED Madrid
Spain
VSanz@DIA.UNED.ES

Abstract

This article introduces a new Modelica library, FIR-lib, developed for the mixed quantitative and qualitative simulation of physical systems. Qualitative sub-models are built using the *Fuzzy Inductive Reasoning (FIR)* paradigm.

Whereas Modelica has been designed for modeling physical systems from first principles, some systems do not lend themselves to this kind of modeling, either because they are too poorly understood (no meta-knowledge is available yet) or because they are so complex that capturing their behavior in a detailed fashion would be a hopeless undertaking.

Use of the new library is demonstrated by means of two examples, a simple hydraulic control system (a textbook example) and a model of the human cardiovascular system.

Keywords: fuzzy inductive reasoning; inductive modeling; qualitative modeling; mixed quantitative and qualitative simulation; cardiovascular system

1 Introduction

Modelica has been designed as an environment for modeling physical systems from first principles in an object-oriented fashion.

Yet, there exist systems that don't lend themselves easily to this type of modeling, either because the meta-laws governing their dynamic behavior are not fully understood, or because these systems are too complex to be described with complete details.

In both of these cases, we need a tool that can capture dynamic behavior *inductively*, i.e., from observations, rather than *deductively*, i.e., from first principles.

Typical tools that are used for such purposes include *artificial neural networks* and *fuzzy modelers*. In this paper, we propose the use of a fuzzy modeling approach called *Fuzzy Inductive Reasoning (FIR)* [2,5].

In FIR, observations of input/output behavior of an unknown system are fuzzified (discretized with fuzzy membership functions associated with each class). A fuzzy rule base of dynamic relations between inputs and outputs is then automatically synthesized. The fuzzy rule base constitutes the qualitative model of the system. It is subsequently used to infer qualitative behavior of the system in a qualitative simulation step. The qualitative simulation results, so-called episodes, are then defuzzified (quantified) to trajectory behavior using the information contained in the fuzzy membership functions.

We sometimes encounter systems that are partially understood, i.e., the meta-laws describing some of its subsystems are well-known, whereas those describing other subsystems are unknown or only incompletely known.

In such cases, it is useful to be able to simulate such systems using a mixed quantitative and qualitative simulation environment. A (synthetic) example model is shown in Fig.1.

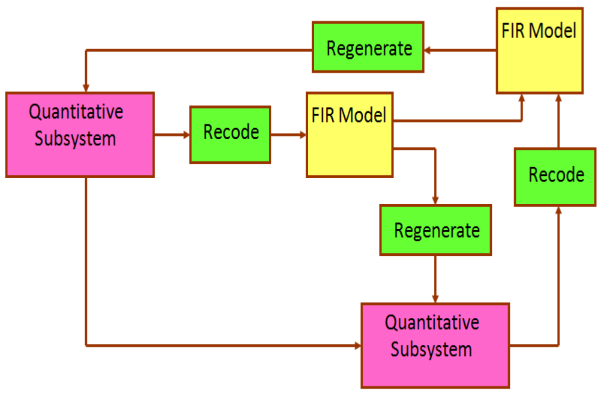


Fig.1: Mixed quantitative and qualitative model

The pink boxes of Fig.1 represent quantitative subsystems, whereas the yellow boxes represent qualitative subsystems. Quantitative signals can be converted to qualitative signals (i.e., fuzzified) using the green *Recode* block, whereas qualitative signals can be converted to quantitative signals (i.e., defuzzified) using the green *Regenerate* block.

2 Qualitative Variables

Qualitative variables are variables that assume qualitative values. Variables of a dynamical system are functions of time. The behavior of a dynamical system is a description of the values of its variables over time. The behavior of quantitative variables is usually referred to as trajectory behavior, whereas the behavior of qualitative variables is commonly referred to as episodical behavior. Qualitative simulation can thus be defined as the process of inferring the episodical behavior of a qualitative dynamical system or model.

Qualitative variables are frequently interpreted as an ordered set without distance measure [1]. It is correct that 'warm' is "larger" (warmer) than 'cold,' and that 'hot' is "larger" (warmer) than 'warm.' Yet, it is not true that:

'warm' – 'cold' = 'hot' – 'warm'

or, even more absurdly, that:

'hot' = 2 · 'warm' – 'cold'

No subtraction operator is defined for qualitative variables.

Whereas many qualitative simulation engines treat also the independent variable, *time*, as a qualitative variable, FIR does not. FIR simulates the behavior of qualitative states as functions of a quantitative *time* variable.

Without this feature, FIR would not be capable of dealing with mixed quantitative and qualitative models.

3 Fuzzy Inductive Reasoning

The Fuzzy Inductive Reasoning (FIR) methodology consists of four primary modules. The *Recode* module converts (fuzzifies) quantitative variables into qualitative variables; the *Optmask* module determines inductively a qualitative model relating sets of observations of input and output behavior; the *Forecast* module performs a qualitative simulation by inferring the episodical (qualitative) future behavior of a set of output variables given a set of input variables and a qualitative model; and finally the *Regenerate* module converts (defuzzifies) qualitative variables into quantitative variables.

3.1 Fuzzification

Recoding denotes the process of converting a quantitative variable to a qualitative variable. In general, some information is lost in the process of recoding. Obviously, a temperature value of 97^oF contains more information than the value 'hot.' Fuzzy recoding avoids this problem. Fig.2 shows the fuzzy recoding of a variable called "systolic blood pressure."

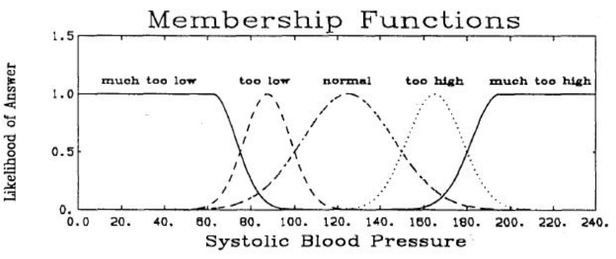


Fig.2: Fuzzy recoding

For example, a quantitative systolic blood pressure of 135.0 is recoded into a qualitative class value of 'normal' with a fuzzy membership value of 0.895, and a side value of 'right.' Thus, a single quantitative value is recoded into a qualitative triple. Any systolic blood pressure with a quantitative value between 100.0 and 150.0 will be recoded into the qualitative value 'normal.' The fuzzy membership function denotes the value of the bell-shaped curve shown on Fig.2, always a value between 0.5 and 1.0, and the side function indicates whether the quantitative value is to the left or to the right of the maximum of the currently active fuzzy membership function. Obviously, the qualitative triple contains the same information as the original quantitative variable. The quantitative value can be regenerated accurately from the qualitative triple, i.e., without any loss of information.

The shape of the fuzzy membership functions can be chosen either Gaussian or triangular, and the landmarks, i.e., the values of the variable to be recoded that separate neighboring classes from each other, can be either user-specified, or they can be determined by the FIR software itself using a variety of different approaches, such as the equal partitioning method [8], whereby the landmarks are chosen such that each class of the recoded variable contains the same number of samples.

3.2 Fuzzy Modeling

A qualitative model determines a relationship between the class values of a set of input variables and that of an output variable. FIR encodes the qualitative model using a so-called *optimal mask*.

A mask denotes a relationship between a set of variables. For example, let us consider the following raw data model consisting of five variables, namely two input variables, u_1 and u_2, and three output variables, y_1, y_2, and y_3, that are recorded at different instants of time:

$$\begin{array}{l} \textit{time} \\ 0.0 \\ \delta t \\ 2\cdot\delta t \\ 3\cdot\delta t \\ \vdots \\ (n_{\text{rec}}-1)\cdot\delta t \end{array} \quad \begin{array}{c} \begin{array}{ccccc} u_1 & u_2 & y_1 & y_2 & y_3 \end{array} \\ \begin{pmatrix} \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \ldots & \ldots & \ldots & \ldots & \ldots \end{pmatrix} \end{array}$$

Each column of the raw data model lists the class values of one qualitative variable recorded at different instants of time, and each row lists the class values of all qualitative variables recorded simultaneously. The raw data matrix is accompanied by a fuzzy membership matrix and a side matrix of identical dimensions.

A *mask* denotes a relationship between these variables. For example, the mask:

$$\begin{array}{c|ccccc} {}_t\backslash^x & u_1 & u_2 & y_1 & y_2 & y_3 \\ t-2\delta t & 0 & 0 & 0 & 0 & -1 \\ t-\delta t & 0 & -2 & -3 & 0 & 0 \\ t & -4 & 0 & +1 & 0 & 0 \end{array}$$

denotes the following relationship pertaining to the five variable system:

$$y_1(t) = f(y_3(t\text{-}2\delta t), u_2(t\text{-}\delta t), y_1(t\text{-}\delta t), u_1(t))$$

The single positive element in the mask, always located in the last row, denotes the position of the model output. The negative elements denote the positions of the model inputs. The example mask has four inputs. The sequence in which they are enumerated is immaterial. They are usually enumerated from left to right and top to bottom. Thus, the mask is simply a matrix representation of the qualitative relationship relating model inputs to the model output.

The mask must have the same number of columns as the raw data matrix. The number of rows of the mask is called the *depth* of the mask. The mask can be used to map a dynamic relationship onto a static relationship. To this end, the mask is shifted over the raw data matrix. Selected inputs and outputs can be read out from the raw data matrix and can be written on a single row next to each other. Fig.3 illustrates this process.

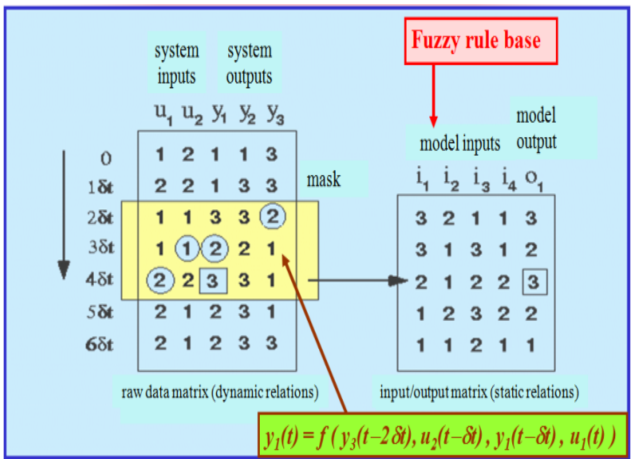

Fuzzy rule base
system inputs
system outputs
u1 u2 y1 y2 y3
model inputs
model output
i1 i2 i3 i4 o1
mask
raw data matrix (dynamic relations)
input/output matrix (static relations)
y1(t) = f (y3(t−2δt), u2(t−δt), y1(t−δt), u1(t))

Fig.3: Flattening dynamic relationships

After the mask has been applied to the raw data matrix, the formerly dynamic episodical behavior has become static, i.e., the relationship is now contained within a single row:

$$o_1(t) = f(i_1(t), i_2(t), i_3(t), i_4(t))$$

The resulting matrix is called *input/output matrix*. Each row of the input/output matrix represents a fuzzy rule.

How is the mask selected? A *mask candidate matrix* is constructed, in which negative elements denote positions of potential model inputs, and the single positive element denotes the position of the model output. A good mask candidate matrix for the aforementioned five variable system might be:

$$\begin{array}{c|ccccc} {}_t\backslash^x & u_1 & u_2 & y_1 & y_2 & y_3 \\ t-2\delta t & -1 & -1 & -1 & -1 & -1 \\ t-\delta t & -1 & -1 & -1 & -1 & -1 \\ t & -1 & -1 & +1 & 0 & 0 \end{array}$$

A mask candidate matrix is an ensemble of all acceptable masks. The optimal mask selection algorithm determines the best among all masks that are compatible with the mask candidate matrix. The mask shown before is one such mask. The optimal mask is the one mask that maximizes the forecasting power of the inductive reasoning process. To this end, the mask selection algorithm optimizes a mask quality metric that is a combination of a Shannon entropy reduction metric (making the input/output matrix as deterministic as possible) and an observation ratio metric (ensuring that most input/output patterns have been observed at least five times) [5].

3.3 Fuzzy Simulation

Once the optimal mask has been determined, it can be applied to the given raw data matrix resulting in a particular input/output matrix. Since the input/output matrix contains functional relationships within single rows, the rows of the input/output matrix can now be sorted in alphanumerical order. The result of this operation is called the *behavior matrix* of the system. The behavior matrix is a finite state machine. For each combination of input values, it shows, which output is most likely to be observed.

Forecasting (simulation) is now a straightforward procedure. The mask is simply shifted further down beyond the end of the raw data matrix, future inputs are read out from the mask, and the behavior matrix is used to determine the future output, which can then be copied back into the raw data matrix. In fuzzy forecasting, it is essential that, together with the qualitative output, also a fuzzy membership value and a side value are forecast. Thus, fuzzy forecasting predicts an entire qualitative triple, from which a quantitative variable can be regenerated whenever needed.

3.4 Defuzzification

Once the qualitative output episode has been determined, a quantitative trajectory can easily be constructed by the reverse operation of fuzzy recoding. The class, membership, and side values are simply recombined to produce a real-valued signal.

4 FIR Software

Originally, the FIR algorithms had been coded in Fortran and were made available as a CTRL-C library [6]. Mixed quantitative and qualitative simulations were performed in ACSL, which could invoke the Fortran routines of the *Recode*, *Forecast*, and *Regenerate* subroutines directly, i.e., the qualitative models were constructed in CTRL-C, but mixed simulations were run in ACSL [5].

When CTRL-C died, the algorithms were recoded in C, and CTRL-C was replaced by Matlab as the interactive matrix manipulation environment.

There are currently two separate Matlab toolboxes available implementing the FIR algorithms. SAPS-II [5] offers a command-driven interface. The user invokes the four blocks of the FIR methodology by writing m-files. Visual-FIR [9] offers a menu-driven interface. Here, the user doesn't write any code, but selects combinations of algorithms from a set of pull-down menus. SAPS-II is more general, but Visual-FIR is easier and faster to use.

Both toolboxes can be used to run purely qualitative simulations directly under Matlab. Yet, mixed quantitative and qualitative simulations cannot be run in this fashion. Thus when ACSL died, we lost our ability to run mixed simulations.

This is where the new FIRlib fits in. The software allows us to once again run mixed quantitative and qualitative simulations, replacing ACSL by Modelica.

Just like the former ACSL implementation, FIRlib currently offers *Recode*, *Forecast*, and *Regenerate* modules only. There is no need to offer an *Optmask* module in the software, as the qualitative model is being generated off-line. Hence also with FIRlib, the qualitative models are being created using either SAPS-II or Visual-FIR. Future versions of FIRlib may offer an *Optmask* module also for convenience.

FIRlib offers currently two implementations of the FIR algorithms. In one of them (native SAPS), the formerly C-coded algorithms were translated into Modelica. The other (external SAPS) invokes C-coded routines.

For small examples, there is little difference between the two versions. Yet, the cardiovascular system model will not run efficiently in Dymola using the native SAPS modules. The FIR algorithms operate on very large data tables that Dymola converts to individual variables. Hence the cardiovascular system model, when using native SAPS, generates 200,000 scalar variables, whereas only 4000 variables are generated when the external SAPS modules are invoked.

5 A Simple Textbook Example

We shall demonstrate the use of FIRlib by means of a simple position control system involving a hydraulic motor with a servo-valve. The control system is shown in Fig.4.

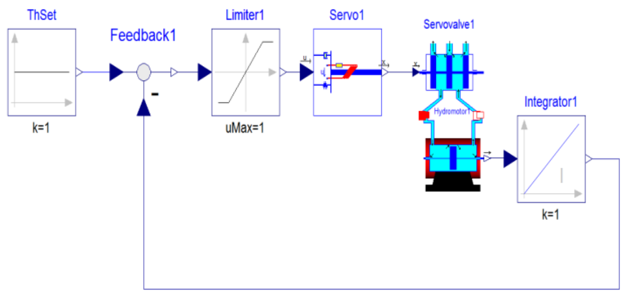


Fig.4: A position control system

The servo-valve and the hydraulic motor models were composed using the hydraulic sub-library of BondLib [3]. The hydraulic motor model is shown in Fig.5.

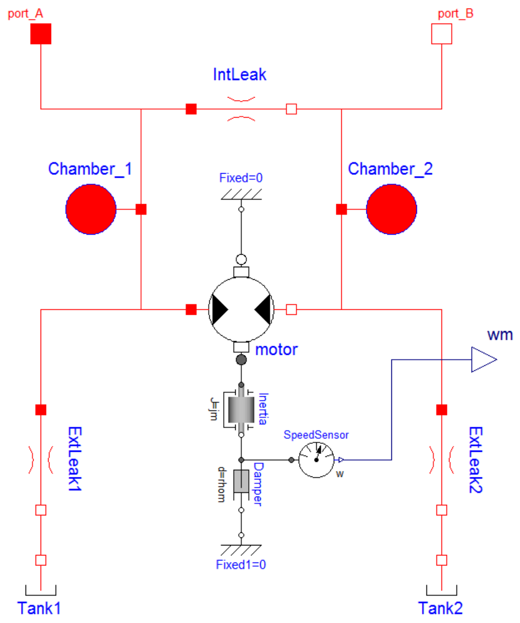


Fig.5: Hydraulic motor

We want to replace the entire hydraulic part of the model by a qualitative model. We assume that the mechanical torque, τ, of the hydraulic motor is a function of the actuator signal, u, and the angular velocity of the motor, ω.

$$\tau = f(u, \omega)$$

Therefore, we need to recode (fuzzify) these three signals. This is done in Modelica using FIRlib, as shown in Fig.6.

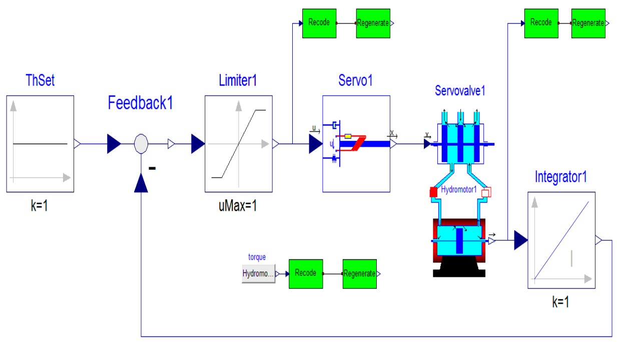


Fig.6: Fuzzification of three signals

The *Recode* block converts a real-valued signal to a qualitative triple. The FIR connector, at the output of the *Recode* block, contains three signals representing the class, membership, and side values of the fuzzified signal. The fuzzified signals were immediately defuzzified (regenerated) again so that Dymola can then be used to plot the signal and verify that fuzzification/defuzzification were done correctly. This is shown in Fig.7.

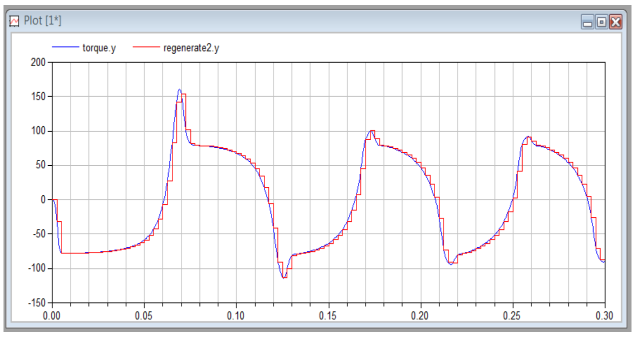


Fig.7: Torque signal, original and regenerated

The three recoded signals were then exported to Matlab. In Matlab, the raw data matrix (or rather, the three matrices containing the class, membership, and side values) was constructed, and the SAPS-II toolbox was used to generate the optimal mask and, from it, the input/output matrix and the behavior matrix.

The optimal mask and the corresponding behavior matrix (actually three matrices) were then re-imported into Dymola to be used in a mixed quantitative and qualitative simulation. The mixed model is shown in Fig.8.

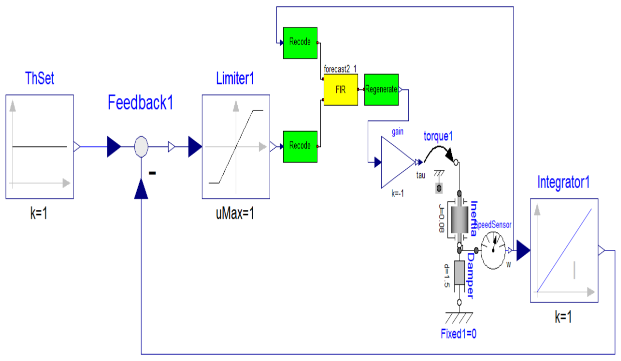


Fig.8: Mixed quantitative and qualitative model

The yellow *FIR* block represents the qualitative simulation (forecasting) engine. It takes the recoded (fuzzified) actuator and angular velocity signals and, using table look-up and interpolation in the behavior matrix, estimates the correct value of the torque in the form of a qualitative signal. The green Regenerate block then converts the qualitative triple back to a real-valued quantitative signal that can be used by regular Modelica models.

Fig.9 shows the motor angle trajectories of the original purely quantitative simulation and the mixed quantitative and qualitative simulation.

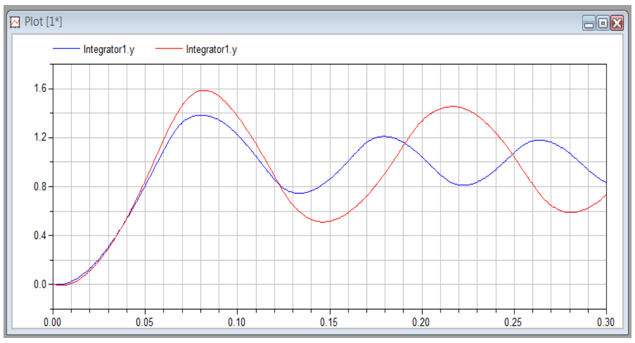


Fig.9: Motor angle trajectories

The (red) motor angle trajectory computed by the mixed simulation is a little more sluggish and a little less stable than that of the purely quantitative simulation (blue), because we didn't sample fast enough.

6 The Cardiovascular System

The human cardiovascular system is composed of two parts.

The hemodynamics describe the blood flow through the heart and the blood vessels. This part is well understood. It functions like any other hydraulic system with a pump and some pipes, with valves and containers of liquid. The hemodynamics can thus be well described by differential equations, and consequently, a quantitative model of the hemodynamics is adequate.

On the other hand, we need to describe also the control signals that operate on the hemodynamics. Control signals determine how fast the heart beats, how much the chambers contract, etc. The functioning of these nervous control signals is less well understood, and consequently, a qualitative model of the central nervous system control of the cardiovascular system may be more suitable.

6.1 Hemodynamics

The hemodynamics model has been presented at a previous conference [4]. It is built in BondLib using encapsulated bond graphs [3,7]. Although the bond graphs themselves are only seen at the bottom layer of the hierarchy, whereas all higher layers are built using symbols that medical professionals understand, the connectors are bond graph connectors everywhere. In order for this to work, all container models end in junctions, whereas all transporter models end in bonds. In this way, by following the rule that container and transporter models must always toggle, there is no need to fully wrap [7] the bond graph models, as was done in the hydraulic sub-library.

The heart model is shown in Fig.10.

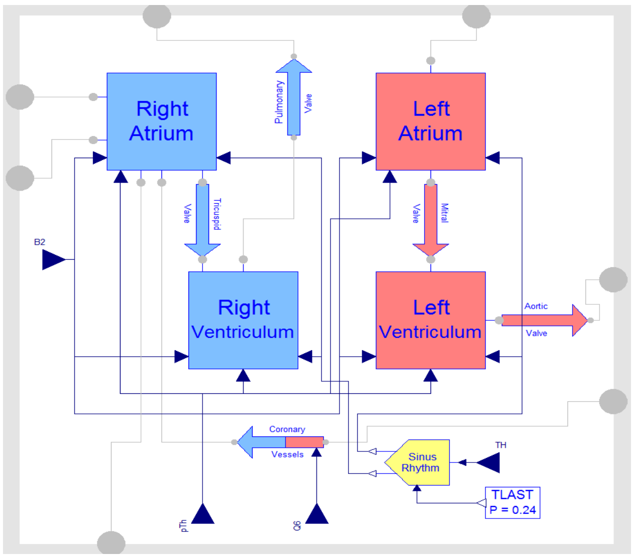


Fig.10: Model of the human heart

The model contains four container models representing the four heart chambers, as well as five transporter models. Four of them represent the four heart valves, the tricuspid and pulmonary valves carrying (blue) venous blood, and the mitral and aortic valves carrying (red) arterial blood. Also included is a model of the coronary blood vessels that are responsible for the oxygenation of the heart muscle. The yellow sinus rhythm block calculates the trigger impulses that lead to the contraction of the four heart chambers. It is controlled by the heart rate controller, one of the central nervous system control functions of the heart. The left chambers are shown on the right side of the graph, because this is what a heart surgeon experiences when he or she operates on a patient.

The heart is embedded in the thorax, shown in Fig.11.

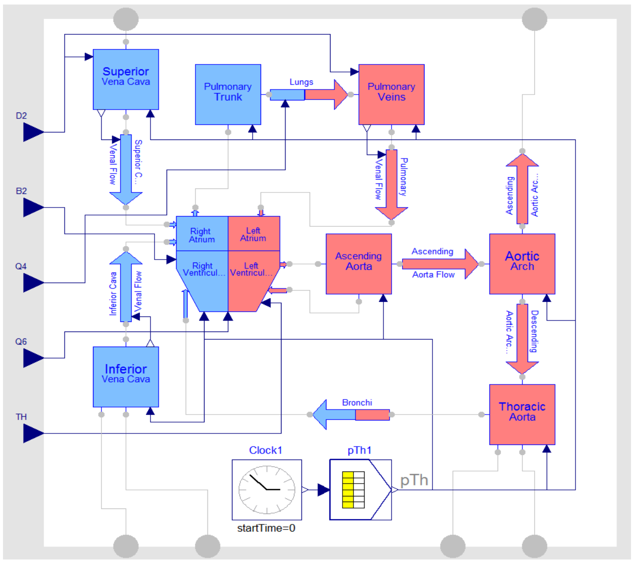


Fig. 11: Model of the thorax

The thorax model contains the heart and all of its external connections. Also included are models of the lungs and the bronchi. The tabular block at the bottom calculates the thoracic pressure that stems from the breathing. As the lungs expand, there is less space available for the heart and the blood vessels, and consequently, they experience an external pressure.

The overall hemodynamics model is shown in Fig.12.

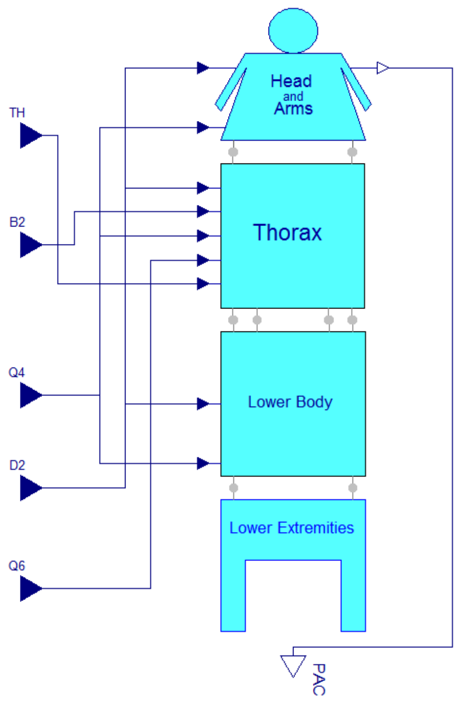


Fig.11: Model of the hemodynamics

The hemodynamics model contains models of the thorax, the head and arms (brain and blood vessels of the upper extremities), the lower body (abdomen and stomach), as well as the blood vessels of the legs.

The hemodynamics are controlled by five control signals denoting the heart rate, the myocardiac contractility, the peripheric resistance, the venous tone, and the coronary resistance.

It is assumed that all five control signals are functions of the same variable, namely the carotid sinus blood pressure, *PAC*, i.e., the arterial pressure in the brain.

6.2 Central Nervous System Control

Five separate single-input/single-output (SISO) qualitative FIR models are to be identified that each calculate one of the five control signals as a function of the carotid sinus blood pressure.

The data needed for the identification of the five FIR models are here not collected from a fully quantitative simulation (as in the previous case), but rather, they are obtained through measurements from real patients having a heart catheter for some reason or other (invasive procedure). The patients gave their consent to perform a number of so-called Valsalvæ maneuvers [10,11,12], a breathing test that excites the entire cardiovascular system. Data were collected from 10 different patients, each performing five Valsalvæ maneuvers. In the experiments described here, we used the data of one patient only. Four of the five Valsalvæ maneuvers (4800 data records) were used to identify the five controller models, and the final 1200 data records (the final maneuver) were used for model validation.

Fig.12 shows the recorded data of the venous tone controller signal of one patient during one Valsalvæ maneuver.

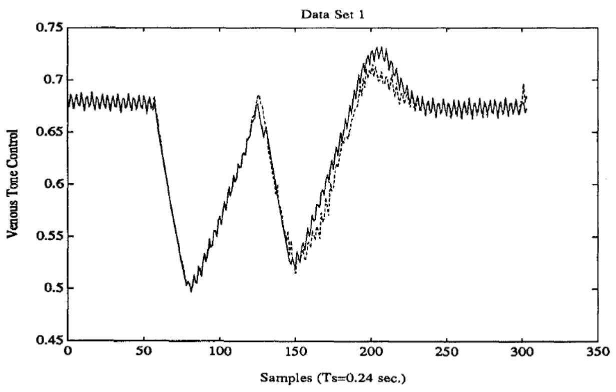


Fig.12: Venous tone controller signal

The large and low-frequency oscillation is caused by the breathing pattern of the Valsalvæ maneuver, whereas the superposed small and high-frequency oscillation is caused by the beating of the heart.

The Valsalvæ maneuver shown is the one that was not used for model identification. Superposed with the measurement data is the forecast obtained by the qualitative FIR model.

The five models were identified using the SAPS-II toolbox, and also the simulation was performed using the same Matlab toolbox. As this is a purely qualitative simulation, there was no need to perform the simulation in Modelica using FIRlib.

6.3 Mixed Quantitative and Qualitative Simulation of the Human Cardiovascular System

The top-level model is shown in Fig.13. The pink box on the right-hand side represents the hemodynamics model, whereas the green box on the left-hand side represents the central nervous system control containing the five (yellow) qualitative FIR models representing the five controllers.

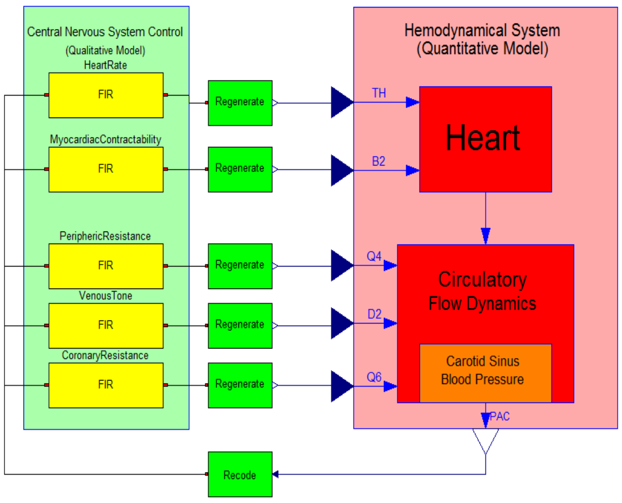
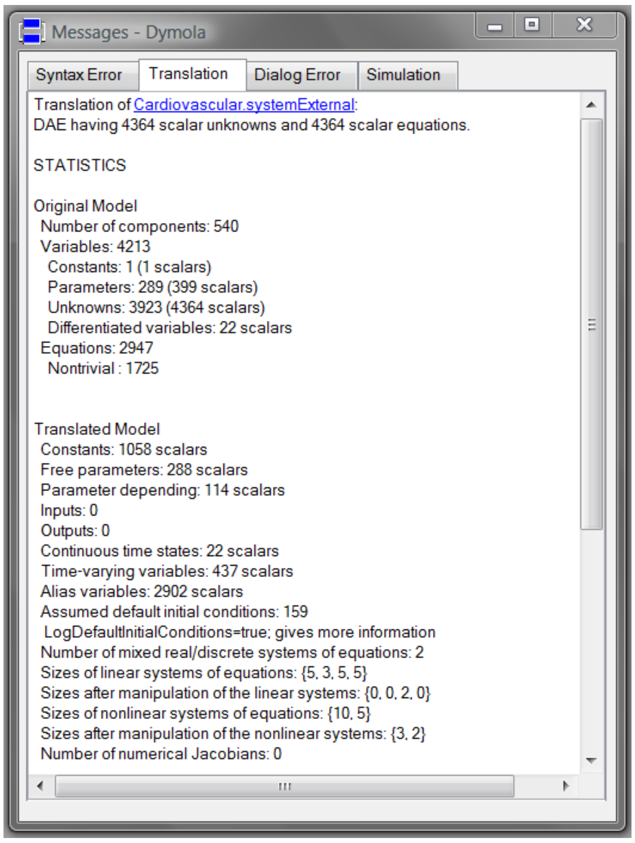


Fig.13: Cardiovascular system model

The green (Recode, Regenerate) and yellow (FIR) blocks are those of the external SAPS sub-library, i.e., the C-coded FIR algorithms are being used.

The model was then compiled. The translation log is shown in Fig.14.

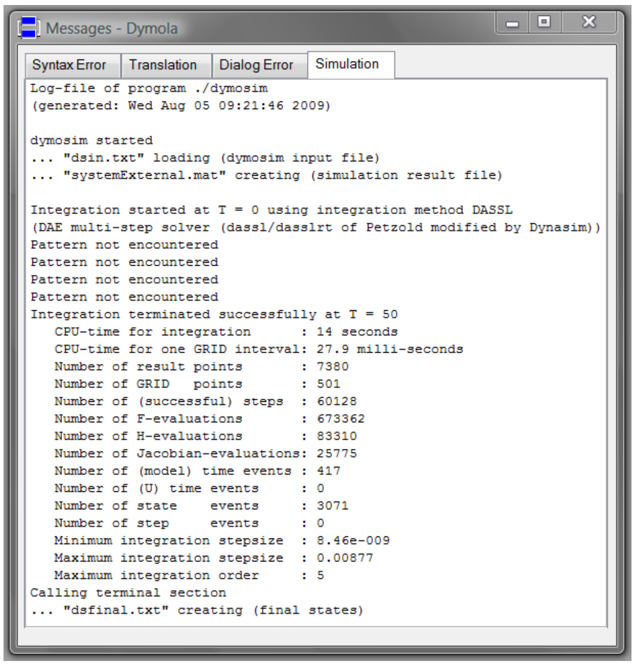


```
Translation of Cardiovascular.systemExternal:
DAE having 4364 scalar unknowns and 4364 scalar equations.

STATISTICS

Original Model
 Number of components: 540
 Variables: 4213
  Constants: 1 (1 scalars)
  Parameters: 289 (399 scalars)
  Unknowns: 3923 (4364 scalars)
  Differentiated variables: 22 scalars
 Equations: 2947
  Nontrivial : 1725


Translated Model
 Constants: 1058 scalars
 Free parameters: 288 scalars
 Parameter depending: 114 scalars
 Inputs: 0
 Outputs: 0
 Continuous time states: 22 scalars
 Time-varying variables: 437 scalars
 Alias variables: 2902 scalars
 Assumed default initial conditions: 159
  LogDefaultInitialConditions=true; gives more information
 Number of mixed real/discrete systems of equations: 2
 Sizes of linear systems of equations: {5, 3, 5, 5}
 Sizes after manipulation of the linear systems: {0, 0, 2, 0}
 Sizes of nonlinear systems of equations: {10, 5}
 Sizes after manipulation of the nonlinear systems: {3, 2}
 Number of numerical Jacobians: 0
```

Fig.14: Translation log

The flattened model contained originally 4364 scalar variables and equations. After code optimization, 22 state variables and 437 algebraic variables remained.

Fig.15 shows the simulation log.



```
Log-file of program ./dymosim
(generated: Wed Aug 05 09:21:46 2009)

dymosim started
... "dsin.txt" loading (dymosim input file)
... "systemExternal.mat" creating (simulation result file)

Integration started at T = 0 using integration method DASSL
(DAE multi-step solver (dassl/dasslrt of Petzold modified by Dynasim))
Pattern not encountered
Pattern not encountered
Pattern not encountered
Pattern not encountered
Integration terminated successfully at T = 50
   CPU-time for integration      : 14 seconds
   CPU-time for one GRID interval: 27.9 milli-seconds
   Number of result points       : 7380
   Number of GRID   points       : 501
   Number of (successful) steps  : 60128
   Number of F-evaluations       : 673362
   Number of H-evaluations       : 83310
   Number of Jacobian-evaluations: 25775
   Number of (model) time events : 417
   Number of (U) time events     : 0
   Number of state    events     : 3071
   Number of step     events     : 0
   Minimum integration stepsize  : 8.46e-009
   Maximum integration stepsize  : 0.00877
   Maximum integration order     : 5
Calling terminal section
... "dsfinal.txt" creating (final states)
```

Fig.15: Simulation log

The simulation took 14.0 seconds of real time to complete 50 seconds of simulated time. Four times during the simulation, one of the controllers encountered a pattern that had not been recorded in the training database. In those cases, no prediction was possible, and therefore, the software simply retained the previous prediction value.

Fig.16 shows the thoracic pressure, *pTh*, generated by a table look-up function inside the thorax model.

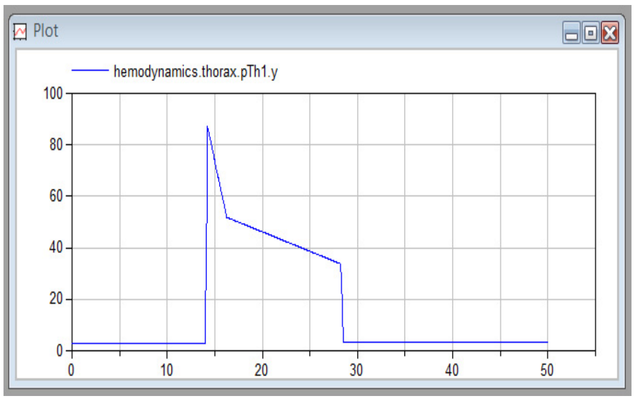


Fig.16: Thoracic pressure

The graph shows the simulated "Valsalvæ" maneuver. The "patient" is not breathing during 14 seconds, then "he" inhales sharply, holds "his" breath more or less for another 14 seconds, then exhales sharply again.

The resulting carotid sinus pressure, *PAC*, as calculated by the hemodynamics model, is shown in Fig.17.

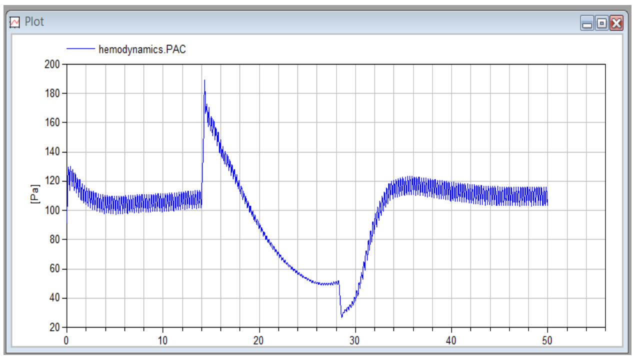


Fig.17: Carotid sinus pressure

The high-frequency oscillation is caused by the heartbeat, calculated in the sinus rhythm box of the heart model. The low-frequency oscillation is the hemodynamic response to the simulated breathing pattern.

Fig.18 shows the venous tone control signal as calculated by the corresponding FIR model in response to the simulated breathing pattern.

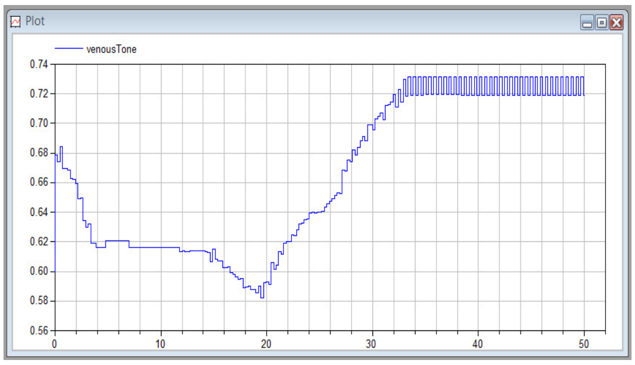


Fig.18: Venous tone control signal

7 Conclusions

In this paper, we have demonstrated how mixed qualitative and quantitative models can be simulated in Modelica using the new FIRlib library. The qualitative models make use of fuzzy inductive reasoning, a non-parametric inductive approach to modeling continuous-time systems by means of fuzzy logic. The approach was demonstrated by a small textbook example involving a hydraulic position control system. A model of the human cardiovascular system served as a larger example. In that model, the hemodynamics were described using quantitative models derived from first principle, whereas the nervous central system control functions were modeled by use of qualitative FIR models.

References

[1] Babbie, E.: *The Practice of Social Research*, 5th Edition, Wadsworth Publishing Company, Belmont, CA, 1989

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

[3] Cellier, F.E. and Nebot, A.: The Modelica Bond Graph Library, In: *Proc. 4th International Modelica Conference*, Hamburg, Germany (2005) Vol.1, 57-65

[4] Cellier, F.E. and Nebot, A.: Object-oriented Modeling in the Service of Medicine, In: *Proc. 6th Asia Simulation Conference*, Beijing, China (2005) 33-40

[5] Cellier, F.E., Nebot, A., Mugica, F., and de Albornoz, A.: Combined Qualitative/Quantitative Simulation Models of Continuous-time Processes Using Fuzzy Inductive Reasoning Techniques. In: *International Journal of General Systems* (1996) Vol. 24(1-2), 95-116

[6] Cellier, F.E. and Yandell, D.W.: SAPS-II: A New Implementation of the Systems Approach Problem Solver, In: *Intl. J. General Systems* (1987) Vol. 13(4), 307-322

[7] Cellier, F.E. and Zimmer, D.: Wrapping Multi-bond Graphs: A Structured Approach to Modeling Complex Multi-body Dynamics, In: *Proc. 20th European Conference on Modeling and Simulation*, Bonn, Germany (2006) 7-13

[8] Escobet, A., Huber, R.M., Nebot, A., and Cellier F.E.: Enhanced Equal Frequency Partition Method for the Identification of a Water Demand System, In: *Proc. AI, Simulation and Planning in High Autonomy Systems*, Tucson, Arizona (2000) 209-215.

[9] Escobet, A., Nebot, A., and Cellier, F.E.: Visual-FIR: A Tool for Model Identification and Prediction of Dynamical Complex Systems, In: *Simulation Modeling Practices and Theory* (2008) Vol. 16(1), 76-92

[10] Nebot, A.: *Qualitative Modeling and Simulation of Biomedical Systems Using Fuzzy Inductive Reasoning*, Ph.D. Dissertation, Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain, 1994

[11] Nebot, A., Cellier, F.E., and Vallverdú, M.: Mixed Quantitative/Qualitative Modeling and Simulation of the Cardiovascular System, In: *Computer Methods and Programs in Biomedicine* (1998) Vol. 55(2), 127-155

[12] Vallverdú, M.: *Modelado y Simulación del Sistema de Control Cardiovascular en Pacientes con Lesiones Coronarias*, Ph.D. Dissertation, Institut de Cibernètica, Universitat Politècnica de Catalunya, Barcelona, Spain, 1993

Author Biographies

François E. Cellier received his BS degree in electrical engineering in 1972, his MS degree in automatic control in 1973, and his PhD degree in technical sciences in 1979, all from the Swiss Federal Institute of Technology (ETH) Zurich. Dr. Cellier worked at the University of Arizona as professor of Electrical and Computer Engineering from 1984 until 2005. He then returned to his home country of Switzerland. Dr. Cellier's main scientific interests concern modeling and simulation methodologies, and the design of advanced software systems for simulation, computer aided modeling, and computer-aided design. Dr. Cellier has authored or co-authored more than 200 technical publications, and he has edited several books. He published a textbook on Continuous System Modeling in 1991 and a second textbook on Continuous System Simulation in 2006, both with Springer-Verlag, New York.

Victorino Sanz received his MS degree in computer science in 2004 from the Universidad Politécnica de Madrid. He is currently a Ph.D. student in systems engineering and automatic control at the Universidad Nacional de Educación a Distancia (UNED) in Madrid. His research focuses on the development of several Modelica libraries for discrete-event system and hybrid system modeling and simulation.