Inductive Modeling

- In this lecture, we shall study yet more general techniques for identifying complex non-linear models from observations of input/output behavior.
- These techniques make an attempt at mimicking human capabilities of ***vicarious learning***, i.e., of learning from observation.
- These techniques should be perfectly ***general***, i.e., the algorithms ought to be capable of capturing an arbitrary functional relationship for the purpose of reproducing it faithfully.
- The techniques will also be totally ***unintelligent***, i.e., their capabilities of generalizing patterns from observations are almost non-existent.

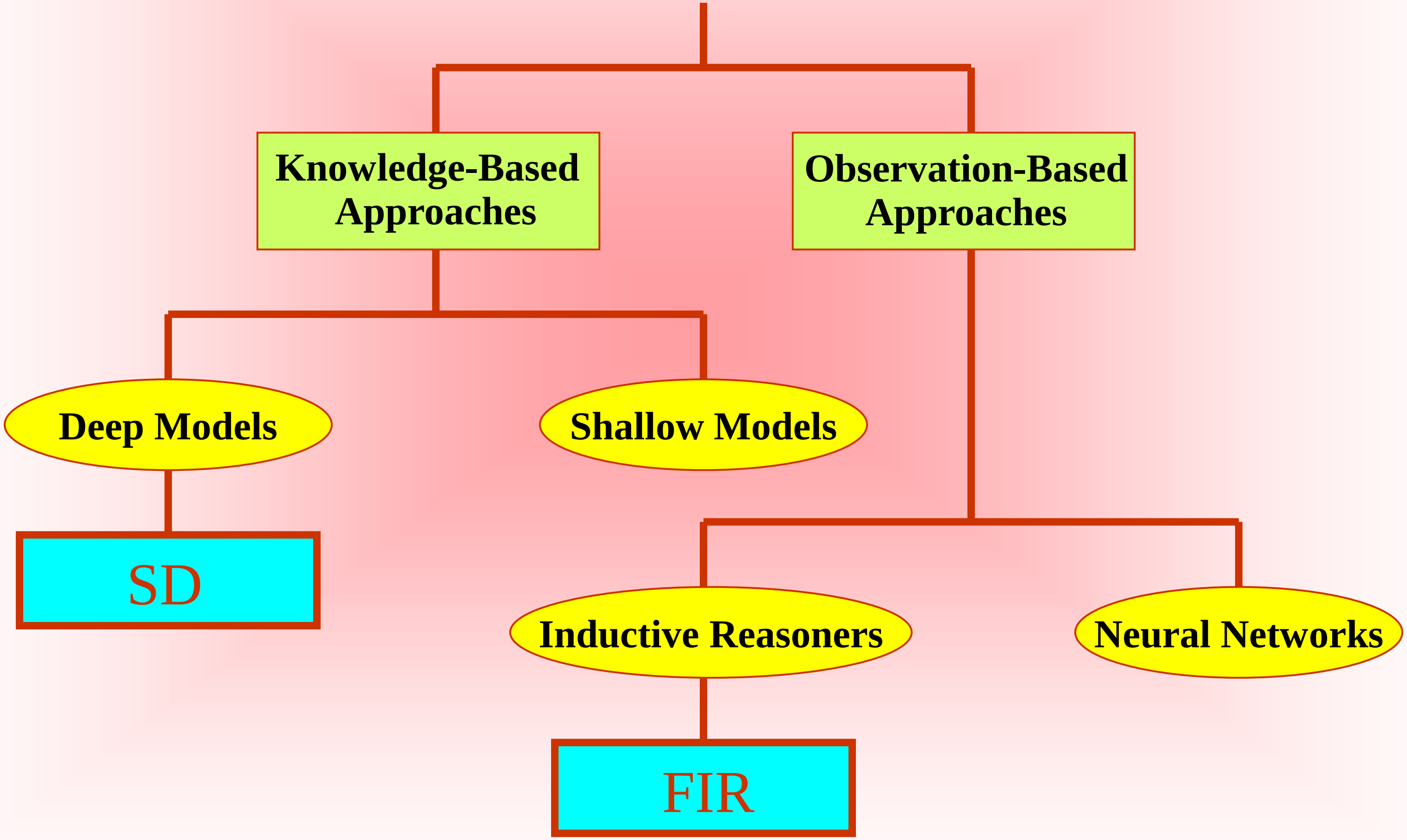
Table of Contents

- Knowledge-based vs. observation-based modeling
- Taxonomy of modeling methodologies
- Observation-based modeling and optimization
- Observation-based modeling and complexity
- Artificial neural networks
- Parametric vs. non-parametric modeling
- Quantitative vs. qualitative modeling
- Fuzzy modeling
- Fuzzy inductive reasoning
- Cardiovascular system

Knowledge-based *vs.* Observation-based Modeling

- Until now, we have almost exclusively embraced modeling techniques that were based on ***a priori* knowledge**.
- Only on a very few occasions have we created ***models from observations***.
- The one time that we really tried to do this, namely when we created a ***Lotka-Volterra model*** of the ***larch bud moth***, we were not overly successful in our endeavor.
- Yet, when we use ***a priori* knowledge**, such as when we model a resistor using the equation: $u = R \cdot i$, we are not really ***making models*** – we are only using models that had been made for us by someone else.

Taxonomy of Modeling Methodologies



Observation-based Modeling and Optimization

- Any ***observation-based modeling methodology*** is closely linked to ***optimization***.
- Let us look once more at our ***Lotka-Volterra models***:

$$\begin{aligned}\dot{P}_{pred} &= -a \cdot P_{pred} + k \cdot b \cdot P_{pred} \cdot P_{prey} \\ \dot{P}_{prey} &= c \cdot P_{prey} - b \cdot P_{pred} \cdot P_{prey}\end{aligned}$$

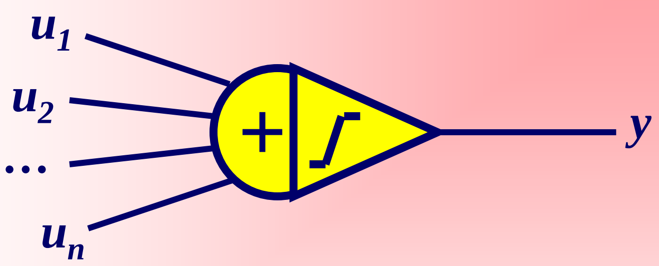
- When we used this structure to model the population dynamics of the ***larch bud moth***, we mapped the ***observational knowledge*** available onto the ***parameters*** of the ***Lotka-Volterra equations***, i.e., a, b, c, k, x_{prey_0}, and x_{pred_0}.
- ***Modeling*** here meant to ***identify these parameter values***, i.e., to ***minimize the error*** between the ***observed and simulated behavior*** by means of ***optimization***.

Observation-based Modeling and Complexity

- ***Observation-based modeling*** is very important, especially when dealing with ***unknown*** or only ***partially understood systems***. Whenever we deal with new topics, we really have no choice, but to model them ***inductively***, i.e., by using available observations.
- The less we know about a system, the more general a modeling technique we must embrace, in order to allow for all eventualities. If we know nothing, we must be prepared for anything.
- In order to model a totally unknown system, we must thus allow a ***model structure*** that can be ***arbitrarily complex***.

Artificial Neural Networks (ANN) I

- One very popular approach to model systems from observations is by use of ***artificial neural networks (ANNs)***.
- ***ANNs*** are modeled after the neurons of the human brain.



$$x = w' \cdot u + b$$
$$y = \text{activation}(x)$$

w' is the ***weight vector***

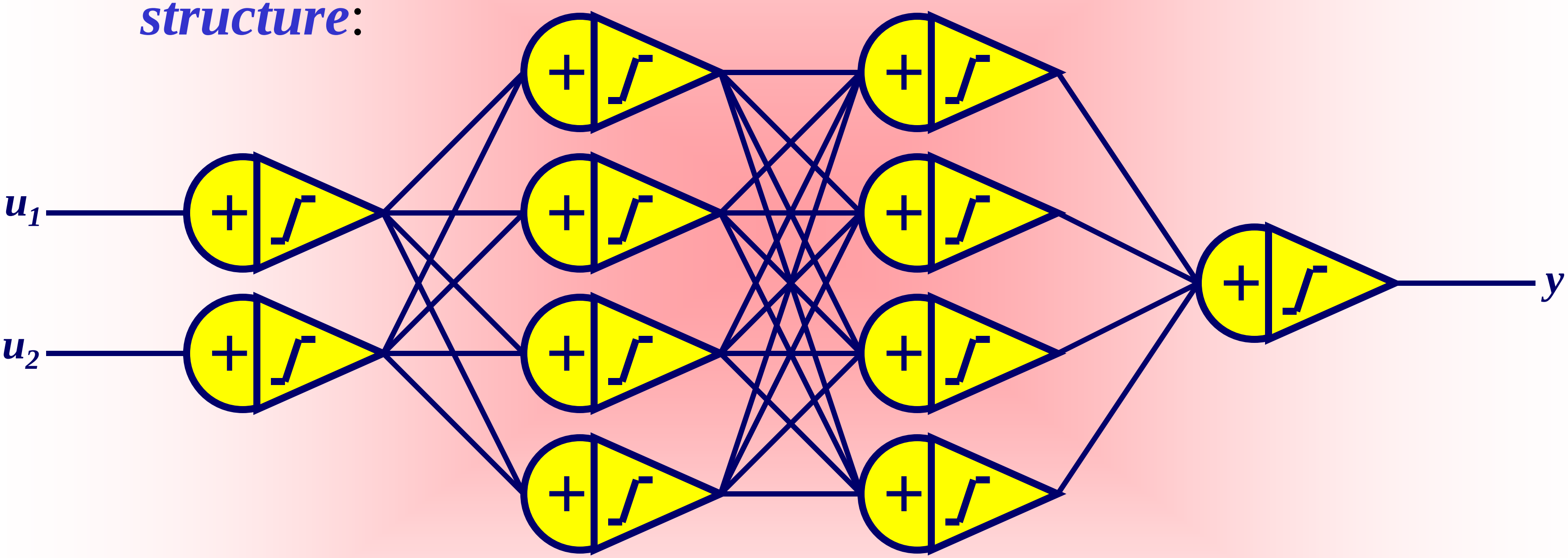
b is the ***bias***

activation() is a non-linear ***activation function***, usually shaped after the ***logistic equation***, e.g.:

$$y = \text{sigmoid}(x) = \frac{1.0}{1.0 + \exp(-x)}$$

Artificial Neural Networks (ANN) II

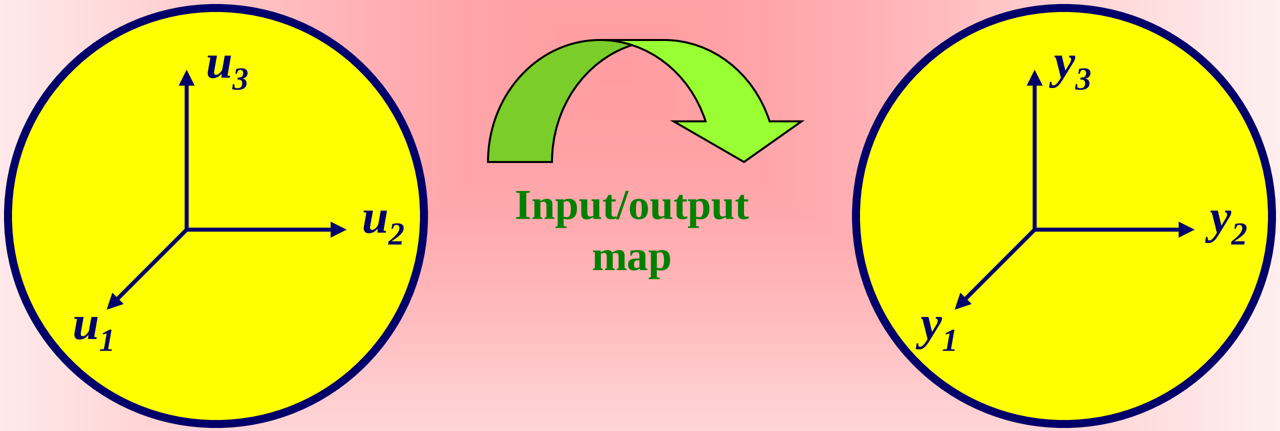
- Many such ***neurons*** are grouped together in a ***matrix structure***:



- The ***weight matrices*** and ***bias vectors*** between neighboring ***network layers*** store the information about the function to be modeled.

Artificial Neural Networks (ANN) III

- It can be shown that an ***ANN*** with a ***single hidden layer*** and enough neurons on it can learn ***any function*** with a compact domain of the input variables.



- With at least ***two hidden layers***, even arbitrary functions with holes in their input domains can be learnt.

Parametric *vs.* Non-parametric Models I

- ***ANNs*** are ***parametric models***. The observed knowledge about the system under study is mapped on the (potentially very large) set of parameters of the ***ANN***.
- Once the ***ANN*** has been ***trained***, the original knowledge is no longer used. Instead, the learnt behavior of the ***ANN*** is used to make predictions.
- This can be dangerous. If the ***testing data***, i.e. the input patterns during the use of the already trained ***ANN*** differ significantly from the ***training data*** set, the ***ANN*** is likely to predict garbage, but since the original knowledge is no longer in use, is unlikely to be aware of this problem.

Parametric *vs.* Non-parametric Models II

- ***Non-parametric models***, on the other hand, always refer back to the original training data, and therefore, can be made to reject testing data that are incompatible with the training data set.
- The ***Fuzzy Inductive Reasoning (FIR)*** engine that we shall discuss in this lecture, is of the ***non-parametric*** type.
- During the ***training*** phase, ***FIR*** organizes the observed patterns, and places them in a ***data base***.
- During the ***testing*** phase, ***FIR*** searches the data base for the five most similar training data patterns, the so-called ***five nearest neighbors***, by comparing the new input pattern with those stored in the data base. ***FIR*** then predicts the new output as a weighted average of the outputs of the five nearest neighbors.

Quantitative *vs.* Qualitative Models I

- ***Training a model*** (be it parametric or non-parametric) means ***solving an optimization problem***.
- In the ***parametric*** case, we have to solve a ***parameter identification*** problem.
- In the ***non-parametric*** case, we need to ***classify the training data***, and store them in an optimal fashion in the data base.
- Training such a model can be excruciatingly slow.
- Hence it may make sense to devise techniques that will help to speed up the training process.

Quantitative *vs.* Qualitative Models II

- How can the speed of the optimization be controlled? Somehow, the ***search space*** needs to be reduced.
- One way to accomplish this is to convert ***continuous variables*** to equivalent ***discrete variables*** prior to optimization.
- For example, if one of the variables to be looked at is the ambient temperature, we may consider to classify temperature values on a spectrum from ***very cold*** to ***extremely hot*** as one of the following discrete set:

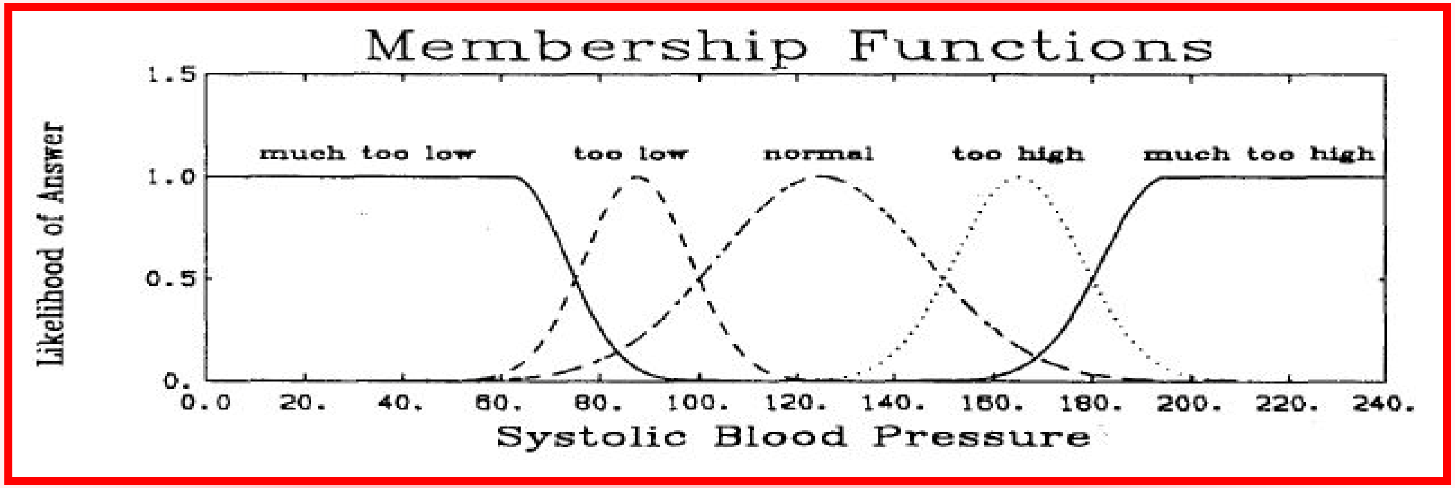
temperature = { freezing, cold, cool, moderate, warm, hot }

Qualitative Variables

- A variable that only assumes one among a set of discrete values is called a ***discrete variable***. Sometimes, it is also called a ***qualitative variable***.
- Evidently, it must be cheaper to search through a ***discrete search space*** than through a ***continuous search space***.
- The problem with ***discretization schemes***, such as the one proposed above, is that a lot of potentially valuable detailed information is being lost in the process.
- To avoid this pitfall, ***L. Zadeh*** proposed a different approach, called ***fuzzification***.

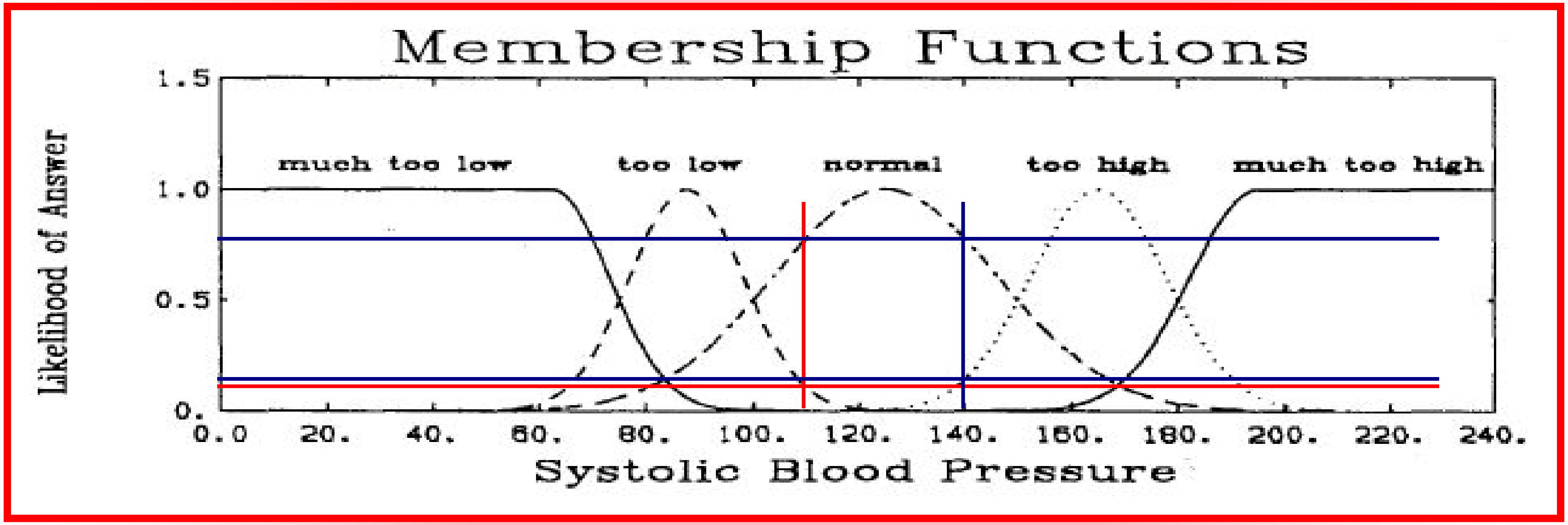
Fuzzy Variables I

- ***Fuzzification*** proceeds as follows. A continuous variable is ***fuzzified***, by decomposing it into a ***discrete class value*** and a ***fuzzy membership value***.



- For the purpose of ***reasoning***, only the ***class value*** is being considered. However, for the purpose of ***interpolation***, the ***fuzzy membership value*** is also taken into account.
- ***Fuzzy variables*** are not ***discrete***, but they are also referred to as ***qualitative***.

Fuzzy Variables II



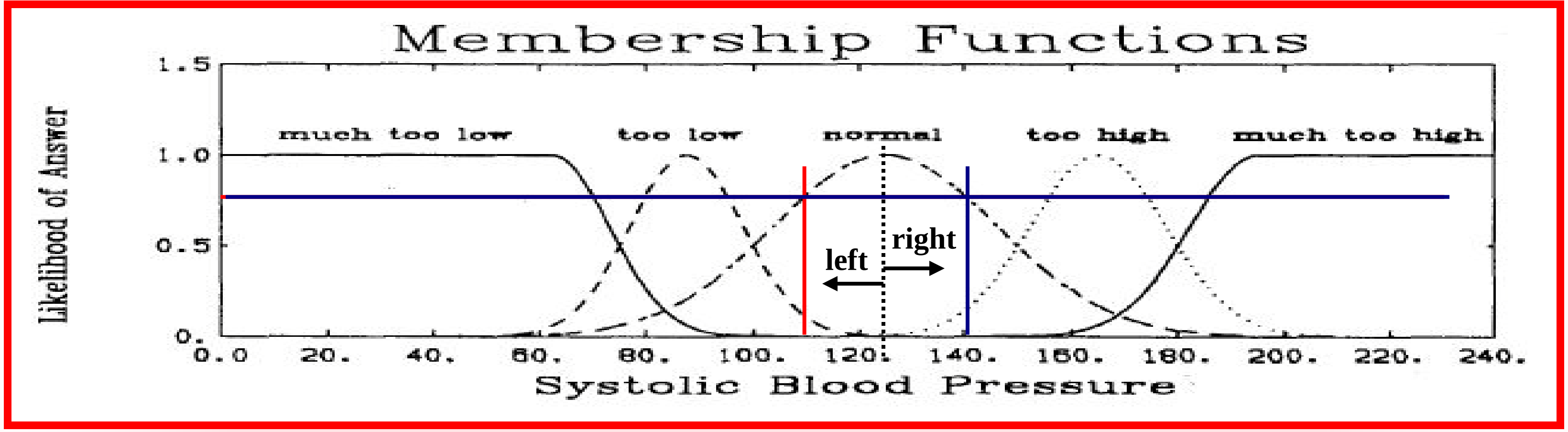
Systolic blood pressure = 110
⇒ { normal, 0.78 } ∩ { too low, 0.15 }

Systolic blood pressure = 141
⇒ { normal, 0.78 } ∩ { too high, 0.18 }

{ Class, membership } pairs of ***lower likelihood*** must be considered as well, because otherwise, the mapping would not be unique.

Fuzzy Variables in FIR

FIR embraces a slightly different approach to solving the uniqueness problem. Rather than mapping into ***multiple fuzzy rules***, FIR only maps into a ***single rule***, that with the largest likelihood. However, to avoid the aforementioned ***ambiguity problem***, FIR stores one more piece of information, the “***side value***.” It indicates, whether the data point is to the left or the right of the peak of the fuzzy membership value of the given class.



Systolic blood pressure = 110
⇒ { normal, 0.78, left }

Systolic blood pressure = 141
⇒ { normal, 0.78, right }

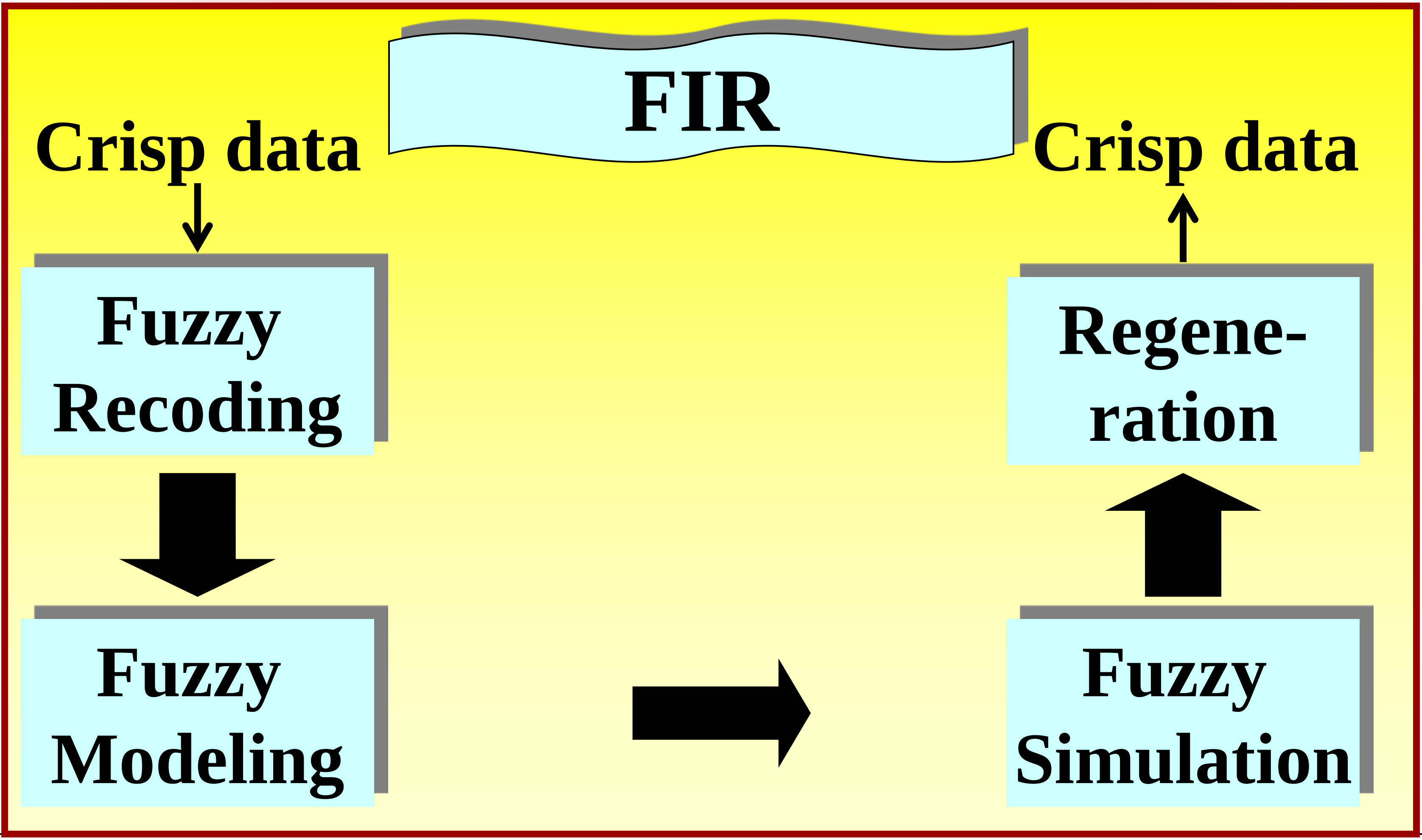
Neural Networks *vs.* Inductive Reasoners

Neural Networks	Fuzzy Inductive R.
Quantitative	Qualitative
Parametric	Non-parametric
Adaptive	Limited Adaptability
Slow Training	Fast Setup
Smooth Interpolation	Decent Interpolation
Wild Extrapolation	No Extrapolation
No Error Estimate	Error Estimate
Unsafe / Gullible	Robust / Self-critical

Fuzzy Inductive Reasoning (FIR) I

- Discretization of quantitative information *(Fuzzy Recoding)*
- Reasoning about discrete categories *(Qualitative Modeling)*
- Inferring consequences about categories *(Qualitative Simulation)*
- Interpolation between neighboring categories using fuzzy logic *(Fuzzy Regeneration)*

Fuzzy Inductive Reasoning (FIR) II



FIR

Crisp data

Fuzzy Recoding

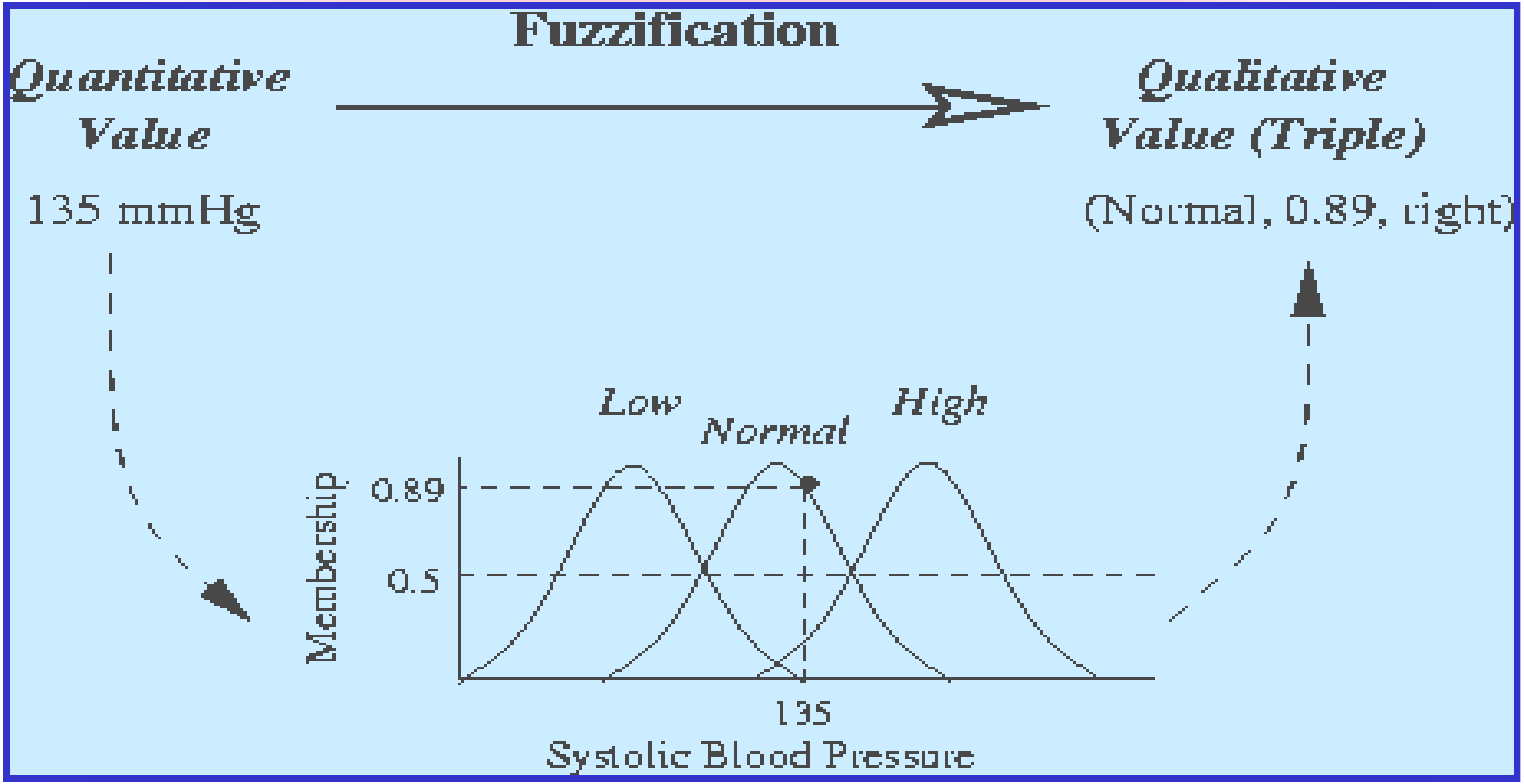
Fuzzy Modeling

Fuzzy Simulation

Regeneration

Crisp data

Fuzzification in FIR



Fuzzification

Quantitative Value

135 mmHg

Qualitative Value (Triple)

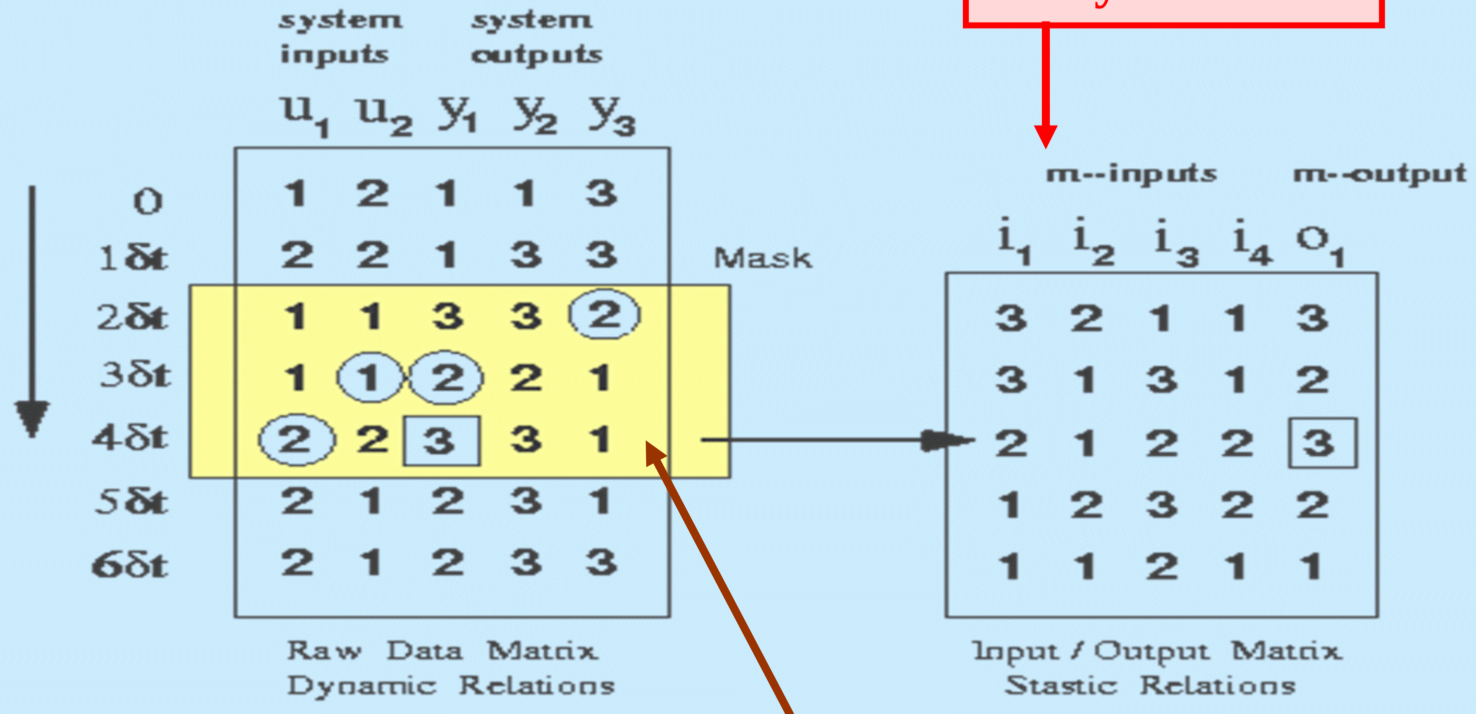
(Normal, 0.89, right)

Qualitative Modeling in FIR I

- Once the data have been recoded, we wish to determine, which among the possible set of input variables best represents the observed behavior.
- Of all possible input combinations, we pick the one that gives us as deterministic an input/output relationship as possible, i.e., when the same input pattern is observed multiple times among the training data, we wish to obtain output patterns that are as consistent as possible.
- Each input pattern should be observed at least five times.

Qualitative Modeling in FIR II

Fuzzy rule base



	u_1	u_2	y_1	y_2	y_3
0	1	2	1	1	3
$1\delta t$	2	2	1	3	3
$2\delta t$	1	1	3	3	2
$3\delta t$	1	1	2	2	1
$4\delta t$	2	2	3	3	1
$5\delta t$	2	1	2	3	1
$6\delta t$	2	1	2	3	3

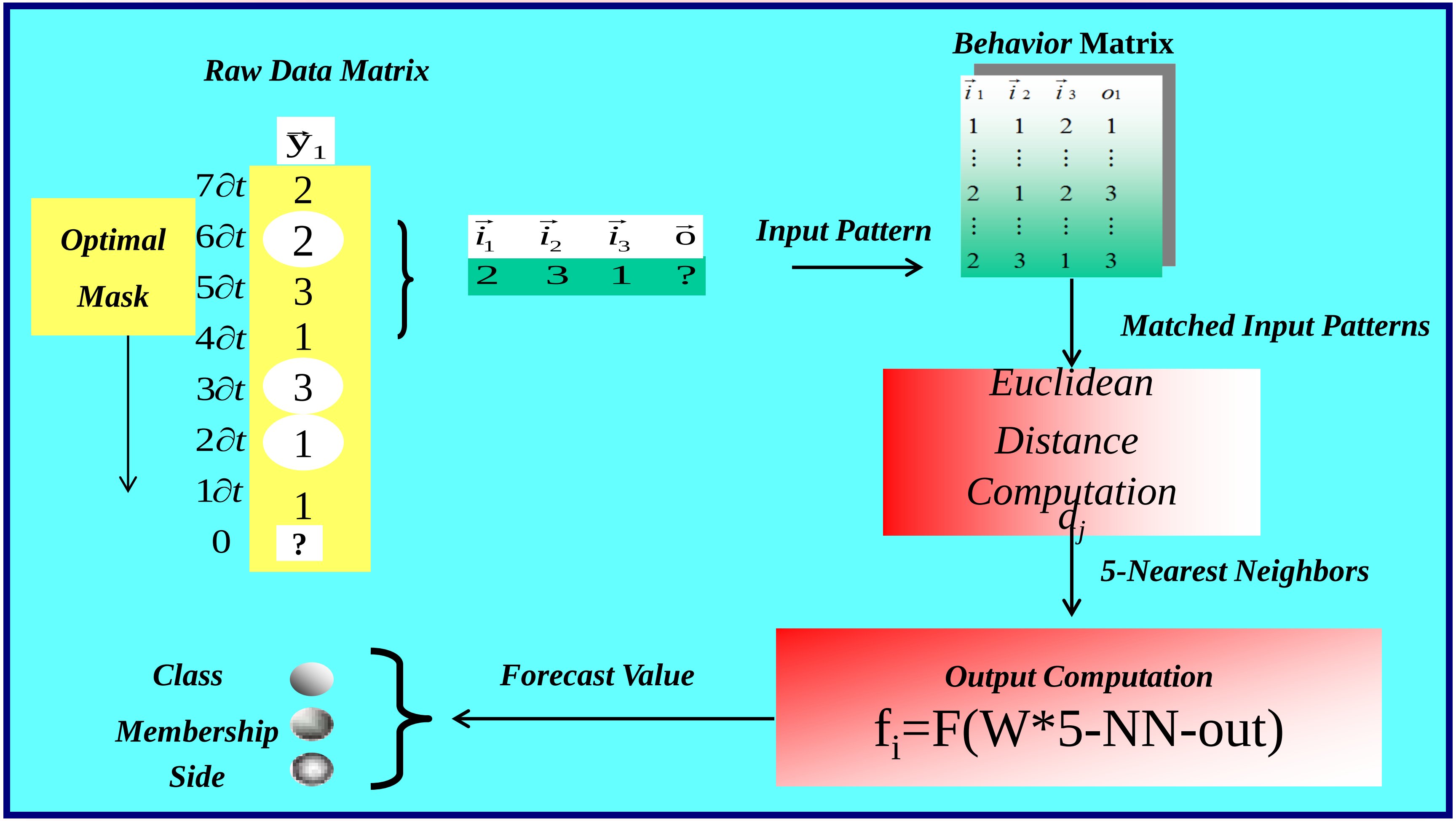
i_1	i_2	i_3	i_4	o_1
3	2	1	1	3
3	1	3	1	2
2	1	2	2	3
1	2	3	2	2
1	1	2	1	1

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

Qualitative Modeling in FIR III

- The ***qualitative model*** is the ***optimal mask***, i.e., the set of inputs that best predict a given output.
- Usually, the ***optimal mask*** is ***dynamic***, i.e., the current output depends both on current and past values of inputs and outputs.
- The optimal mask can then be applied to the training data to obtain a set of ***fuzzy rules*** that can be alphanumerically sorted.
- The ***fuzzy rule base*** is our ***training data base***.

Qualitative Simulation in FIR



Raw Data Matrix

	$\vec{y}_1$
$7\partial t$	2
$6\partial t$	2
$5\partial t$	3
$4\partial t$	1
$3\partial t$	3
$2\partial t$	1
$1\partial t$	1
0	?

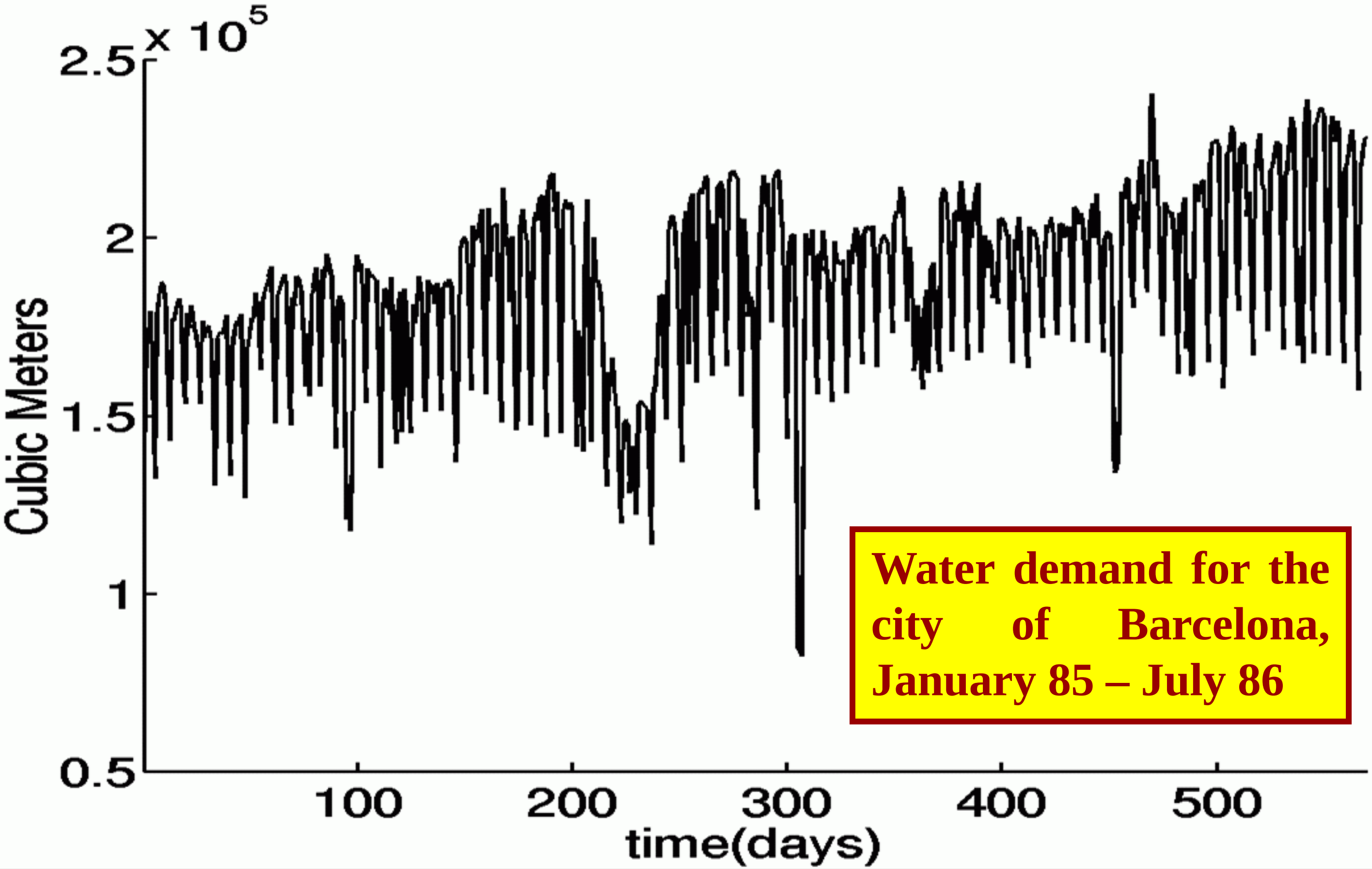
Input Pattern

$\vec{i}_1$	$\vec{i}_2$	$\vec{i}_3$	$\vec{o}$
2	3	1	?

Behavior Matrix

$\vec{i}_1$	$\vec{i}_2$	$\vec{i}_3$	o_1
1	1	2	1
⋮	⋮	⋮	⋮
2	1	2	3
⋮	⋮	⋮	⋮
2	3	1	3

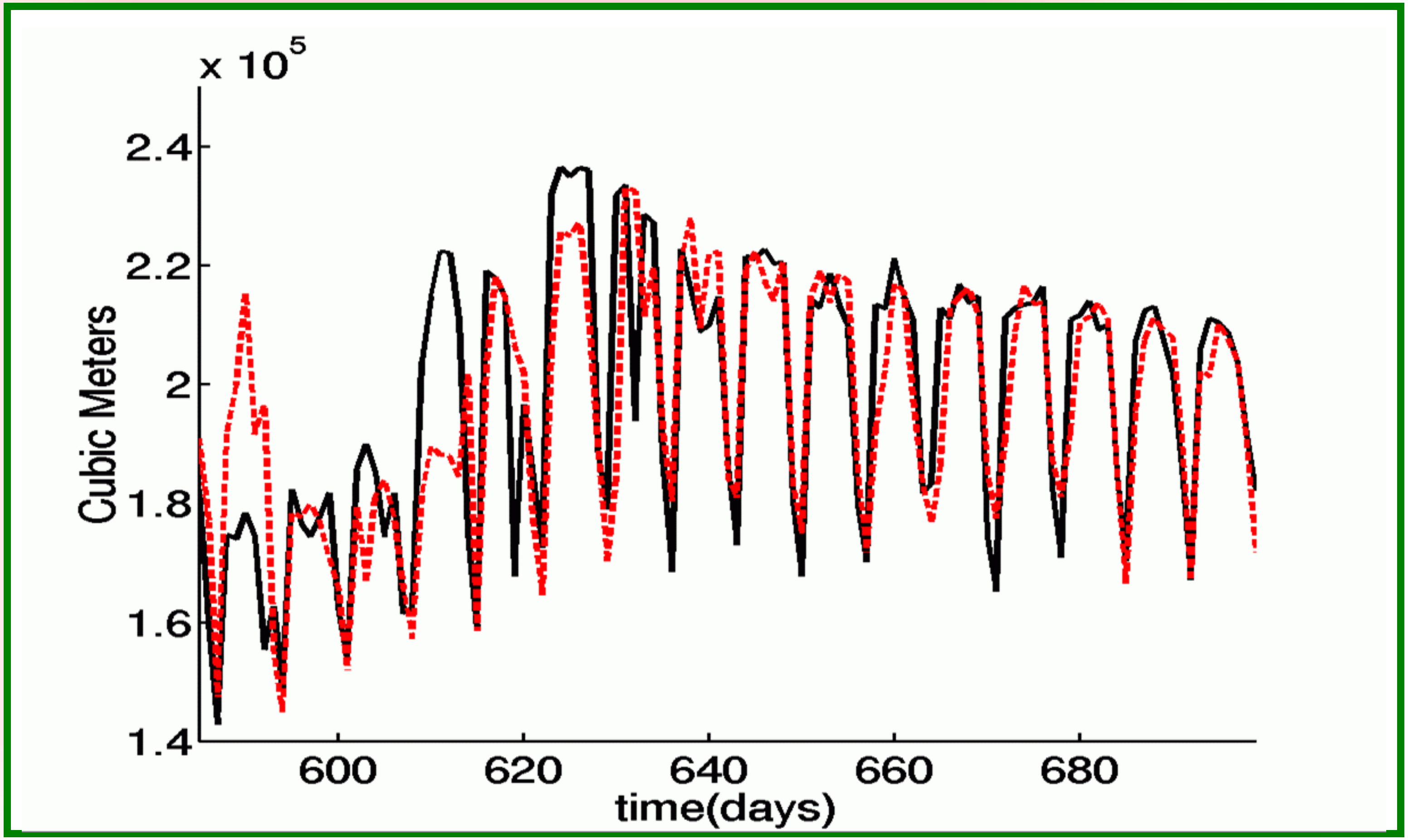
Time-series Prediction in FIR



Water demand for the city of Barcelona, January 85 – July 86

Simulation Results I

$$\begin{array}{l} \text{time} \\ t-14\partial t \\ \cdots \\ t-7\partial t \\ \cdots \\ t-\partial t \\ t \end{array} \begin{pmatrix} u_1 \\ -1 \\ 0 \\ -2 \\ 0 \\ -3 \\ +1 \end{pmatrix}$$

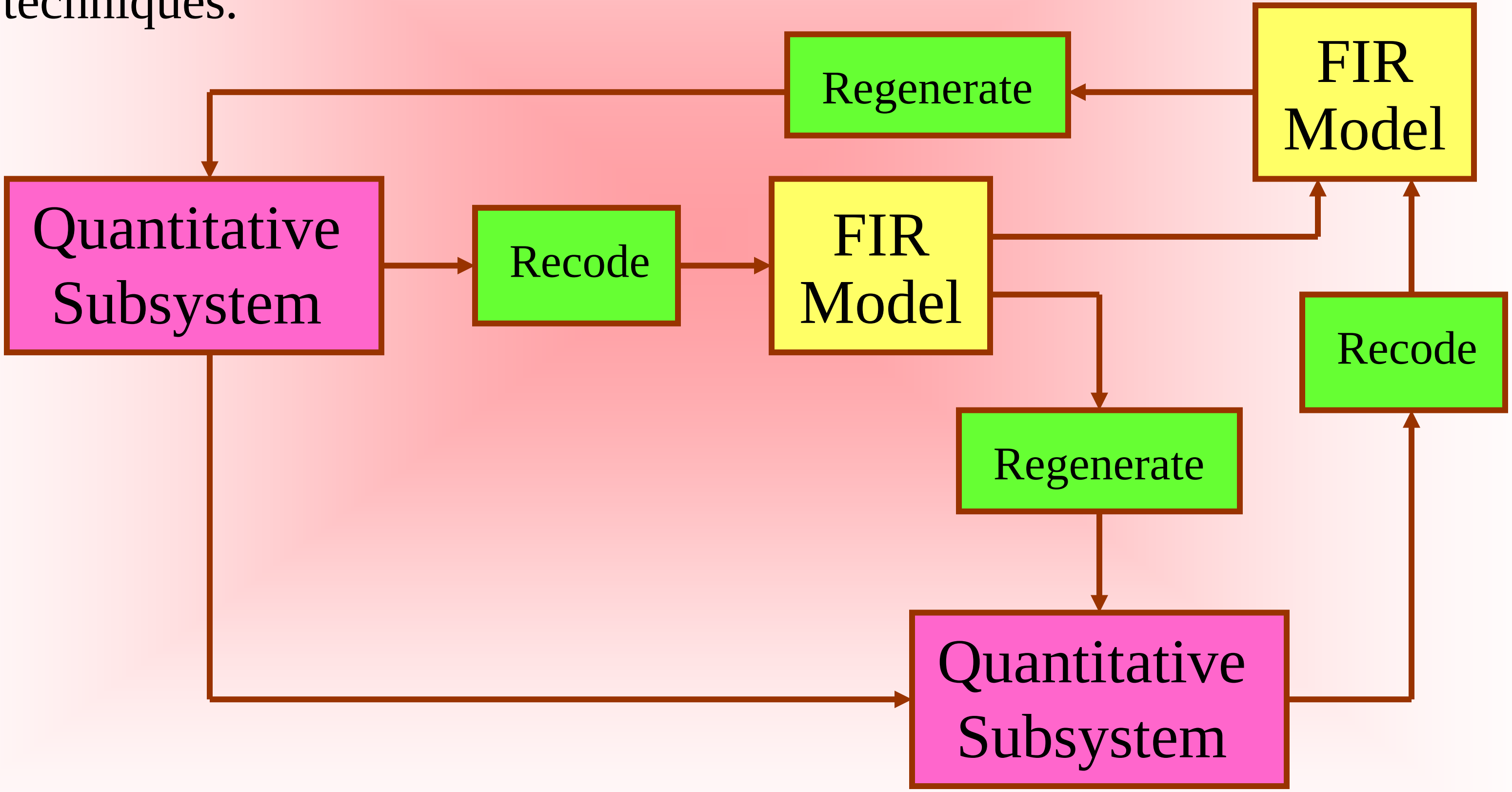


Quantitative *vs.* Qualitative Modeling

- ***Deductive Modeling Techniques***
  - * have a large degree of validity in many different and even previously unknown applications
  - * are often quite imprecise in their predictions due to inherent model inaccuracies
- ***Inductive Modeling Techniques***
  - * have a limited degree of validity and can only be applied to predicting behavior of systems that are essentially known
  - * are often amazingly precise in their predictions if applied carefully

Mixed Quantitative & Qualitative Modeling

- It is possible to combine qualitative and quantitative modeling techniques.



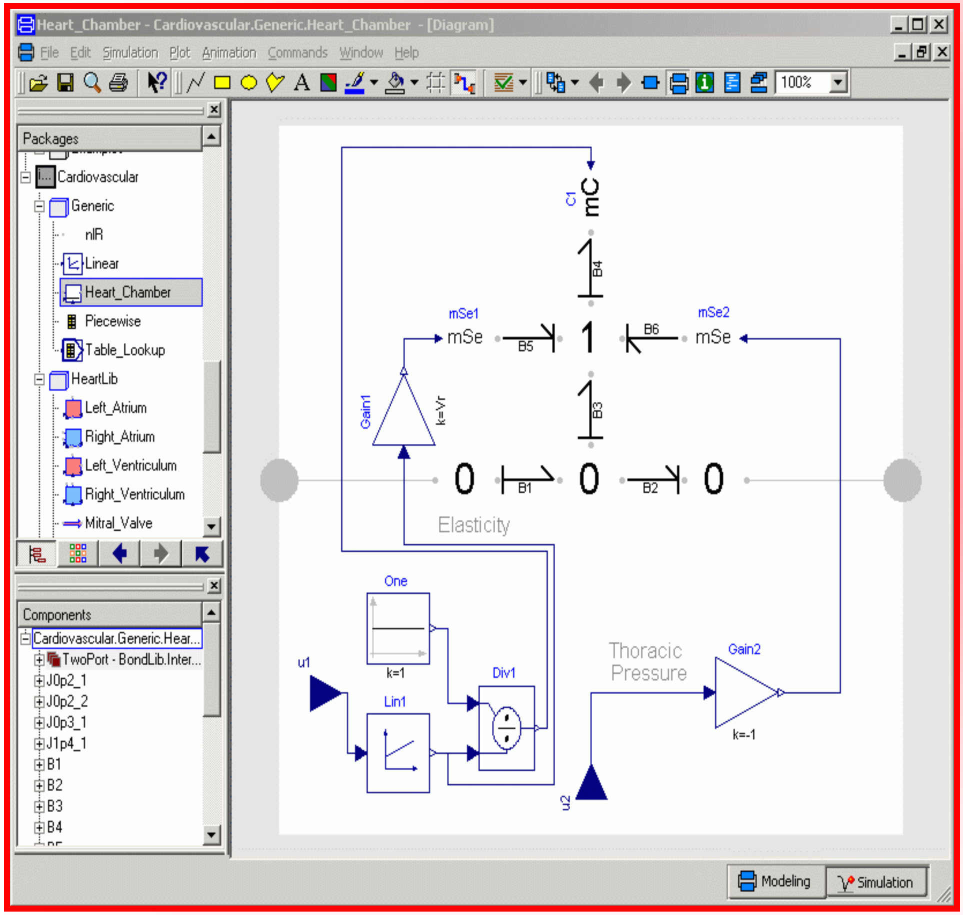
Application: Cardiovascular System I

- Let us apply the technique to a fairly complex system: the ***cardiovascular system*** of the human body.
- The cardiovascular system is comprised of two subsystems: the ***hemodynamic system*** and the ***central nervous control***.
- The ***hemodynamic system*** describes the flow of blood through the heart and the blood vessels.
- The ***central nervous control*** synchronizes the control algorithms that control the functioning of both the heart and the blood vessels.

Application: Cardiovascular System II

- The ***hemodynamic system*** is essentially a hydrodynamic system. The heart and blood vessels can be described by pumps and valves and pipes. Thus ***bond graphs*** are suitable for its description.
- The ***central nervous control*** is still not totally understood. ***Qualitative modeling*** on the basis of observations may be the tool of choice.

The Hemodynamic System I



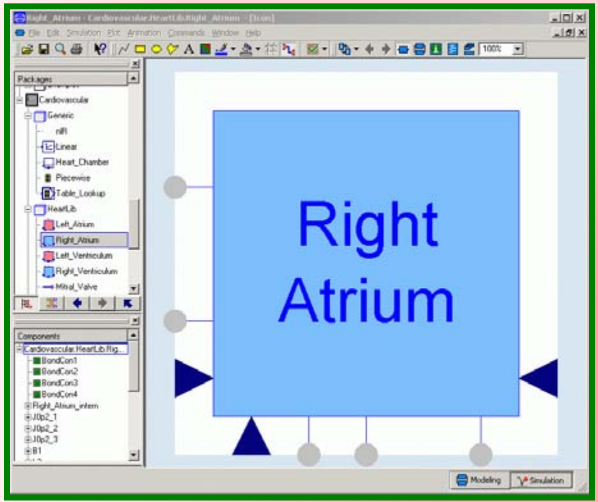
The ***heart chambers*** and ***blood vessels*** are containers of blood. Each container is a storage of mass, thus contains a ***C-element***.

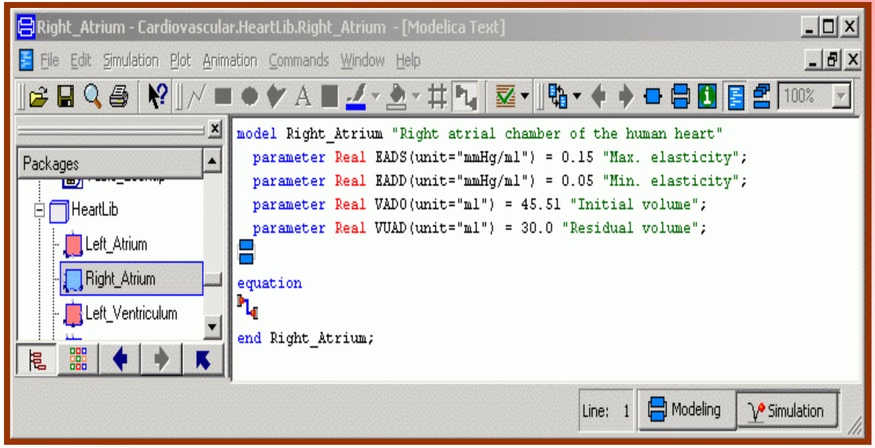
The ***C-elements*** are partly non-linear, and in the case of the heart chambers even time-dependent.

The ***mSe-element*** on the left side represents the time-varying pressure caused by the contracting heart.

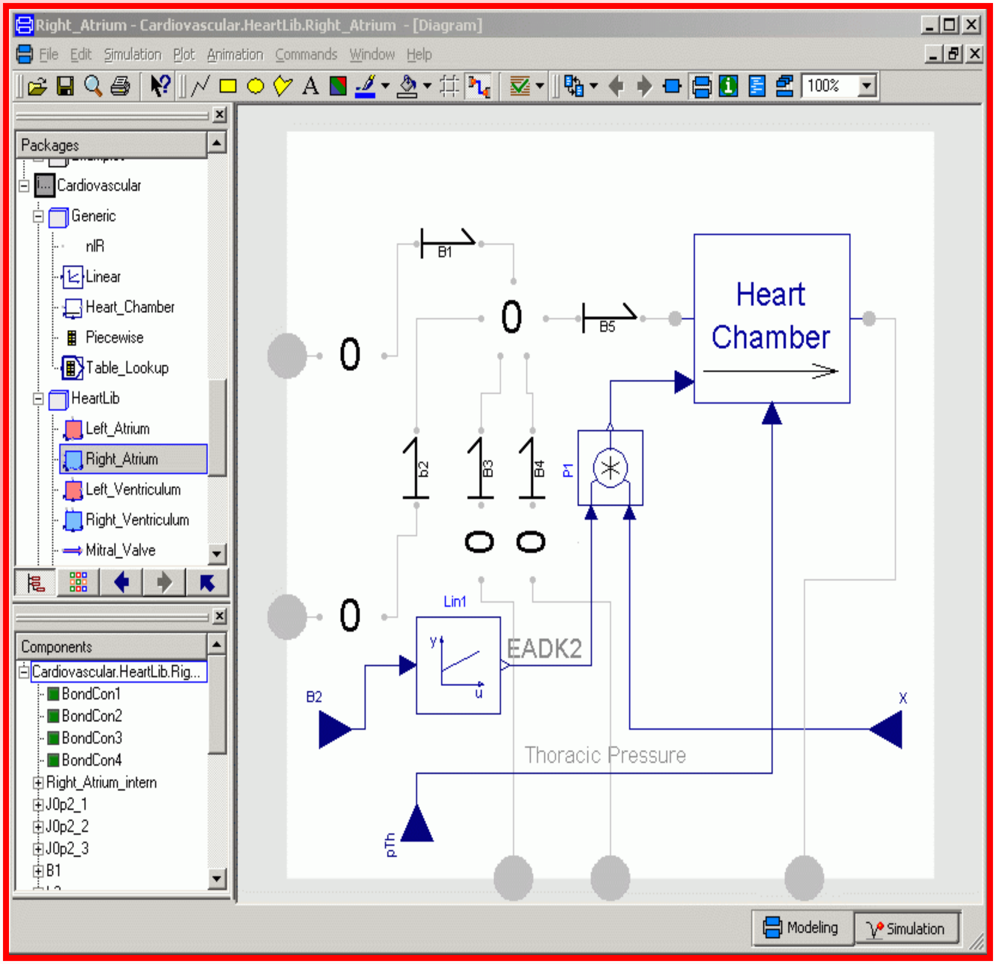
The ***mSe-element*** on the right side represents the thoracic pressure, which is influenced by the breathing.

The Hemodynamic System II

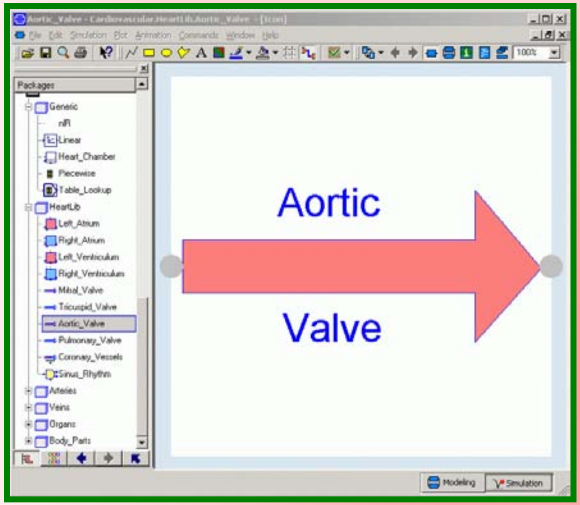


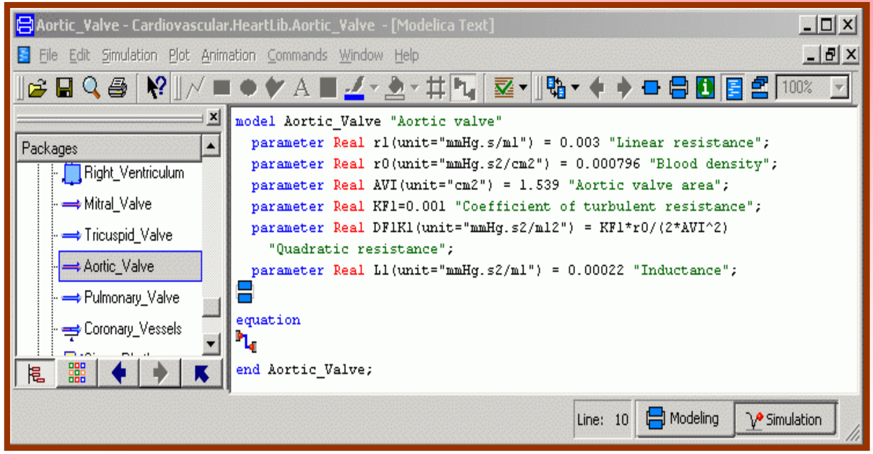

Right_Atrium - Cardiovascular.HeartLib.Right_Atrium - [Modelica Text]
File Edit Simulation Plot Animation Commands Window Help
100%
Packages
HeartLib
Left_Atrium
Right_Atrium
Left_Ventriculum
Line: 1
Modeling
Simulation

```
model Right_Atrium "Right atrial chamber of the human heart"
  parameter Real EADS(unit="mmHg/ml") = 0.15 "Max. elasticity";
  parameter Real EADD(unit="mmHg/ml") = 0.05 "Min. elasticity";
  parameter Real VAD0(unit="ml") = 45.51 "Initial volume";
  parameter Real VUAD(unit="ml") = 30.0 "Residual volume";

equation

end Right_Atrium;
```

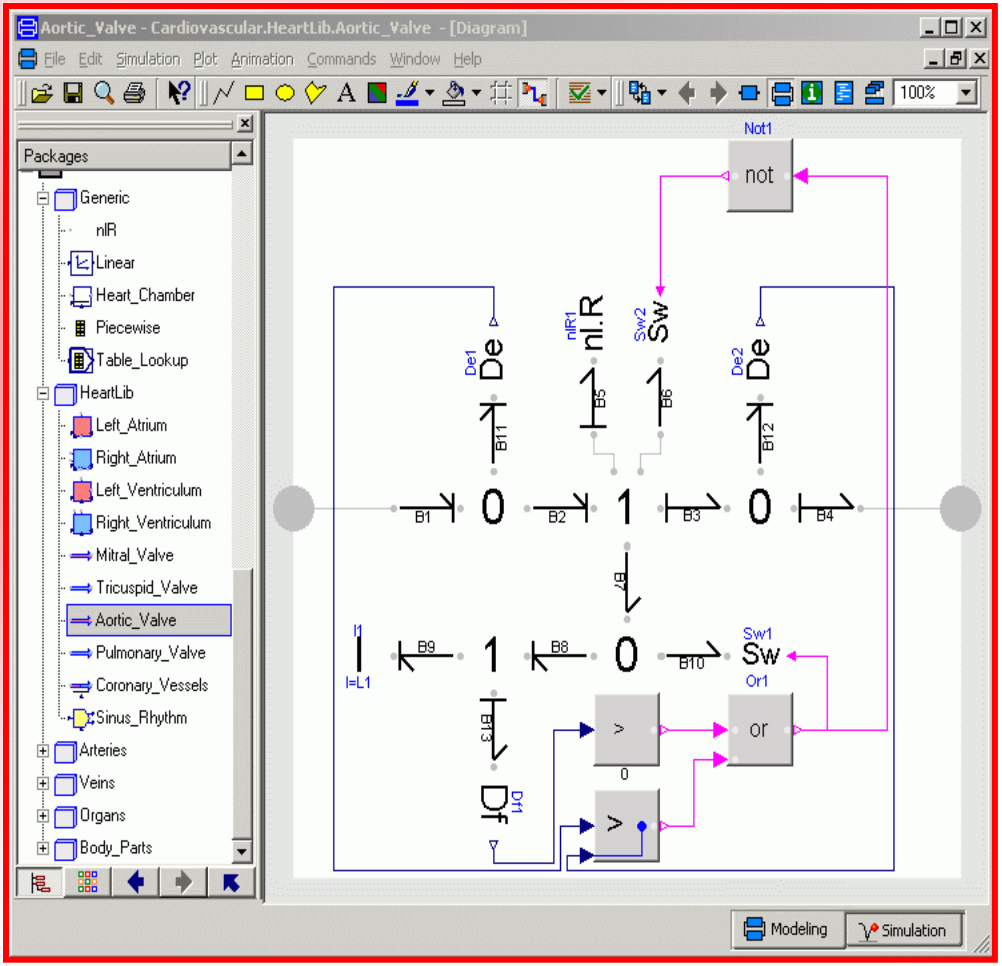


The Hemodynamic System III




Aortic_Valve - Cardiovascular.HeartLib.Aortic_Valve - [Modelica Text]
File Edit Simulation Plot Animation Commands Window Help
100%
Packages
Right_Ventriculum
Mitral_Valve
Tricuspid_Valve
Aortic_Valve
Pulmonary_Valve
Coronary_Vessels
Line: 10
Modeling
Simulation

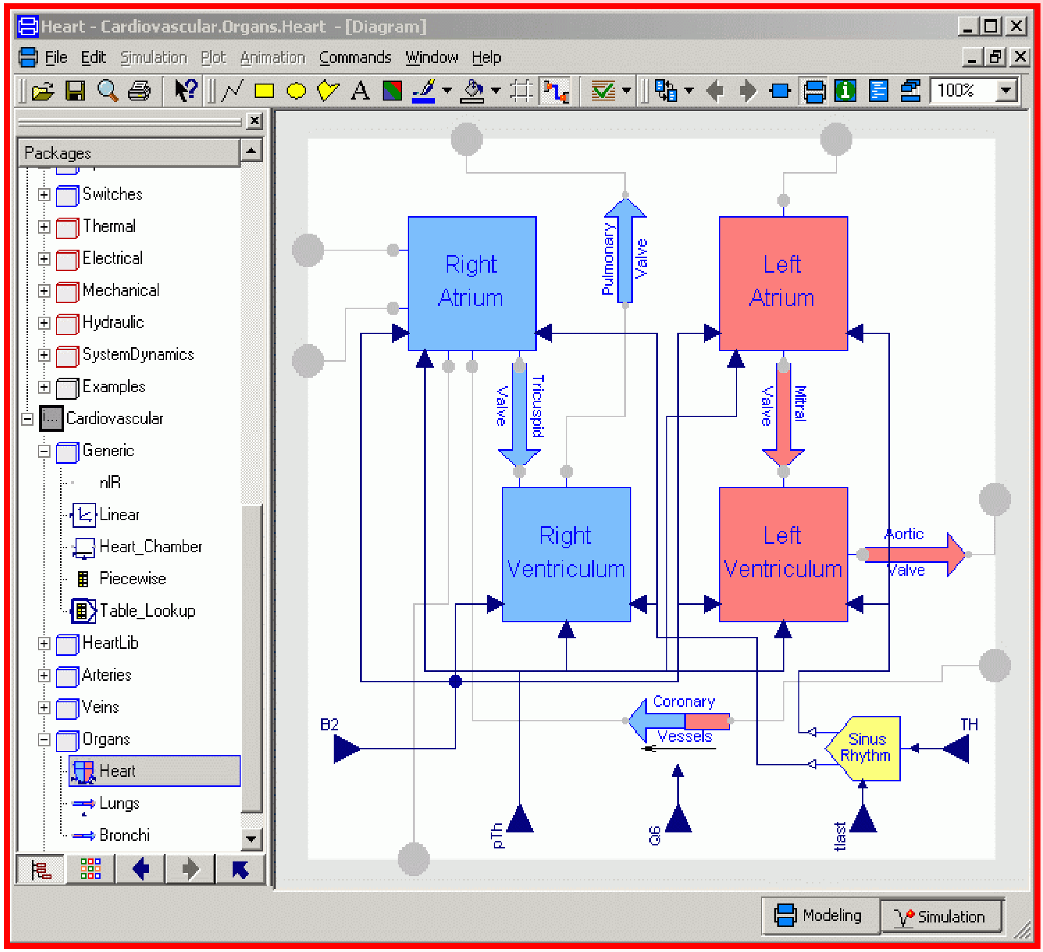
```
model Aortic_Valve "Aortic valve"
  parameter Real r1(unit="mmHg.s/ml") = 0.003 "Linear resistance";
  parameter Real r0(unit="mmHg.s2/cm2") = 0.000796 "Blood density";
  parameter Real AVI(unit="cm2") = 1.539 "Aortic valve area";
  parameter Real KF1=0.001 "Coefficient of turbulent resistance";
  parameter Real DF1K1(unit="mmHg.s2/ml2") = KF1*r0/(2*AVI^2)
    "Quadratic resistance";
  parameter Real L1(unit="mmHg.s2/ml") = 0.00022 "Inductance";

equation

end Aortic_Valve;
```



The Hemodynamic System IV

- All ***containers*** are drawn as ***boxes***. They end in ***0-junctions***.
- All ***flows*** between containers are drawn as ***arrows***. They end in ***bonds***.
- As long as containers and flows toggle, they can be connected together without bonds in between.
- Some of the ***flows*** contain inductors, others only resistances. Some of them also contain valves, which are represented by ***Sw-elements***.

The Heart

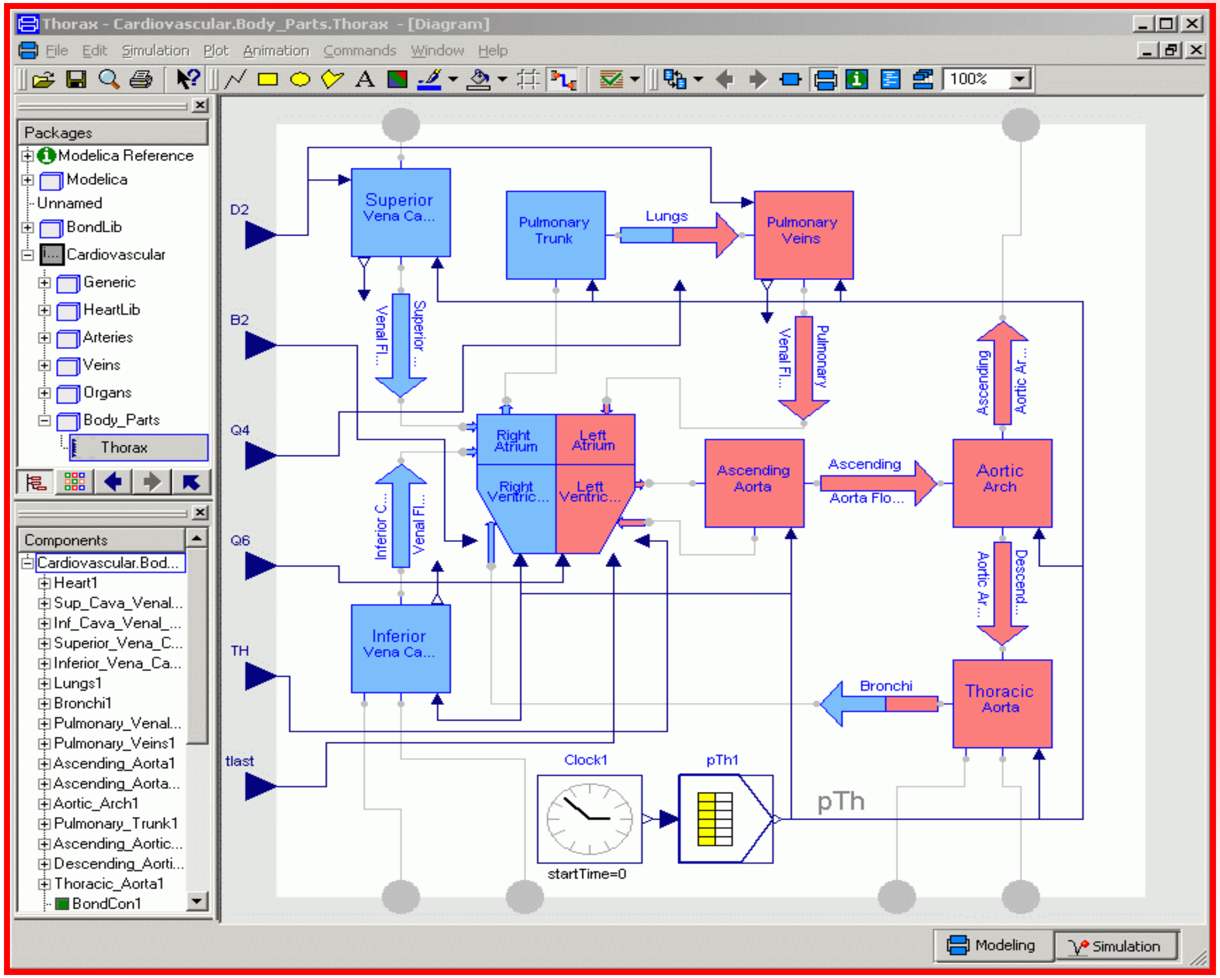


The ***heart*** contains the four chambers, as well as the four major heart valves, the ***pulmonary*** and ***aorta valves*** at the exits of the ventricula, and the ***mitral*** and ***triscuspid valves*** between the atria and the corresponding ventricula.

The ***sinus rhythm*** block programs the contraction and relaxation of the heart muscle.

The ***heart muscle*** flow symbolizes the coronary blood vessels that are responsible for supplying the heart muscle with oxygen.

The Thorax



The *thorax* contains the *heart* and the *major blood vessels*.

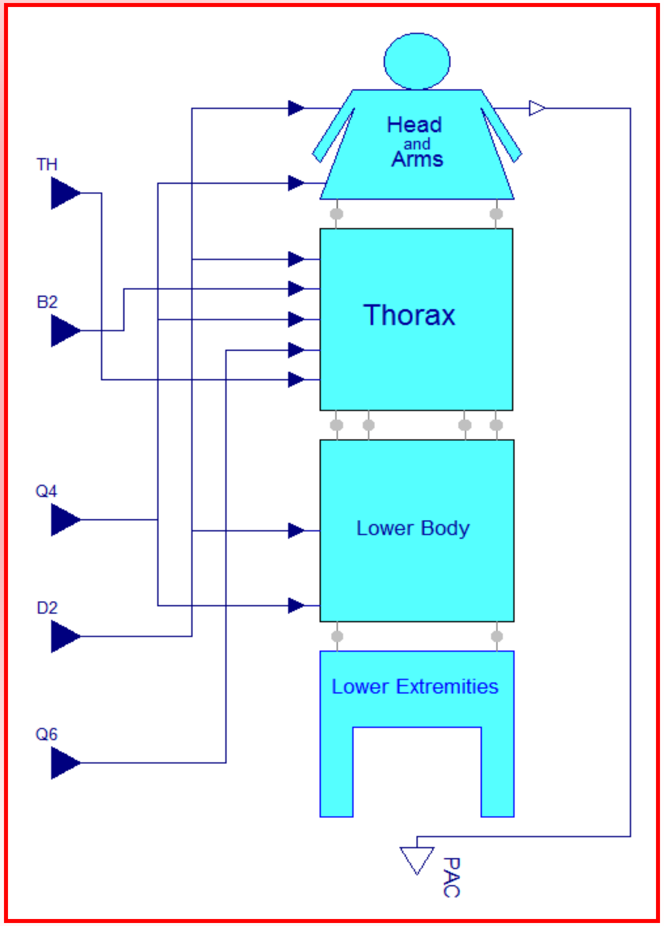
The *table lookup* function at the bottom computes the thoracic pressure as a function of the breathing.

The *arterial blood* is drawn in red, whereas the *venous blood* is drawn in blue.

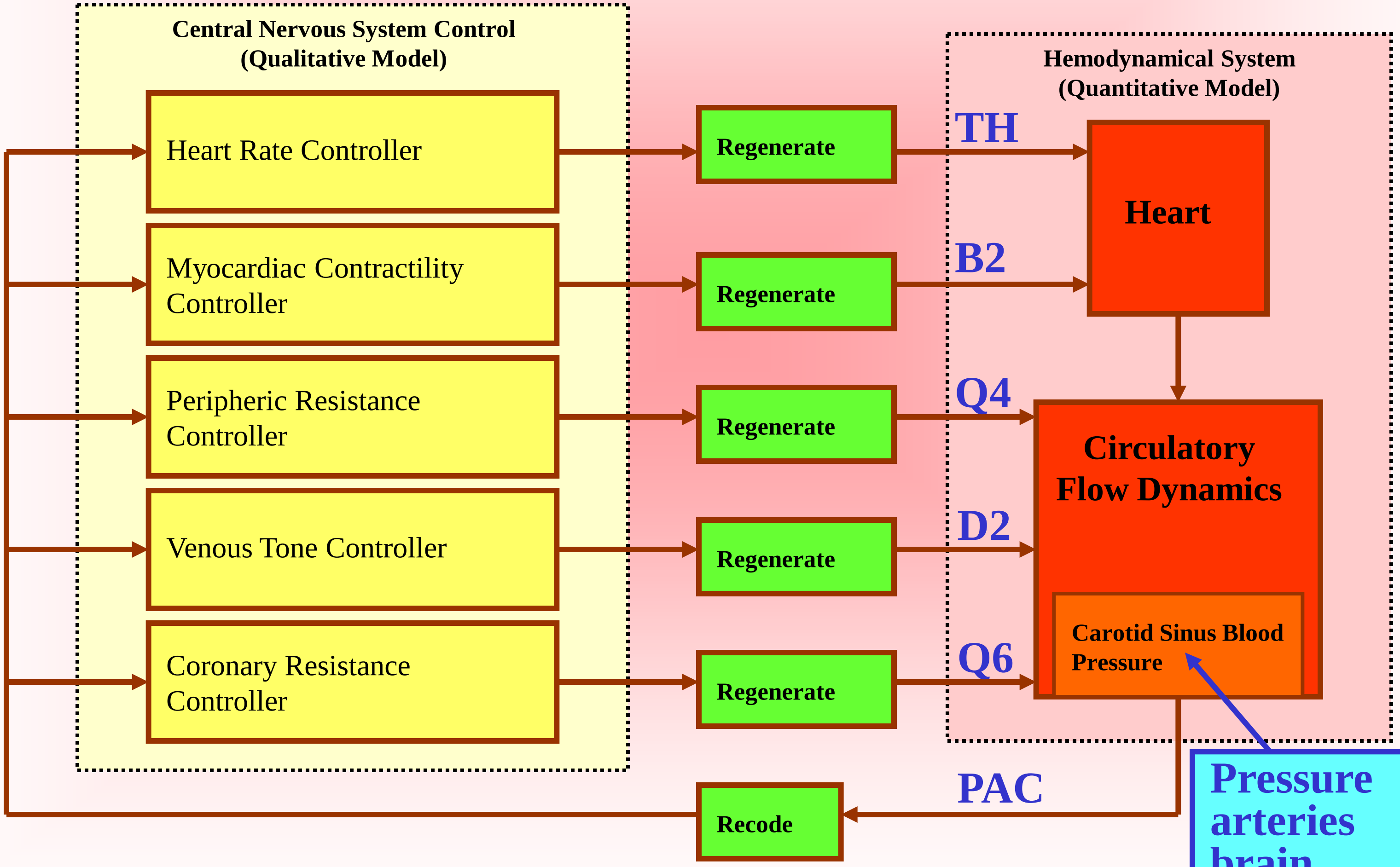
Shown on the left are the *central nervous control signals*.

The Body Parts

- In similar ways, also the other parts of the circulatory system can be drawn. These include the ***head and arms*** (the ***brachiocephalic trunk and veins***), the ***abdomen*** (the ***gastrointestinal arteries and veins***), and the ***lower limbs***.
- Together they form the ***hemodynamic system***.
- What is lacking still are the ***central nervous control*** functions.

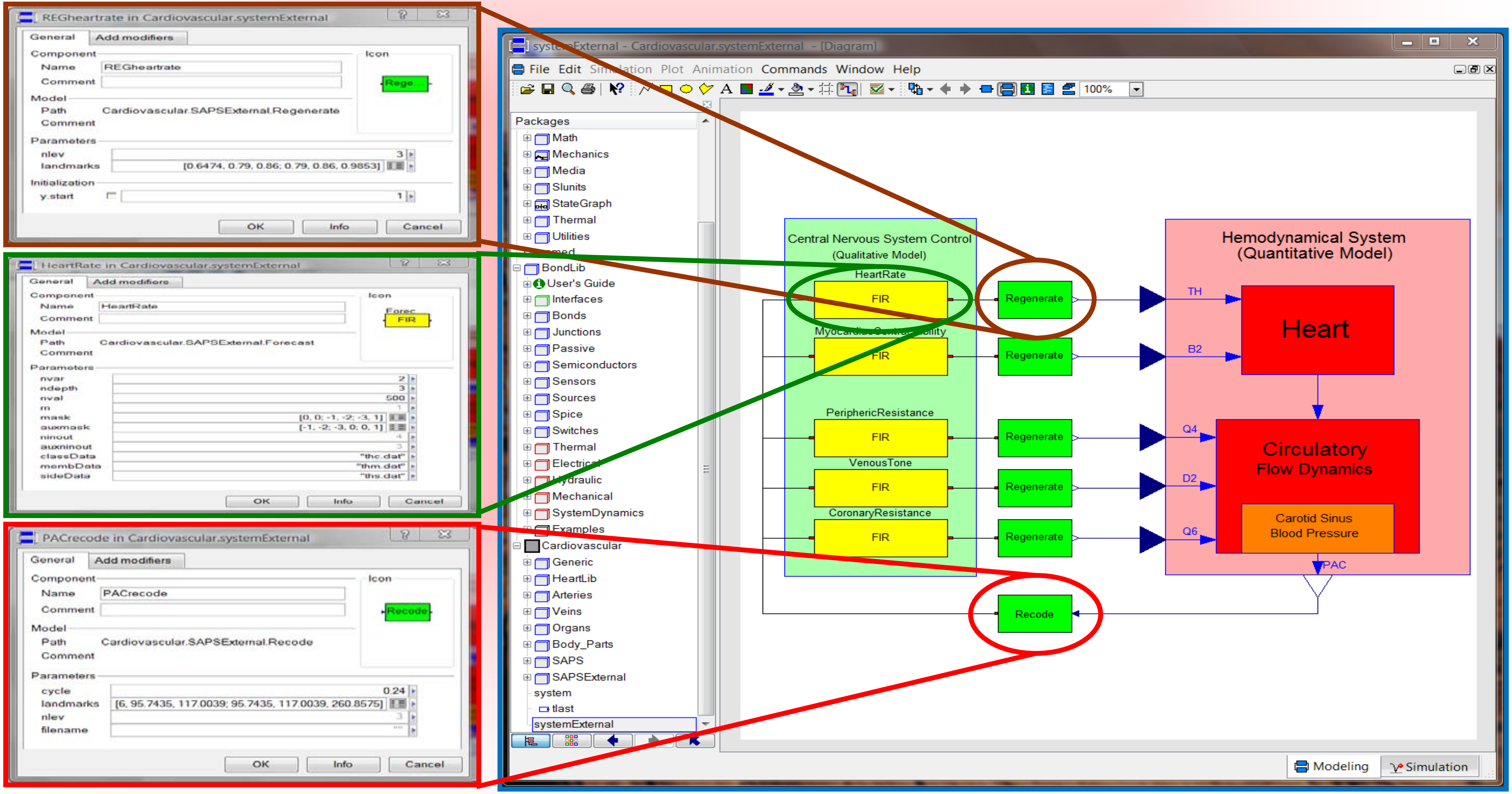


The Cardiovascular System I


Central Nervous System Control
(Qualitative Model)
Heart Rate Controller
Myocardiac Contractility Controller
Peripheric Resistance Controller
Venous Tone Controller
Coronary Resistance Controller
Regenerate
Regenerate
Regenerate
Regenerate
Regenerate
Recode
Hemodynamical System
(Quantitative Model)
TH
B2
Q4
D2
Q6
PAC
Heart
Circulatory Flow Dynamics
Carotid Sinus Blood Pressure

Pressure of the arteries in the brain.

The Cardiovascular System II



REGheartrate in Cardiovascular.systemExternal

Component
Name: REGheartrate
Comment:
Model
Path: Cardiovascular.SAPSExternal.Regenerate
Comment:
Parameters
nlev: 3
landmarks: [0.6474, 0.79, 0.86; 0.79, 0.86, 0.9853]
Initialization
y.start: □ 1

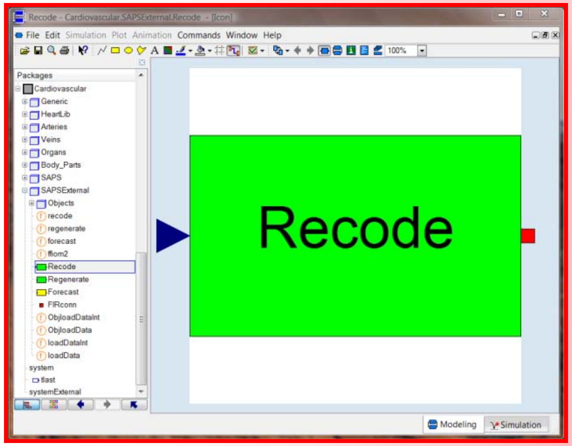
HeartRate in Cardiovascular.systemExternal

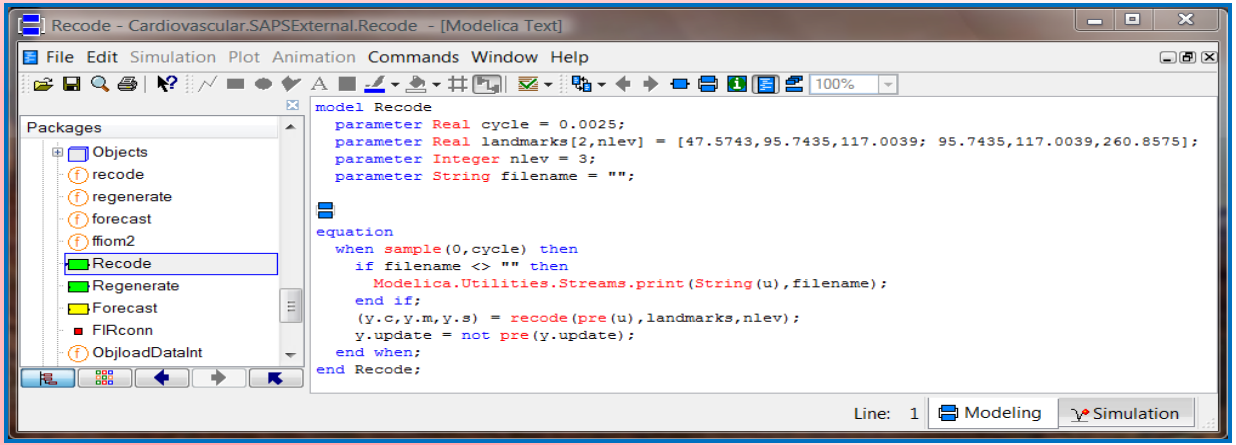
Component
Name: HeartRate
Comment:
Model
Path: Cardiovascular.SAPSExternal.Forecast
Comment:
Parameters
nvar: 2
ndepth: 3
nval: 500
m: 1
mask: [0, 0; -1, -2; -3, 1]
auxmask: [-1, -2; -3, 0; 0, 1]
ninout: 4
auxninout: 3
classData: "thc.dat"
membData: "thm.dat"
sideData: "ths.dat"

PACrecode in Cardiovascular.systemExternal

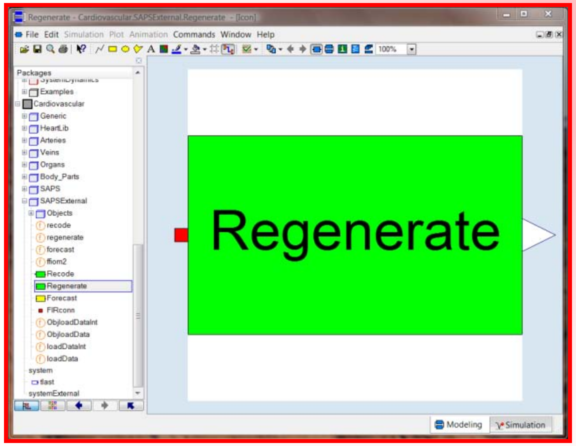
Component
Name: PACrecode
Comment:
Model
Path: Cardiovascular.SAPSExternal.Recode
Comment:
Parameters
cycle: 0.24
landmarks: [6, 95.7435, 117.0039; 95.7435, 117.0039, 260.8575]
nlev: 3
filename: ""

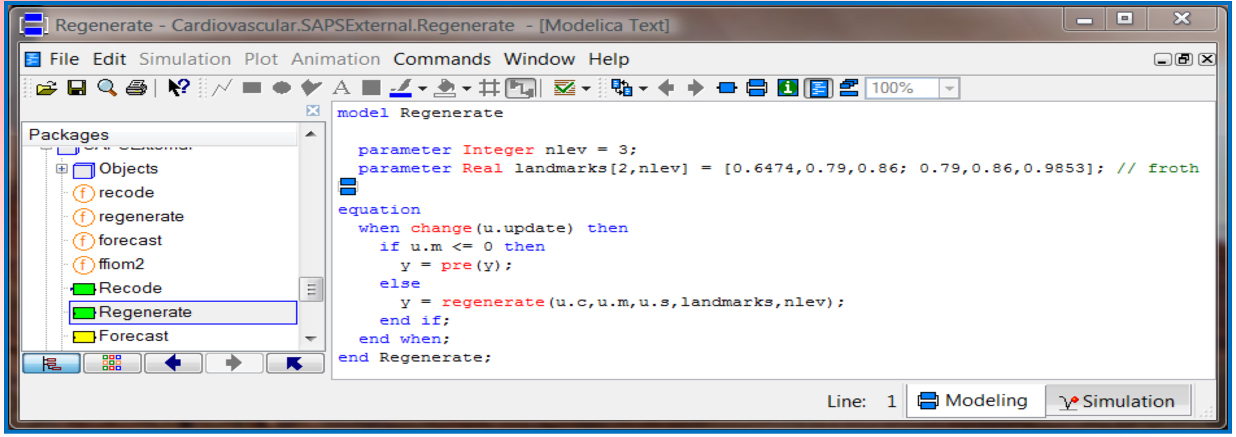
Recode and Regenerate





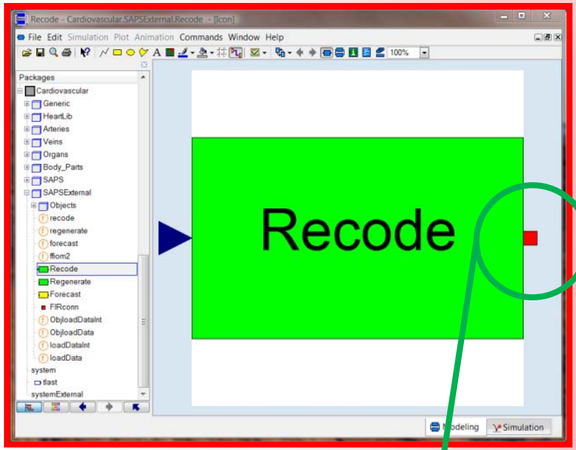
```
model Recode
  parameter Real cycle = 0.0025;
  parameter Real landmarks[2,nlev] = [47.5743,95.7435,117.0039; 95.7435,117.0039,260.8575];
  parameter Integer nlev = 3;
  parameter String filename = "";

equation
  when sample(0,cycle) then
    if filename <> "" then
     Modelica.Utilities.Streams.print(String(u),filename);
    end if;
    (y.c,y.m,y.s) = recode(pre(u),landmarks,nlev);
    y.update = not pre(y.update);
  end when;
end Recode;
```

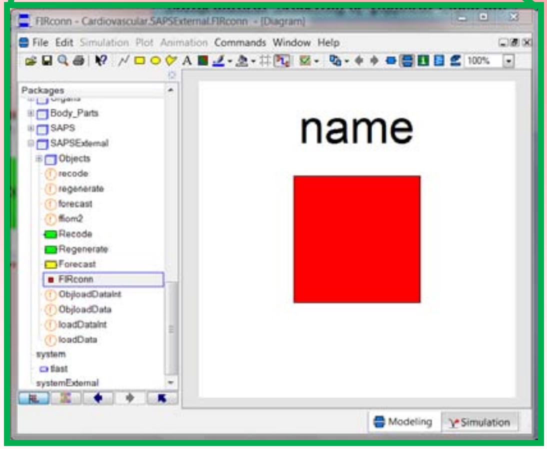


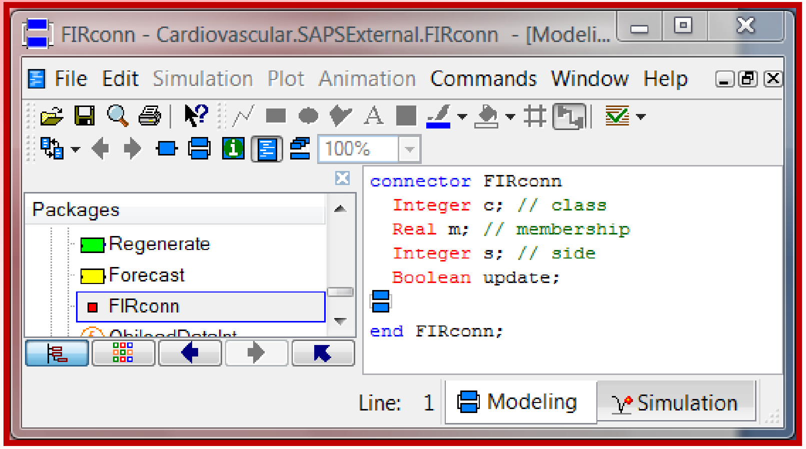

Regenerate - Cardiovascular.SAPSExternal.Regenerate - [Modelica Text]
File Edit Simulation Plot Animation Commands Window Help
100%
Packages
Objects
recode
regenerate
forecast
ffiom2
Recode
Regenerate
Forecast
Line: 1
Modeling
Simulation

```
model Regenerate

  parameter Integer nlev = 3;
  parameter Real landmarks[2,nlev] = [0.6474,0.79,0.86; 0.79,0.86,0.9853]; // froth

equation
  when change(u.update) then
    if u.m <= 0 then
      y = pre(y);
    else
      y = regenerate(u.c,u.m,u.s,landmarks,nlev);
    end if;
  end when;
end Regenerate;
```

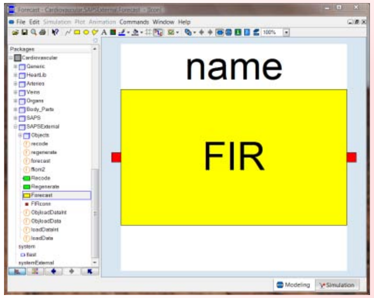
The FIR Connector





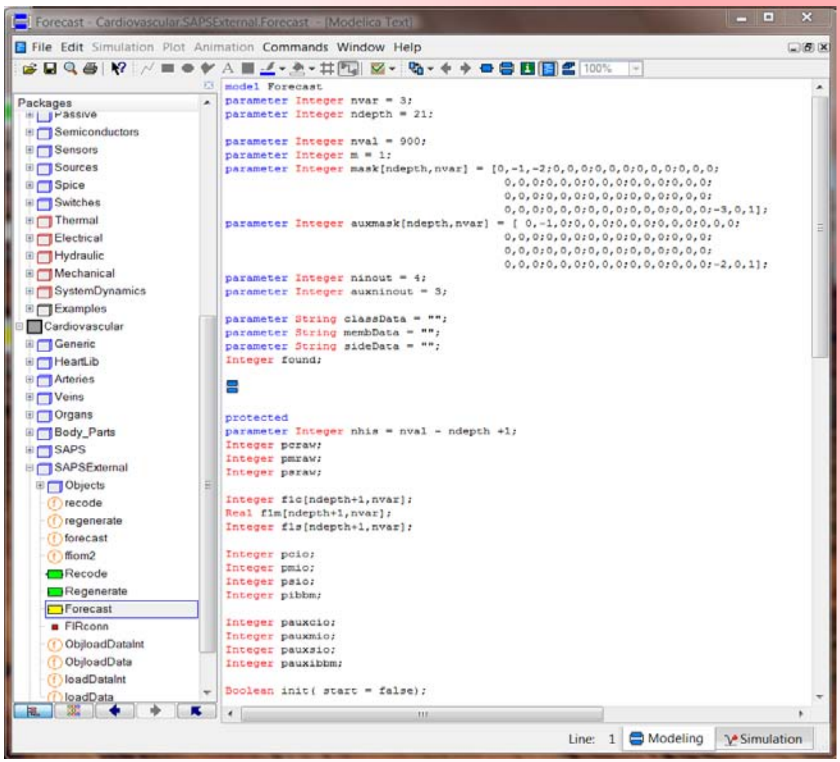

FIRconn - Cardiovascular.SAPSExternal.FIRconn - [Modeli...
File Edit Simulation Plot Animation Commands Window Help
100%
Packages
Regenerate
Forecast
FIRconn
Line: 1
Modeling
Simulation

```
connector FIRconn
  Integer c; // class
  Real m; // membership
  Integer s; // side
  Boolean update;

end FIRconn;
```

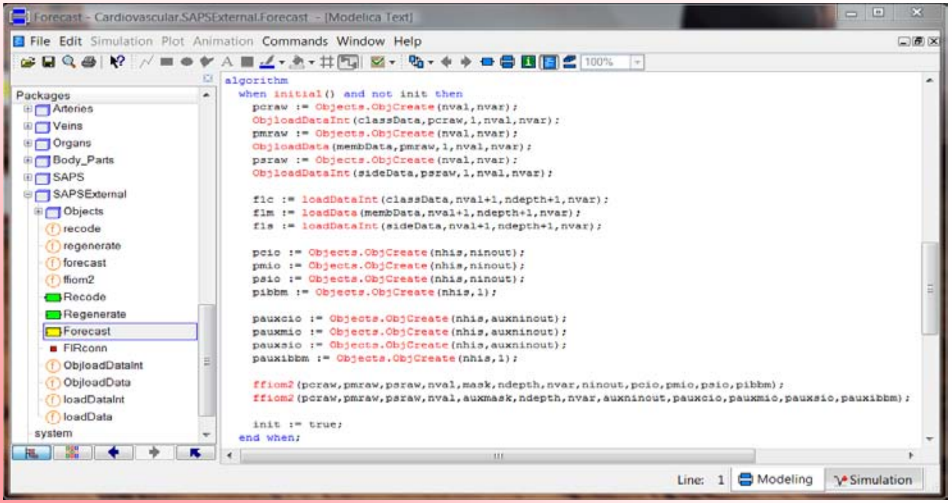


Forecast

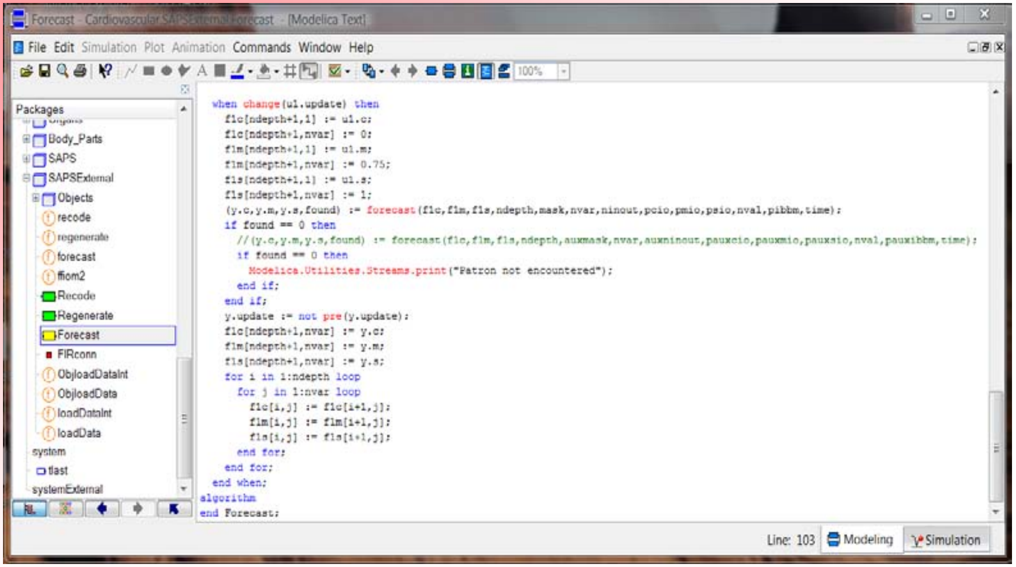
… a bit ugly …



```
model Forecast
parameter Integer nvar = 3;
parameter Integer ndepth = 21;

parameter Integer nval = 900;
parameter Integer m = 1;
parameter Integer mask[ndepth,nvar] = [0,-1,-2;0,0,0;0,0,0;0,0,0;0,0,0;
                                        0,0,0;0,0,0;0,0,0;0,0,0;0,0,0;
                                        0,0,0;0,0,0;0,0,0;0,0,0;0,0,0;
                                        0,0,0;0,0,0;0,0,0;0,0,0;0,0,0;-3,0,1];
parameter Integer auxmask[ndepth,nvar] = [ 0,-1,0;0,0,0;0,0,0;0,0,0;0,0,0;
                                        0,0,0;0,0,0;0,0,0;0,0,0;0,0,0;
                                        0,0,0;0,0,0;0,0,0;0,0,0;0,0,0;
                                        0,0,0;0,0,0;0,0,0;0,0,0;0,0,0;-2,0,1];
parameter Integer ninout = 4;
parameter Integer auxninout = 3;

parameter String classData = "";
parameter String membData = "";
parameter String sideData = "";
Integer found;


protected
parameter Integer nhis = nval - ndepth +1;
Integer pcraw;
Integer pmraw;
Integer psraw;

Integer flc[ndepth+1,nvar];
Real flm[ndepth+1,nvar];
Integer fls[ndepth+1,nvar];

Integer pcio;
Integer pmio;
Integer psio;
Integer pibbm;

Integer pauxcio;
Integer pauxmio;
Integer pauxsio;
Integer pauxibbm;

Boolean init( start = false);
```

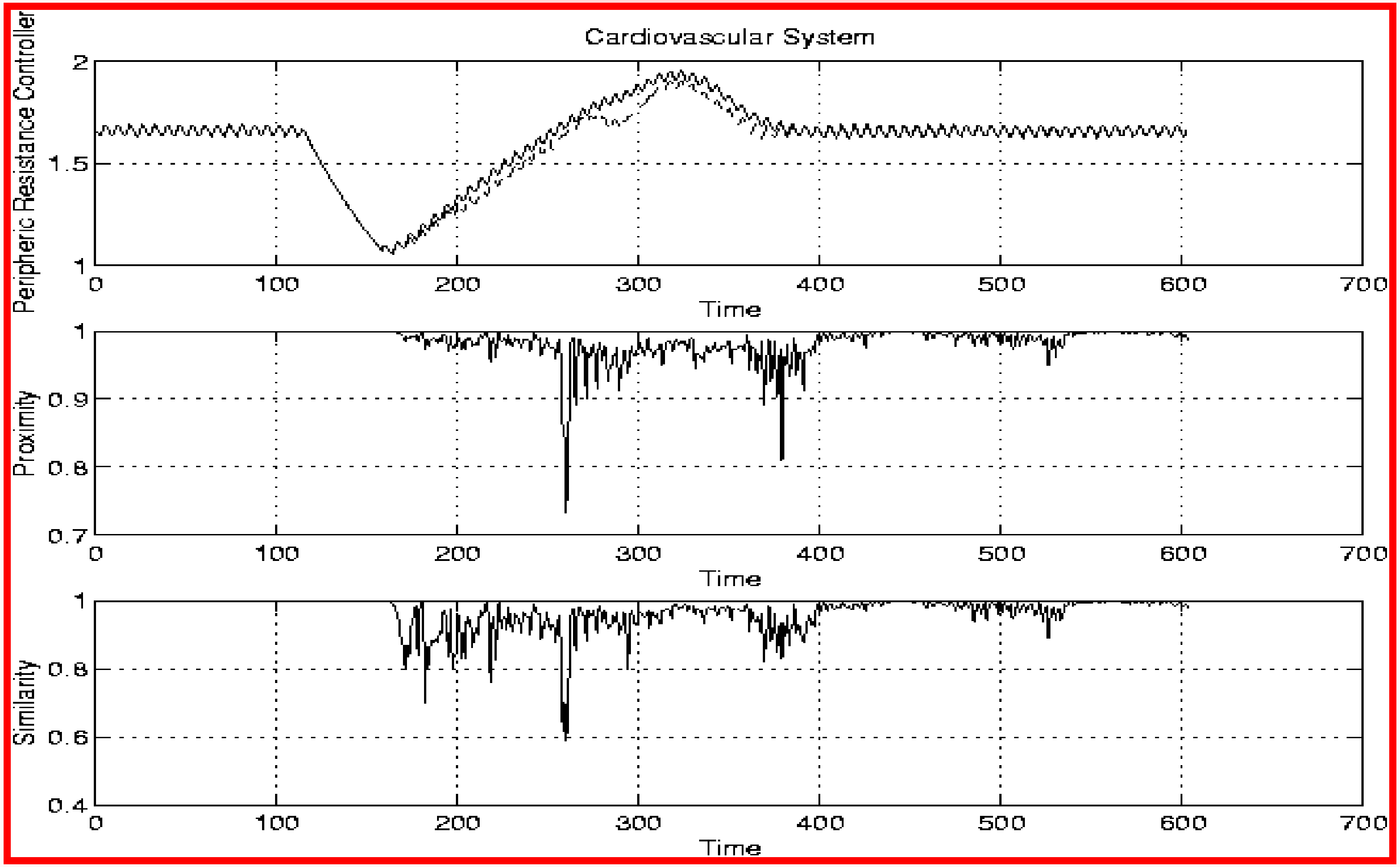


```
algorithm
  when initial() and not init then
    pcraw := Objects.ObjCreate(nval,nvar);
    ObjloadDataInt(classData,pcraw,1,nval,nvar);
    pmraw := Objects.ObjCreate(nval,nvar);
    ObjloadData(membData,pmraw,1,nval,nvar);
    psraw := Objects.ObjCreate(nval,nvar);
    ObjloadDataInt(sideData,psraw,1,nval,nvar);

    flc := loadDataInt(classData,nval+1,ndepth+1,nvar);
    flm := loadData(membData,nval+1,ndepth+1,nvar);
    fls := loadDataInt(sideData,nval+1,ndepth+1,nvar);

    pcio := Objects.ObjCreate(nhis,ninout);
    pmio := Objects.ObjCreate(nhis,ninout);
    psio := Objects.ObjCreate(nhis,ninout);
    pibbm := Objects.ObjCreate(nhis,1);

    pauxcio := Objects.ObjCreate(nhis,auxninout);
    pauxmio := Objects.ObjCreate(nhis,auxninout);
    pauxsio := Objects.ObjCreate(nhis,auxninout);
    pauxibbm := Objects.ObjCreate(nhis,1);

    ffiom2(pcraw,pmraw,psraw,nval,mask,ndepth,nvar,ninout,pcio,pmio,psio,pibbm);
    ffiom2(pcraw,pmraw,psraw,nval,auxmask,ndepth,nvar,auxninout,pauxcio,pauxmio,pauxsio,pauxibbm);

    init := true;
  end when;
```



```
  when change(u1.update) then
    flc[ndepth+1,1] := u1.c;
    flc[ndepth+1,nvar] := 0;
    flm[ndepth+1,1] := u1.m;
    flm[ndepth+1,nvar] := 0.75;
    fls[ndepth+1,1] := u1.s;
    fls[ndepth+1,nvar] := 1;
    (y.c,y.m,y.s,found) := forecast(flc,flm,fls,ndepth,mask,nvar,ninout,pcio,pmio,psio,nval,pibbm,time);
    if found == 0 then
      //(y.c,y.m,y.s,found) := forecast(flc,flm,fls,ndepth,auxmask,nvar,auxninout,pauxcio,pauxmio,pauxsio,nval,pauxibbm,time);
      if found == 0 then
        Modelica.Utilities.Streams.print("Patron not encountered");
      end if;
    end if;
    y.update := not pre(y.update);
    flc[ndepth+1,nvar] := y.c;
    flm[ndepth+1,nvar] := y.m;
    fls[ndepth+1,nvar] := y.s;
    for i in 1:ndepth loop
      for j in 1:nvar loop
        flc[i,j] := flc[i+1,j];
        flm[i,j] := flm[i+1,j];
        fls[i,j] := fls[i+1,j];
      end for;
    end for;
  end when;
algorithm
end Forecast;
```

Simulation Results II



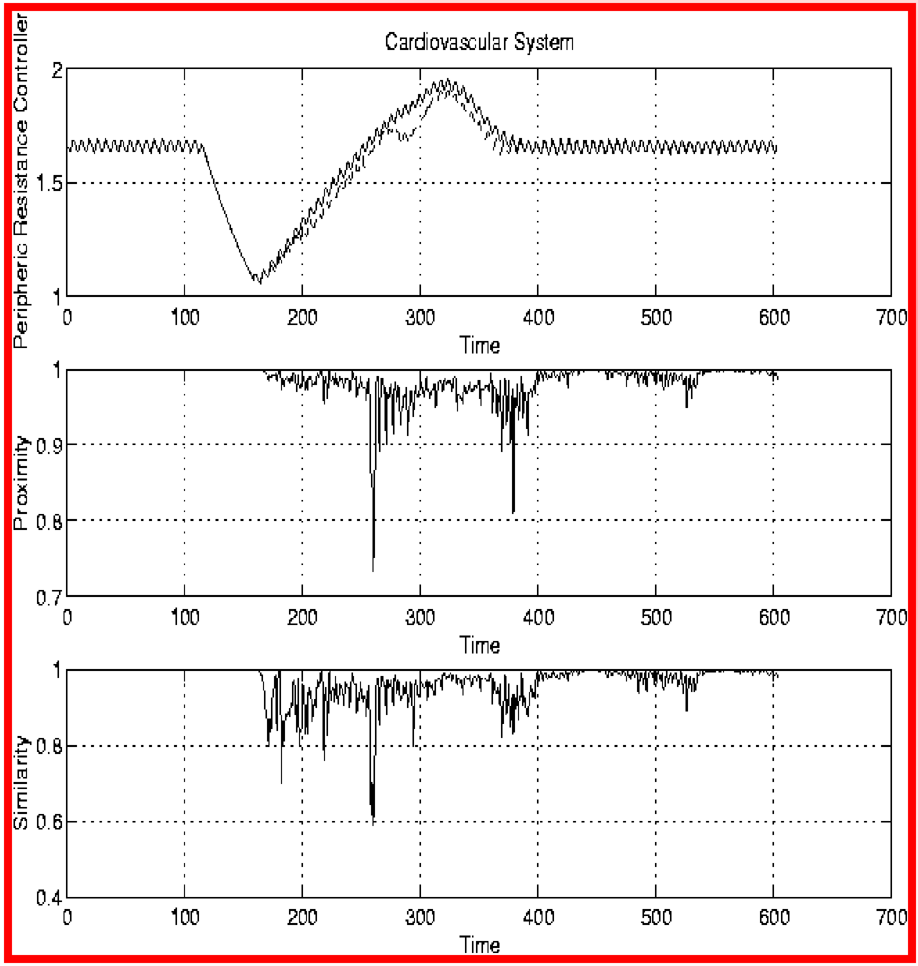
Discussion I

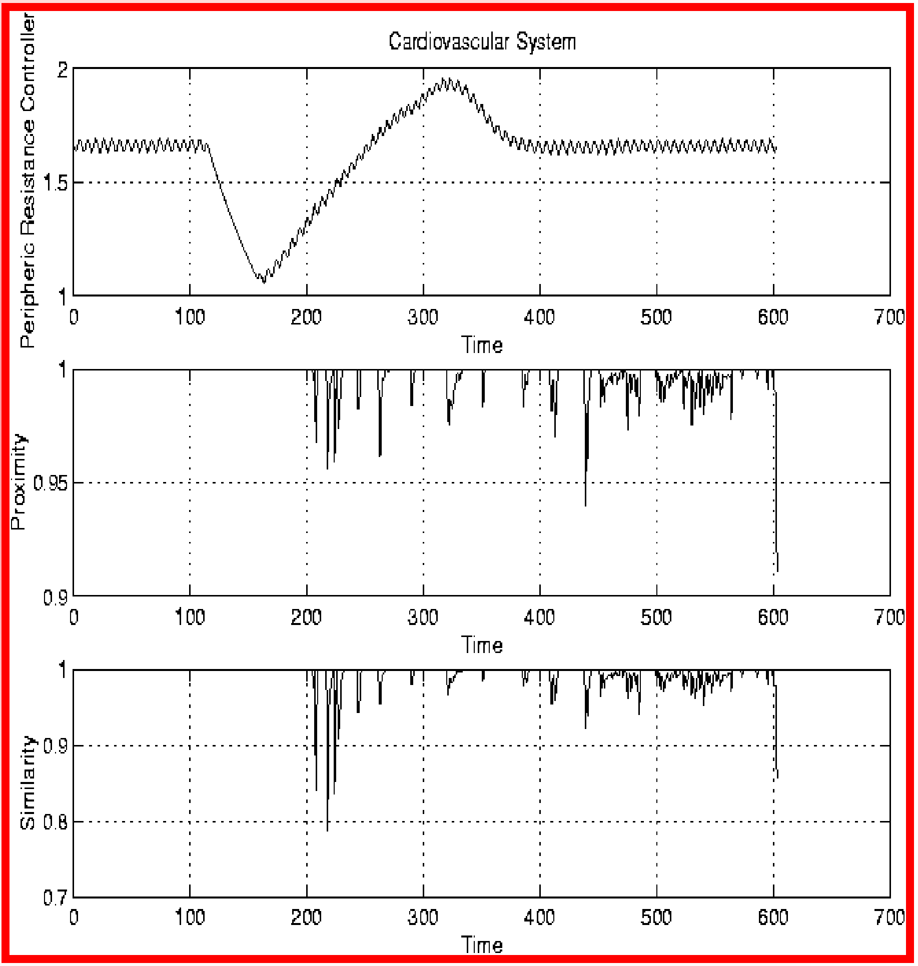
- The top graph shows the ***peripheric resistance controller***, ***Q4***, during a ***Valsalva maneuver***.
- The true data are superposed with the simulated data. The simulation results are generally very good. However, in the center part of the graph, the errors are a little larger.
- Below are two graphs showing the ***estimate of the probability of correctness*** of the prediction made. It can be seen that ***FIR*** is aware that the simulation results in the center area are less likely to be of high quality.

Discussion II

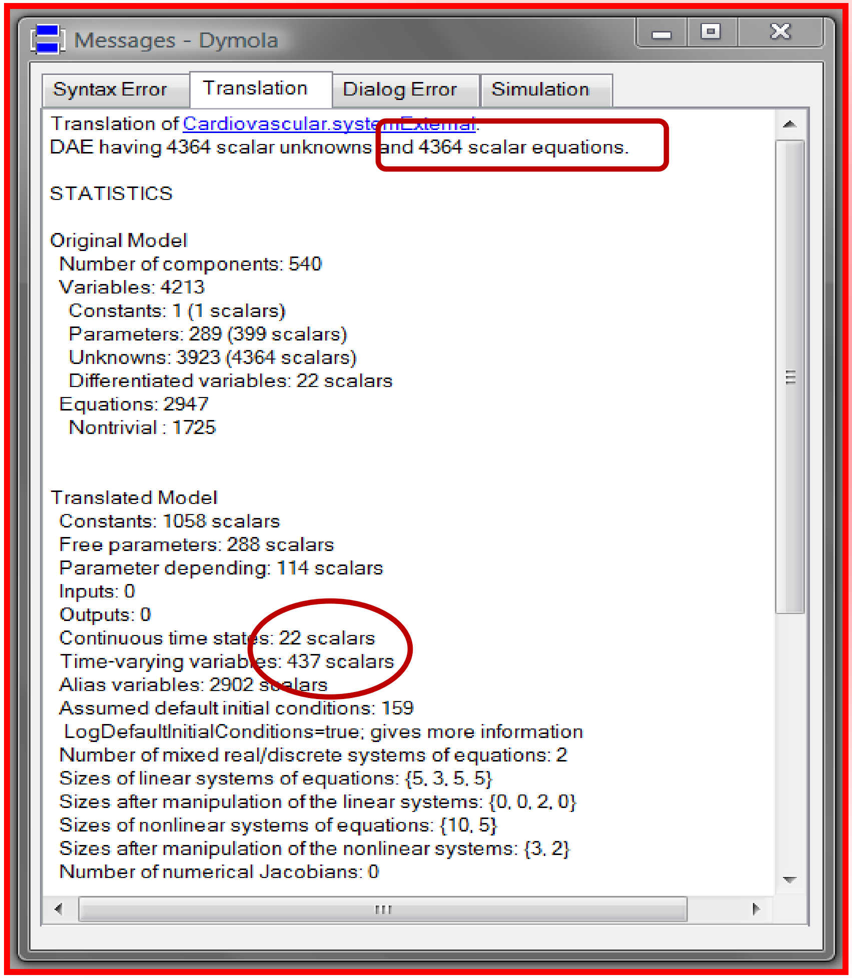
- This can be exploited. Multiple predictions can be made in parallel together with estimates of the likelihood of correctness of these predictions.
- The predictions can then be kept that are accompanied by the ***highest confidence value***.
- This is shown on the next graph. Two different models (***sub-optimal masks***) are compared against each other. The second mask performs better, and also the confidence values associated with these predictions are higher.

Simulation Results III

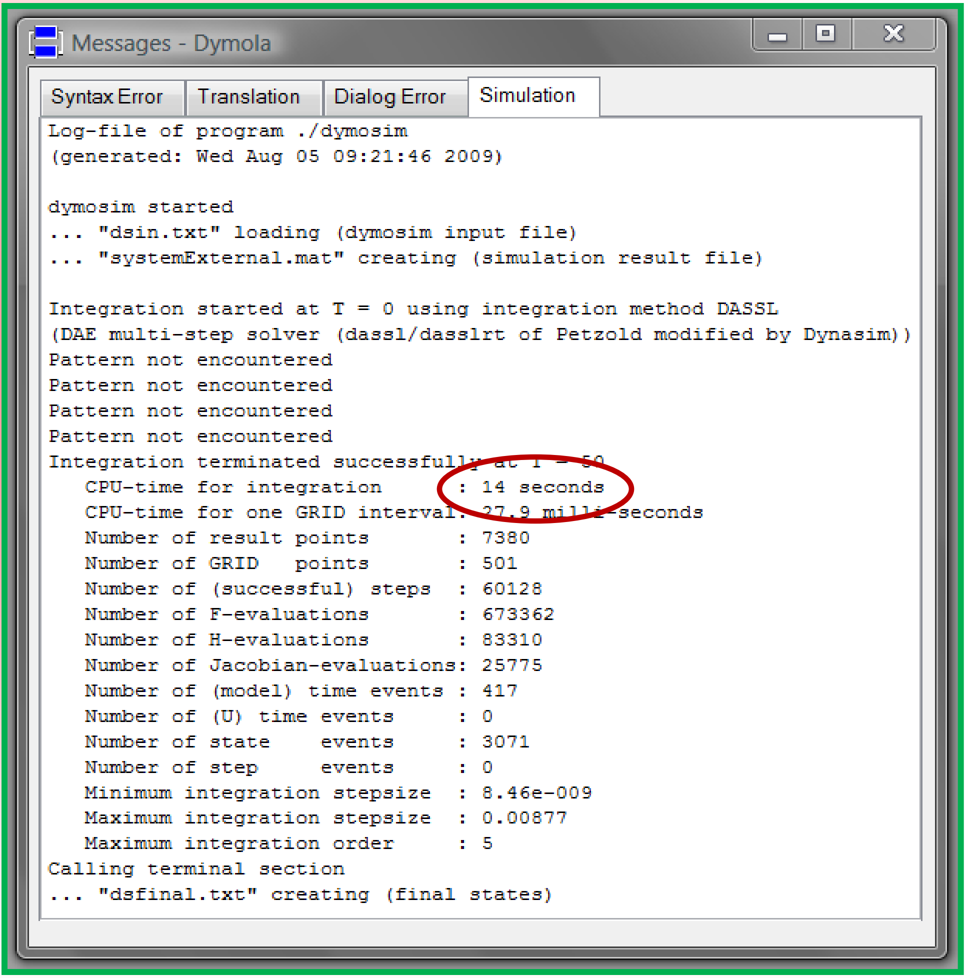




Simulation Results IV



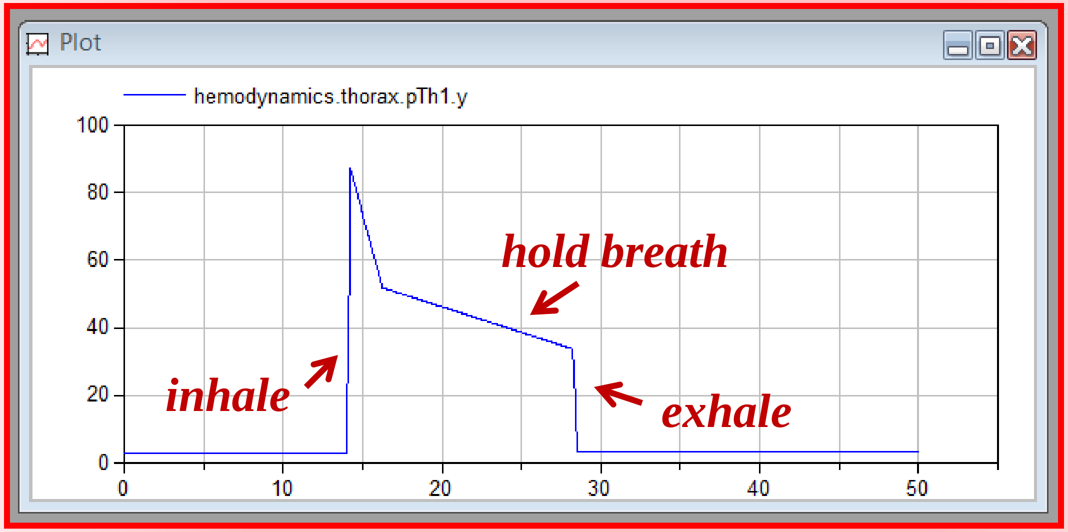
```
Translation of Cardiovascular.systemExternal:
DAE having 4364 scalar unknowns and 4364 scalar equations.

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
```



```
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)
```

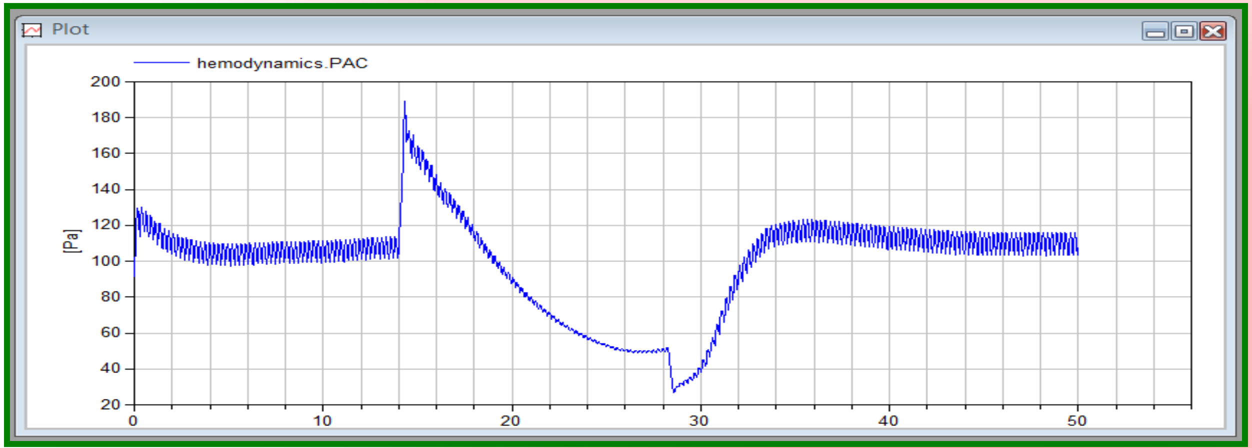
Simulation Results V

Valsalvæ maneuver

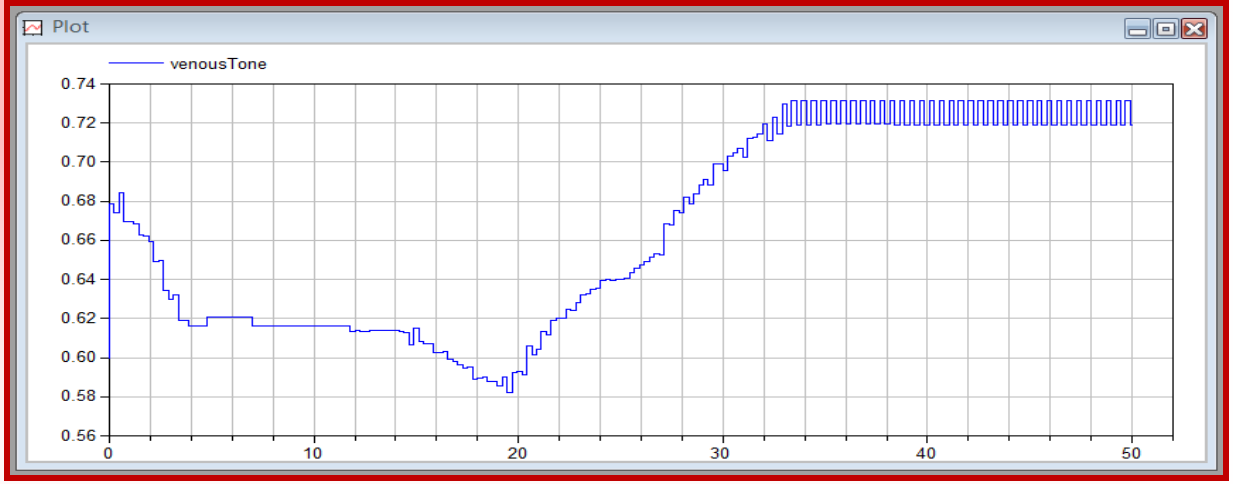


- As the patient inhales, the lungs expand, leaving less "empty space" in the thorax, thereby increasing the thoracic pressure on blood vessels and organs.

Simulation Results VI



Carotid sinus blood pressure (PAC) as simulated using the mixed quantitative and qualitative FIR model.



Regenerated venous tone control signal (D2) as determined by one of the five qualitative FIR (SISO) controllers.

Conclusions I

- ***Quantitative modeling***, i.e. modeling from first principles, is the appropriate tool for applications that are well understood, and where the ***meta-laws*** are well established.
- ***Physical modeling*** is most desirable, because it offers most insight and is most widely extensible beyond the range of previously made experiments.
- ***Qualitative modeling*** is suitable in areas that are poorly understood, where essentially all the available knowledge is in the observations made and is still in its raw form, i.e., no meta-laws have been extracted yet from previous observations.

Conclusions II

- ***Fuzzy modeling*** is a highly attractive ***inductive modeling*** approach, because it enables the modeler to obtain a ***measure of confidence*** in the predictions made.
- ***Fuzzy inductive reasoning*** is one among several approaches to fuzzy modeling. It has been applied widely and successfully to a fairly wide range of applications both in engineering and in the soft sciences.
- ***Qualitative models*** cannot provide insight into the functioning of a system. They can only be used to predict their future behavior, as long as the behavioral patterns stay within their observed norms.

Industrial Applications

- ***Cardiovascular System Modeling*** for Classification of Anomalies.
- ***Anaesthesiology Model*** for Control of Depth of Anaesthesia During Surgery.
- ***Shrimp Growth Model*** for ***El Remolino*** Shrimp Farm in Northern Mexico.
- ***Prediction of Water Demand*** in Barcelona and Rotterdam.
- ***Design of Fuzzy Controller*** for Tanker Ship Steering.
- ***Fault Diagnosis*** of Nuclear Power Plants.
- ***Prediction of Technology Changes*** in Telecommunication Industry.

References I

- Cellier, F.E. (1991), *Continuous System Modeling*, Springer-Verlag, New York, Chapter 13.
- Cellier, F.E. (1991), *Continuous System Modeling*, Springer-Verlag, New York, Chapter 14.
- Cellier, F.E., A. Nebot, F. Mugica, and A. de Albornoz (1996), "Combined Qualitative/Quantitative Simulation Models of Continuous-Time Processes Using Fuzzy Inductive Reasoning Techniques," *Intl. J. General Systems*, **24**(1-2), pp.95-116.

References II

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

- Cellier, F.E. and V. Sanz (2009), "Mixed Quantitative and Qualitative Simulation in Modelica," *Proc. 7th Intl. Modelica Conference*, Como, Italy, pp. 86-95.

- Cellier, F.E. (2009), *The Dymola Cardiovascular System Model*, Version 2.1.

Ph.D. Dissertations

- ***Nebot, A.*** (1994), *Qualitative Modeling and Simulation of Biomedical Systems Using Fuzzy Inductive Reasoning*.
- ***Mugica, F.*** (1995), *Diseño Sistemático de Controladores Difusos Usando Razonamiento Inductivo*.
- ***de Albornoz, A.*** (1996), *Inductive Reasoning and Reconstruction Analysis: Two Complementary Tools for Qualitative Fault Monitoring of Large-Scale Systems*.
- ***López, J.*** (1999), *Qualitative Modeling and Simulation of Time Series Using Fuzzy Inductive Reasoning*.
- ***Mirats, J.M.*** (2001), *Large-Scale System Modeling Using Fuzzy Inductive Reasoning*.