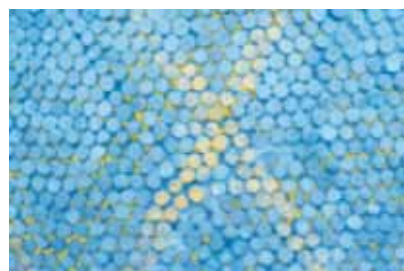
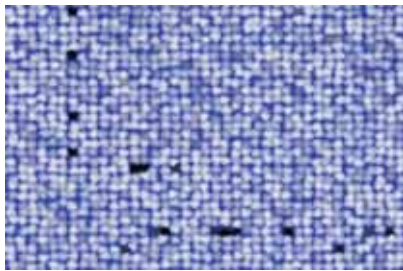
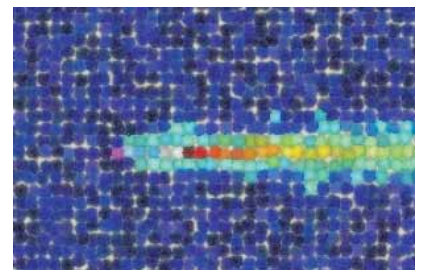
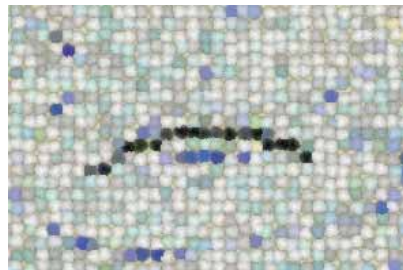
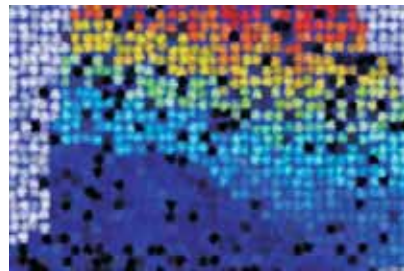
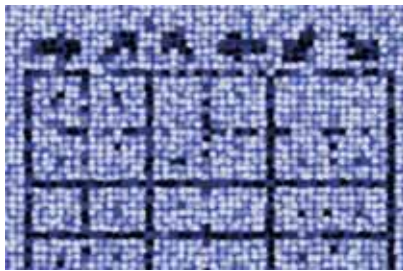


# SNE SIMULATION NEWS EUROPE



Volume 16, Number 3-4

December 2006, ISSN 0929-2268



Journal on Developments and  
Trends in Modelling and Simulation  
Membership Journal for Simulation  
Societies in EUROSIM





Dear readers,

The year 2006 came up with changes in SNE and for SNE. SNE started not only with a new layout, also some structural changes took place. In 2006 we introduced for SNE a new numbering with volumes - in 2006 SNE Volume 16, with four issues (regularly one single regular issue, one single special issue, and one regular double issue). This remembers, that we are editing and publishing SNE since 16 years - an occasion to thank all who have supported us and have made SNE a big success. We are proud having started the SNE Special Issues with SNE 16/2 'Parallel and Distributed Simulation', to be continued in 2007 with Special Issue 17/2 'Validation and Verification'.

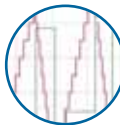
The SNE Editorial Board is increasing, and it co-operates with IPCs from Simulation Conferences, to suggest conference papers for publication in SNE in extended and revised form. This issue publishes e.g. revised contributions from Modelica 2006 Conference (Vienna), and from ASIM 2006 Conference (Hannover). Furthermore, more space is available for Technical Notes, Short Notes and Benchmark Notes.

Another reorganisation is in progress - the reorganisation of the ARGESIM Comparisons, which have evolved to ARGESIM Benchmarks on Modelling and Simulation. This issue presents first time the new two-page layout for five benchmarks solutions - now called Benchmark Notes, and starts a series with contributions on theory and background of the benchmarks - beginning with 'Cellular Automata Models for SIR-type Epidemics' - C17. Furthermore we start with revisions and updates of benchmarks - this issue with 'C9R Extended Fuzzy Control' and 'C19R Pollution in Groundwater Flow'.

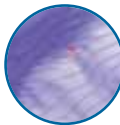
This issue presents in the News Section instead of reports of the societies only quick information on the societies in a new table format. This format will be continued in the next issues, integrated in the reports of the societies. The News Section publishes an obituary - it is a sad duty to announce the death of Len Dekker, the father of EUROSIM.

Felix Breiteneker, editor-in-chief; Felix.Breiteneker@tuwien.ac.at

## SNE 16/3-4 in Five Minutes



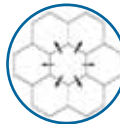
*Quantised State System Simulation in Dymola/Modelica using the DEVS Formalism* (TN) - presents features for discrete modelling in Modelica - **page 3**



*Two-dimensional Finite Element Model for Thermodynamic and Continuum Mechanical Processes within the Snow Cover* (TN) - introduces into dynamics of snow movement - **page 13**



*Advanced Modeling and Simulation Techniques in MOSILAB: A System Development Case Study* (TN) - presents Modelica-based hybrid modelling techniques - **page 19**



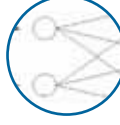
*Cellular Automata Models for SIR-type Epidemics* (TN) - compares CA models with ODE, PDE and DE models - supplemental info for the homonymous benchmark - **page 27**



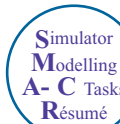
*Modelling and Identification of a Laboratory Helicopter* (TN) - presents modelling, control and identification in mechatronics - **page 37**



*Model-oriented Data-driven Architecture for Microsimulation* (SN) - describes use of large data systems for microsimulation studies - **page 43**



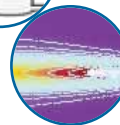
*Model-based Learning Classifiers for Surface Inspection Problems* (SN) - compares modelling methods for quality inspection - **page 47**



*ARGESIM Benchmarks: Revised Definitions, Extended Solutions, and Supplemental Information* (BN) - introduces the reorganised ARGESIM benchmark series - **page 57**  
ARGESIM Benchmark Section



- defines the revised benchmark C9R Extended Fuzzy Control of a Two-Tank System - **page 59**

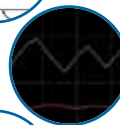


- updates the benchmark C19R Pollution in Groundwater Flow with Spatially Distributed Modelling - **page 63**



ARGESIM Benchmark Solutions

- C7 Constrained Pendulum with MOSILAB - **page 67**



- C9 Fuzzy Control with Dymola - **page 69**



- C9 Fuzzy Control with MATLAB/Simulink - **page 71**



- C9 Fuzzy Control with AnyLogic - **page 73**



- C9R Extended Fuzzy Control with MATLAB/Simulink - **page 75**



- C17 SIR-type Epidemic' using MAPLE - **page 77**



- C19R Pollution in Groundwater Flow' using COMSOL - **page 79**



Book Reviews and Journal News

- Five book reviews  
- Introduction of SNE Special Issue Parallel and Distributed Simulation Methods and Environments  
- Call for next SNE Special Issue Validation and Verification

- **page 51 - 56**



EUROSIM Society Quick Info  
In Memoriam Len Dekker  
in the News Section

- **8 pages**

# Maple<sup>TM</sup> 10

## Explore...Teach...Connect...

Entdecken Sie die Mathematik mit Maple, einem der mächtigsten analytischen Rechensysteme der Welt, mit einer erweiterbaren mathematischen Programmiersprache, mit 2D- und 3D-Visualisierungen oder mit selbst entworfenen grafischen Oberflächen...

Unterrichten Sie Mathematik mit Maplet-Tutoren und Visualisierungs-Routinen, die speziell für Studenten entworfen wurden und mit kostenlosen Kursmaterialien aus dem Maple Application Center...

Schlagen Sie Brücken zu MATLAB<sup>®</sup>, Visual Basic<sup>®</sup>, Java<sup>™</sup>, Fortran und C, durch den Export nach HTML, MathML<sup>™</sup>, XML, RTF, LaTeX, POV-Ray<sup>™</sup> oder über das Internet mit Hilfe von TCP/IP-Sockets.




















[www.scientific.de](http://www.scientific.de) · [maple@scientific.de](mailto:maple@scientific.de)



*scientific* COMPUTERS



## Content

-  Quantised State System Simulation in Dymola/Modelica using the DEVS Formalism; T. Beltrame, F. E. Cellier ... 3
-  Two-dimensional Finite Element Model for Thermodynamic and Continuum Mechanical Processes within the Snow Cover; H. Teufelsbauer ... 13
-  Advanced Modeling and Simulation Techniques in MOSILAB: A System Development Case Study; Ch. Nytsch-Geusen, T. Ernst, A. Nordwig, P. Schwarz, P. Schneider, M. Vetter, Ch. Wittwer, A. Holm, T. Noudui, J. Leopold, G. Schmidt, A. Mattes ... 19
-  Cellular Automata Models for SIR-type Epidemics; G. Schneckenreither, F. Breiteneker, N. Popper, G. Zauner ... 27
-  Modelling and Simulation of a Laboratory Helicopter Epidemics; G. Karer and B. Zupančič, ... 37
-  Model-oriented Data-driven Architecture for Microsimulation; I. Sinka, I. Molnar ... 43
-  Model-based Learning Classifiers for Surface Inspection Problems; S. Seichter, F. Breiteneker, Ch. Eitzinger ... 47
-  Book / Journal Reviews, Book / Journal ... 51
-  ARGESIM Benchmarks on Modelling and Simulation: Revised Definitions, Extended Solutions, and Supplemental Information; F. Breiteneker ... 57
-  'Extended Fuzzy Control of a Two-Tank System' - ARGESIM Benchmark C9R; I. Škrjanc; ... 59
-  'Pollution in Groundwater Flow' - ARGESIM Benchmark C19 with Spatially Distributed Modelling; F. Judex, F. Breiteneker, G. Höfinger ... 63
-  Three Structural Different Modelling Approaches to ARGESIM Comparison C7 'Constrained Pendulum' using the Modelica-Simulator MOSILAB; G. Zauner, F. Breiteneker ... 67
-  Directly Programmed Fuzzy Control in ARGESIM Benchmark C9 'Fuzzy Control of a Two-Tank System' using Dymola; A. Sodja ... 69
-  A Discrete Model Approach to ARGESIM Benchmark C9 'Fuzzy Control of a Two-Tank System' with MATLAB, Simulink and Fuzzy Control Toolbox; L. Wallentin, R. Hausleitner, M. Paier, G. Zauner ... 71
-  A Java-supported Approach to ARGESIM Benchmark C9 'Fuzzy Control of Two-Tank System' by Anylogic; Th. Mair, R. Leidenfrost, A. Platschek, S. Tauböck ... 73
-  A Classic Solution to ARGESIM Benchmark C9R 'Extended Fuzzy Control' using MATLAB/Simulink and Fuzzy Control Toolbox; I. Škrjanc, F. Breiteneker ... 75
-  A Solution to ARGESIM Benchmark C17 'SIR-type Epidemic' using Numerical Programming Features in MAPLE; Š. Emrich, H. Glavanovits, T. Khorzad ... 77
-  A FEM - based Approach to ARGESIM Benchmark C19R 'Pollution in Groundwater Flow' using COMSOL Multiphysics; H. Teufelsbauer ... 79
-  Quick Info on EUROSIM and EUROSIM Societies (addresses, websites, officers, etc.)

In Memoriam Len Dekker

8 pages

## SNE Editorial Boards

**SNE** - Simulation News Europe is advised by two Editorial Boards. The *SNE Editorial Board*, a scientific editorial board, is taking care on peer reviewing and handling of Technical Notes, Short Notes, Software Notes, Book Reviews, and Benchmark Notes (*SNE Notes Section*). The *SNE News Editorial Board* (see *SNE News Section*) is responsible for reports from EUROSIM, EUROSIM societies, International Societies, and for Industry News.

### SNE Editorial board

Felix Breiteneker (Editor-in-Chief), Vienna Univ. of Technology, [Felix.Breiteneker@tuwien.ac.at](mailto:Felix.Breiteneker@tuwien.ac.at)  
 Peter Breedveld, University of Twente, Div. Control Engineering, [P.C.Breedveld@el.utwente.nl](mailto:P.C.Breedveld@el.utwente.nl)  
 Francois Cellier, ETH Zurich, Inst. f. Computational Science [fcellier@inf.ethz.ch](mailto:fcellier@inf.ethz.ch),  
 Russell Cheng, Fac. of Mathematics / OR Group, Univ. of Southampton, [rhc@maths.soton.ac.uk](mailto:rhc@maths.soton.ac.uk)  
 Rihard Karba, University of Ljubljana, Fac. Electrical Engineering, [rihard.karba@fe.uni-lj.si](mailto:rihard.karba@fe.uni-lj.si)  
 David Murray-Smith, University of Glasgow, Fac. Electrical & Electronical Engineering; [d.murray-smith@elec.gla.ac.uk](mailto:d.murray-smith@elec.gla.ac.uk)  
 Horst Ecker, Vienna Univ. of Technology, Inst. f. Mechanics, [Horst.Ecker@tuwien.ac.at](mailto:Horst.Ecker@tuwien.ac.at)  
 Thomas Schriber, University of Michigan, Business School [schriber@umich.edu](mailto:schriber@umich.edu)  
 Sigrid Wenzel, University Kassel, Inst. f. Production Technique and Logistics, [S.Wenzel@uni-kassel.de](mailto:S.Wenzel@uni-kassel.de)

**SNE Contact:** SNE - Editors /ARGESIM  
 c/o Inst. f. Analysis and Scientific Computation  
 Vienna University of Technology  
 Wiedner Hauptstrasse 8-10, 1040 Vienna, AUSTRIA  
 Tel + 43 - 1- 58801-10115 or 11455, Fax - 42098  
[sne@argesim.org](mailto:sne@argesim.org); [WWW.ARGESIM.ORG](http://WWW.ARGESIM.ORG)

**SNE Quiz.** Thee cover page shows eight modified pictures, seven alienated pictures from this issue, and one picture from a real object with only changed colours. Readers are invited to send in solutions with at least six identified pictures. The winner (most identifications) will get a book from the ARGESIM Report Series.

### Editorial Information - Impressum

**SNE** Simulation News Europe ISSN 1015-8685 (0929-2268).  
**Scope:** Technical Notes and Short Notes on developments in modelling and simulation in various areas /application and theory) and on benchmarks for modelling and simulation, membership information for EUROSIM and Simulation Societies.  
**Editor-in-Chief:** Felix Breiteneker, Mathematical Modelling and Simulation, Inst. f. Analysis and Scientific Computing, Vienna University of Technology, Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria; [Felix.Breiteneker@tuwien.ac.at](mailto:Felix.Breiteneker@tuwien.ac.at)  
**Layout:** A. Breiteneker, ARGESIM TU Vienna / Linz; [Anna.Breiteneker@liwest.at](mailto:Anna.Breiteneker@liwest.at)  
**Printed by:** Grafisches Zentrum, TU Vienna, Wiedner Hauptstrasse 8-10, 1040, Vienna, Austria  
**Publisher:** ARGESIM / ASIM  
 ARGESIM, c/o Inst. for Scientific Computation, TU Vienna, Wiedner Hauptstrasse 8-10, A-1040 Vienna, Austria, and ASIM - German Simulation Society), c/o BAUSCH-GALL GmbH, Wohlfartstr. 21b, 80939 München

© ARGESIM / ASIM 2006

# TECHNICAL NOTES

## Quantised State System Simulation in Dymola/Modelica using the DEVS Formalism

Tamara Beltrame, VTT Industrial Systems, Finland; [Tamara.Beltrame@vtt.fi](mailto:Tamara.Beltrame@vtt.fi)

François E. Cellier, ETH Zurich, Switzerland; [FCellier@inf.ethz.ch](mailto:FCellier@inf.ethz.ch)

Continuous-time systems can be converted to discrete -event descriptions using the Quantised State Systems (QSS) formalism. Hence it is possible to simulate continuous-time systems using a discrete-event simulation tool, such as a simulation engine based on the DEVS formalism. A new Dymola library, ModelicaDEVS, was developed that implements the DEVS formalism. DEVS has been shown to be efficient for the simulation of systems exhibiting frequent switching operations, such as flyback converters. ModelicaDEVS contains a number of basic components that can be used to carry out DEVS simulations of physical systems. Furthermore, it is also possible - with some restrictions - to combine the two simulation types of ModelicaDEVS and Dymola (discrete-event and discrete-time simulation) and create hybrid models that contain ModelicaDEVS as well as standard Dymola components.

### Introduction

Since Dymola/Modelica was primarily designed to deal with continuous physical problems, numerical integration is central to its operation, and therefore, the search for new algorithms that may improve the efficiency of simulation runs is justified.

Toward the end of the nineties, a new approach for numerical integration by a *discrete-event formalism* has been developed by Zeigler et al. ([13]): given the fact that all computer-based continuous system simulations have to undergo a discretisation of one form or another – as digital machines are not able to process raw continuous signals – the basic idea of the new integration approach was to replace the discretisation of time by a quantisation of state. The DEVS formalism turned out to be particularly well suited for implementing such a state quantisation approach, given that it is not limited to a finite number of system states, which is in contrast to many other discrete-event simulation techniques. The *Quantised State Systems* (QSS) introduced by Kofman in 2001 ([6]) improved the original quantised state approach of Zeigler by avoiding the problem of ever creating illegitimate models, and hence gave rise to efficient DEVS simulation of large and complex systems.

The simulation of a continuous system by a (discrete) DEVS model comes with several benefits: When using discretisation of time, variables have to be updated synchronously<sup>1</sup>. Thus, the time steps have to be chosen according to the variable that changes the fastest, otherwise a change in that variable could be missed. In a large system where probably very slow

but also very fast variables are present, this is critical to computation time, since the slow variables have to be updated too often. The DEVS formalism however allows for asynchronous variable updates, whereby the computational costs can be reduced significantly: every variable updates at its own speed; there is no need anymore for an adaptation to the fastest one in order not to miss important developments between time steps. This property could be extremely useful in stiff systems that exhibit widely spread eigenvalues, i.e., that feature mixed slow and fast variables. The DEVS formalism is very well suited for problems with frequent switching operations such as electrical power systems. Given that the problem of iteration at discontinuities does not apply anymore, it even allows for real-time simulation.

For hybrid systems with continuous-time, discrete-time, and discrete-event parts, a discrete-event method provides a ‘unified simulation framework’: discrete-time methods can be seen as a particular case of discrete-event methods ([6]), and continuous-time parts can be transformed in a straightforward manner to discrete-time/discrete-event systems. When using the QSS approach of Kofman in order to transform a continuous system into a corresponding discrete system, there exists a closed formula for the global error bound ([2]), which allows a mathematical analysis of the simulation.

<sup>1</sup> Note that this is not true for methods with dense output. However, the above statement holds for the majority of today’s integration methods, since they rarely make use of dense output.

Since the mid seventies, when Zeigler introduced the DEVS formalism ([11]), there have emerged several DEVS implementations, most of them designed to simulate discrete systems. However, one simulation/modelling software system aimed at simulating continuous systems is PowerDEVS ([8]): it provides a library consisting of block diagram component models that can be used for modelling any system described by ODE's (DAE's), thereby allowing for the simulation of continuous systems.

The implementation of ModelicaDEVS has been kept close to the PowerDEVS simulation software. Hence ModelicaDEVS can be considered a re-implementation of PowerDEVS in Modelica.

## 1 Continuous System Simulation with DEVS

### 1.1 The DEVS Formalism

The DEVS formalism has been introduced by Zeigler in 1976 ([11]). It was the first methodology designed for discrete-event system simulation based on system theory. A DEVS model has the following structure:

$$M = \langle X, Y, S, \delta_{int}(s), \delta_{ext}(s, e, x), \lambda(s), ta(s) \rangle$$

where the variables have the following meaning (see also [2], Chapter 11):  $X$  represents all possible inputs,  $Y$  represents the outputs, and  $S$  is the set of states. The variable  $e$  indicates the amount of time the system has already spent in the current state.  $\delta_{ext}(s, e, x)$  is the external transition that is executed after an external event has been received.  $\delta_{int}(s)$  is the internal transition that is executed as soon as the system has spent in its current state the time indicated by the time-advance function.

$ta(s)$  is the so-called *time advance function* that indicates how much time has to pass until the system undergoes the next internal transition. The time-advance function is often represented by variable  $\sigma$  which holds the value for the amount of time that the system has to remain in its current state in the absence of external events. The  $\lambda$ -function is the output function. It is executed prior to performing an internal transition. External transitions do not produce output.

Figure 1 illustrates the functioning of a DEVS model: the system receives input (top graph), changes its states according to the internal and external transitions (center graph), and produces output (bottom graph).

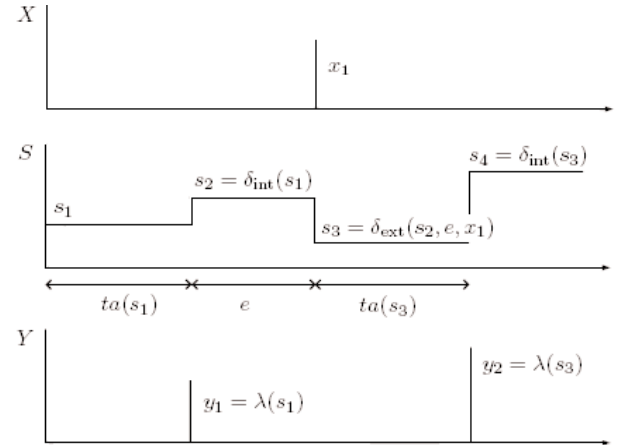


Figure 1: Trajectories in a DEVS model.

In theory, DEVS models can describe arbitrarily complex systems. The only drawback is that the more complex the system is, the more difficult it will be to set up the correct transition functions describing the system. Fortunately, complex systems can be broken down into simpler submodels that are easier to handle.

The fact that DEVS is closed under coupling [2] makes such an approach viable. Figure 2 illustrates this concept: the model  $N$  consists of two coupled atomic models  $M_a$  and  $M_b$ .  $N$  can be said to wrap  $M_a$  and  $M_b$  and is indistinguishable from the outside from an atomic model.

### 1.2 Quantised State Systems

For a system to be representable by a DEVS model, it must exhibit an input/output behaviour that is describable by a sequence of events. In other words, the DEVS formalism is able to model any system with piecewise constant input/output trajectories, since piecewise constant trajectories can be described by events ([2]).

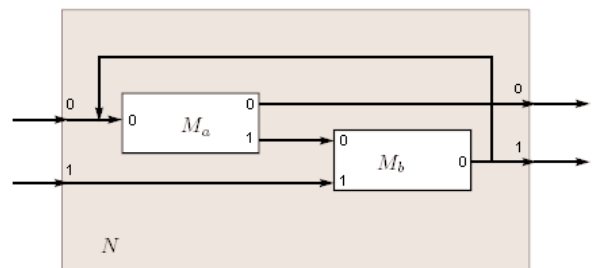


Figure 2: Coupled DEVS models [2].

Continuous state variables are being quantised. Consider the following system represented by the state-space description:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t)$$

where  $\mathbf{x}(t)$  is the state vector and  $\mathbf{u}(t)$  is the input vector, i.e. a piecewise constant function. The corresponding quantised state system has the following form:

$$\dot{\mathbf{q}}(t) \approx \mathbf{f}(\mathbf{q}(t), \mathbf{u}(t), t)$$

where  $\mathbf{q}(t)$  is the (componentwise) quantised version of the original state vector  $\mathbf{x}(t)$ . A simple quantisation function could be:

$$\mathbf{q}(t) = \text{floor}(\mathbf{x}(t))$$

Unfortunately, the transformation of a continuous system into its discrete counterpart by applying an arbitrarily chosen quantisation function can yield an illegitimate model<sup>2</sup>. Thus, the quantisation function has to be chosen carefully, such that it prevents the system from switching states with an infinite frequency.

This property can be achieved by adding hysteresis to the quantisation function [6], which leads to the notion of a Quantised State System (QSS) as introduced by Kofman [6] providing legitimate models that can be simulated by the DEVS formalism.

A hysteretic quantisation function is defined as follows [2]: Let  $Q = \{Q_0, Q_1, \dots, Q_r\}$  be a set of real numbers where  $Q_{k-1} < Q_k$  with  $1 \leq k \leq r$ . Let  $\Omega$  be the set of piecewise continuous trajectories, and let  $x \in \Omega$  be a continuous trajectory. The mapping  $b: \Omega \rightarrow \Omega$  is a hysteretic quantisation function if the trajectory  $q = b(x)$  satisfies:

$$q(t) = \begin{cases} Q_m & \text{if } t = t_0 \\ Q_{k+1} & \text{if } x(t) = Q_{k+1} \wedge q(t^-) = Q_k \wedge k < r \\ Q_{k-1} & \text{if } x(t) = Q_k - \varepsilon \wedge q(t^-) = Q_k \wedge k < 0 \\ q(t^-) & \text{otherwise} \end{cases}$$

$$\text{and } m = \begin{cases} 0 & \text{if } x(t_0) < Q_0 \\ r & \text{if } x(t_0) \geq Q_r \\ j & \text{if } Q_j \leq x(t_0) < Q_{j+1} \end{cases}$$

<sup>2</sup> Definition in [2]: ‘A DEVS model is said to be legitimate if it cannot perform an infinite number of transitions in a finite interval of time.’ Illustrative examples of illegitimate models can be found in [2] and [6].

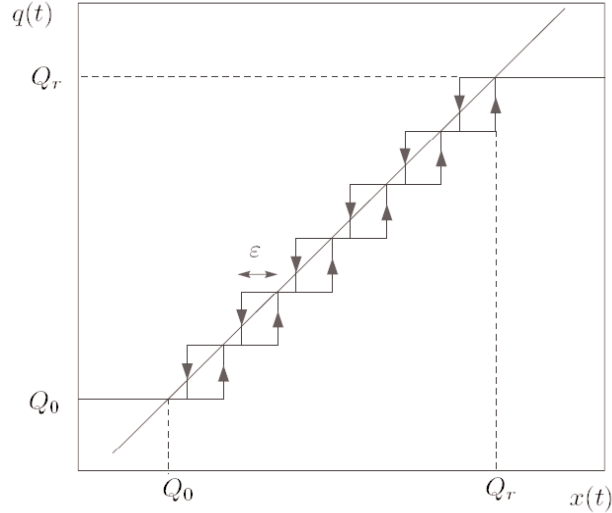


Figure 3: Quantisation function with hysteresis [2].

The discrete values  $Q_i$  and the distance  $Q_{k+1} - Q_k$  (usually constant) are called the quantisation levels and the quantum, respectively. The boundary values  $Q_0$  and  $Q_r$  are the upper and the lower saturation values, and  $\varepsilon$  is the width of the hysteresis window. Figure 3 shows a quantisation function with uniform quantisation intervals.

The QSS described above is a first-order approximation of the real system trajectory. Kofman however has also introduced second- and third-order approximations that may reduce the error made by the approximation. These systems are referred to as QSS2 ([7]) and QSS3 ([9]), respectively.

## 2 ModelicaDEVS

The average block of the ModelicaDEVS library exhibits the following basic structure:

```

1 block SampleBlock
2 extends ModelicaDEVS.Interfaces. ... ;
3 parameter Real ... ;
4
5 protected
6 discrete Real lastTime(start=0);
7 discrete Real sigma(start=...);
8 Real e;
9 Boolean dext;
10 Boolean dint;
11 [...other variable declarations...]
12
13 equation
14 dext = uEvent;
15 dint = time>=pre(lastTime)+pre(sigma);

```



```

16
17 when { dint } then
18   yVal[ 1 ] = ...;
19   yVal[ 2 ] = ...;
20   yVal[ 3 ] = ...;
21 end when;
22 yEvent = edge(dint);
23
24 when { dint, dext } then
25   e=time-pre(lastTime);
26   if dint then
27     [..internal transition behaviour..]
28   else
29     [..external transition behaviour..]
30   end if;
31   lastTime=time;
32 end when;
33
34 end SampleBlock;

```

The following sections will offer more insight into the reasons for this specific block structure.

In accordance with the PowerDEVS implementation, ModelicaDEVS event ports (connectors) consist of four variables representing the coefficients to the first three terms (constant, linear, and quadratic) of the function's Taylor series expansion, and a Boolean value that indicates whether a block is currently sending an event.

Dense output can then be approximated as:

$$y_{out} = y_0 + y_1 \cdot (t - t_{last}) + y_2 \cdot (t - t_{last})^2$$

whereby the coefficient of the quadratic term of the Taylor series,  $y_2 = yVal[ 3 ]$ , is only used by the third-order accurate method, QSS3, whereas the linear term,  $y_1 = yVal[ 2 ]$ , is used by QSS2 and QSS3.

Let us now consider a small example in order to gain increased insight into the role of the Boolean variable of the port. Let us assume a two-block system consisting of block A and block B, where the only input port of block B is connected to the only output port of block A. Every block features a variable `dext` accompanied by an equation

```
dext = uEvent;
```

where `uEvent` is the Boolean component of the connector that represents an input event. Suppose now that block A produces an output event at time  $t=3$ . At this precise instant, it updates its output vector with the appropriate values (the coefficients of the Taylor series) and sets `A.yEvent` to true:

```

when dint then
  yVal[ 1 ] = ...; //new output value 1
  yVal[ 2 ] = ...; //new output value 2
  yVal[ 3 ] = ...; //new output value 3
end when;
yEvent = edge(dint);

```

Still at time  $t=3$ , block B notices that now `B.uEvent` has become true (note that `B.uEvent = A.yEvent` because the two blocks are connected), and therefore `dext` has become true, also. Consequently, Block B is executing its external transition [4].

A DEVS model must contain code to perform internal and external transitions, as well as execute the time advance and output functions at the appropriate instants. All of these functions have to be explicitly or implicitly present in the ModelicaDEVS blocks.

The *time advance function* is normally represented by a variable `sigma`. It is a popular trick in DEVS to represent the current value of the time advance function by `sigma` [2].

The internal transition is executed when `dint` is true. An internal transition depends only on `sigma`. Hence the value of `dint` can be calculated as:

```
dint = time >= pre(lastTime) + pre(sigma);
```

where `lastTime` holds the time of the last execution of a transition (internal or external).

The external transition is executed when `dext` is true. The variable `dext` is defined as follows:

```
dext = uEvent;
```

The internal and external transitions are represented by a `when` statement. The reason for packing the internal and external transitions into a single `when` statement instead of having two separate `when` statements, one representing the internal transition and the lambda function, the other one representing the external transition, is due to a rule of the Modelica language specification that states that equations in different `when` statements may be evaluated simultaneously.

Hence, if there are two `when`-statements each containing an expression for a variable  $X$ ,  $X$  is considered overdetermined. This circumstance would cause a syntactical problem with variables that have to be updated both during the internal and the external transition and thus would have to appear in both `when`-statements.

For this reason, we need to have a `when` statement that is active if either `dint` or `dext` becomes true. Subsequently, an additional discrimination is done *within* the `when`-statement, determining whether it was an internal (`dint` is true) or an external transition (`dext` is true) that made the `when`-statement become active, and as the case may be, updating the variables with the appropriate value.



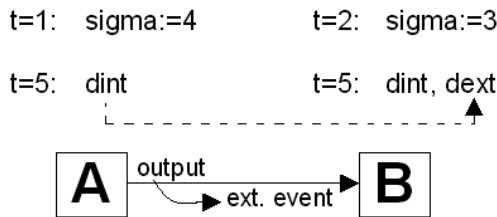


Figure 4: Concurrent events at block B.

The *lambda function* is executed right before an internal transition. Lines 17-22 of the ‘block basic structure’ code (beginning of Section 2) constitute the typical lambda function part, containing a *when-state* statement and a separate instruction for the *yEvent* variable. The right hand side of the equations in the lambda function normally depends on *pre()* values of the used variables. This is due to the fact that the lambda function has to be executed *prior* to the internal transition. The variable *yEvent* has to be true in the exact instant when an internal transition is executed and false otherwise. This behaviour is obtained by using the Modelica *edge()* operator.

There is one particular situation that can occur in a model that requires special attention: let us assume two connected blocks, where both block A and block B have to execute an internal transition simultaneously (Figure 4). Whereas block A simply executes its internal transition, block B is confronted with the problem of concurrent events: from block A it receives an external event, but at the same time, it was about to undergo its own internal transition. Which event should be processed first? This question is to be answered by the priority settings between the two blocks.

In our simple two-block example there are only two possible priority orderings with the following consequences: either block A is prior to block B, and block A will produce the output event before block B executes the internal transition (block B will first execute an external transition triggered by the external event it received from block A), or block B is prior to block A, such that block B will first undergo its internal transition and receive the external event right afterwards, when A will be allowed to execute its internal transition.

The problem of block priorities can be solved in two ways: by an explicit, absolute ordering of all components in a model (e.g., a list), or by letting every block determine itself whether it processes the external or the internal event first, in case both of them occur simultaneously.

ModelicaDEVS implements the latter approach. As can be seen in the ‘block basic structure’ code, internal transitions take always priority over external transitions (line 26: the code checks first whether *dint* is true).

The reason for this choice is quite simple. As internal events are processed before external events, and since internal events are accompanied by output events, the variable *yEvent* can be computed as a function of *dint* alone. If we were to force external events to be processed before internal events, we would need to make sure that *yEvent* is only set true in the case that the internal event is not accompanied by a simultaneous external event. Thus *yEvent* would now be a function of both *dint* and *dext*. Yet, *dext* is a function of *uEvent*. Thus, if ModelicaDEVS blocks were connected in a circular fashion, as this is often the case, an algebraic loop in discrete (Boolean) variables would be created, which would get the Dymola compiler into trouble.

By forcing the internal events to always take preference over external events, ModelicaDEVS blocks can be interconnected in an arbitrary fashion without ever creating algebraic loops in the Boolean event indication variables. Note that since Dymola/Modelica is already aimed at object-oriented modelling, which includes the re-use of multi-component models as parts of larger models, the issue of hierarchically coupled models did not require any special treatment in ModelicaDEVS.

Dymola can trigger two types of events: state events that require iteration to locate the event time, and time events that make Dymola ‘jump’ directly to the point in time when the time event takes place. The only expressions responsible for activating the *when* statements in the models, namely:

```
dext = uEvent;
```

and:

```
dint = time >= pre(lastTime) + pre(sigma);
```

both trigger time events and hence avoid the computationally more expensive state events.

An earlier version of ModelicaDEVS used an approach that triggered mostly state events. Inspired by the book of Fritzson ([4]), a number of small modifications have been applied that converted all state events to time events. Performance comparisons carried out between the two versions showed that the time-event approach is roughly four times faster than an equivalent approach triggering state events.

### 3 Results

#### 3.1 Efficiency

In order to compare the run-time efficiency of ModelicaDEVS to other simulation software systems (PowerDEVS and standard Dymola), a system with frequent switching operations was modelled using each of the three tools (PowerDEVS, ModelicaDEVS and Dymola), and the execution times of the three codes were compared against each other.

The chosen system is the flyback converter example presented in [5]. The flyback converter can be used to transform a given input voltage to a different output voltage. It belongs to the group of DC-DC converters. A very simple electrical circuit with a voltage source connected to the primary winding of the converter and a load to its secondary winding looks as shown in Figure 5. Figure 6 shows the first two milliseconds of a simulation run of the flyback converter circuit given in Figure 5. The rapid switching is a result of the high switching rate of the ideal switch.

The flyback converter is described by a set of acausal equations in Dymola. However, in order to be able to model the flyback converter in either ModelicaDEVS or PowerDEVS, the behaviour of the converter needs to be converted to a causalised block diagram<sup>3</sup>, which then can be modelled using component models of the PowerDEVS/ModelicaDEVS libraries.

Figure 7 shows the flyback converter model built in ModelicaDEVS. The structure of this block diagram is also valid for the PowerDEVS model.

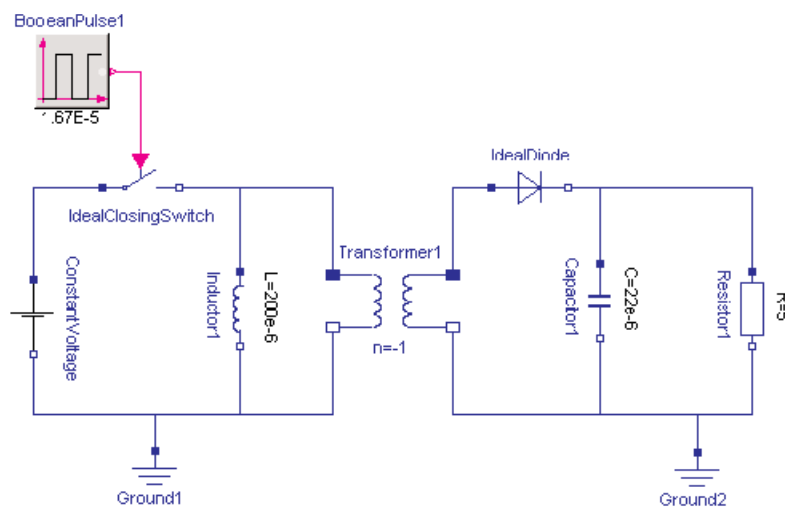


Figure 5: The flyback converter in Dymola.

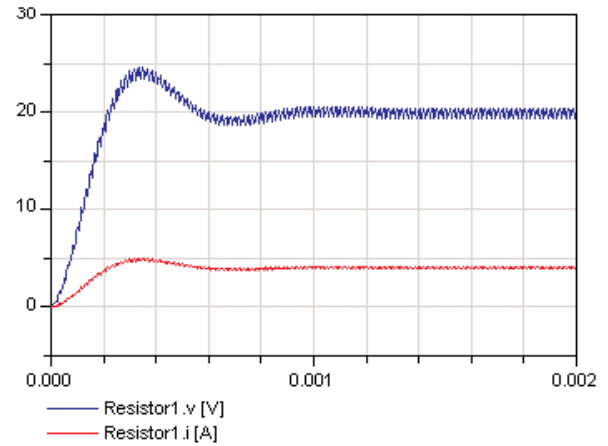


Figure 6: The flyback converter output.

	CPU time [s]	time events	result points
<b>Dymola</b>	0.062	239	738
<b>M-DEVS</b>	QSS1 3.55	6363	11829
	QSS2 0.688	958	2299
	QSS3 0.656	833	2164
<b>P-DEVS</b>	QSS1 0.064	N/A	N/A
	QSS2 0.019	N/A	N/A
	QSS3 0.018	N/A	N/A

Table 1: Execution efficiency comparison.

Table 1 provides the average simulation CPU time for a simulation of 0.002 seconds of the flyback converter model in standard Dymola, ModelicaDEVS, and PowerDEVS, respectively. The Dymola and ModelicaDEVS model were simulated setting the numerical integration method to LSODAR<sup>4</sup>. Testing has been carried out on an IntelCeleron 2.6 GHz Laptop with 256MB RAM. The resulting CPU time may vary from one computer system to another, but the relative ordering is expected to remain the same.

<sup>3</sup> For more details on the causalising process in the flyback converter example, see [1].

<sup>4</sup> Although ModelicaDEVS does not make use of LSODAR directly, the event handling behaviour of Dymola is somewhat influenced by the selection of the numerical integration algorithm.

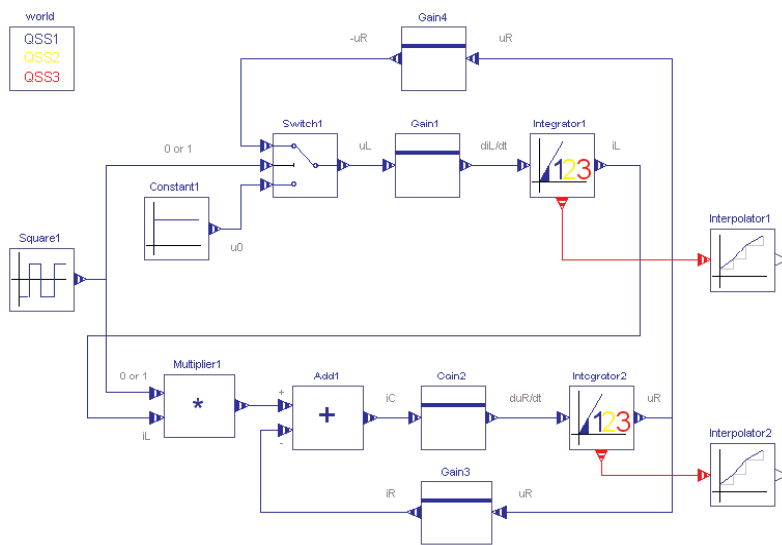


Figure 7: The ModelicaDEVS flyback converter.

Table 1 shows a clear ordering of the three different systems in terms of performance: PowerDEVS is faster than Dymola, which in turn is faster than ModelicaDEVS.

First, it needs to be remarked that standard Dymola simulates this model very efficiently. The switching (BooleanPulse) block leads to time events only, whereas the diode should lead to state events. Yet, this is not the case.

Switching at the input leads immediately to a switching of the diode as well. Since Dymola iterates after each event to determine a consistent set of initial conditions, the switching of the diode is accomplished at once without need of first triggering a state event.

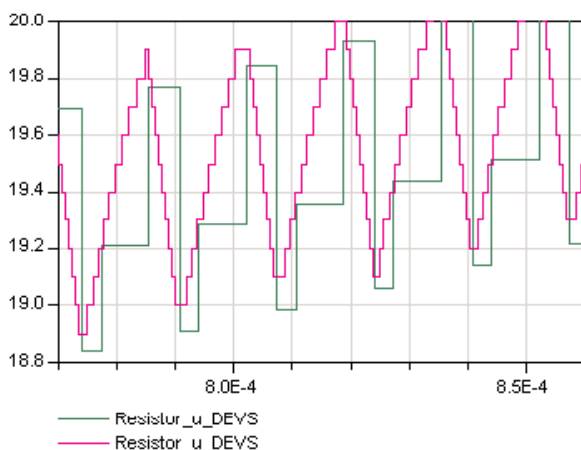


Figure 8: QSS3 simulation vs. QSS1 simulation.

Second, the model is quite trivial. The execution time is almost entirely dictated by the number of time events handled. What happens in between events is harmless in comparison. Standard Dymola performs exactly one time event per switching. In contrast, ModelicaDEVS performs considerably more time events. Time events take here the role of integration steps.

Figure 8 shows the constant term of the Taylor series expansion of the load voltage as a function of time for QSS1 and QSS3. QSS1 requires a new time event as soon as the constant output no longer represents the true output, whereas QSS3 requires an event only, when the second-order accurate Taylor series expansion no longer approximates the true output.

QSS1 requires roughly eight times as many events as QSS3, and is therefore between five and six times slower. Yet, even QSS3 requires roughly three times as many events as standard Dymola. In addition, the ModelicaDEVS model contains roughly three times as many variables as the standard Dymola model. All of these variables are being stored at every event. Consequently, QSS3 is roughly nine times slower than standard Dymola. Yet, QSS3 in PowerDEVS is roughly three times *faster* than standard Dymola for comparable accuracy.

A comparison between PowerDEVS and ModelicaDEVS is not straightforward. PowerDEVS implements Zeigler’s hierarchical simulator [12], whereas ModelicaDEVS operates on simultaneous equations and synchronous information flow [10]. Consequently, PowerDEVS suffers from requiring message passing to implement the communication between blocks, but enjoys the advantage of only having to process those equations that are directly involved with the event being handled. In contrast, ModelicaDEVS needs to visit all equations of all blocks whenever an event takes place. Which variables are to be updated in each case is decided by Boolean expressions associated with the various when-statements.

Yet the true difference in speed has probably more to do with the event handling itself. Dymola has been designed for optimal speed in the simulation of continuous models and for optimal robustness in handling hybrid models.

The algorithms implemented in Dymola for robust event handling are important in the context of hybrid modelling. In the context of a pure discrete-event simulation, these algorithms are an overkill.

For example, in a pure discrete-event simulation there is no need for iteration after each event to determine a new consistent set of initial conditions. In Dymola, many variables are being stored internally in order to allow LSODAR to integrate continuous state equations correctly across discontinuities. In a pure discrete-event simulation, variables need to be stored for output only.

### 3.2 Mixed Systems

Mixed systems contain both Dymola and Modelica blocks. Figure 9 shows an example of a simple electrical circuit modelled in Dymola, and in a mixed version with a ModelicaDEVS capacitor.

Figure 10 illustrates the implementation of the ModelicaDEVS capacitor. On its outside, this block looks like a normal electrical Dymola component, but internally it consists of ModelicaDEVS blocks that model the behaviour of a capacitor: The Gain block multiplies the incoming signal by the value of  $1/C$ , where  $C$  is specified by a parameter, and passes it on to the Interpolator.

Taken as a whole, the ModelicaDEVS blocks constitute nothing more than the well known capacitor formula

$$v = \frac{1}{C} \int i \, dt$$

Unfortunately, it is not as straightforward as it may seem at first glance to replace a component from the Dymola standard electrical library by its ModelicaDEVS equivalent: since the electrical components do not assume a certain data flow direction (they are described by acausal equations), whereas the ModelicaDEVS components do (DEVS components feature input and output ports), the ModelicaDEVS capacitor must turn acausal equations into causal ones. It assumes the capacitive current  $i$  to be given, and hence computes the capacitive voltage  $v$ . Note that such a capacitor would not work anymore correctly if we were to connect it to a voltage source instead of a current source.

An even more severe problem is caused by the SamplerTime block applying the  $\text{der}()$  operator to the signal that it receives through its input port:

```
du=der(u);
when sample(start,period) then
yVal[ 1]=u;
yVal[ 2]=if method>1 then du else 0;
yVal[ 3]=if method>2 then der(du) else 0;
end when;
```

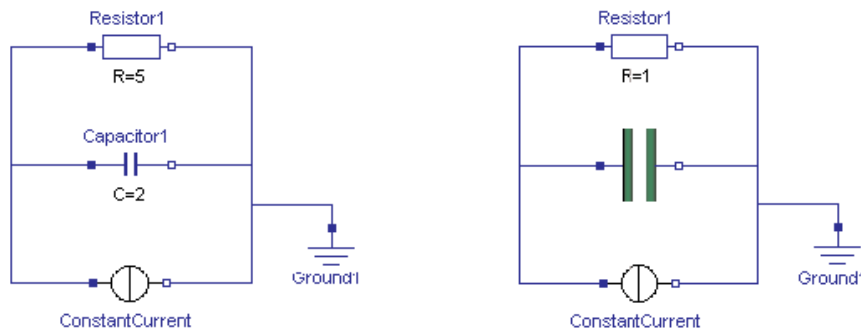


Figure 9: Two versions (Dymola and Dymola/ModelicaDEVS) of a simple electrical circuit.

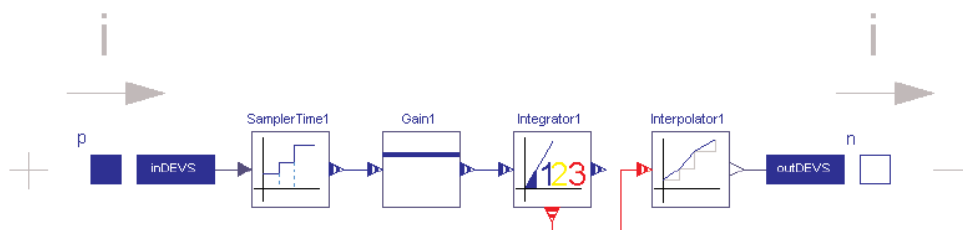


Figure 10: The internal structure of the ModelicaDEVS capacitor.



Given that the input of the `SamplerTime` block depends algebraically on the output of the `Interpolator` in the DEVS capacitor, Dymola would have to differentiate discrete variables, which it is unable to do. An attempt to solve this problem was made using Dymola's 'User specified Derivatives' feature described in the Dymola User's Manual ([3]): functions for the first and second derivatives have been inserted into the `Interpolator`, but due to unknown reasons, this did not resolve the issue either.

In order to be able to perform mixed simulations nonetheless, another trick has been applied: supplementary to the standard ModelicaDEVS `SamplerTime` block that uses the Modelica `der()` operator, an additional block has been programmed: the `SamplerTimeNumerical` block avoids the problem caused by the `der()` operator by means of the `delay()` function that is used to differentiate the input variable numerically. Instead of the first and second derivatives of the input signal, the `SamplerTimeNumerical` returns a numerical approximation:

```
Du = delay(pre(u), D);
D2u = delay(pre(u), 2*D);

yVal[ 1] = pre(u);
yVal[ 2] = if method>1 then
    (pre(u) - Du) / D
    else 0;
yVal[ 3] = if method>2 then
    (pre(u) - 2*Du + D2u) / (D*D)
    else 0;
```

Using the new sampler block, the mixed simulation could be carried out without any problems, and the results differ only slightly from the simulation with conventional Dymola components (see Figure 11).

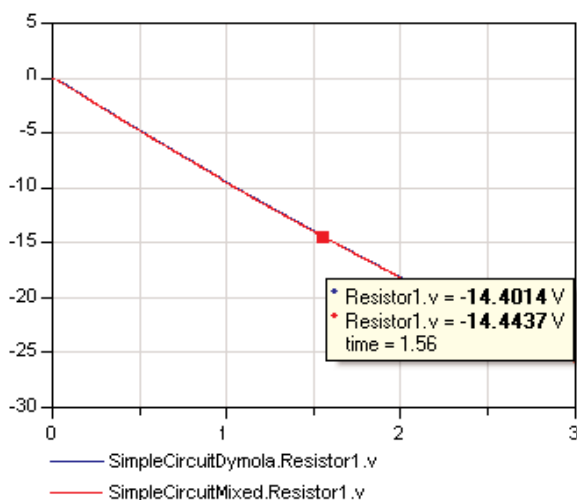


Figure 11: Standard Dymola (blue) and mixed (red) simulation of the simple electrical circuit (Figure 9).

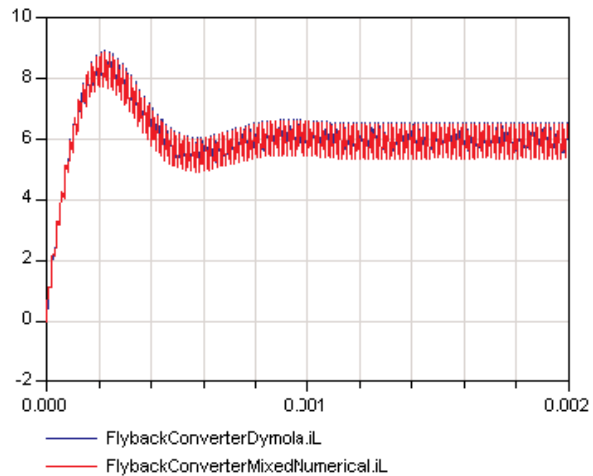


Figure 12: Standard Dymola (blue) and mixed (red) simulation of the flyback converter.

### 3.3 Hybrid Systems

Hybrid systems contain mixed integration methods: standard Modelica integrators and ModelicaDEVS Integrator blocks. An example of a hybrid system is for instance an electrical circuit with at least one ModelicaDEVS capacitor/inductor (using the ModelicaDEVS Integrator block) and at least one Dymola capacitor/inductor (using the Modelica `der()` operator).

The flyback converter of Section 3.1, where the capacitor in the secondary winding is replaced by an equivalent ModelicaDEVS capacitor, may serve as an example of a hybrid system.

Note that the ModelicaDEVS capacitor applies numerical differentiation in order not to obtain 'DAE index reduction' error messages (see previous section).

Figure 12 shows the output of the mixed simulation compared to the result of the standard Dymola simulation.

Just as it was the case with the simpler example of Section 3.2, the output of the hybrid simulation differs only slightly from the Dymola simulation. Thus, it is also possible to perform not only accurate<sup>5</sup> mixed simulations, but also hybrid simulations.

<sup>5</sup> Note that due to the numerical differentiation used in the `SamplerTimeNumerical` block, the result is not as accurate as if analytical differentiation had been used. However, the accuracy is sufficient for most purposes, and also adjustable through selection of the width parameter,  $D$ .

## 4 Conclusions

A new Dymola/Modelica library implementing a number of Quantised State System (QSS) simulation algorithms has been presented. ModelicaDEVS duplicates the capabilities of PowerDEVS. The graphical user interfaces of both tools are practically identical.

However, the underlying simulators are very different. Whereas PowerDEVS implements Zeigler's hierarchical DEVS simulator, ModelicaDEVS operates on simultaneous equations and synchronous information flows.

The embedding of ModelicaDEVS within the Dymola/Modelica environment enables users to mix DEVS models with other modelling methodologies that are supported by Dymola and for which Dymola offers software libraries.

Unfortunately, ModelicaDEVS is much less efficient in runtime performance than PowerDEVS. The loss of runtime efficiency is probably caused by Dymola's event handling algorithms that have been designed for optimal robustness in the context of hybrid system simulation rather than runtime efficiency in the context of pure discrete-event system simulation.

Although ModelicaDEVS offers a full implementation of a DEVS kernel and can therefore be used for the simulation of arbitrary discrete-event systems, the modelling blocks that have been made available so far in ModelicaDEVS are geared towards the simulation of continuous systems using QSS algorithms.

## References

- [1] T. Beltrame: *Design and Development of a Dymola/Modelica Library for Discrete Eventoriented Systems using DEVS Methodology*. MS Thesis, Institute of Computational Science, ETH Zurich, Switzerland, 2006.
- [2] F. E. Cellier, E. Kofman: *Continuous System Simulation*. Springer-Verlag, New York, 2006.
- [3] Dynasim AB: *Dymola Users' Manual, Version 6.0*. Lund, Sweden, 2006.
- [4] P. Fritzson: *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Wiley-Interscience, New York, 2004.
- [5] J.S.Glaser,F.E. Cellier, A.F. Witulski: *Object-Oriented Switching Power Converter Modeling Using Dymola With Event-Handling*. Proc. OOS'95, SCS Object-Oriented Simulation Conference, Las Vegas, NV, pp.141-146, 1995.
- [6] E. Kofman, S. Junco: *Quantised State Systems: A DEVS Approach for Continuous Systems Simulation*. Transactions of SCS, 18(3), pp.123-132, 2001.
- [7] E. Kofman: *A Second Order Approximation for DEVS Simulation of Continuous Systems*. Simulation, 78(2), pp.76-89.
- [8] E. Kofman, M. Lapadula, E. Pagliero: *PowerDEVS: A DEVS-based Environment for Hybrid System Modeling and Simulation*. Technical Report LSD0306, LSD, Universidad Nacional de Rosario, Argentina, 2003.
- [9] E. Kofman: *A Third Order Discrete Event Method for Continuous System Simulation*. Latin American Applied Research, 36(2), pp.101-108.
- [10] M. Otter, H. Emqvist, Mattsson: *Hybrid Modeling in Modelica Based on the Synchronous Data Flow Principle*. CACSD'99, IEEE Symposium on Computer-Aided Control System Design, Hawaii, pp.151-157, 1999.
- [11] B. P. Zeigler: *Theory of Modeling and Simulation*. John Wiley & Sons, New York, 1976.
- [12] B. P. Zeigler: *Multifaceted Modelling and Discrete Event Simulation*. Academic Press, London, 1964.
- [13] B. P. Zeigler, J.S. Lee: *Theory of Quantized Systems: Formal Basis for DEVS/HLA Distributed Simulation Environment*. SPIE Proceedings, Vol. 3369, pp.49-58, 1998.

**Corresponding author:** Tamara Beltrame  
Tamara Beltrame, VTT, Industrial Systems, VM3  
PO Box 1000, 02150 Espoo, Finland  
[Tamara.Beltrame@vtt.fi](mailto:Tamara.Beltrame@vtt.fi)  
François E. Cellier, Institute of Computational Science  
ETH Zurich, 8092 Zurich, Switzerland  
[fcellier@inf.ethz.ch](mailto:fcellier@inf.ethz.ch)

Received: October, 10, 2006

Accepted: December 5, 2006

This contribution has been published first in Ch. Kral, A. Haumer (eds): *Proceedings of the 5th International Modelica Conference 2006, Vienna, Austria* (publishers The Modelica Association ([www.modelica.org](http://www.modelica.org)) and arsenal research ([www.arsenal.ac.at](http://www.arsenal.ac.at)), 2006; pp 73-82. It has been suggested for publication in SNE by SNE's Editorial Board.

# Two-dimensional Finite Element Model for Thermodynamic and Continuum Mechanical Processes within the Snow Cover

Harald Teufelsbauer, University of Natural Resources and Applied Life Sciences, Vienna  
*harald.teufelsbauer@boku.ac.at*

The goal of this work is to create a snow cover model which can be used in praxis. In addition, this model offers the possibility to calculate two dimensional cross sections of slopes. Thus, small structural differences within the snow cover can be located. The two dimensional snow pack modelling should also allow the analysis of continuum mechanic correlations, better than the one dimensional model. Furthermore, it should offer the possibility to combine it with a snow drift model. A two dimensional snow pack model demands an extrapolation of the measured data, recorded by means of automatic gauging stations. By dint of an implicit Finite Element Method heat transfer and settlement can be calculated. The geometry can be adapted at any time step of the simulation process and it consequently fits with the cross section of the snow cover. For the evaluation of the simulated results, data of winter 1998/1999, 2004/05 and 2005/06 are available.

## Introduction

Modelling of physical processes within the snow pack is one of the major goals of snow science. A number of one-dimensional snow pack models like SNOWPACK ([2], [3], [4]) and CROCUS ([5]) already exists. In Austria physical snow pack modelling has started a couple of years ago. The goal is to create a simulation tool which can be easily combined with further wind and snow drift simulations. Two-dimensional modelling requires the processing of time variable geometries. The model describing differential equations are solved with the Finite Element Method which is implemented in MATLAB. Right now the work is focused on the calculation of snow temperatures and on a settlement simulation which includes the variation of density because of mechanical snow pack deformations.

## 1 Geometry and Model Input Data

### 1.1 Choosing Cross Section Geometry

It is provided to place an intersection along a slope and to calculate two dimensional snow-profiles along this cross section (Figure 1). The selected cross section of the slope shows a two-dimensional cut geometry, which is being triangulated using the Delaunay-Algorithm. Thereby it is regarded that the elements close to the snow surface are getting more and more smoothly (Figure 2).

### 1.2 Extrapolation of Measured Data

A two dimensional snow pack model requires the extrapolation of automatic gauging station measurements to other points of the surrounding area. Some data, for example air temperature or humidity will vary just a little within a narrow radius of the gauging station.

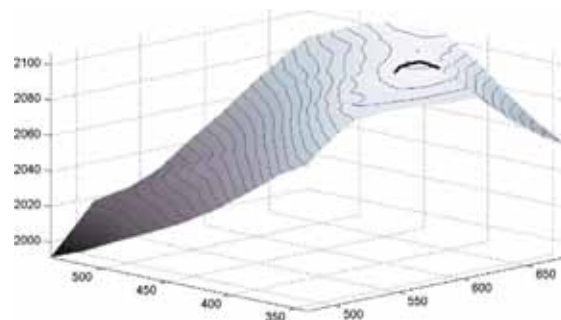


Figure 1: 3-D profile and selection of the two dimensional cut of the snow pack (marked with line across the ridge marks).

Compared to this, there are long wave emission data which are highly dependent on the snow surface temperature or the portion of short wave irradiation which depends on the exposition and the slope.

For a precise snow pack simulation exact input data are required. However, some measuring instruments react very sensitive to atmospheric influence and often provide unusable measurements at unfavourable weather conditions. For this reason the test series must be checked and mistakes must be corrected before a simulation can be started.

### Short wave radiation

The calculation of the wave angle  $\varphi$  between solar irradiation and the terrain's normal vector is essential for the calculation of the energy input [9]. By means of the wave angle the percentage of the measured short wave radiation which hits the particular area can be calculated on every single day of the year at any time. Furthermore it can be derived if a calculation point is located in the shadow of another point by using a digital elevation model (Figure 3). The intensity  $I(\varphi)$  of the incoming solar radiation is defined as  $I(\varphi) = \cos(\varphi)$ .

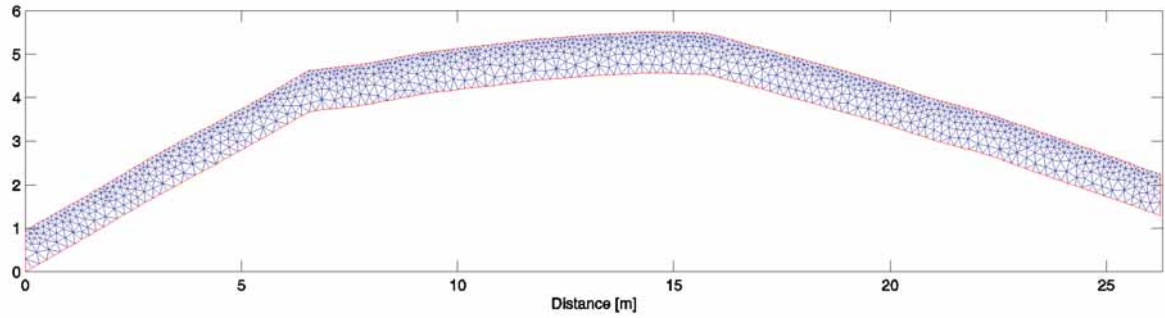


Figure 2: Triangulation of the two-dimensional cross section geometry.

Short wave radiation is measured by using two pyranometers. One pyranometer aiming downward measures the reflexion of short wave radiation on top of the snow surface whereas the second pyranometer aiming upwards measures the incoming radiation to a horizontal area. The albedo  $a$  refers to the quotient of irradiation and emittance:

$$a = \frac{kw_{out}}{kw_{in}}$$

The amount of reflected irradiation  $kw_{out}$  and thus also the albedo depends on the angle of insolation of the sun, the structure of the snow surface and on the degree of pollution of the snow surface. Problems while measuring short wave radiation occur at a low sun level since the energy flow is located in the error of measurement. Often the upper part of the radiation measuring instrument is covered by fresh fallen snow, it is frosted or steamed up. Unfortunately there is no possibility to calculate short wave radiation instead of measuring it. The only way to check the measured data is to compare incoming and reflected radiation. If unrealistic albedo values arise, an error of measuring should be considered. Since the measurements of the lower part of the radiation measuring device are often more reliable a relationship to the radiation can be established by means of the albedo.

In order to extrapolate the measuring data to the area, the radiation measured by the upper pyranometer has to be converted to a reference value  $kw_{ref}$ , representing the measuring of a pyranometer which would be permanently directed normal to the incoming sun rays. The angle  $\varphi_{ref}$  defines the angle between the normal vector of a horizontal plain and the incoming solar radiation.

The amount of energy  $E_{in}$  which finally arrives the snow surface is calculated as follows:

$$kw_{ref} = \frac{kw_{in}}{\cos(\varphi_{ref})}$$

$$E_{in} = (1-a) \cdot I(\varphi) \cdot kw_{ref}$$

The calculation of  $kw_{ref}$  provides good results, except for the time of sunrise and sunset. This time span is approximated by a bell-shaped curve.

The intensity of radiation within the snow pack decreases exponentially with penetration depth. Hence the source term of the heat equation  $Q$  is a function of penetration depth  $p_d$  and incoming net energy  $E_{in}$ :

$$Q(p_d) = u_1 \cdot E_{in} \cdot \exp(-u_2 \cdot p_d)$$

### Long wave radiation

The measurement of long wave radiation is carried out with non-ventilated pyrrometers, which are very sensitive against atmospheric conditions. Thus it has been tried to replace the measured values by calculated ones. Thereby it is paid attention that only robust data are used for the calculation. Long wave emittance  $lw_{out}$  can be calculated as a function of the snow surface temperature  $T_s$  [°K] by using the Stefan Boltzmann Equation:

$$lw_{out} = \sigma \cdot T_s^4 \quad \text{with } \sigma = 5.669 \cdot 10^{-8} \left[ \frac{W}{m^2} \right]$$

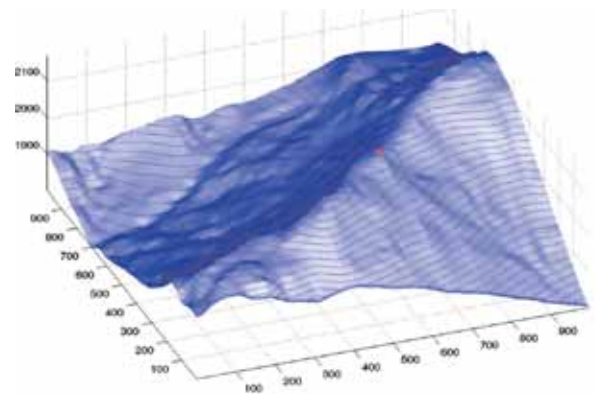


Figure 3: Calculation of intensity  $I$  and shadows with respect to exposition and slope, (2005/02/04; 13:00).



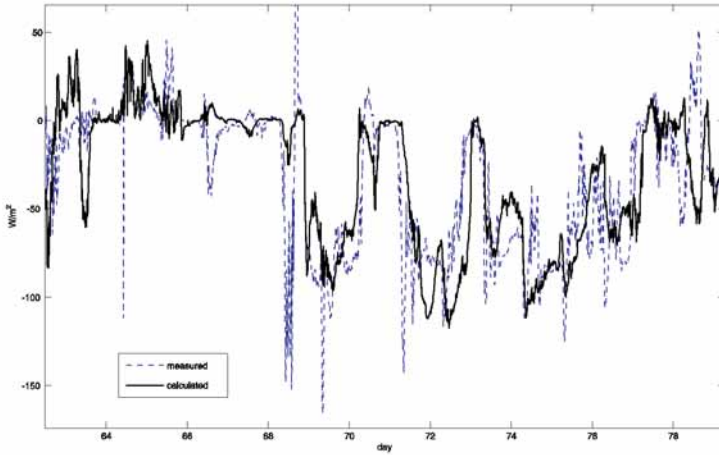


Figure 4: comparison between measured and calculated long wave radiation balance.

Calculating the long wave irradiation  $lw_{in}$  is more difficult. It strongly depends on air temperature, air humidity and sky cover. Since sky cover cannot be measured, the calculation is limited to an empiric link of air humidity  $rh$  and air temperature  $LT$  [°K]:

$$lw_{in} = 0.8 \cdot \frac{\min(\max(35, rh), 85)}{100} \cdot \sigma \cdot LT^4 + 90$$

The simulation of the snow surface temperature points out that a simulation with calculated long wave balance shows a better correlation to the measured surface temperatures than a simulation with the measured long wave balance. Consequently the long wave radiation measurements are only used as evaluation data and the snow temperature simulation can be run without long wave radiation measurements.

## 2 Temperature Simulation

Modelling the temperature distribution within the snow cover is based on the classic heat equation:

$$\rho_s c_s \frac{\partial T(x, y, t)}{\partial t} = \nabla \cdot (k_s(x, y, t) \nabla T(x, y, t)) + Q(x, y, t)$$

The parameters of the heat equation are combined of snow density  $\rho_s$ , specific heat capacity of snow  $c_s$ , thermal conductivity  $k_s$  and a source term  $Q$ . A lot of field and laboratory tests have already been performed to calculate thermal conductivity of snow [1] [3], [6], [7], [10]. They vary from simple empiric assumptions to detailed microstructural analyses. In most theories thermal conductivity is a function of snow density.

The boundary conditions of the heat equation are divided into three different types.

If the temperature on the surface boundary of the area is known one should prefer to use Dirichlet boundary conditions  $\Gamma_D$ . A possibility to describe a heat flow between snow surface and atmosphere is offered by Neumann boundary conditions  $\Gamma_N$  or rather hybrid boundary conditions  $\Gamma_H$ . If using this hybrid condition, not only heat flux via atmospheric radiation is considered, but also convective influences just like temperature difference between environment and surface coupled with wind velocity:

$$T = T_D \quad \text{on } \Gamma_D$$

$$k_s \frac{\partial T}{\partial n} = g \quad \text{on } \Gamma_N$$

$$k_s \frac{\partial T}{\partial n} = g + v \cdot (T_{ext} - T) \quad \text{on } \Gamma_H$$

The geometry of the two dimensional snow pack simulation has four separate selectable boundaries (see Figure 2). The boundary between snow cover and soil is described by a Dirichlet boundary condition. The boundary between snow surface and atmosphere is also defined as a Dirichlet boundary condition if the temperature of the snow surface is known, otherwise hybrid boundary conditions will be accepted. The left and right margins will be described by symmetrical conditions defined by a Neumann boundary condition whereas heat flow is zero. The hybrid boundary condition considers the following heat fluxes:

$$k \frac{\partial T}{\partial n} = lw + q_{latent} + q_{sensible}$$

$$lw = lw_{out} - lw_{in}$$

$$q_{sensible} = f(v_{wind}) \cdot (T_{air} - T_s)$$

$$q_{latent} = n \cdot v_{wind} \cdot \frac{L^{i/w} \cdot \rho_{air}}{\rho_{air}} (e_s^w(T_{air}) \cdot rh - e_s^l(T_s))$$

$$e_s^{i/w}(T) = p_t \cdot \exp\left(\frac{L^{i/w}(T - T_t)}{R_v \cdot T_t \cdot T}\right)$$

$Li = 238$ [kJ/kg] ...	latent heat of sublimation
$Lw = 2256$ [kJ/kg] ...	latent heat of vaporisation
$rh$ ...	relative humidity
$v_{wind}$ ...	wind speed
$p_t = 610,5$ [Pa] ...	triple point pressure
$T_t = 273,16$ [K] ...	triple point temperature
$R_v = 461,9$ [J/(kg K)] ...	specific gas constant
$n$ ...	empirical constant

The heat equation is solved by using a Finite Element Method with implicit time integration. Furthermore the calculation of the sensible heat flux  $q_{sensible}$  does not require a measurement of the surface temperature  $T_s$  due to implicit implementation.

To conserve linearity, the calculation of the terrestrial emittance  $lw_{out}$  is based on the surface temperature, calculated one time step before. The calculation of the latent heat  $q_{latent}$  is assumed from [4].

### 3 Settlement and Densification of Snow Pack

The calculation of the settlement and a combined snow densification can be determined by continuum mechanical correlations. The settlement is modelled independently of grain shape and water vapour diffusion inside the snow cover. The calculated densities are therefore just a result of the volume change during the settlement.

#### 3.1 Calculation of Snow Settling and Creeping

The settlement can be described by viscous and elastic deformations whereas the elastic part is very small. The correlation between viscosity  $\eta$ , stress  $\sigma$  and strain rate  $\dot{\epsilon}$  is described by

$$\dot{\epsilon} = \frac{1}{\eta} \sigma$$

Furthermore, correlations between stress tensor  $\sigma$ , vector of body force  $q$ , vector of displacement  $u$  and strain rate  $\dot{\epsilon}$  are defined by structural mechanics:

$$\begin{aligned} \text{div } \sigma = -q &\Leftrightarrow \nabla \cdot \begin{pmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{yx} & \sigma_{yy} \end{pmatrix} = - \begin{pmatrix} q_x \\ q_y \end{pmatrix} \\ &\Leftrightarrow \underbrace{\begin{pmatrix} \frac{\partial}{\partial x} & 0 & \frac{\partial}{\partial y} \\ 0 & \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{pmatrix}}_{=d^T} \cdot \begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{pmatrix} = - \begin{pmatrix} q_x \\ q_y \end{pmatrix} \end{aligned}$$

$$\begin{aligned} \epsilon = d \cdot u &\Leftrightarrow \dot{\epsilon} = d \cdot \dot{u} \\ &\Leftrightarrow \begin{pmatrix} \dot{\epsilon}_{xx} \\ \dot{\epsilon}_{yy} \\ 2\dot{\epsilon}_{xy} \end{pmatrix} = \begin{pmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{pmatrix} \cdot \begin{pmatrix} \dot{u}_x \\ \dot{u}_y \end{pmatrix} \end{aligned}$$

The body force  $(q_x, q_y)$  represents the force acting on each grid point, respectively each mass point in  $x$ - and  $y$ -direction.

By combination of the last three relations the movement  $u = (u_x, u_y)$  of the snow pack can be calculated. The rate of the snow pack settlement depends on the chosen viscosity  $\eta$ , which is determined empirically:

$$\eta = \begin{cases} (h_1 \cdot \rho)^{(h_2 - h_3 \cdot T)} & \text{for } \rho < \rho_g \\ \left( \frac{\rho_s}{\rho_g} \right)^5 \cdot (h_1 \cdot \rho)^{(h_2 - h_3 \cdot T)} & \text{for } \rho \geq \rho_g \end{cases}$$

The value  $\rho_g \sim 150 \text{ kg/m}^3$  is a threshold density, where the rate of settlement decreases very fast. The remaining parameters  $h_1, h_2, h_3$  are describing the influence of temperature  $T_s$  (in  $^{\circ}\text{C}$ ) and snow density  $\rho_s$ .

Besides the settlement by the tare weight of the snow cover, a snow drift simulation is added too. It is taken to account that snow drift increases with growing wind speed and declines with growing snow density:

$$sh_{drift} = 1,15 \cdot \frac{dt}{\rho_s^{2,5}} \cdot (\max(v_{start}, v_{wind}) - v_{start})^{0,8}$$

The parameter  $v_{start} \sim 4 \text{ m/s}$  is the threshold value, snow drift occurs and  $dt$  is the data logger's time interval of saving averaged wind speed values, measured every second.

#### 3.2 Calculation of Snow Density

The densification of the snow pack is directly caused by its movement. If snow is assumed to be compressible, the continuity equation with  $\vec{u}$  being velocity of settlement has the following form:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) = 0$$

The equations of heat transfer, settlement and mass conservation are solved using the Finite Element Method. The mesh is generated with a Delaunay algorithm and just one mesh is used, solving all equations. If the cross section geometry changes due to snow fall, snow drift or melting a new mesh must be generated.

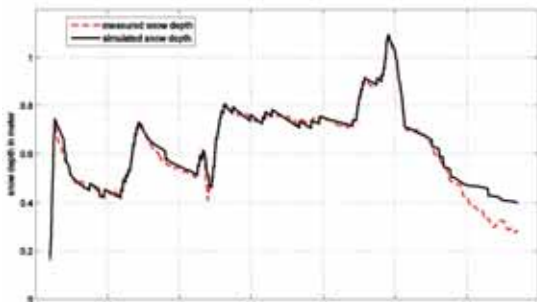


Figure 5: Calculated and measured snow depth from 2005/01/23 to 2005/03/16.

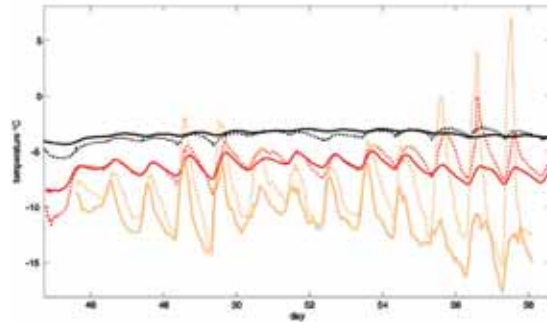


Figure 6: Calculated and measured snow temperatures in three different heights from 2005/02/14 to 2005/02/23.

## 4 Evaluation

For evaluating the snow pack model, automatic gauging stations are measuring data of the snow pack and atmosphere. These measurements allow to compare snow temperatures and total snow depth of the spatial point where the gauging station is installed. Hence the evaluation of a whole two dimensional cross section is not possible because measurement instruments are too expensive.

The following plots (Figure 5, Figure 6, and Figure 7) show the comparison between measurements and calculations. Dotted lines represent measured values and filled lines show the simulation results.

### 4.1 Evaluation of Snow Settlement and Snow Drift

The evaluation of snow settlement and snow drift can be done as comparison between measured and simulated total snow depth at the spatial point of the gauging station (Figure 5). The calculation starts with the measured snow depth at the beginning of the simulation run.

At any simulation time step the amount of fresh fallen snow is added. The simulation results indicate that the total snow depth can be calculated relative exactly. The consideration of snow drift shows a clear improvement in the accuracy of the results. Further improvements are expected from the modelling of melting processes, particularly at the end of winter.

### 4.2 Evaluation of Snow Temperatures

The comparison of the simulated and measured temperature distribution within the snow cover shows extensively good results. To compare the simulation with real snow temperatures, the automatic gauging station measures the vertical temperature gradient approximately all 20 cm.

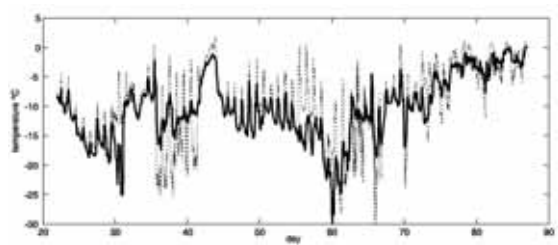


Figure 7: Calculated and measured snow surface temperatures from 2005/01/23 to 2005/03/16.

Figure 6 shows the comparison of snow temperatures, measured in 32 cm, 52 cm and 72 cm over soil. Due to a better presentation of the data, just an extract of 8 days is shown in Figure 6.

The evaluation of snow temperatures shows that near the snow surface the difference between measurement and calculation is higher than near the ground. This fact can be explained by a high influence of short wave radiation in the upper 30 cm of the snow pack. On the one hand simplification in the modelling cause some deviations and on the other hand erroneous measurements can be observed.

Temperature sensors near the snow surface can be heated by penetrating short wave radiation which results in raised temperatures, sometimes even over 0° C.

If using hybrid boundary conditions, the snow surface temperature is a result of the heat equation. The quality of the calculation of the surface temperature depends strongly on the accuracy of the measurements of atmospheric data like wind speed, short and long wave radiation, relative humidity air temperatures and so on. The comparison between measured and simulated snow surface temperatures is shown in Figure 7.

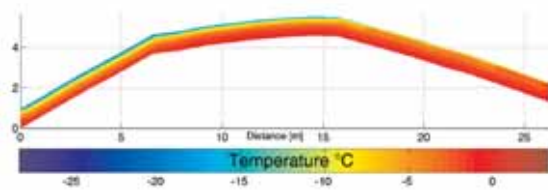


Figure 8: Illustration of different temperatures between north and south directed slopes.

The result of the two-dimensional simulation of the cross section which is shown in (Figure 2), illustrates temperature differences between south and north directed slopes (Figure 8).

## 5 Outlook

There are still a lot of enhancements that can be done, like the modelling of snow metamorphism, melting processes and surface hoar. The two-dimensional snow cover model will be linked to a wind simulation to get better input data for the simulation. Further improvements are expected by terrestrial laser scan technology [8], which allows a three dimensional scanning of mountainsides. By dint of this new technology more precise snow pack cross sections can be measured.

## Acknowledgments

This work is realised with the financial support given by 'Torrent and Avalanche Control Service'. This project has also been supported by the 'Skilifts Lech am Arlberg' who provided infrastructural devices.

## References

- [1] E. E. Adams, A. Sato: *Model for effective thermal conductivity of a dry snow cover composed of uniform ice spheres*. Annals of Glaciology 18 1993, 300-304, 1993.
- [2] P. Bartelt, M. Lehning: *A physical Snowpack model for the Swiss avalanche warning, Part I: numerical model*. Cold Reg. Sci. Technol. 35, 123 – 145, 2002.
- [3] P. Bartelt, M. Lehning: *A physical Snowpack model for the Swiss avalanche warning, Part II: snow microstructure*. Cold Reg. Sci. Technol. 35, 147-167, 2002.
- [4] P. Bartelt, M. Lehning: *A physical Snowpack model for the Swiss avalanche warning, Part III: meteorological forcing, thin layer formation and evaluation*. Cold Reg. Sci. Technol. 35, 169-184, 2002.
- [5] E. Brun, E. Martin, V. Simon, C. Gendre, C. Coleou: *An energy and mass model of snow cover suitable for operational avalanche forecasting*. Journal of Glaciology, Vol. 35, No. 121, 333-342, 1989.
- [6] T. J. L. McComb, A.B. Rimmer, M. L. B. Rodgers, K. E. Turver, A. F. Vivkers: *A mathematical model for the prediction of temperature in a dry snow layer*. Cold Reg. Sci. Technol. 20, 247-259, 1992.
- [7] E. M. Morris: *Modelling the flow of mass and energy within a snowpack for hydrological forecasting*. Annals of Glaciology 4, 198-203, 1983.
- [8] A. Prokop: *Hangbezogene Ermittlung der flächigen Schneehöhenverteilung mittels Laserscanners*. Journal of Torrent, Avalanche, Landslide and Rock Fall Engineering nr. 154, 2005.
- [9] C.-J. Winter, R.L. Sizmann, L. L. Vant-Hull: *Solar Power Plants*. Springer Verlag, 17-27, 1991.
- [10] T. Yamazaki, J. Kondo, T. Skuraoka, T. Nakamura: *A one-dimensional model of the evolution of snow cover characteristics*. 18, 22-26, 1993.

**Corresponding author:** Harald Teufelsbauer,  
Department of Structural Engineering  
& Natural Hazard  
Institute of Mountain Risk Engineering, University  
of Natural Resources and Applied Life Sciences,  
Peter Jordanstrasse 82, 1190 Vienna, Austria  
[harald.teufelsbauer@boku.ac.at](mailto:harald.teufelsbauer@boku.ac.at)

Received: October 15, 2006

Revised: November 20, 2006

Accepted: November 30, 2006



## Advanced Modeling and Simulation Techniques in MOSILAB: A System Development Case Study

Ch. Nytsch-Geusen, T. Ernst, A. Nordwig, Fraunhofer Inst. FIRST, Berlin; [christoph.nytsch@first.fhg.de](mailto:christoph.nytsch@first.fhg.de)  
P. Schwarz, P. Schneider, M. Vetter, Ch. Wittwer, A. Holm, T. Nouidui, J. Leopold,  
G. Schmidt, A. Mattes, Fraunhofer Gesellschaft, Germany

The design and the optimisation of complex technical systems can be supported efficiently by using simulation methods and tools. For this reason, the generic simulation tool MOSILAB (Modelling and Simulation Laboratory) is being developed by a consortium of six Fraunhofer institutes in the GENSIM project. For the modelling process, MOSILAB uses the object- and equation- oriented model description language Modelica<sup>®</sup>, with a backwards-compatible extension to incorporate elements for describing model structure dynamics. In this article we will illustrate how MOSILAB's advanced modelling and simulation techniques support the user, with the help of two case studies: a complex energy system and a cutting tool system. Thus, the case studies illustrates very different uses of MOSILAB.

### 1 Case studies

#### 1.1 Complex Energy System

The case study of a solar heating system will demonstrate MOSILAB's advanced modelling and simulation techniques, such as model-based development, model structure dynamics, external simulator coupling, or the distributed execution of simulation experiments. The considered system model includes a solar energy plant model, a building model, a model for the control strategy and an environment model for the climate parameters (see Figure 1).

The solar energy plant model consists of a primary solar cycle with the collector field, the solar pump and some tubes. The solar energy is transferred by a counterflow heat exchanger to the secondary storage cycle, where a storage pump loads the thermal storage. A discrete two-point controller switches on both mentioned pumps, if the temperature difference between the collector output is higher than the fluid temperature in the lowest point of the storage. The other side of the storage provides the building model with heating energy by a heating cycle.

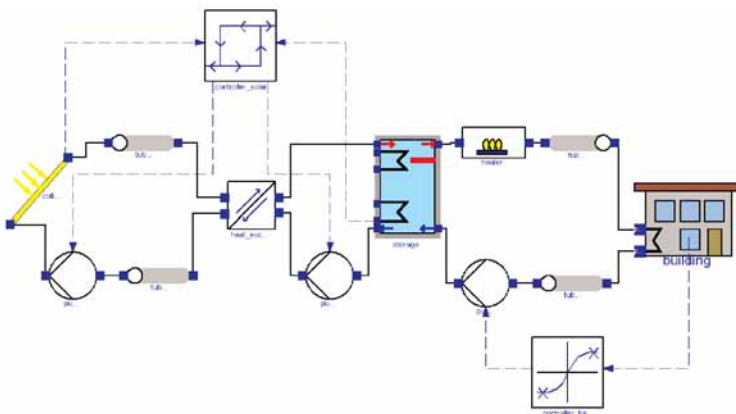


Figure 1: Energy system for solar heating system.

A continuous controller regulates the mass flow between zero and a maximum value, subject to the difference of the current room temperature and the set room temperature. An auxiliary heater delivers additional thermal energy, if the set flow temperature is not achieved by the storage output temperature.

#### 1.2 Cutting Tool System

At present high performance and high precision cutting tools often are designed as modular systems with a complex mechanical behaviour. Static and dynamic tool deformations or deflections affect the reliability of the technological process as well as the quality of the workpiece. Tool designers need convenient modelling techniques to predict the tool behaviour under working conditions in order to optimise the tool design. Simulation can also help users to choose the most suitable tool and to optimise the cutting conditions.

In the case study of a machining tool, MOSILAB is coupled with two external domain-specific FEM (Finite-Element-Methods) simulators. To simulate the behaviour of high performance cutting machine tools, the machining processes are considered to determine the loading conditions, the tool deformation, the cutting edge displacement and possible malfunctions caused by overloads. Nonlinear effects at interfaces between components of a modular cutting tool are also included. The mechanical and thermal tool loads undergo changes while complex workpiece geometries are machined e.g. dies and molds. Detailed knowledge of occurring forces and temperatures caused by chip building for any section of the tool path enables an adjustment of the process parameters to the specific cutting conditions. Thus, the feed rate is optimised by using FEM and analytically based simulation approaches.

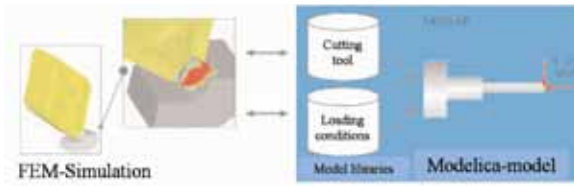


Figure 2: Coupling of FEM-simulators and MOSILAB for cutting tool systems.

This coupled consideration of tool loading and the corresponding tool behaviour enables the choice of the most suitable tool, an estimation of the workpiece quality and provides significant improvements in the efficiency of machining operations.

### 3 Model-based Development with the MOSILAB-IDE

An integrated development environment offers users support at every step of the simulation – from model building to simulation to post-processing [6]. In order to the traditional component diagrams (compare Figure 3), which give overview about the structures of the plant- or submodels, further UMLH - diagram types are available.

The class diagrams are used to organise classes and their relationships in libraries. Statechart diagrams can be used to model reactive behaviour of components, e.g. the drivers for model structural dynamics. An integrated meta-model ensures model consistency for all diagram types [8].

Thus, the behaviour of a solar thermal plant during ‘normal operation’ or in different unscheduled states can be represented in an integrated state-dependent structure variable model. Unscheduled states could be the plant behaviour whilst damaged pumps or self-activated pressure control valves, when the solar collector becomes overheated.

To support a gap-free model-based development process, a code generator plug-in can be used to produce native embedded system code for controller relevant submodels. With this feature, a newly designed controller algorithm can be tested in combination with the virtual model of the controlled system. After successful testing, the same controller algorithm can work on the real controller hardware.

Technically, the description of such controller models uses Modelica’s block and algorithm concepts. Each block implementation can be automatically transformed into controller code for the target operating system. The approach was tested on the embedded Linux derivate BOSS [2].

### 4 Use of Model Structural Dynamics

Using model structural dynamics [1], MOSILAB is able to adapt the model description, depending on the model state. One example of MOSILAB’s flexibility is its capability to switch between simulation models varying in local resolution. We have chosen the application of a 1D-thermal storage model, embedded in the solar heating system model to illustrate this advantage.

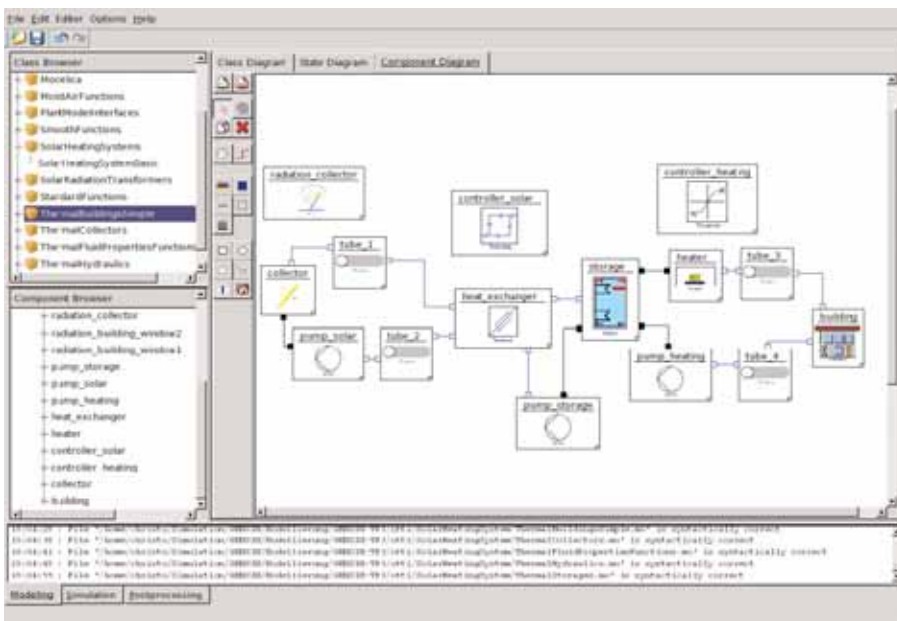


Figure 3: Solar heating system as component diagram in the MOSILAB-IDE.

During periods of low collector temperatures or when the storage pump is off, the thermal stratification in the storage can be calculated sufficiently with few numerical nodes ( $n_{\text{zones}} = 4$ ). When hot water enters the storage, it is necessary to use a storage model with substantially more numerical nodes ( $n_{\text{zones}} = 12$ ) for the thermal gradient calculation (Figure 4).

The following code fragments of the system model show the implemented strategy for switching between both models.

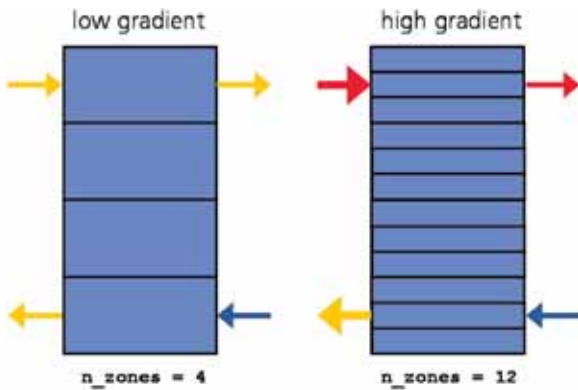


Figure 4: Structural variable storage model, which uses a different number of zones in dependency of the current thermal layering.

The first part includes the declaration of the component models:

```
model SolarHeatingSystem
...
ThermalCollectorDynamic collector(...);
Pump pump_solar(...), pump_storage(...),
  pump_heating(...);
Tube tube1(...), tube2(...), tube3(...),
  tube4(...);
HeatExchangerCounterflow heat_exchanger(...);
TwoPointController controller_solar(...);
TanhController controller_heating(...);
FlowHeater heater(...);
ThermalBuildingHeatEx building(...);
dynamic Storage storage, tempStorage;
event Boolean finer(start=false);
```

The dynamic parts of the system model are marked with the prefix **dynamic**, in our use case the storage model. Further, the Boolean-variable **finer** has the prefix **event**, which is needed to trigger the replacement from the coarser to the finer storage model.

```
equation
finer = pre(finier) or
  collector.out.T-pump_storage.in.T > 3.0
  and controller_solar.out > 0;
.....
```

The first equation in the **equation** section is true, if the difference of the collector temperature and the temperature in the lowest storage zone exceeds 3 K (and the solar pump is on). Then the storage model has to switch from 4 to 12 zones for a better reproduction of the thermal gradient.

The following code illustrates that only the static **connect** equations are available in the **equation** section. All dynamic connections between the storage model and its surrounding components are not closed:

```
// controller solar and storage cycle
collector.out.T = controller_solar.in1;
pump_storage.in.T = controller_solar.in2;
pump_solar.alpha = controller_solar.out;
pump_storage.alpha = controller_solar.out;

// controller heating cycle
building.T_air = controller_heating.in2;
273.15 + 20.0 = controller_heating.in1;
pump_heating.alpha = controller_heating.out;
...
// solar circle:
connect(collector.out, tube1.in);
connect(tube1.out, heat_exchanger.in1);
connect(heat_exchanger.out1, tube2.in);
connect(tube2.out, pump_solar.in);
connect(pump_solar.out, collector.in);
// storage solar circle:
// no static connect between
// heat_exchanger.out2 and storage.in_supply1
// no static connect between
// storage.out_supply1 and pump_storage.in
connect(pump_storage.out, heat_exchanger.in2);
// heating circle:
// no static connect between
// storage.out_load_1 and heater.in
connect(heater.out, tube3.in);
connect(tube3.out, building.in);
connect(building.out, tube4.in);
connect(tube4.out, pump_heating.in);
// no static connect between
// pump_heating.out and storage.in_load1
...
```

In the **statechart** section, which is responsible for the model structure dynamics, the states of the system model (startState, lowResolution, highResolution) are declared and the transitions between the states (startState → lowResolution, lowResolution → highResolution) are modelled:

```
statechart
state SolarHeatingSystemBasic
  extends State;
  State lowResolution, highResolution;
  State startState(isInitial = true);

  entry action
    storage := new Storage(n_zones = 4,
                          volume = 30.0,
                          ...);

  end entry;
```

At the beginning of the simulation experiment (startState → lowResolution) the storage model is added to the system model and the connections of the storage model to its surrounding components are closed.

```
transition startState -> lowResolution
add(storage);
connect(heat_exchanger.out2,
  storage.in_supply1);
connect(storage.out_supply1,
  pump_storage.in);
```

```

connect(storage.out_load1, heater.in);
connect(pump_heating.out,
        storage.in_load1);
end transition;

```

If the transition `lowResolution → highResolution` is triggered by the variable `finer` during the simulation experiment, the connections from the storage model are cut by using `disconnect(a.p, b.p)` and the old storage model is removed.

```

transition lowResolution → highResolution
event finer action
  disconnect(heat_exchanger.out2,
             storage.in_supply1);
  disconnect(storage.out_supply1,
             pump_storage.in);
  disconnect(storage.out_load1, heater.in);
  disconnect(pump_heating.out,
             storage.in_load1);
  remove(storage);

```

Now a new storage model is instantiated with `new` in a higher resolution (`n_zones = 12`). The start values of the new storage model are determined from the current state of the old storage model:

```

tempStorage := new Storage(n_zones = 12,
                           volume = 30.0,
                           ...);
tempStorage.content.T_zone[ 1] :=
    storage.content.T_zone[ 1];
tempStorage.content.T_zone[ 2] :=
    storage.content.T_zone[ 1];
tempStorage.content.T_zone[ 3] :=
    storage.content.T_zone[ 1];
tempStorage.content.T_zone[ 4] :=
    storage.content.T_zone[ 2];
...
tempStorage.content.T_zone[10] :=
    storage.content.T_zone[ 4];

```

Then the new storage model substitutes the old model, must be added to the system model and the connection to its adequate components are closed again.

```

storage := tempStorage;
add(storage);
connect(heat_exchanger.out2,
        storage.in_supply1);
connect(storage.out_supply1,
        pump_storage.in);
connect(storage.out_load1, heater.in);
connect(pump_heating.out,
        storage.in_load1);
end transition;
end SolarHeatingSystem_SC;
end SolarHeatingSystem;

```

The simulation experiment with MOSILAB for this system model for a summer day is shown in Figure 5. The diagram shows the implemented behaviour. During the morning hours, the solar controller switches the pumps on first (third curve).

Two hours later the temperature between the collector output and the temperature in the lowest layer of the storage is greater than 3 K. (This temperature is equal to the input temperature of the storage pump.) As a result, MOSILAB exchanges the coarse storage model with the higher-resolution model (`n_zones = 4 → n_zones = 12`, fourth curve).

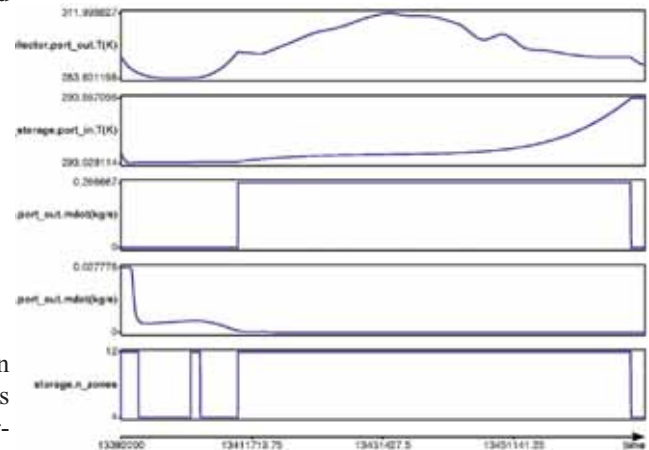


Figure 5: Simulation experiment for a summer day: MOSILAB switches to the detailed model, when hot water enters the storage model and its thermal gradient has to be recalculated in a finer resolution.

## 5 Numerical Coupling with External Simulators

Building on the MOSILAB platform, reusable components for simulator coupling have been developed within the GENSIM project. The components support integration with standard tools, such as MATLAB/Simulink or FEMLAB/COMSOL Multiphysics and also domain-specific FEM-Tools such as MARC and DEFORM. This represents a departure from and an improvement upon the typical separate handling of system simulation and FEM (Finite Element Method) simulation.

### 5.1 MATLAB / Simulink

MOSILAB offers an optional generic interface for MATLAB/Simulink [3]. Thus, it is possible to develop control strategies for embedded systems within MATLAB/Simulink and combine them with a Modelica model of the mixed-continuous discrete system environment. In this scenario each subsystem is modelled in with the appropriate modelling paradigm within adequate simulation engineering tools.

For a smooth integration of both modelling views, a proxy object is introduced in each view. Within a view, the proxy object represents the wrapped simula-



tor which is realised in the other view. This leads to symmetric model perspectives, which are close to the mental model of the engineer.

In MATLAB/Simulink a generic MOSILAB proxy model can be imported and parameterised via the block parameter dialog (Figure 6).

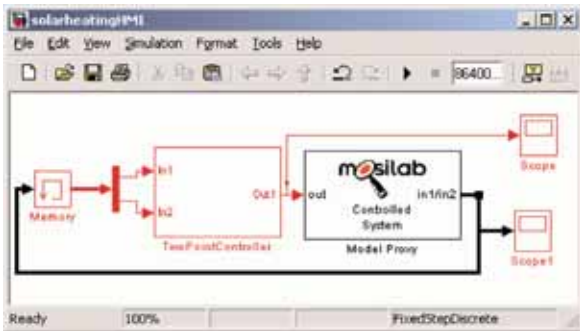


Figure 6: Simulink with an embedded MOSILAB model.

The controlled system model itself (in the case study, the solar energy plant and the building model) is developed using MOSILAB and will be associated with this proxy model, which is shown in the following code fragment:

```
block RemoteModel
  constant Boolean isRemoteModel=true;
  parameter Integer nInp, nOutp;
  input Real inp[nInp];
  output Real outp[nOutp];
end RemoteModel;
```

The constant `isRemoteModel` indicates the presence of a further simulator/driver behind this model. Thus, the numeric algorithms can handle the input and output vectors correctly. The number of input and output variables can be given by `nInp` and `nOutp`. The vectors itself are given by `inp` and `outp`. The following code illustrates the direct use of this generic remote interface within a Modelica model:

```
model SolarHeatingSystem
  ThermalCollectorDynamic collector
  Pump pump_solar(...);
  StorageSimple storage(...);
  ...
  // the Simulink interface model
  RemoteModel ctrl_solar(nInp=2, nOutp=1);
  ...
equation
  ...
  // controller solar cycle
  collector.port_out.T = ctrl_solar.inp[1];
  storage.content.T_zone[4]=ctrl_solar.inp[2];
  pump_solar.alpha = ctrl_solar.outp[1];
  pump_storage.alpha = ctrl_solar.outp[1];
```

```
// solar cycle:
connect(collector.out, tube1.in);
connect(tube1.out, heatexchanger.in1);
connect(heat_exchanger.out1, tube2.in);
connect(tube2.out,pump_solar.in);
connect(pump_solar.out,collector.in);
// storage solar cycle:
connect(heatexchanger.out2,storage.in_supply1);
connect(storage.out_supply1,pump_storage.in);
connect(pump_storage.out,heat_exchanger.in2);
...
end SolarHeatingSystems;
```

In this configuration the simulation is driven by MATLAB/Simulink as the master simulator. Figure 7 illustrates a coupled simulation experiment for the solar heating system during a simulation period of one week in spring. The top screen shows the output signal of the discrete controller, calculated in MATLAB/Simulink. This signal switches the solar pump depending on the temperature difference between the collector output temperature and the temperature in lowest level within the water storage. The bottom screen illustrates the dynamic behaviour of the controlled system, calculated in MOSILAB, for the same time period.

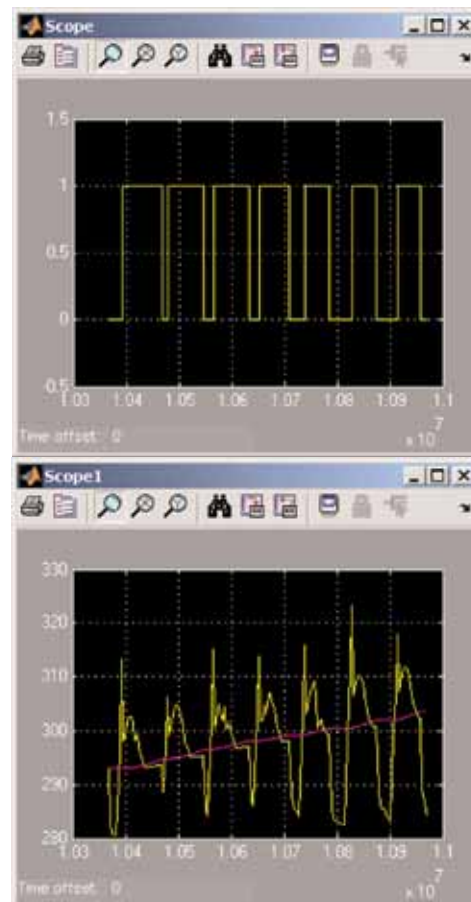


Figure 7: Coupled simulation of MOSILAB with MATLAB/Simulink.

The curves represent both state variables, which are the input signals of the controller (collector output and storage temperature).

## 5.2 FEMLAB / COMSOL Multiphysics

One other aspect within the project was the development of a numeric coupling between the simulators MOSILAB and FEMLAB [4]. For simulator couplings which incorporate FEMLAB, two basic principles exist:

1. Coupling within the MATLAB Framework – here the MATLAB engine is used in a C-program or a dedicated coupling model is implemented based on the MEX-interface.
2. FEMLAB is used as a stand-alone simulator – within FEMLAB Java-API models can be loaded, the simulation can be controlled, and the data exchange can be organised.

The second principle is used for this implementation. Figure 8 illustrates the basic structure of the coupling.

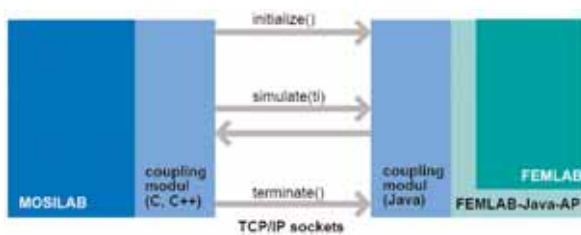


Figure 8: Numerical coupling between MOSILAB and FEMLAB/ COMSOL Multiphysics.

The communication between the two sides is handled by TCP/IP sockets. This extends the usage of the simulator coupling for a distributed computer environment. Due to a lean coupling implementation, the communication time for the data exchange is much shorter than the simulation time for simple FEM models. This allows an effective simulation including realistic transient boundary conditions even in combination with models requiring small simulation time steps.

Hence, simulator coupling is suitable for a wide range of applications. This enables the analysis of control systems with a detailed consideration of the controlled process. Furthermore, components in a complex system can be analysed in detail using the MOSILAB-FEMLAB environment, e.g. the multi-dimensional flow within the heat storage as a part of a solar heating system.

## 5.3 MARC

The finite element code MARC can be used to model complex nonlinear mechanical and thermo-mechanical structures such as machining tools consisting of different components with contact and friction problems. Thus, it is possible to simulate complex system behaviour which cannot be adequately described by analytical functions. For example, nonlinear load dependent contact behaviour between tool components may result in nonlinear tool deformations which require an expensive finite element analysis (FEA). Because of long computing times with FEA the coupling between MOSILAB and MARC will be offline in most cases. For that purpose a special interface has been developed. The coupling of MOSILAB with MARC will enable to opt between analytical models for relatively simple cutting tools or the more complex FEM-models. That way it is possible to optimise the accuracy of the description of the tool behaviour and the expense of the calculations.

Finite element analyses will run outside of MOSILAB and the input and output streams to respectively from the MOSILAB databases will be realised by *readData()* and *writeData()* commands. Using that interface it will also be possible to use predefined FEM models from a tool model library without special knowledge of finite element modelling. The loading conditions required by analytical or finite element analyses are provided by the simulation of the cutting process (see chapter 5.4).

## 5.4 DEFORM

Originally developed for metal forming processes, the FEM-tool DEFORM is also suited for simulation of the chip formation during the machining process. DEFORM is advantageous for an efficient handling of the mesh distortion, which is caused by the high strains within the chip formation zone. Through the remeshing function it is possible to generate a new mesh and to transfer the interpolated values for each node. Thereby the program is able to simulate the mechanical and thermo-mechanical behaviour.

In addition, a simplified and fast model for the chip building process in Modelica was developed, which is based on analytical equations (e.g. cutting force calculation according to Kienzle [5]). First of all, the parameters of the simplified Modelica model have to be calculated with a large number of detailed DEFORM simulations. As a result, the fast Modelica model can be used in the area of validity of these DEFORM calculations.

The cooperation between MOSILAB and DEFORM for the determination of these parameters has been fully automated. First, a routine for automated pre- and post-processing for DEFORM was developed. The execution of DEFORM by an external program is possible using the text mode of the software. This enables set up and run of simulations without going through the graphic user interface. The routine needs initial input information about tool geometry and machining parameters provided in a text file. To transfer amongst others, the values for the angles of the cutting wedge or for the feed rate and width of cut from MOSILAB the commands `readData()` and `writeData()` are applied.

```
model DataExchange
  model Kienzle
  ...
end Kienzle;

parameter String fname = "inputData.txt" ;
parameter String fnameOut =
"outputData.txt";
Kienzle k;
algorithm
  when initial() then
    readData(fname, k);
  end when;
  when terminal() then
    writeData(fnameOut, k);
  end when;
end DataExchange;
```

These loose coupling of MOSILAB with DEFORM helps to combine the advantages of both methods: First, the short computation time, when solving analytical equations in Modelica and second, the manifold possibilities by analysing the chip building process through FEM-Analysis.

## 6 Distributed Execution of Simulation Experiments

Simulators developed using MOSILAB can be generated in various configurations – from a ‘barebone’ variant suitable for constrained environments, such as embedded systems, to a regular desktop application, to a web service for distributed simulation.

### 6.1 Simulator Services and Interoperability

MOSILAB follows a service-based architectural style. For all configurations except the minimal one, the simulators generated by MOSILAB are created as services communicating through a standard interface. The standard interface is based on the W3C/OASIS web services protocol suite (most importantly, HTTP and SOAP), which allows MOSILAB-developed simulators to be controlled from a wide variety of software

environments such as Java, C++, C#.NET, MATLAB, Python, Perl, and Ruby. MOSILAB also supports a more bandwidth-efficient proprietary stream command interface, a direct C++ API, and a Python API. The Python layer abstracts from the underlying transport mechanism; i.e. the same Python experiment script can be used to control a simulator running as a local subprocess and communicating via OS standard I/O pipes, or to control a simulation web service running on a remote machine but having been generated from the same Modelica model (see Figure 9).

These interfaces are all manifestations of one and the same abstract protocol (called *MOSILAB unified steering protocol*), which is only expressed in different programming languages. The generic interfaces to other simulators described in section 5 have been developed using these interfacing options specific to MOSILAB, in addition to Modelica’s standard external function interface.

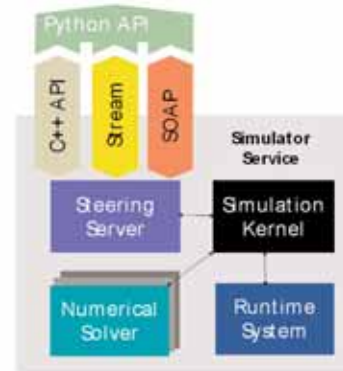


Figure 9: MOSILAB steering interface options.

### 6.2 Speeding up Parameter Studies by Distributed Simulation

Often, the system design task at hand requires a large number of simulation runs with differing parameter values, e.g. to obtain knowledge about the system’s behaviour under parameter variations (‘robust design’), or to approximate a certain desired property of the system being designed (‘optimisation’). In the system model from the case study, it makes sense to consider variations of the model parameters ‘collector area’, ‘heat store volume’ or ‘building orientation’, as well as parameters of the controller model. The following Figures 10 and 11 illustrate a variation of the collector area parameter.

Variations of multiple parameters lead to multidimensional variant spaces, the size of which (i.e. the total number of simulation runs needed) soon becomes impractical, due to the sheer computation time needed. Statistics-based methods exist to achieve a substantial reduction of the variant space with only a marginal loss of result quality, but even with such methods in place, a large number of necessary simulation experiments are likely to remain.

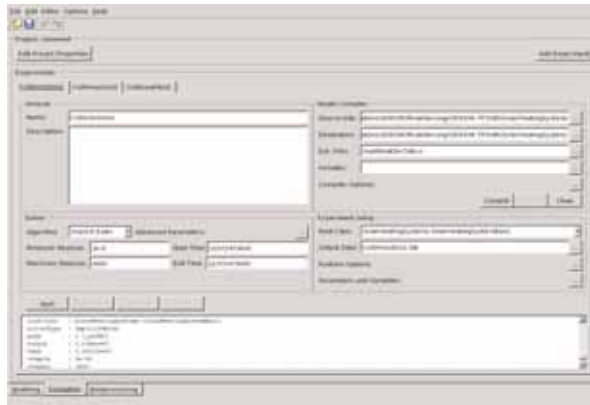


Figure 10: Multiple simulation experiments in the MOSILAB-IDE for varying the collector area.

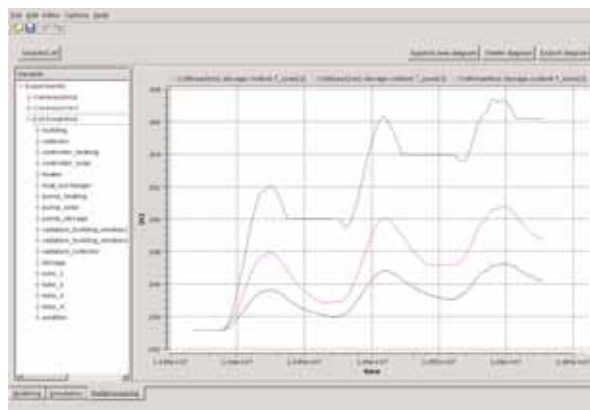


Figure 11: Impact of different collector areas on the storage temperature during a period of 3 days.

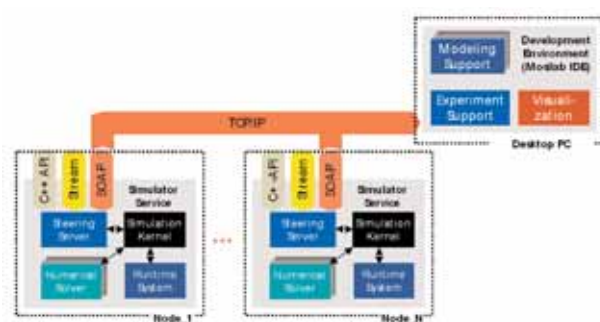


Figure 12: Executing simulator services on the grid.

MOSILAB's service-based architecture allows for distribution of simulation experiments as independent, parallel jobs in clusters and computational grids, thus empowering the user to make optimal use of the computational resources available. The individual distributed simulators can nevertheless be interactively controlled and supervised from the MOSILAB-IDE (see Figure 12).

For very large numbers of parallel experiments, central steering limits scalability, and interactive supervision becomes impractical. In this case, MOSILAB-generated simulators can be distributed on the grid as independent batch jobs. For more information on MOSILAB and grid computing, see [7].

## References

- [1] C. Nytsch-Geusen, et al.: *MOSILAB: Development of a Modelica based generic simulation tool supporting model structural dynamics*. Proc. 4th International Modelica Conference TU Hamburg-Harburg, 2005.
- [2] A. Nordwig, et al.: *Codegenerierung aus Simulationsmodellen von heterogenen technischen Systemen am Beispiel einer Pendelsteuerung*. VSEK-Report, FKZ01ISC65, 2005.
- [3] A. Nordwig: *Coupling of Modelica and Matlab/Simulink models*. Technical Report, Fraunhofer FIRST 2006.
- [4] C. Clauß, et al.: *Simulatorkopplung mit FEMLAB*. Proceedings of the 1th FEMLAB Conference, Frankfurt am Main, 2005.
- [5] W. König: *Fertigungsverfahren – Band 1: Drehen, Fräsen, Bohren*. VDI-Verlag, 1990.
- [6] MOSILAB-Homepage: [www.mosilab.de](http://www.mosilab.de)
- [7] T. Ernst, et al.: *MOSILAB: Modelica Simulation from Desktop to Grid*. 2. Workshop 'Grid-Technologie für den Entwurf technischer Systeme', Dresden, 2006.
- [8] A. Nordwig: *Integration von Sichten für die objektorientierte Modellierung hybrider Systeme*. Verlag dissertation.de, ISBN 3-89825-692-8, 2003.

## Corresponding author:

Christoph Nytsch-Geusen  
Fraunhofer-Institut fuer Rechnerarchitektur und  
Softwaretechnik FIRST  
Kekuléstrasse 7, 12489 Berlin, Germany  
[christoph.nytsch@first.fhg.de](mailto:christoph.nytsch@first.fhg.de)

Received: October 10, 2006

Revised: November 11, 2006

Accepted: November 19, 2006



## Cellular Automata Models for SIR-type Epidemics

Günter Schneckeneither, Felix Breitenecker, Vienna University of Technology, Austria  
`{gschneck, fbreiten}@osiris.tuwien.ac.at`

Nikolas Popper, Günther Zauner, 'Die Drahtwarenhandlung' - Simulation Services, Vienna, Austria

This contribution compares different modelling approaches for quantitative evolution of epidemics. Investigations start with the classical SIR - Model (Susceptible - Infected - Recovered) of ODE type by Kermack and McKendrick, which emphasises on a homogeneous population, neglecting spatial distributions. As alternative, the contribution discusses Cellular Automata (CA) models. After a short introduction to CAs, Lattice Gas Cellular Automata (LCGA) models are investigated, and relations between parameters of the ODE model and parameters of these CA models are derived. The LCGA models allow deep insight into spatial inhomogenities, and allow for testing different vaccination strategies - shown by simulation experiments, based on MATLAB implementations. By extending the classical CA models, the contribution presents Stochastic CA models, which show good coincidence with classical LGCA models, but require less computational effort. Furthermore it is shown, that a difference equation model derived by summations in LGCA models, corresponds with an Euler discretisation of the classical ODE model.

### Introduction

The classical *Susceptible-Infected-Recovered* model (SIR model), which was proposed in 1926 by Kermack and McKendrick, describes the quantitative evolution of epidemics in a homogeneous population by means of ordinary differential equations. The fact, that the spread of infectious diseases is related to the contact behaviour of individuals and follows particular patterns, is not taken into account (Section 1). This is problematic in so far as in particular direct propagation between human individuals bears a great epidemiological danger but also allows advanced methods for confining an outbreak. A great number of model approaches discharge the idea of a homogeneous population and tend towards a more or less detailed imitation of socio-economic, sociologic and demographic features. A promising approach, which regards motion and social interaction of individuals, is modelling by cellular automata (CA).

It is of interest to investigate properties of CA models (Lattice Gas CA models) and relations to the ODE model, and to use them for testing different vaccination strategies (Section 2). The ARGESIM Comparison C17 *Temporal and Spatial Evolution SIR-type Epidemics* ([1]), discusses some of these aspects, and more details and backgrounds are given in this contribution (Section 2). For instance, a difference equation summarised from a LCGA model with homogeneous population, is proven to be equivalent to an Euler discretisation of the ODE model (Section 3).

Interestingly, another type of CA models, stochastic CA models shows advantages (Section 4). They allow to model spatial inhomogeneities as in case of the LCGA models, but implementations need much less code (implementation examples given; Sections 2, 4).

### 1 Differential Equations Model

Under several assumptions concerning the population and the disease, the system of ordinary differential equations is rather simple to derive ([3]). The basic assumptions are, that the population is an isolated *homogeneous* composition of susceptible, infected and recovered individuals ( $S, I, R$ ) and that the duration of the infection is equal to the period of contagiousness (no incubation period):

$$\begin{aligned}\frac{\partial S(t)}{\partial t} &= -r \cdot S(t) \cdot I(t) \\ \frac{\partial I(t)}{\partial t} &= r \cdot S(t) \cdot I(t) - a \cdot I(t) \\ \frac{\partial R(t)}{\partial t} &= a \cdot I(t)\end{aligned}\tag{1}$$

The flow between the three groups is completely determined by the parameters  $r$  and  $a$ . The *rate of infection*  $r$  can be interpreted as the probability of contagion when a susceptible and an infected individual come into contact for a certain period of time. Therefore, if the gain in the number of infected is  $r \cdot I \cdot S$ , all individuals of the population must come into contact during one time unit (not necessarily contagious contact!). This corresponds to Daley and Gani: 'If the individuals in a population mix homogeneously, the rate of interaction between two different subsets of the population is proportional to the product of the numbers in each of the subsets concerned' (see [4]). In this probabilistic approach,  $r \cdot I$  is the expected number of contagious contacts per susceptible individual and time unit and accordingly corresponds to the probability that a susceptible individual becomes infected during one time unit.

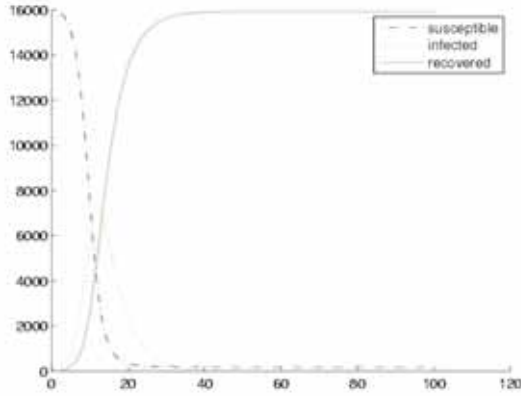


Figure 1: Simulation results for SIR ODE model, with infection rate  $r = 0.6 \cdot 10^4$ , recovery rate  $a = 0.2$ , and initial populations  $S(0) = 16,000$ ,  $I(0) = 100$ ,  $R(0) = 0$ .

The *recovery rate*  $a$  is a medical or biological parameter and does not depend on the contact behaviour of the population. If  $1/a$  is the average duration of the contagious period, then  $a \cdot I$  is the fraction of infected individuals, who will recover during one time unit. This means that the duration of the infection is in some sense geometrically distributed ([3]). Figure 1 shows simulation results for a certain scenario.

## 2 Lattice Gas Cellular Automata Models

The classical SIR-ODE model ‘assumes that the concentration of susceptibles and infected is [always] spatially homogeneous’ ([5]), which actually is not the case when the infections happen through contact between individuals. This deficiency can only be partially balanced by changing the infection rate and makes this model unsuitable for observing vaccination strategies. In order to introduce spatial inhomogeneities different methods like partial differential equations (PDEs), cellular automata or network based models can be used.

### 2.1 A Short Overview on Cellular Automata

Cellular automata (CA) were introduced around 1950 and consist of a finite number  $n$  of equal cells, which are arranged in a lattice structure  $L$ . The cells can hold a finite number of states. For every cell the so-called *automaton rule* calculates a new  $(t+1)$  cell-state according to the present  $(t)$  states of the cell itself and its neighbourhood cells. The *neighbourhood* consists of a certain selection of cells, which lie closest to the cell in the lattice  $L$  ([6]).

For the one-dimensional CA the neighbourhood consist of  $2 \cdot r$  cells and accordingly the automaton rule is an arbitrary function  $F$  of  $2 \cdot r + 1$  arguments.

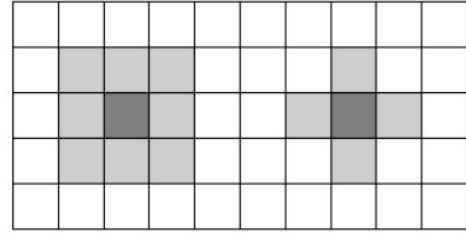


Figure 2: Moore and Von Neumann neighbourhood.

If  $a_i(t)$  represents the state of the  $i$ -th cell in the lattice at time  $t$ , the evolution of the CA is described by the following equations for all  $i = 0, \dots, n$  ([6]):

$$a_i(t+1) = F(a_{i-r}(t), \dots, a_i(t), \dots, a_{i+r}(t))$$

For two-dimensional CA two very common definitions of the neighbourhood are presented here, the Moore neighbourhood and the Van Neumann neighbourhood (Figure 2):

$$N_{i,j}^{Moore} = \{ (k, l) \in L : |k - i| \leq r \wedge |l - j| \leq r \}$$

$$N_{i,j}^{VanNeumann} = \{ (k, l) \in L : |k - i| + |l - j| \leq r \}$$

‘Despite of their simple update rules cellular automata can display complex behaviour which is a prerequisite to use them as a simulation tool for physical (biological, chemical, ...) phenomena like, for example, fluid flow. CA are very easy to implement and are especially well suited for massively parallel computers because of the local character of the update rules. By construction they are unconditionally numerically stable.’- (see [6]).

**Stochastic cellular automata.** In regard of the structure of the neighbourhood, stochastic CA form an extension of ordinary CA. For each two cells on the lattice an interaction coefficient  $\iota$ , which describes the likelihood of interaction, is introduced ([7]). A prototype for  $\iota$  is for example the inverse of the Euclidean distance.

According to the likelihood of interaction a certain number (compare  $2 \cdot r$  for the ordinary CA) of neighbourhood cells is selected (Figure 3). This happens through stochastic methods (function  $P$ ). On those neighbourhood cells the automaton rule  $F$  is applied:

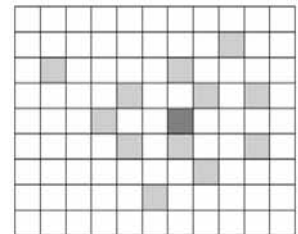


Figure 3: A neighbourhood in a stochastic CA.

$$a_{i,j}(t+1) = F(P(\iota_{(i,j),(k,l)}) \cdot a_{k,l}(t), (k, l) \in L)$$

**Lattice gas cellular automata (LCGA)** emerged in 1973. The observed two-dimensional region or domain is discretised on a lattice of squares (Hardy - de Pazzis - Pomeau automaton; HPP-LGCA) or hexagons (Frisch - Hasslacher - Pomeau automaton; FHP-LGCA) for example. On the lattice particles (gas, molecules) with certain attributes are moving. Particles are residing in some cell, whereby the state of a cell indicates the attributes of a particles, which resides in a particular cell. An update of the CA ‘simulates’ then the movement of the particles, by updating the cell states (attributes of particles) correspondingly.

For the investigated case, the SIR - CA model, it is sufficient that the cells contain four or six state attributes, indicating the direction of movement of a particles residing in the cell. After one time step, which represents one time unit, all particles leave the cell in these directions and move on to the bordering cells, according to different rules (Figure 4). *Interaction* of particles may happen in cells or on movement. In this way the automaton describes dynamic movements of the particles within the observed domain. The movements can be either random or determined by certain *transition rules*, which depend on the collisions of the particles, previous states, and the boundary conditions.

According to [6] for the HPP-LGCA only one deterministic collision configuration, which conserves mass and momentum, is possible: ‘When two particles enter a node from opposite directions and the other two [positions] are empty a head-on collision takes place which rotates both particles by 90° in the same sense’. In all other situations the particles pass through the cell without changing the direction. A particle, which hits the border of the array, is reflected.

FHP-LGCA collision configurations, which only change the directions of 2- and 3-particle collisions and conserve mass and momentum, are summarised under the term FHP-I. There, for 2-particle head-on and 3-particle star-shaped collisions the particles are deflected

by 60° clockwise or counterclockwise by chance.

In all other situations the particles pass through the cell without changing the direction. The boundary conditions are again composed of reflections.

## 2.2 LGCA Model for SIR-type Epidemics

An LGCA modelling approach can easily be adapted to population dynamics, and to the special case of SIR dynamics: the geographical region is discretised on a lattice, and the individuals in this region are represented by particles, residing in cells, and move according to the transition rules of an LGCA. For a LGCA-based SIR model it is sufficient that the cells contain four or six ‘positions’, on which susceptible, infected, or recovered individuals can be placed. *Interaction* – propagation of the infection in this case – is restricted to the particles in the same cell. *Propagation* of the infectious disease happens only within the cells, where susceptible and infected individuals come into contact.

Adapting the transition rules allows modelling demographic and socioeconomic features of a population. Biological or medical features of the observed disease are influenced by the infection rate  $r$  and recovery rate  $a$ . The number of contacts per time unit, which depends on the density of a population, can be influenced by changing the size of the lattice. All these impacts describe epidemiological properties.

In principle, the LGCA - based SIR model is similar to a PDE model, discretised in space and time, with unit-step time discretisation, and by space discretisation with rectangular or hexagonal grid. A further development of LGCA modelling is the Lattice-Boltzmann Method (LBM), where diffusion is modelled by a complex LGCA - approach. There, under certain conditions, the LMB solution converges to the PDE solution.

**Adaptation of parameters.** Summing up at each time instant  $k$  in the LGCA-SIR model the number of susceptible, infected and recovered individuals, gives discrete time courses  $S_{CA}(k)$ ,  $I_{CA}(k)$ , and  $R_{CA}(k)$  for the individuals, which ideally should coincide with the solution of the ODE-SIR model:

$$S_{CA}(k) \approx S(t=k), I_{CA}(k) \approx I(t=k), R_{CA}(k) \approx R(t=k)$$

In order to reach this aim, parameters for the LGCA-SIR model must be chosen accordingly to the parameters of the ODE-SIR model.

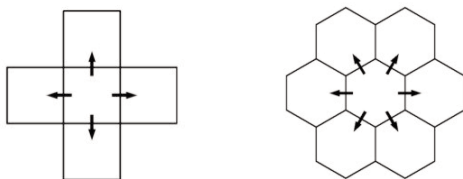


Figure 4: Square (HPP) and hexagonal (FHP), LGCA-neighbourhood.

On a lattice of  $N$  cells (temporary and spatially bounded contact spaces), the probability that a susceptible individual becomes infected is in average  $1 - (1 - r)^{\frac{I}{N}}$ , because the expected number of infected individuals within each cell is  $I/N$ . Taylor series expansion gives a linearisation for small infection rates  $r$  (see [1], [3]):

$$1 - (1 - r)^{\frac{I}{N}} \approx r \cdot \frac{I}{N}$$

Consequently, to obtain the same rate of infections in the LGCA-SIR model as in the ODE-SIR model, the probability of contagion in the LGCA should be changed to  $r_{CA} \approx r \cdot N$  for both the FHP and the HPP LGCA-SIR model.

If  $(1/a)$  is the average duration of the infection, then the probability of recovery during one time step is  $a$ . This means, the duration of the infection in the discrete automaton is geometrically distributed (memory less) with parameter  $a$ . In the LGCA model it is also possible to implement different distributions for the infectious period. The same is true for incubation periods.

### 2.3 Implementation of a LGCA-SIR Model in MATLAB

Cellular automata become more and more used for distributed simulation. Efficient implementations of CA simulation require appropriate data structures and appropriate propagation functions (evolution operators). MATLAB is a very suitable tool for CA implementation, but the efficiency depends heavily on the choice of data structures and nesting of propagation function. In the following an example for an efficient implementation for an model of FHP type is developed.

Observing a lattice of  $n \times n$  cells leads to  $6 \cdot n^2$  positions altogether, because in the FHP model each cell contains six positions. This is also the maximum number of individuals on the domain. In the computer program for example 0, 1, 2 or 3 is assigned to each position if there is no individual, a susceptible individual, an infected individual, or a recovered individual. To store the state of the lattice a  $n^2 \times 6$  Matrix  $L$  is used. The  $n^2$  rows of the matrix  $L$  represent the cells. In each cell one particle will move right, up-right, up-left, ..., down-right in the next time step. These particles (individuals) are always placed in the first, second, ..., sixth column of the matrix (Figure 5).

To calculate the lattice at  $t+1$ , it is necessary to perform the *movements* of the particles (according to their positions), the *propagation* of the disease including recoveries (according to the parameters and the

	→	↗	↖	←	↙	↘
1	0	1	1	0	1	0
2	1	0	2	1	1	3
3	1	1	1	1	1	0
4	3	0	1	2	0	1
5	1	1	0	1	1	1

Figure 5: MATLAB implementation of CA matrix  $L$ , representing the state of the automaton at time  $t$ .

individuals within the cell) and the *determination* of the new moving directions (according to the transition rules and the boundary conditions). This is done by applying an *evolution operator* on each cell  $L(i, :)$ :

```
cellstate1=update(L(i,:));
cellstate2=propagation(cellstate1,r,a);
cellstate3=transition(cellstate2);
```

For every cell, the function `update` loads the particles from the appropriate positions of the bordering cells in the lattice at time  $t$  (from  $L$ ) and stores each of them on the same position of the observed cell in the lattice at  $t+1$  (into matrix  $L_2$ ). Even though this function does not contain very much program code, the implementation is rather complicated. For the FHP model not only the boundary conditions of the automaton, but also the special arrangement of the cells have to be taken into account.

In the subfunction `propagation` the probabilistic approach discussed before can be useful. MATLAB provides a function `binornd(n,p)`, which generates a binomial distributed random number with parameters  $n$  and  $p$ :

```
for k=1:6
    if cell(k) == 1
        % I...number of infected in cell
        cell(k)=1+sign(binornd(I,r));
    elseif cell(k) == 2
        cell(k)=2+binornd(1,a);
    end
end
```

The subfunction `transition` rearranges the particles within the cells according to the directions, in which they should move in the next time step.

After all  $n^2$  cells have been calculated,  $L$  is replaced by  $L_2$ . For graphical output of the lattice, functions like `imagesc` can be used. Susceptible, infected and recovered individuals are then marked by different colours.



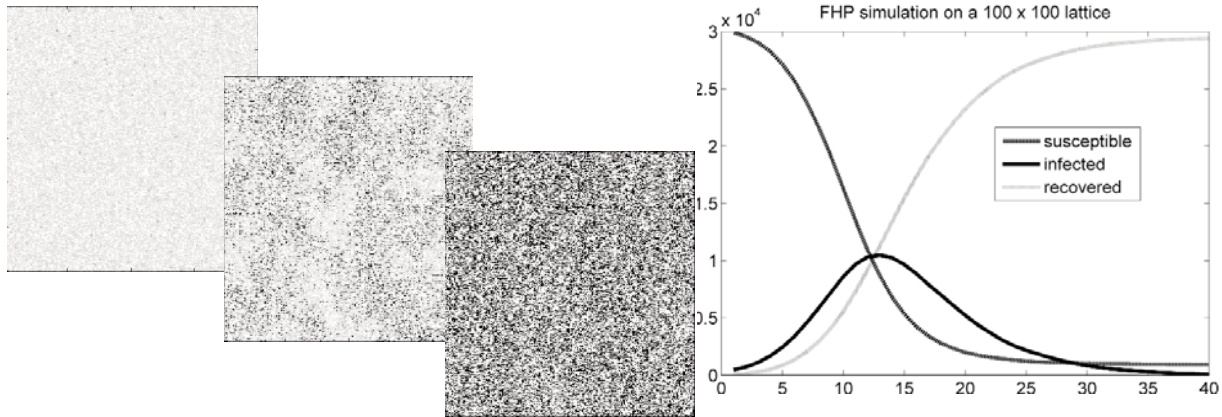


Figure 6: LGCA simulation with homogeneous initial conditions: the left three pictures are lattices at  $t=0$ ,  $t=10$ , and  $t=20$  (grey particles mark susceptible or infected individuals and black particles mark recovered individuals); right picture shows summed up individuals  $S_{CA}(k)$ ,  $I_{CA}(k)$ , and  $R_{CA}(k)$  over time.

## 2.4 Influence of Spatial Inhomogeneities

Results of CA-SIR simulations (Figure 6) show the development of spatial patterns, together with the summed up individuals (with very good coincidence with the ODE-SIR model results). Nevertheless, the benefit of the CA-SIR model is the development of spatial patterns, to be discussed in the following.

**Initial conditions.** Because infections in the LGCA happen through contact between individuals, the course of the epidemic highly depends on the initial conditions and the density of the population. Simulations show, that the qualitative and quantitative differences between the FHP-type LGCA-SIR model and the ODE-SIR model are minimal, if the initial conditions of the CA assume full density of the population (no empty positions in the CA) and that the infected individuals are uniformly distributed on the lattice. This corresponds to the idea that homogeneous mixing in the continuous model is always perfect. The remaining difference is a faster spread in the continuous model ([1]).

**Subgroupings.** Generally the feature of local disease propagation in the LGCA leads to slight subgroupings of infected individuals (to be seen in Figure 5, CA lattice at  $t=10$ ). In such epidemic areas the probability of infection is higher than in other areas of the lattice. When no susceptibles are left in these epidemic areas, the spread of the disease can only happen, if infected individuals move into areas with susceptible individuals or vice versa. Consequently the spread of the disease partially depends on the (slow) motion of the particles. This is also the reason why transition rules, which lead to a stronger mixing (e.g. FHP-I rules), accelerate the epidemic (*epidemic areas* vs. *homogeneous mixing*).

**Transition rules.** FHP-I transition rules for example deliver rather good mixing of the individuals and consequently the behaviour of the simulations is qualitatively and quantitatively closer to the continuous model [1]. Transition rules, which ‘dictate direction changes more often [on the other hand] lead to a diffusive type of motion’ of the individuals, which does in general not correspond to the reality [5]. The peak of diffusive motion is reached by introducing totally random transition rules and results in a slower spread (Figure 8).

**Neighbourhood.** Simulations show that the FHP model delivers faster spread than the HPP model (see solutions to ARGESIM Comparison C17, [1] and [2]) even though the overall ratio of infections is the same for both models. The reasons for this difference (Figure 8) must therefore lie in the structure of the lattice and can not depend on the size of the cells. The distinction is that the cell neighbourhood in the FHP automaton is larger than in the HPP automaton (Figure 4). Presumably in general a larger neighbourhood favours faster spread. Because a large neighbourhood corresponds to more dynamic, faster and wide ranged movements of the individuals, this idea is also true for epidemics in real life.

## 2.5 LGCA - Vaccination Strategies

Vaccination strategies in the classical ODE-SIR model are always spatially homogeneous. The corresponding strategy for the LGCA model would be a uniform distribution of the vaccinated (recovered) individuals among the population. More interesting in the context of LGCA are strategies, which intend to use spatial inhomogeneities. In [5] for example a region of infected individuals in the centre of the lattice surrounded by a barrier of vaccinated individuals is observed.

A simpler approach for the same strategy is to place the infected particles in one half or in smaller fractions of the domain as suggested in [1].

In accordance with simulation experiments performed in ARGESIM Benchmark C17, Task B ([1] and [2]), in the following efficiency and significance of vaccination strategies is discussed. For comparison, a FHP lattice of  $100 \times 100$  cells with a population of 20.100 individuals of whom 16.000 are susceptible, 100 infected and 4.000 vaccinated (recovered) is observed. The recovery rate  $a$  is 0.2 and the overall rate of infection  $r$  is  $0.6 \cdot 10^{-4}$ , therefore  $r_{CA} = r \cdot n^2 = 0.6$  is the infection rate in the LGCA-SIR model. The motion of the particles is determined by FHP-I transition rules.

Several vaccination policies are applicable:

- Uniform distribution of the vaccinated individuals on the whole domain.
- Vaccinations inside the epidemic area.
- Vaccinations outside the epidemic area.
- Placing the vaccinated individuals at the border of the epidemic area ('barrier strategy').

**Homogeneous initial distribution of infected.** In the first experiment the 100 infected individuals are situated in one half of the domain. Simulations with each of the four vaccination strategies show, that the course of the epidemic is always the same. When the barrier strategy is applied, a closer look on the lattice tells, that the barrier is too small in order to confine the outbreak. Because the infected individuals are grouped together with many susceptibles, the duration of the 'epidemic wave' is rather long. When the barrier begins to diffuse, still many new infections take place and the disease can easily spread to the rest of the domain. The result of all four strategies is a damping of the speed of the spread.

**Concentrated initial distribution of infected.** In simulations, assuming that the 100 infected individuals are situated in one sixth of the domain, differences between the strategies become visible (Fig.7). Vaccinations inside the epidemic area deliver a good damping in the beginning, because the ratio of vaccinated individuals is rather high in this area and fewer infections take place.

But when infected individuals leave the area, the disease begins to spread at full speed to the rest of the domain (epidemic wave). Uniformly distributed vaccinations cannot influence the course of the epidemic, because the ratio of vaccinated individuals is always too small. The same is true for vaccinations outside the epidemic area.

If the barrier strategy is applied, the disease spreads at full speed all over the area containing the infected individuals. Because the epidemic wave is not as intense as in the first experiment with the barrier strategy, only very few infected pass on through to the other side. The outbreak reaches a second climax when the epidemic spreads to the rest of the domain. In comparison to vaccinations inside the epidemic area, the barrier strategy even accelerates the epidemic (Figure 7). On the other hand, a barrier of 8.000 vaccinated individuals instead of 4.000 can stop the outbreak by quarantining the infected individuals in one sixth of the domain.

## 2.6 Remarks on Vaccination Strategies

In regions containing vaccinated individuals, the spread of the disease is generally slower. A sole application of the barrier strategy can either accelerate or confine the epidemic, depending on the size of the epidemic area and the barrier. Combinations of these strategies can deliver better results. In the continuous model (1) the assumption of a positive number of recovered individuals in the beginning has absolutely no effect on the course of the epidemic.

The initial conditions assumed in the last simulations are not very realistic:

- The detection of the spread of an infectious disease must not necessarily happen if only very few individuals (100) are concerned.
- It is very unlikely that the occurrence of infections is restricted to a small region (one sixth).
- Vaccinating a large number of individuals (4.000 or 8.000) is often impossible.

But even a damping in the speed of the spread, as it was reached in the first experiment, can be useful.

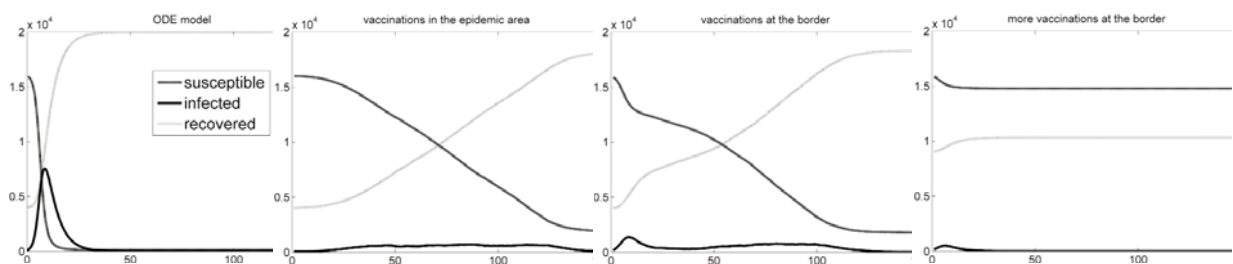


Figure 7: Different vaccination strategies: uniform vaccination (ODE model), vaccination in the epidemic area, vaccination at the border, larger number of vaccinations at the border (from left to right).

An example showing the advantage of the barrier strategy is given in [5]. Other results in [5] are: To improve the outcomes of the calculations, the mean of a larger number of simulations can be used. ‘The uniform strategy is not better than no vaccination’. A measure for the severity of an outbreak is for example the following formula:

$$\frac{R(\infty) - R(0)}{R + I + S}$$

A method for directly calculating the necessary number of vaccinations in order to prevent an epidemic outbreak was given in [4]: ‘The fraction of the population that must be vaccinated to prevent an epidemic outbreak’ is  $1 - (a/r_{CA})$ , which in this case would be  $\frac{2}{3}$ .

National vaccination plans, as they are suggested by the WHO, intend the use of quarantines and a chronological order for vaccinating certain subsets of the population. Medical (in medical facilities) and administrative staff (police, military, government, ...) for example enjoy preference. For modelling these types of vaccination strategies, different approaches are more suitable.

### 3 Difference Equations Model

The probability of infection within a cell can be approximated by using the average number of infected individuals in the cell  $I/N$ . This leads to an expected overall number of susceptibles to become infected during one time step of

$$S(t) \cdot (1 - (1 - \tilde{r})^{\frac{I(t)}{N}})$$

Accordingly the number of all susceptible, infected and recovered individuals *on the lattice* after one time step/unit is given (approximated) by the following difference equations:

$$\begin{aligned} S(t+1) &= S(t) - S(t) \cdot (1 - (1 - \tilde{r})^{\frac{I(t)}{N}}) \\ &= S(t) \cdot (1 - \tilde{r})^{\frac{I(t)}{N}} \\ I(t+1) &= I(t) + S(t) \cdot (1 - (1 - \tilde{r})^{\frac{I(t)}{N}}) - a \cdot I(t) \\ &= I(t) \cdot (1 - a) + S(t) \cdot (1 - (1 - \tilde{r})^{\frac{I(t)}{N}}) \\ R(t+1) &= R(t) + a \cdot I(t) \end{aligned} \quad (2)$$

Taylor series expansion from section 2.2 shows that

$$1 - (1 - \tilde{r})^{\frac{I(t)}{N}} \approx \tilde{r} \cdot \frac{I(t)}{N}$$

This simplification leads to a new system of difference equations, which again describes the evolution of the epidemic on the whole lattice:

$$\begin{aligned} S(t+1) &= S(t) \cdot (1 - \tilde{r} \cdot \frac{I(t)}{N}) \\ I(t+1) &= I(t) \cdot (1 - a) + S(t) \cdot \tilde{r} \cdot \frac{I(t)}{N} \\ R(t+1) &= R(t) + a \cdot I(t) \end{aligned} \quad (3)$$

Employing  $\tilde{r} \cdot \frac{I(t)}{N} = r \cdot I(t)$  on (3) again delivers equations for the whole lattice:

$$\begin{aligned} S(t+1) &= S(t) \cdot (1 - r \cdot I(t)) \\ I(t+1) &= I(t) \cdot (1 - a) + S(t) \cdot r \cdot I(t) \\ R(t+1) &= R(t) + a \cdot I(t) \end{aligned} \quad (4)$$

At the same time this system of difference equations (4) is the discretisation of the classical model (1) and can serve as Euler approximation for the ODEs in (1).

#### 3.1 Connection between the ODE and the LGCA Model

It is possible to proof, that the LGCA model is an extension of the ODE model by spatial properties of disease propagation. By randomly rearranging the lattice in the automaton in each update step, the spatial properties become eliminated and the behaviour of the automaton converges towards the continuous model, as discussed in ARGESIM Benchmark C17, Task C.

At a certain moment  $t$ , there are in average

$$s(t) = \frac{S(t)}{N}, \quad i(t) = \frac{I(t)}{N}, \quad r(t) = \frac{R(t)}{N}$$

susceptible, infected, and recovered individuals in each cell. To derive a difference equation system for the *average* number of individuals  $A = (S + I + R) / N$  in each cell, these substitutions are employed on (3):

$$\begin{aligned} s(t+1) &= s(t) \cdot (1 - \tilde{r} \cdot i(t)) \\ i(t+1) &= i(t) \cdot (1 - a) + s(t) \cdot \tilde{r} \cdot i(t) \\ r(t+1) &= r(t) + a \cdot i(t) \end{aligned} \quad (5)$$

Obviously (5) and (4) are equivalent (actually only in the sense of applying (5) on every cell containing the average  $A$  of individuals).

As stated before the spatial component of the cellular automaton can be deployed by rearranging the individuals on the lattice after every time step. Simulations show that in this case the behaviour is very close to (4). In a purely theoretical approach the lattice could be rearranged very often to obtain a uniform distribution of susceptible, infected, recovered individuals  $A$  in each cell. This situation would correspond to the difference equations in (5). In this sense ‘the solution of the difference equations (4) serves as upper bound for the automaton’ ([2]).



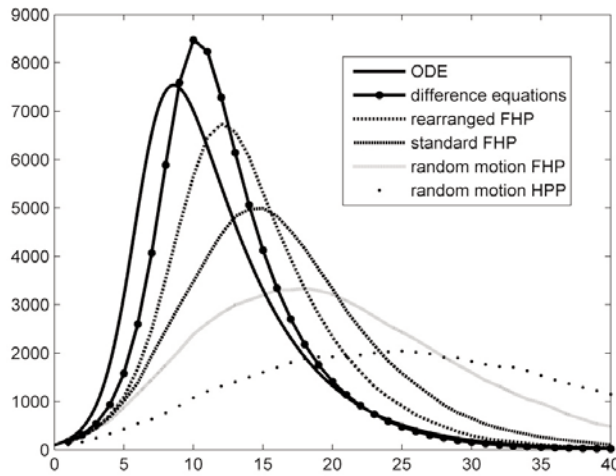


Figure 8: Infected individuals from different modelling techniques.

Rearranging the lattice more and more often leads to a continuous dispersal of the structure of the lattice; simultaneously reducing the step size, ends in the continuous model.

#### 4 Stochastic Cellular Automata Models

According to sections 2.4 and 3.1 it is clear that the qualitative and quantitative behaviour of the automaton model highly depends on the structure of the cell-neighbourhood. For the LGCA interaction is restricted to four resp. six individuals. The velocity of each individual is always one discrete movement between neighbouring cells. *Stochastic Cellular Automata* (section 2.1) allow a more flexible definition of the neighbourhood. The number of contacts per individual and time step and the distance between interacting individuals (compare ‘velocity’) can be arbitrary values.

In the stochastic cellular automaton model the cells are arranged on a square lattice; they can represent single individuals and can hold the states  $S$ ,  $I$  or  $R$ . For evolution of the automaton each cell establishes contact with  $k$  different cells, which are situated at any position on the lattice. The selection of the contact-cells can depend on the likelihood of interaction or on any appropriate rule.

In order to model a different number of contacts for every individual – instead of a fix number of  $k$  contacts –, an additional parameter for every cell can be used. This would allow for example distributing the number of contacts on the lattice. The interaction coefficient  $\iota$ , which describes the likelihood of contact, must depend on ‘the distance, demographics and socioeconomic features’ ([7]).

In [7] an exponential *decay function* with parameter  $\lambda$  is used to stochastically determine the contacts.  $\lambda$  controls the structure of the neighbourhoods and influences the occurrence of subgroupings.

A way to model a social partition of the population in the stochastic CA would be through an *interaction function*, which favours for example horizontal contacts and establishes fewer vertical contacts.

#### 4.1 Implementation of Stochastic CA in MATLAB

In order to establish contact between individuals/cells in the computer program, in [7] a so-called *bounding-box* algorithm is presented. This algorithm randomly selects the contact cells within certain square areas around the cell. For every new contact the area grows and accordingly the likelihood of interaction with distant cells is smaller (decay).

For reasons of simplification and comparability, a stochastic CA with a uniform distribution of the likelihood of interaction  $\iota$  is presented here first. This means that the  $k$  contact cells are randomly chosen within just one single area, which is defined by a circle with radius  $d$  around the cell.

In epidemic areas many infected individuals/cells try to establish the full number of  $k$  contacts, which can lead to more than  $k$  contacts per time step for a susceptible individual in the same region. To avoid this, the number of contacts is counted for each cell on the lattice (Matrix  $c$ ). On the other hand, if already many contacts have been established, the remaining infected cells may not find any more contact cells. Consequently the number  $c$  of tries to establish  $k$  contacts must be bounded ( $c \leq 2 \cdot k$  for example).

For every infected cell on the lattice ( $n \times n$ ), the program must select  $k$  contact cells within the radius  $d$  and perform the propagation rules on them. Because of the structure of the lattice, the contact distance should be at least  $\sqrt{2}/2 \approx 0.71$  (Figure 9).

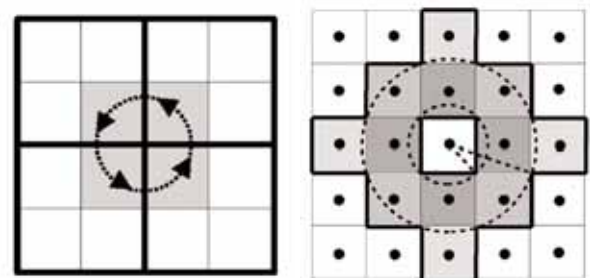


Figure 9: Neighbourhood in the cyclic HPP automaton and in the stochastic CA with radius  $d$ .



An efficient MATLAB implementation is given in the following:

```
C=zeros(n,n); % counts the contacts
                % of each cell

% for each cell on the lattice:
if L(i,j) == 2
    c=0; % counts the number of tries

    while (C(i,j) < k) && (c < 2*k)
        c=c+1;
        % select cell within the radius:
        mod=0.71+rand*(d-0.715);
        arg=rand*2*pi;
        ii=round(mod*cos(arg));
        jj=round(mod*sin(arg));

        % checking if the contact-cell
        % is outside the domain:
        if (i+ii < 1) || (i+ii > n) || ...
            ...(j+jj < 1) || (j+jj > n)
            % nothing;
        % checking if the contact-cell
        % has more than k contacts:
        elseif C(i+ii,j+jj) >= k
            % nothing;
        else
            % count contact (in C):
            C(i+ii,j+jj)=C(i+ii,j+jj)+1;
            C(i,j)=C(i,j)+1;
            % if the cell is susceptible:
            if L2(i+ii,j+jj) == 1
                L2(i+ii,j+jj)=1+binornd(1,r);
            end
        end
    end
end
L2(i,j)=2+binornd(1,a);
end
```

After all cells have been updated,  $L$  is replaced by  $L2$  and  $C$  is set back to  $\text{zeros}(n,n)$ .

In order to have exponential decay of the likelihood of interaction, the variable `mod` must be defined e.g. as  $\exp(d \cdot c \cdot \text{rand}/4) - 0.28$ . In this case the interpretation of parameter  $d$  is different.  $d=1$  delivers rather linear growth of the radius for less than 10 contacts. Bigger values deliver strong exponential growth of the radius and accordingly a huge neighbourhood.

#### 4.2 Evaluation of the Stochastic CA SIR Model

For simulations and comparison with other models, the parameters  $d$  and  $k$  have to be chosen carefully. For both parameters an increase results in faster spread. But there are qualitative differences between the effects, which may only become visible after many time steps.

**Modified HPP LGCA vs. Stochastic CA.** For comparison, a population of  $N=40.000$  individuals with rates  $a=0.2$  and  $r=6 \times 10^{-5}$  is observed.

The inhomogeneous initial condition consists e.g. of a homogeneous mixing of equally many (10.000) susceptible and infected individuals in one half of the domain and 20.000 susceptible in the other half.

Individuals in the HPP LGCA ( $n=100 \rightarrow \tilde{r}=0.6$ ) do not have a fixed neighbourhood for all time steps as in the stochastic CA ( $n=200 \rightarrow \tilde{r}=0.6$ ). Consequently, finding the corresponding parameters for the stochastic automaton is not straight forward. A restriction of the motion of each particle in the HPP automaton to four cells (rotational cyclic movement, Figure 9) minimises the overall interaction area of a particles. In this situation each particle has  $4 \times 3 = 12$  direct contacts every four time steps. The corresponding parameters for the stochastic CA would be approximately  $k=3$  and  $d=2$ . But in this case the spread in the stochastic CA is a little bit slower than in the modified HPP LGCA. An increase of  $d$  to 3.5 gives exactly the same qualitative and quantitative behaviour for both models.

Another way to speed up the spread of disease in the stochastic CA is to by change  $k$  to 4 instead. But in this case the stochastic automaton shows too fast spread.

For the **Stochastic CA with decay function** (as suggested in section 4.1), and for initial conditions, which assume, that the infected individuals are situated in the centre of the domain, parameter values  $k=3$  and  $d=1.8$  result in a very good correspondence with results from the modified HPP LGCA model.

**Stochastic CA with decay function vs. random-motion-FHP LGCA.** Basis for investigations are parameter values  $n=200$ ,  $\tilde{r}=0.6$  for the Stochastic CA model with decay function, and  $n=81$ ,  $\tilde{r}=0.6$  for the random-motion FHP LGCA model. The initial conditions assume about 20.000 uniformly distributed susceptibles and 20 infected in the centre. The values  $k=6$  and  $d=1.3$  show very good correspondence of the qualitative and quantitative behaviours (Figure 10).

**Difference Equations model vs. Stochastic CA.** For comparison, the following initial values are chosen:  $S(0)=39.000$ ,  $I(0)=1.000$ ,  $R(0)=0$ , with parameters  $a=0.2$  and  $r=6 \times 10^{-5}$ .

The difference equations (4) assume perfect homogeneous mixing of the population and the likelihood of contact being the same for all individuals. This would mean that all the cells in the stochastic CA establish contact with each other, requiring the following parameter values:

$$k=4.000, d \geq \sqrt{200^2 + 200^2}$$

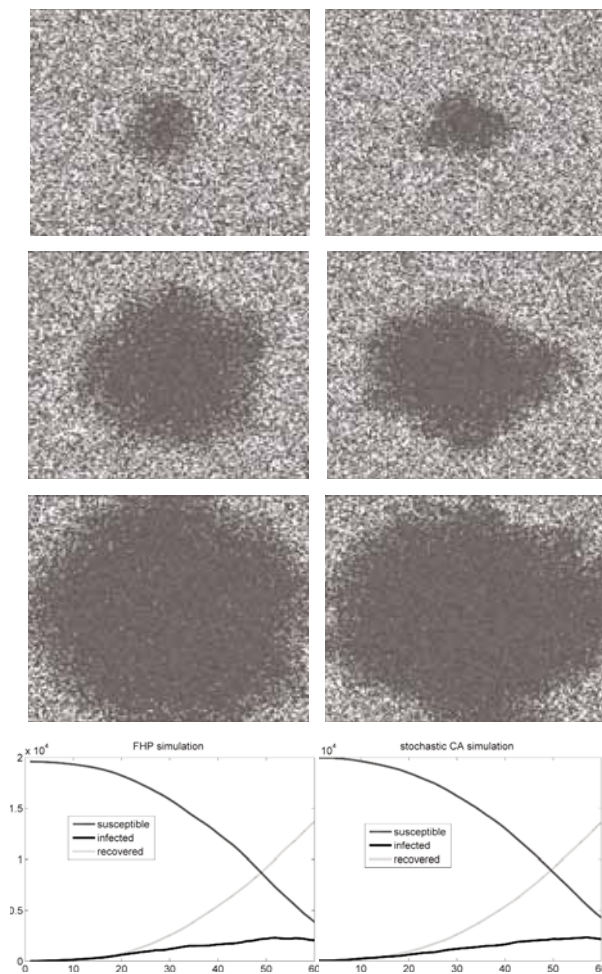


Figure 10: Simulation results with the random-motion FHP CA model ( $n = 81$ ), at left, and with stochastic CA model ( $n = 200$ ) with decay function ( $d = 13$ ), at right; lattices shown for  $t = 20$ ,  $t = 40$ , and  $t = 60$ .

Simulations with such high parameters would require extremely much computational effort. If  $k$  should be a lower value, the infection rate must be  $(N/k)$  - times higher in order to preserve the same overall rate of infections. This yields to the infection rate  $\tilde{r} = r \cdot \frac{N}{k}$  for the stochastic CA. Simulations show, that parameter values  $k=20$ ,  $d=10$ , and  $\tilde{r}=0.1$  give results, which correspond acceptably.

#### 4.3 Remarks on the Stochastic CA Model

The stochastic CA model needs much less code than the LGCA models, and the simulations run faster, especially if only few infected individuals are left and the number of contacts per time step is not too high (at least in so far as straight forward implementations are concerned). A more flexible definition of the neighbourhood and of the contact behaviour is possible, but motion of individuals is neglected.

Anti-epidemic strategies can not be directly translated between LGCA models and the stochastic CA model. A way to model such strategies in the stochastic CA would be vaccinating individuals and reducing the number of contacts or the radius of interaction. These policies intend to change certain features of the population (compare quarantining), which is not the case for simple vaccination strategies in the LGCA model. As in the LGCA model, incubation periods can be implemented. For handling the parameters and adapting demographic and socioeconomic features, fuzzy control can be useful.

#### References

- [1] H. Hötzendorfer, N. Popper, F. Breiteneker: *Temporal and Spatial Evolution of a SIR-type Epidemic – ARGESIM Comparison C17 – Definition*. SNE 41/42, pp 42-44, December 2004.
- [2] H. Hötzendorfer, F. Breiteneker: *A Directly Programmed Implementation of ARGESIM Comparison C17 'SIR-type Epidemic' using MATLAB*. SNE 41/42, pp 45, December 2004.
- [3] D. Smith, L. Moore: *The SIR Model for Spread of Disease*. Duke University, [www.math.duke.edu/education](http://www.math.duke.edu/education)
- [4] L. Zager: *Epidemics and Networks – A Little Survey*. EECS, MIT, February 2006.
- [5] H. Fukš, A. Lawniczak: *Individual-based Lattice Model for Spatial Spread of Epidemics*. Discrete Dynamics in Nature and Society, vol. 6, no. 3, pp. 191-200, 2001.
- [6] D. A. Wolf-Gladrow: *Lattice-Gas Cellular Automata and Lattice Boltzmann Models*. Springer, 2000.
- [7] S. Venkatachalam, A. R. Mikler: *Towards Computational Epidemiology: Using Stochastic Cellular Automata in Modeling Spread of Diseases*. Proc. 4th Annual International Conference on Statistics, Mathematics and Related Fields, Honolulu, HI, January 2005.

**Corresponding authors:** Günter Schneckenreither, F. Breiteneker; {gschneck, fbreiten}@osiris.tuwien.ac.at  
Institute for Analysis and Scientific Computing  
Vienna University of Technology  
Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria

Nikolas Popper, Günther Zauner  
'Die Drahtwarenhandlung' - Simulation Services  
Neustiftgasse 57-59, 1070 Vienna, Austria

Received: September 20, 2006

Revised: November 12, 2006

Accepted: November 30, 2006

# Modelling and Identification of a Laboratory Helicopter

G. Karer and B. Zupančič, University of Ljubljana  
*gorazd.karer@fe.uni-lj.si*

In this paper modelling and identification of a laboratory helicopter with the final aim to design a control system is presented. The CE 150 made by Humusoft is a laboratory helicopter designed for studying system dynamics and control engineering principles. First, the helicopter set-up is depicted. Next, theoretical and empirical modelling is systematically described. Identification of the necessary parameters is tackled and the results are presented. Finally, model validation is discussed and a simple control approach is proposed.

## Introduction

Modelling and simulation are very important approaches for designing control systems. Therefore, laboratory set-ups, which model real processes, and mathematical models have a significant role [1–3]. The CE150 is a laboratory helicopter made by Humusoft [4]. It is used for studying system dynamics and control engineering principles from the theoretical point of view and enables a wide range of practical experiments in the fields of modelling, simulation and control. The goal of modelling and identification is to prepare a basis for the students' laboratory assignments, such as designing a multivariable controller that ensures satisfactory control in the whole operating range. There are two well known modelling approaches: theoretical and experimental. Usually, both approaches have to be combined, which is also the case in modelling of the laboratory helicopter.

## 1 The Laboratory Helicopter Set-up

The laboratory helicopter set-up (see Figure 1) comprises a helicopter body carrying two motors, which drive the main and the tail propeller of the helicopter, and a servomechanism, which shifts the centre of gravity by moving a weight along the helicopter's horizontal axis. The helicopter body is connected to a base so that two degrees of freedom are enabled:

- rotation around horizontal axis  $\rightarrow$  pitch angle  $\psi$ ;
- rotation around the vertical axis  $\rightarrow$  azimuth  $\varphi$ .

The axes of the main and tail rotor and the vertical and horizontal helicopter axis are perpendicular to each other. The helicopter model can be represented as a non-linear multi-variable system with three inputs (measured in machine units  $[-1, 1]$ ):

- $u_1$  – voltage driving the main motor;
- $u_2$  – voltage driving the tail motor;
- $u_3$  – position of the servomechanism (weight);

and two outputs (measured in radians):

- $\psi$  – pitch angle;
- $\varphi$  – azimuth.

From now on, let us presume that the weight never moves from the neutral position during operation of the helicopter ( $u_3=0$ ). Otherwise, we should model the movement as a disturbance that affects both the centre of gravity and the moment of inertia of the helicopter body. However, this is beyond the scope of this paper.

The inputs  $u_1$  and  $u_2$  are measured in machine units ranging from  $-1$  to  $1$ . An interface unit, which connects the helicopter and the computer, converts the inputs from machine units to appropriate voltage values that drive the motors. Output  $\psi$  denotes the pitch angle, i.e. the angle between the vertical axis and the longitudinal axis of the helicopter body, whereas  $\varphi$  denotes the azimuth, i.e. the angle in the horizontal plane between the longitudinal axis of the helicopter body and its zero position. Both angles are measured in radians.

The voltage driving the main motor  $u_1$  and the voltage driving the tail motor  $u_2$  affect both the pitch angle  $\psi$  and the azimuth  $\varphi$ , therefore we can say that the mentioned interactions make the system multivariable. However, it is possible to fix one (or both) degree of freedom by tightening the intended screw(s) in the helicopter base when needed.

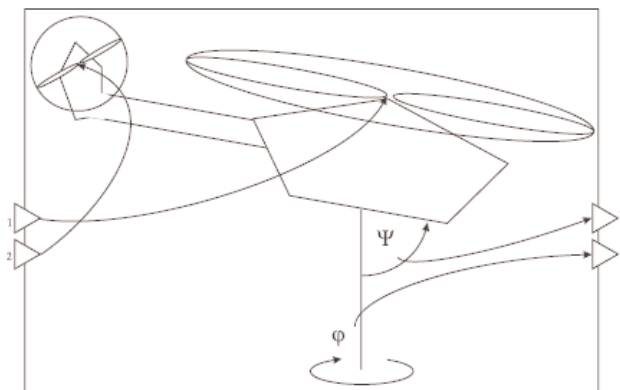


Figure 1: The laboratory helicopter set-up.

There are analogue connections between the helicopter and the interface unit, which converts the signals from analogue to digital and vice versa. The interface unit is connected to a computer via a multifunction input/output card. All the experiments are done in Matlab-Simulink environment using Real Time Toolbox [5].

## 2 Theoretical Modelling

When modelling a system it is important to find a balance between simplicity and complexity of the model, according to its purpose and operating conditions. The model has to be clear, concise and flexible, yet it must consider all the relevant sub-processes in the system. Modelling of dynamic systems is a cyclic process, therefore usually many iterations are needed before a satisfactory model is obtained [6]. Sometimes validation of a particular sub-system gives unsatisfactory results.

Hence, another approach has to be considered and some of the previously neglected properties have to be taken into account. In the following section, modelling procedure will be described. Obviously, only the last iteration of the procedure is presented in the paper and the model is proposed in its final form.

### 2.1 Modelling of Sensors

Both angle sensors are linear, so the modelling is rather straightforward. Signals from the sensors are denoted by  $y_\Psi$  and  $y_\varphi$ .

$$\Psi = k_\Psi \cdot y_\Psi + y_{\Psi 0} \quad (1)$$

$$\varphi = k_\varphi \cdot y_\varphi + y_{\varphi 0} \quad (2)$$

### 2.2 Vertical Plane Dynamics

We start with torque balance equation around the horizontal axis.

$$J_1 \ddot{\Psi} = \tau_1 - \tau_{F_g} - \tau_{B\Psi} - \tau_G \quad (3)$$

Here,  $J_1$  denotes the moment of inertia around the horizontal axis,  $\tau_1$  the torque of the propulsion force of the main propeller,  $\tau_{F_g}$  the torque of the gravitation force of the helicopter body,  $\tau_{B\Psi}$  the friction torque and  $\tau_G$  the gyroscopic effect caused by rotation of the main propeller and rotation of the helicopter body around the vertical axis.

The torque of the propulsion force of the main motor is modelled experimentally.

The static characteristic is derived from the dominant ventilator characteristic as described in eq. (4). The motor-propeller dynamics are relatively fast comparing to the dynamics of the helicopter body. Therefore, it can be modelled as a 2<sup>nd</sup> order transfer function with two equal poles – see eq. (5).

$$\tau_1 = a_1 \cdot u_{1mot}^2 + b_1 \cdot u_{1mot} \quad (4)$$

$$\frac{U_{1mot}(s)}{U_1(s)} = \frac{1}{(T_1 \cdot s + 1)^2} \quad (5)$$

The other torques are defined by basic physical laws:

$$\tau_{F_g} = F_g \cdot l \cdot \sin \Psi = M_g \cdot \sin \Psi \quad (6)$$

$$\tau_{B\Psi} = B_{\Psi 1} \cdot \dot{\Psi} + B_{\Psi 2} \cdot \text{sign}(\dot{\Psi}) \quad (7)$$

$$\tau_G = K_{gyro} \cdot \tau_1 \cdot \dot{\varphi} \cdot \cos \varphi \quad (8)$$

In eq. (6),  $F_g$  is the gravitation force,  $l$  is the lever between the centre of gravity and the horizontal axis of the helicopter body, and  $M_g$  is the appropriate torque. In eq. (7) we assume that the friction torque is a sum of a Coulomb part and a linear part, where the latter is proportional to the angular velocity. The torque caused by the gyroscopic effect (see eq. (8)) is proportional to the product of the angular velocity of the main propeller, the angular velocity of the body around the vertical axis, and the cosine of the pitch angle  $\varphi$ . It is presumed that  $\tau_1$  is proportional to the angular velocity of the main motor. The constant of the gyroscopic coupling is denoted by  $K_{gyro}$ .

### 2.2 Horizontal Plane Dynamics

We start with torque balance equation around the vertical axis.

$$J_2 \ddot{\varphi} = \tau_2 - \tau_{B\varphi} - \tau_r \quad (9)$$

Here,  $J_2$  denotes the moment of inertia around the vertical axis,  $\tau_2$  the torque of the propulsion force of the tail propeller,  $\tau_{B\varphi}$  the friction torque, and  $\tau_r$  the reaction torque caused by the main propeller rotation. Eqs. (10), (11) and (12) are defined in the same manner as in the previous subsection. The other torques are defined using basic physical laws.

$$\tau_2 = a_2 \cdot u_{2mot}^2 + b_2 \cdot u_{2mot} \quad (10)$$

$$\frac{U_{2mot}(s)}{U_2(s)} = \frac{1}{(T_2 \cdot s + 1)^2} \quad (11)$$

$$\tau_{B\varphi} = B_{\varphi 1} \cdot \dot{\varphi} + B_{\varphi 2} \cdot \text{sign}(\dot{\varphi}) \quad (12)$$



The reaction torque caused by the main propeller rotation is also modelled experimentally. The static characteristic is derived from the dominant ventilator characteristic as described in eq. (13). The dynamics are modelled as a transfer function in eq. (14) where the denominator is the same as in eq. (5). Due to the moment of inertia of the main rotor, which affects the reaction torque, numerator dynamics are assumed as well.

$$\tau_r = a_r \cdot u_{lr}^2 + b_r \cdot u_{lr} \quad (13)$$

$$\frac{U_{lr}(s)}{U_1(s)} = \frac{T_{1r} \cdot s^2 + T_{2r} \cdot s + 1}{(T_1 \cdot s + 1)^2} \quad (14)$$

## 2.2 The Whole System – Simulink Model

The whole system – Simulink model Using eqs. (1) – (14) a Simulink block diagram of the whole system depicted in Figure 2 was developed.

## 3 Measurements and Identification of the Parameters

Once the theoretical model of the laboratory helicopter set-up is obtained, 22 parameters have to be determined:

$$k_\psi, y_{\psi 0}, k_\phi, y_{\phi 0}, \\ M_g, J_1, B_{\psi 1}, B_{\psi 2}, J_2, B_{\phi 1}, B_{\phi 2}, \\ T_1, a_1, b_1, T_2, a_2, b_2, \\ T_{1r}, T_{2r}, a_r, b_r \text{ in } K_{gyro}.$$

There are two possible approaches:

- direct measurements of the accessible physical quantities;
- identification, i.e. experimental estimation of the parameters by means of measuring inputs and outputs [7–9].

The **angle sensors**  $k_\psi, y_{\psi 0}, k_\phi, y_{\phi 0}$  can be calibrated by simple angle measurements:

$$k_\psi = \pi / 1024 \text{ [rad]}, \\ y_{\psi 0} = 275 \pi / 1024 \text{ [rad]}, \\ k_\phi = \pi / 1024 \text{ [rad]}, \\ y_{\phi 0} = 0.7 \pi \text{ [rad]}.$$

The **torque of the gravitation force** of the helicopter body  $M_g$  estimated by hanging  $N$  weights weighing  $m_i$  ( $i = 1, \dots, N$ ) on the helicopter body behind and in front of the horizontal axis. We can obtain  $M_g$  by measuring the pitch angle  $\psi$  and the appropriate levers  $l_i$  in steady state:

$$M_g = \frac{\sum_{i=1}^N m_i \cdot g \cdot l_i}{\sin \Psi} \quad (15)$$

In order to mitigate the effect of Coulomb friction on the measurements and thus obtain better accuracy, it is reasonable to carry out many measurements with different weights hung at different places and then calculate the average  $M_g = 0.07088 \text{ N m}$

### Vertical plane dynamics

$$J_1, B_{\psi 1}, B_{\psi 2}$$

The parameters concerning the vertical plane dynamics are estimated by means of identification. When conducting this experiment both motors are turned off and rotation in the horizontal plane is disabled by tightening the corresponding screw.

Due to constrained motion of the pitch angle  $\psi$ , the model base is fixed in the perpendicular position, which means that the helicopter base is tilted 90 degrees so that the vertical axis is put in a horizontal position. Obviously, the new position has to be considered when calibrating the sensor (offset  $\pi/4$ ). The helicopter response to an initial condition (tilt of the helicopter body) is finally recorded.

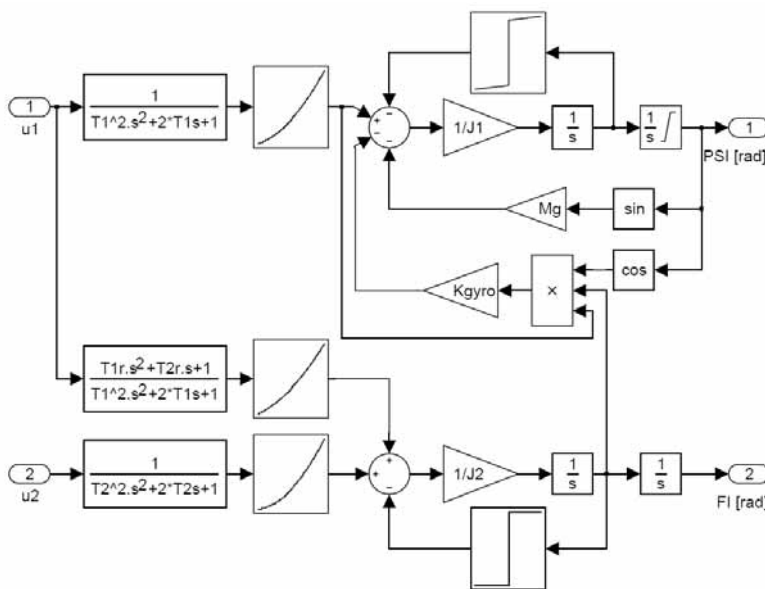


Figure 2: The block diagram of the whole system.

The parameters are identified by offline model adaptation, which is a method suitable for nonlinear identification problems such as the laboratory helicopter. In this case, the minimisation of the criterion function, which takes into account the sum of squares of the difference between the real response  $y_{real}$  and the model response  $y_{mod}$  to the initial condition (see eq. (16)), is carried out using the Nelder-Mead method:

$$\min_{J_1, B_{\psi 1}, B_{\psi 2}} \sum_{k=1}^N (y_{real}(k, \mathbf{u}) - y_{mod}(k, \mathbf{u}, J_1, B_{\psi 1}, B_{\psi 2}))^2 \quad (16)$$

Before the optimisation it is reasonable to roughly estimate the parameters from the response signal, in order not to end up in a local minimum that provides inadequate results. The optimisation returns the following parameters:

$$\begin{aligned} J_1 &= 2.43 \cdot 10^{-3} \text{ kg m}^2 \\ B_{\psi 1} &= 1.53 \cdot 10^{-4} \text{ kg m}^2 \text{ s}^{-1} \\ B_{\psi 2} &= 1.03 \cdot 10^{-4} \text{ kg m}^2 \text{ s}^{-1} \end{aligned}$$

#### Horizontal plane dynamics $J_2, B_{\phi 1}, B_{\phi 2}$

The parameters concerning the horizontal plane dynamics are identified in a similar fashion. Both motors are turned off and rotation in the vertical plane is disabled by tightening the corresponding screw. The model base is fixed in the perpendicular position, therefore the influence of the torque of the gravitation force of the helicopter body has to be considered in the adjusted model as well. The helicopter response to an initial condition (tilt of the helicopter body) is finally recorded. Again, the parameters are identified by offline model adaptation:

$$\begin{aligned} J_2 &= 2.02 \cdot 10^{-3} \text{ kg m}^2 \\ B_{\phi 1} &= 5.50 \cdot 10^{-6} \text{ kg m}^2 \text{ s}^{-1} \\ B_{\phi 2} &= 5.96 \cdot 10^{-4} \text{ kg m}^2 \text{ s}^{-1} \end{aligned}$$

#### Torque of propulsion force of main propeller, $T_1, a_1, b_1$ .

The parameters  $a_1$ , and  $b_1$ , represent the static characteristic of the main motor-propeller subsystem. They can be identified from the measurements of the static characteristic:

$$a_1 \cdot u_1^2 + b_1 \cdot u_1 = M_g \sin \psi$$

Using the least squares method, the following parameters are obtained  $a_1 = 0.1244 \text{ Nm}$ ,  $b_1 = 0.0496 \text{ Nm}$ .

Parameter  $T_1$  is identified by offline model adaptation to a step response signal of the helicopter body  $T_1 = 0.0904 \text{ s}$ .

#### Torque of propulsion force of tail propeller $T_2, a_2, b_2$

In a similar manner as in the previous subsection, but with the helicopter in the perpendicular position, the following parameters are obtained:  $a_2 = 0.1959 \text{ Nm}$ ,  $b_2 = 0.0202 \text{ Nm}$ ,  $T_2 = 0.0567 \text{ s}$ .

#### Reaction torque of propeller rotation $T_{1r}, T_{2r}, a_r, b_r$

In a similar manner as in the previous subsection, again with the helicopter base in the perpendicular position, the following parameters are obtained:

$$\begin{aligned} a_r &= 0.0148 \text{ Nm}, \quad b_r = 0.0108 \text{ Nm}, \\ T_{1r} &= 0.0017 \text{ s}, \quad T_{2r} = 0.1908 \text{ s} \end{aligned}$$

#### Gyroscopic effect $K_{gyro}$

Parameter  $K_{gyro}$  is identified by offline model adaptation. The main motor input  $u_1$  is set to a constant value, so that it can be assumed the angular velocity of the main propeller is constant. In this manner, the system should stabilize at a certain pitch angle  $\psi$ , which should not be too close to  $\pi/4$ .

Next, the body of the helicopter is rotated (by hand, without touching the body) around its vertical axis, so that the angular velocity is approximately constant. The rotation causes the gyroscopic effect, which results in a change of the pitch angle. The signals  $\psi$  and  $\phi$  are recorded. The angular velocity around the vertical axis is derived ( $d\phi/dt$ ) and used for identification as a subsystem input. The result is  $K_{gyro} = 0.3185 \text{ s}$ .

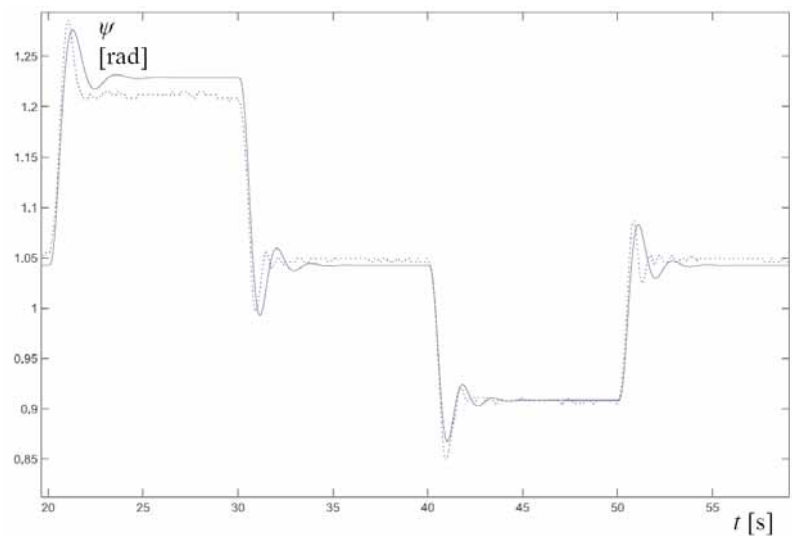


Figure 3: Real system (dotted line) and model response –  $\Psi$ .

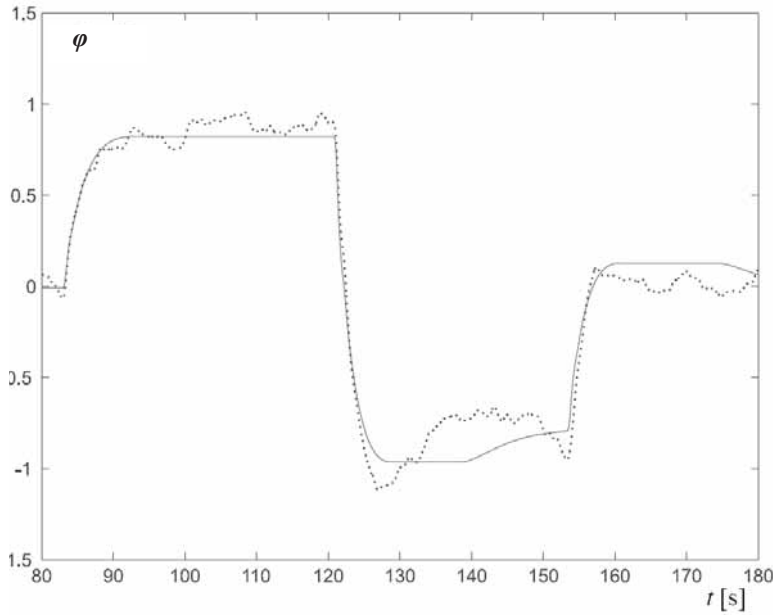


Figure 4: Real system (dotted line) and model response –  $\varphi$ .

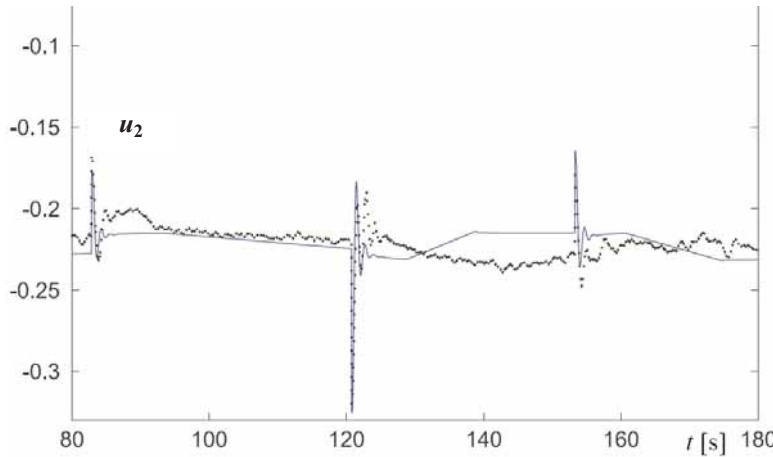


Figure 5: Real system (dotted line) and model response –  $u_2$ .

#### 4 Model Validation

Model validation is carried out in several steps. First, the dynamics in the vertical plane are validated. Since the process is operating in a stable region, an open loop experiment is conducted. In this case, input  $u_1$  is a multi-step signal ( $u_1 = 0,53 \rightarrow 0,56 \rightarrow 0,53 \rightarrow 0,50 \rightarrow 0,53$ ). The comparison between the response of the pitch angle  $\psi$  of the real helicopter (dotted line) and of the model (solid line) is shown in Figure 3.

Due to instability, the horizontal plane dynamics can not be validated in an open loop experiment. Therefore, a controller (a modified PI-D controller is used) has to be provided, which will enable the tracking of an azimuth reference.

The azimuth reference  $\varphi_{ref}$  is a multi-step signal ( $\varphi_{ref} = 0 \rightarrow \pi/4 \rightarrow -\pi/4 \rightarrow 0$ ). The comparison between the response of the azimuth  $\varphi$  of the real helicopter (dotted line) and of the model (solid line) is shown in Figure 4. In both cases the same controller is used. In the case of the real system response, we can see that the azimuth  $\varphi$  is subject to significant external disturbances.

Figure 5 shows the input signal  $u_2$  provided by the controllers in case of the real system (dotted line) and in case of the model (solid line). The input signal  $u_2$  is the average of 20 measurements. This way it is possible to reduce the impact of noise and the effect of drift of the parameters. There is especially noticeable drift of the parameters in the static characteristic of the tail motor-propeller subsystem, therefore it sometimes needs adjustment, in our case by factor  $k = 0.88$ .

From the comparison we can see that the signals are quite similar in the transient states. However, there are some perceivable differences, especially in the interval from 125 s to 155 s, which occur due to the torque of the Coulomb friction.

The Coulomb friction causes the helicopter body to stop rotating not only if the input  $u_2$  is set to a certain exact value, but also if it is set in a narrow band around that value. This means that two slightly different input signals  $u_2$  can cause the same output signal  $\varphi$ . Obviously, the maximum torque difference generated by the two input signals is  $2 \cdot B_{\varphi 2}$ .

Eq. (18) proves that the input signal difference causes a torque difference that is smaller than the mentioned bound:

$$\begin{aligned} \left| \frac{d\tau_2}{du_2} \right|_{u_{2,0} \approx -0,22} \cdot \Delta u_2 \Big|_{\max} &= \\ &= \left| \left( 2 \cdot a_2 \cdot k^2 \cdot |u_{2,0}| + b_2 \cdot k \right) \cdot \Delta u_{2\max} \right| \approx \\ &\approx 0,9 \cdot 10^{-3} \text{ Nm} \leq 1,2 \cdot 10^{-3} \text{ Nm} = 2 \cdot B_{\varphi 2} \quad \square \end{aligned} \quad (18)$$

When assessing the usability of the model, we encountered a simulation problem; namely, because of the nonlinearity caused by Coulomb friction, the simulation runs very slowly, despite the use of the solver for stiff systems (ode23s). Hence, it is reasonable to replace the nonlinear function with a piecewise linear function, i.e. to introduce a very high gain (in our case  $10^8$ ) and an appropriate saturation as a substitute for infinite gain around zero angular velocity. Such modification has practically no influence on the results; however, it does speed up the simulation considerably and what is more, even enables real-time (Matlab 7.0.1 on Celeron 2.4 GHz, 512 MB.5) experiments and thus allows for online comparison of real and simulated signals.

## 5 Control

As mentioned, the developed mathematical model is intended for the design of control systems. In this section, a simple control approach is presented. Two independent modified PI-D controllers for each degree of freedom respectively are used, without considering the multivariable nature of the system. However, such basic approach can still turn out useful for stabilizing the helicopter body in a reference position, as shown in Figure 6.

## 6 Conclusion

In the paper, modelling and identification of a laboratory helicopter was dealt with. The CE 150 laboratory helicopter made by Humusoft was presented as a multivariable system with two inputs and two outputs.

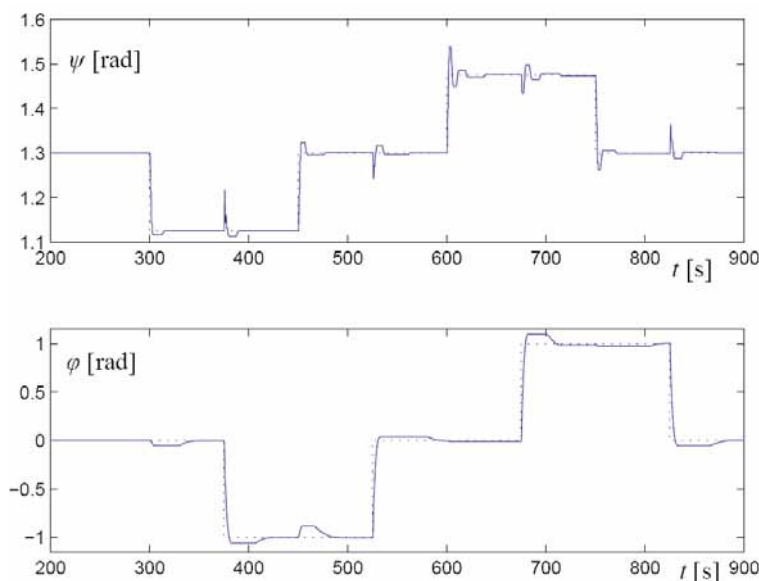


Figure 6: Reference trajectories (dotted line) and closed loop model response –  $\psi$ ,  $\varphi$ .

Next, modelling of the helicopter was systematically tackled by disassembling the system into simpler subsystems, i.e. modelling of the sensors, vertical plane dynamics and horizontal plane dynamics. Furthermore, the vertical and horizontal plane dynamics have been analysed by modelling each relevant torque separately. In addition, measurement and identification of all the parameters needed was illustrated. Finally, validation of the developed mathematical model was treated. The validation results suggest that the developed mathematical model adequately represents the laboratory helicopter. In conclusion, a simple control approach was presented, which involves two independent modified PI-D controllers for each degree of freedom respectively.

## References

- [1] F. E. Cellier: *Continuous System Modelling*. Springer Verlag, New York, 1991.
- [2] O. Egeland, J. T. Gravdahl: *Modeling and Simulation for Automatic Control*. Marine Cybernetics.
- [3] R. Karba: *Modeliranje procesov*. Založba FE in FRI, Ljubljana, 1999.
- [4] Humusoft: *CE 150 helicopter model: User's manual*. Humusoft, Prague, 2002.
- [5] Humusoft: *Real time toolbox for use with SIMULINK: User's manual*. Humusoft, Prague, 2002.
- [6] D. Matko, B. Zupančič, R. Karba: *Simulation and modelling of continuous systems: a case study approach*. Prentice Hall, New York, 1992.
- [7] J. N. Juang: *Applied System Identification*. Prentice Hall, Englewood Cliffs, 1994.
- [8] L. Ljung: *System Identification: Theory for the user*. Prentice Hall, Englewood Cliffs, 1987.

**Corresponding author:** G. Karer  
Faculty of Electrical Engineering,  
University of Ljubljana  
Tržaška 25, 1000 Ljubljana, Slovenia  
[gorazd.karer@fe.uni-lj.si](mailto:gorazd.karer@fe.uni-lj.si)  
[harald.teufelsbauer@boku.ac.at](mailto:harald.teufelsbauer@boku.ac.at)

Received: September 12, 2006

Revised: October 18, 2006

Accepted: October 22, 2006



## SHORT NOTES

### Model-oriented Data-driven Architecture for Microsimulation

Imre Sinka, siDesign, Budapest; Hungary; [imre.sinka@sidesign.hu](mailto:imre.sinka@sidesign.hu)  
Istvan Molnar, Bloomsburg University of Pennsylvania, USA; [molnar@bloomu.edu](mailto:molnar@bloomu.edu)

This paper describes the use of large data systems for microsimulation studies. Basic technologies are discussed, which are able to satisfy all user requirements. First, authors present the relevant characteristics of the microsimulation application field, where model-oriented data-driven architectures are developed, and examine some of the salient problems and opportunities. After a short analysis of specific meta-database and web-service technologies, authors show how selected technologies can be applied to support microsimulation model development. Finally, references are made to the practical use. The paper concludes with a summary of research results and future plans.

#### Introduction

Socioeconomic systems are complex, extremely sensitive and of great social and economic importance. Microsimulation is a method able to handle complex socioeconomic systems by creating and studying a model that makes intensive use of the statistical data of the observed objects. These objects are the so-called micro units of the socioeconomic system; the person, the family or the household. Microsimulation models use simulation techniques in order to study the behaviour of micro level units in time.

Microsimulation models have different data elements: initial model data, intermediate and/or final simulation data; all of these data are stored for further analysis.

Model behaviour in microsimulation models is described using algorithms, which reflect the behaviour processes of the micro units and represent their environment.

By using this method, special care is taken to do the data analysis and the estimation of simulation model parameters. The microsimulation model is working in an experimental framework in order to study the effects of policy changes on the microsimulation model behaviour. A typical application environment of a microsimulation model that could be used by state administration officials for decision-making is presented in Figure 1.

The major characteristic of such architecture – beside its complexity – is that data are stored and processed in

a distributed way in different databases in order to support data maintenance by different, mainly government authorities.

Most of the data are available in different time series, in such a way that data content is hard to define, it might change with the progress of time, and data integrity and accuracy is difficult to maintain.

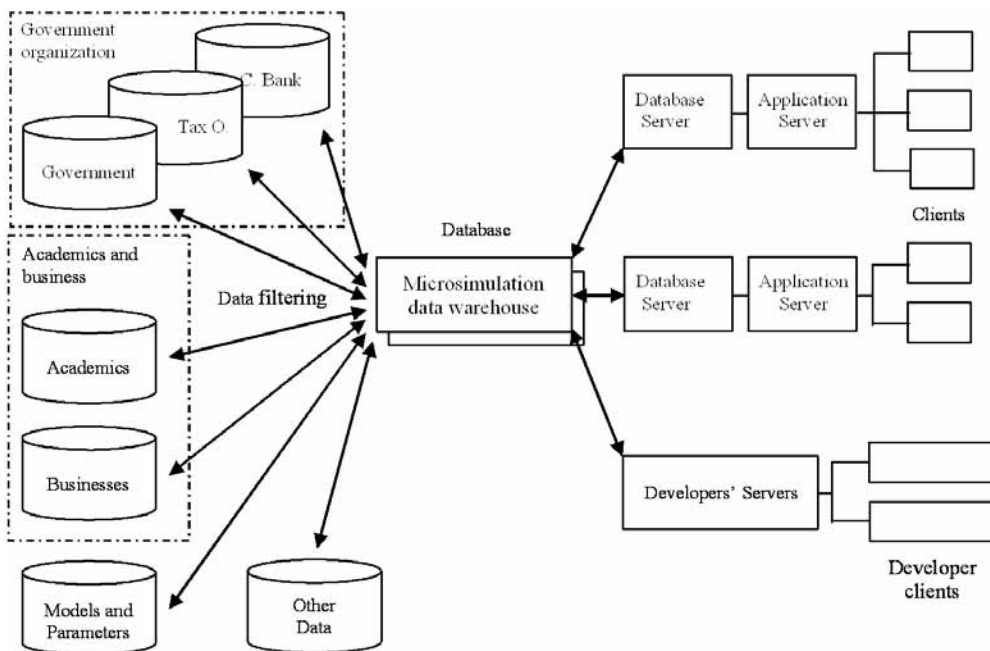


Figure 1: Model-oriented data-driven architecture.

One of the biggest problems could be the management of the same data content under different names and different data content under the same name. Some of the simulation modelling problems are traditionally solved by using synthetic data sets (e.g., merging, imputing), which means that artificial data sources (and methods) are applied instead of traditional system data sampling (for the methodology, see [1] and [2]). In addition to the data of the 'real system', different kind of simulation data are regularly stored and retrieved.

Maintenance and use of complex data-driven architectures, like those presented above, might cause methodological and technical problems. Most of the recently published studies focus on PC-oriented microsimulation applications and provide related data management solutions (see e.g., [3]). Little attention has been paid to the management of large data sets for distributed microsimulation applications. In this paper, authors pursue the approach presented in [4] and extend the database use with new techniques and concepts.

## 1 Applied Technologies

Authors call data systems 'model-oriented', when those were designed and are in use to support the use of a class of mathematical models. In the present case, the data system has been designed and used for microsimulation. Because the data represented in the system come from different sources (i.e. measured, synthetic and simulated data are present), the simultaneous use of data by simulation models can be considered a special type of data-driven application.

There are some general requirements for microsimulation software and software development environments, which help to establish a broad framework for our application development environment:

- Network-oriented data and model access.
- Distributed model development, execution/and data analysis.
- Platform-independent hardware and software solutions based on open standards.
- Data and network security.
- User friendliness, standardised user interfaces.
- Efficiency of software development during the whole software life cycle.

Some of the listed requirements can be at least partially realised using webservice technologies, as suggested in [4]. Others, like network-oriented data access and data analysis, as well as software development efficiency, require different technologies.

The described data systems can be best managed using a centralised management system, which consists of crucial information about all individual microsimulation data available in the data system; a meta-database, which consists of data about microsimulation data. Based on the applied microsimulation models, the microsimulation data meta-database must be model-oriented. The major advantages of meta-databases for microsimulation data consist of, among others, increased data quality and overall cost efficiency.

When using a meta-database for microsimulation data, all used data can be best referred to by using the appropriate data references of the meta-database. Because all methods, which can be executed on the microsimulation database data, can also be considered as data, it is evident that the most efficient solution to store the methods themselves is to store them in database(s), i.e., by creating appropriate method-database(s). Having method-databases requires the same data management practice as using a meta-database; i.e., a meta-database for microsimulation methods must be created.

The model-oriented data-driven microsimulation architecture presented above uses the following technologies:

- **Meta databases:** database of data about microsimulation data (how microsimulation data are collected/generated, accessed, processed, etc.).
- **Data warehouse:** data are stored in a special structure based on data-related dimensions (e.g., time, collected by, data content).
- **Object-oriented data modelling (OODM):** a data modelling paradigm, which applies object-oriented approach and data modelling and best articulated in semantic data modelling (see [5]).
- **Object oriented programming (OOP):** a programming paradigm, described in detail in [6].
- **Service Oriented Architecture (SOA):** Web services are applications implemented as Web-based components, which offer certain functionality to clients via the Internet. The components have well-defined interfaces. Once deployed, Web services can be discovered, used/and reused by consumers (clients, other services or applications) as building blocks. Web service architecture is built on open standards and vendor-neutral specifications.

The architecture is presented in [4] and the publications referred to there.

## 2 Designing Meta-Databases for Microsimulation

Authors consider a fundamental principle of the microsimulation application and development environment that two types of meta-databases are to be developed, one for the microsimulation data and another for the microsimulation model methods.

### 2.1 Meta-database for Microsimulation Data

The meta-database stores data about each microsimulation data into four groups:

- **Data content:** used for identification and usability decisions. The following data are stored, among others: unique id, mnemonic, description, validation, version, usability role, and accessibility.
- **Data access:** access-related data are stored in specific order, depending on the storage medium, the location, the type of access (e.g. local or networked), the applied protocol (e.g., http, ftp, soap), etc.
- **Data processing:** most of the data are used for further calculations; therefore algorithms/methods used for processing these data must be identified.

The full specification of each algorithm/method is given as data content in two different ways: using direct statements (all processing steps of the algorithm are described directly, there are no loosely coupled method calls) and using indirect statements (the algorithm consists of loosely coupled method calls; e.g., Java RMI, Microsoft COM).

- **Other data:** technical data might also be stored e.g., action log, access log, error log.

The architecture with extended meta-database for microsimulation data is presented in Figure 2.

### 2.2 Meta-database for Microsimulation Methods

A meta-database for microsimulation methods can be considered as a separate meta-database. In this case all meta-database data groups, except the data processing, can be used for the method definition. (In special cases, the meta-database for microsimulation methods can be directly integrated into the data processing data group of the meta-database for microsimulation data). The architecture with extended meta-database for microsimulation methods is presented in Figure 3.

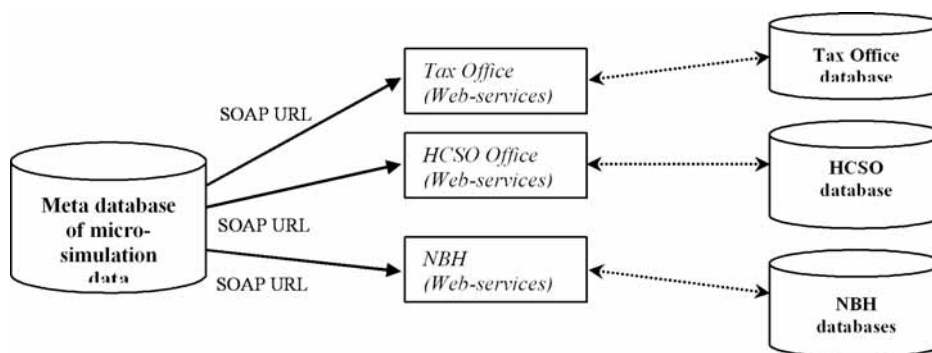


Figure 2: Meta-database of microsimulation data.

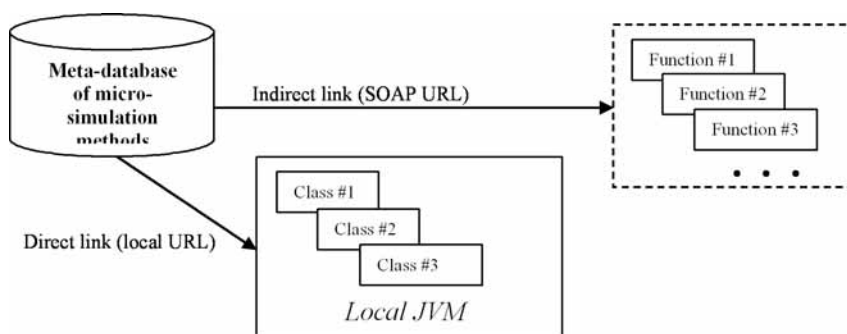


Figure 3: Meta-database of microsimulation methods.

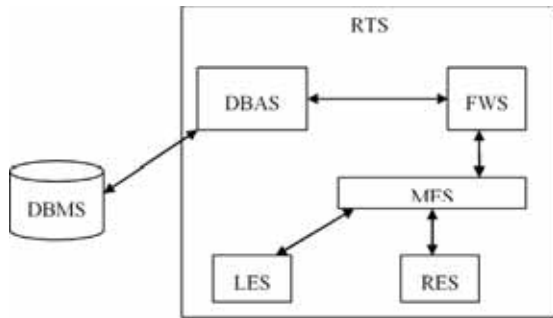


Figure 4: The architecture of the software system based on meta-databases.

### 3 Software Components of the Microsimulation system

As presented above, the meta-databases for microsimulation are characterised by networked remote access and therefore both methods and data are accessed using information stored in meta-databases. Systems described here can be realised using the following major software-components (Figure 4):

- **RunTime System (RTS):** runs all software components. The JBoss Application Server has been used, which is an Open Source J2EE implementation.
- **FramWork System (FWS):** synchronises the components' run, working as the center of the architecture communicating with the components. Implemented in J2EE.
- **Remote Execution System (RES):** runs all loosely coupled methods (for both remote data access and remote method execution). Implemented in J2EE.
- **Local Execution System (LES):** runs local methods. Implemented in J2EE.
- **Messaging System (MES):** manages communication between/among the components. The JBossAS JMS has been used.
- **DataBase Acces System (DBAS):** provides access to all local databases (incl. meta-databases). EIS Connection Pool, which is implemented in the Application Server.
- **DataBase Management System (DBMS):** manages all local databases (incl. meta-databases). Oracle10g XE has been used.

The development of the model-oriented data-driven microsimulation system consists of two phases, which can be realised to a great extent independently from each other. In *phase one* the software system is implemented, while in *phase two* the meta-database(s) and database(s) are populated. Both phases require different team compositions and professional background.

### 4 Conclusions and Final Remarks

The paper presents a generally applicable methodology to handle model-oriented data-driven microsimulation systems and related data management problems in a smart way.

The basic idea of the solution is based on meta-databases, which are also implemented in a distributed environment using webservices. In order to increase the flexibility of data storage and data processing, the relationships stored in databases are described using object-oriented data modelling and programming concepts; the computation engine will use the stored data and methods from the same database.

Using the presented technologies, all relevant user requirements can be satisfied. The experiences with a simple model show good results, on which one can build real word applications. In order to extend the range of applications and provide wide availability of the microsimulation models, administrative regulation of data access must be implemented.

#### References

- [1] R. J. A Little, D. B. Rubin: *Statistical analysis with missing data*. New York, Wiley, 2002.
- [2] D. B. Rubin: *Multiple Imputation for Nonresponse in Surveys*. New York, John Wiley & Sons, 2004.
- [3] H. Sutherland (ed.): *Final report: EROMOD: an integrated European benefit-tax model*. EUROMOD Working Paper No. EM9/01, 2001
- [4] I. Molnar: *Microsimulation Model Development Environments*. Int. Journal of Simulation Systems, Science and Technology, Special Issue, ISSN:1473-804x [Online], ISSN: 1473-8031 [Print], Vol.6 No. 5, 2005; p. 33-41
- [5] P. Gray, K. G. Kulkarni N. W. Paton: *Object-Oriented Databases: A Semantic Data Model Approach*. New York, Prentice-Hall, 1992.
- [6] P. Coad, J. Nicola: *Object-oriented Programming*. New Jersey, Prentice Hall, 1993.

#### Corresponding author:

Istvan Molnar, [imolnar@bloomu.edu](mailto:imolnar@bloomu.edu)  
 Bloomsburg University of Pennsylvania, School of Business, Dep. Computer and Information Systems  
 Bloomsburg, Pennsylvania, 17815, U.S.A.

Imre Sinka, [imre.sinka@sidesign.hu](mailto:imre.sinka@sidesign.hu)  
 siDesign Software design and Consulting,  
 Budapest; Hungary;

Submitted: September 2006, ASIM 2006 Conference  
 suggested for publication in SNE by ASIM 2006 IPC  
 Revised SNE: November 3, 2006  
 Accepted: November 13, 2006



# Model-based Learning Classifiers for Surface Inspection Problems

Simon Seichter, Felix Breitenecker, Vienna University of Technology, Austria;

*simon.seichter@gmail.com*

Christian Eitzinger, Profactor Research Ltd., Austria

In surface inspection problems, it is crucial to model the human decision process well, which can be performed by applying learning methods to appropriate computer software. Pictures of products contain a variable number of irregularities (flaws), each described by a feature vector. The fact that the number of flaws is unknown makes it hard to model the good-bad - decision meaningfully, so different classification methods are employed and compared. There, linear classifiers are fully understood with respect to their properties. Neural networks are a more sophisticated modelling technique which imitate functionality of the human brain by a structure similar to it, whereby any desirable decision boundary can be implemented. Both methods require an input vector of fixed size, so the feature vectors have to be concatenated into one input vector. Recurrent neural networks allow all types of connections, which provides the network with some form of memory. The contribution compares the three methods and gives enhancements for quality control with recurrent neural networks.

## Introduction

In earlier times, a human inspector was indispensable for classifying any produced goods as good or bad. He or she reached a decision, based on his/her knowledge, more or less by closely looking at it. Nowadays, computers with appropriate software can handle this task in an automated process with the use of image processing. The core issue in these surface inspection problems is to model the human decision process sufficiently well. For that purpose, learning methods which are able to learn the difference between good and bad out of training samples are employed.

When wanting to classify a product, a photograph of it, for instance some part of an engine or a synthetic panel surface, is taken. The image can contain an arbitrary number of potential errors (irregularities), for instance dark or bright spots or scratches. For each of these so-called flaws, a number of characteristic features, like area, position, brightness etc., are determined and combined into one feature vector. Thus, one image can be identified with a set of feature vectors. In the present work, it is assumed that this image preprocessing has already been done. The software should be able to decide whether the product is usable or defective by considering all feature vectors of all flaws. During the learning phase, this decision is still corrected or approved by a human operator, but after the system has learned the decisive properties well enough, it works without further supervision.

Generally speaking, a classifier is a mathematical object which takes some input vector, performs certain operations with it and decides out of that to which class it belongs. As the samples only have to be distinguished between good and bad, only two-class classification problems are considered in the following.

Two main problems have to be addressed. On the one hand, the number of flaws per picture is variable. This could be easily handled by processing the flaws sequentially. Here, the effect of the second problem becomes evident: not only do the single flaws have an influence on the classification, but also certain combinations of them, like the relative positions of the flaws, the total area etc. It is desired that the model is able to take these aggregated values into account.

In the present work, these problems are not fully solved. Existing methods are investigated and applied to the classification problems in order to get a feeling of what is already possible. These methods are explained in greater detail in Section 2. A new approach which is very promising is tried out in the last part of the work.

## 1 Samples and Rating of the Samples

The variable number of flaws is responsible for the variable size of the input vector. As most existing classifiers need an input vector of fixed size for producing a classification result, a maximum number of features was set which describe one picture. This determines the dimension of the feature space in which some manifold that separates the samples should be found.

For testing the developed classifiers, test data were artificially generated according to specified rules. The maximum number of features per picture was set to 24, which means that the feature space was 24-dimensional.

Several combinations to achieve that number, from 24 flaws with 1 feature each to 6 flaws with 4 features each, were tried out to form the test datasets.

To get reasonable classification results, the number of training samples should be chosen high enough. This number was set to 20000 for testing purposes. However, in common practical applications, a picture can usually contain up to 50 flaws, with 10 up to 100 features each, which would yield a 500- up to 5000-dimensional feature space. When wanting to train a classifier reasonably in these cases, data-sets of enormous size would be needed, which renders it impossible to apply this in realistic tasks.

The features were generated randomly in the interval  $[-1, 1]$ , with one feature of each flaw being chosen in  $[0, 1]$ , as there is almost always at least one positive feature.

To obtain a realistic model of different aspects under which a human inspector classifies an image, the samples are rated by several rating criteria which are likely to be used in practice. These ratings simulate the human input for testing the implemented algorithms.

Among them are criteria where the relative positions of the flaws plays a role. It will be shown that these are more difficult to handle than those which consider only single flaws. An example for the latter would be the *areamax* criterion, which rates a sample bad if the area of at least one flaw exceeds a maximum threshold value. Though this sounds rather simple, it is of practical importance, as it is often the case that already a single, yet big flaw renders the product unusable as a whole.

Naturally, it is not always that easy, thus also criteria which take combinations of the flaws into account, like the *cluster formation criterion*, are investigated. A product is rated bad by that criterion if too many flaws, regardless of their properties, lie within a small region. This is also of high practical interest, as accumulations of flaws often point out weak spots of a product.

It is desirable to have equally many good and bad samples, as this makes the training algorithms more stable. If the proportion is not well-balanced, it could easily happen that suddenly all samples are classified as good or all as bad. To obtain a proportion near to 1:1, the parameters of the rating criteria were varied.

## 2 Methods Used

Several approaches can be taken to model the human decision making process. The simplest one, which is very common, is to use linear classifiers.

### 2.1 Linear Classifiers

Linear classifiers provide a good basis for comparison of the results as they are fully understood with respect to their properties. They try to find a linear boundary (a straight line, a plane or a hyperplane, according to the dimension) in feature space which separates the samples into two classes. This hyperplane can be implemented mathematically by a weight vector perpendicular to it. The decision is modelled as the sign of the product of the weight vector  $a$  with a given feature vector  $x$ :

$$x \rightarrow \begin{cases} \text{class 1} & \text{if } a^T x \geq 0 \\ \text{class 2} & \text{if } a^T x < 0. \end{cases}$$

Here, the problem why the number of flaws has to be set in advance, becomes obvious: As the weight vector is to be trained and is the same for all samples, all input vectors have to be of the same size, or otherwise the multiplication could not be carried out. Thus, it is necessary to identify each image with one single feature vector. The idea now is to stack up all feature vectors of all flaws and fill the resulting vector with zeros up to the prespecified size of the input vector. This maximum size of the input vector gives an upper bound for the number of flaws in each image, which is a severe restriction to generality.

The question arises how this method is made trainable. The answer is simple: The decision which is obtained as described above is compared to a desired target decision. Depending on the consistency or divergence of these two decisions, the weight vector is altered. This process terminates when and only when all samples are correctly classified. The weight vector can be changed after each presented sample, which is called *on-line training*, or after one pass through all samples, which is referred to as *batch training*. Both training methods have advantages and disadvantages, in particular with respect to stability issues.

### 2.2 Neural Networks

A more elaborate modelling technique which is very promising in classification are neural networks. They try to imitate functionalities of the human brain to solve highly complex problems by resembling part of its structure. This is implemented as a mathematical structure which produces some output by propagating the input over weighted connections through the structure.

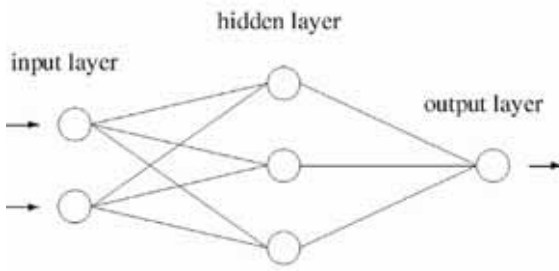


Figure 1: Feed-forward neural network.

The neurons are represented by single numbers, and the connections between them are modelled by weights. These weights are trainable, which enables the network to learn.

In the simplest type of neural networks, feedforward neural networks, the neurons are arranged in layers which are connected by unidirectional connections. There is one input layer, one or more hidden layers, and one output layer. Only a single output neuron is needed to indicate the class (*good* or *bad*). A simple example for such a structure is depicted in Figure 1.

In order to achieve some desired output, the weights are modified. This represents the learning process, which works as follows: The weights are initialised randomly. An input vector  $x$  is presented to the network. The product of  $x$  with the matrix of input-to-hidden weights  ${}_iW$  is calculated to form the input of the hidden neurons, which apply a nonlinear activation function, like hyperbolic tangent to it, resulting in  $y$ . The same procedure is performed with  $y$  to obtain the output of the output units,  $z$ :

$$\begin{aligned} {}_i net &= x \cdot {}_i W & y &= \tanh({}_i net) \\ {}_o net &= y \cdot {}_o W & z &= \tanh({}_o net) \end{aligned}$$

Any difference between actual output and training signal (desired output) corresponds to an error and is measured by the error function

$$E = \frac{1}{2} (z - \text{target})^2.$$

The updates of the weights are determined by differentiating the error function with respect to all network weights, and adjusting the weights by adding a multiple of the result to the old weights:

$$\frac{\partial E}{\partial {}_o W} = \frac{\partial E}{\partial z} \frac{\partial z}{\partial {}_o net} \frac{\partial {}_o net}{\partial {}_o W}$$

With  $\eta$  being the learning rate, the new weight matrix is obtained by

$${}_o W = {}_o W + \eta \frac{\partial E}{\partial {}_o W}.$$

Special care had to be taken of the learning rate: If  $\eta$  is set too low, the training process takes needlessly long; if it set too high, this could cause all pictures to be classified as *good*, or all as *bad*, which is why the learning rate was continually adapted.

This algorithm, which is the most widely used training method for feedforward networks, is called *backpropagation of errors*, as the error is propagated through the structure from the output layer back to the input layer. Thus, the error is assigned to the single weights depending on the influence they have on the result.

The architecture of the network requires an input vector of fixed size, which is why a maximum number of features per picture (corresponding to the number of neurons in the input layer) has to be set. As with linear classifiers, several flaws with several features each can be presented to the network simultaneously by putting them into one big feature vector.

What is different to linear classifiers is that the test data have to be split into a training and a test dataset. At first, the network has to be trained using the training data. After the weights in the network have been fixed, the classification performance can be evaluated. MATLAB provides a powerful toolbox for handling neural networks, which was very helpful for the work. Nevertheless, the simplest training algorithm, *backpropagation of errors*, was implemented in addition to that. The main reason for this is that it was intended to use a generalised version of it, *backpropagation through time*, with recurrent neural networks.

### 2.3 Recurrent Neural Networks

The new concept with this type of networks is that all possible connections between the neurons are admitted. This allows a bidirectional flow of information.

A simple *recurrent network*, or *Elman network*, is used, which contains feedback connections from the neurons in the hidden layer to themselves. In order to train such a network, it can be transformed into a feed-forward network by a technique called *unfolding in time*. The resulting structure is a network with two hidden layers. For every additional time step, a hidden layer is added to the unfolded network structure.

*Backpropagation through time* is the most popular training algorithm for recurrent neural networks, as it is very similar to standard backpropagation. The weight updates are determined by differentiating the error function with respect to the weights. The number of time steps that are taken into account determines how far back the error function has to be differentiated.

The recurrent structure provides the network with some form of memory. Information that was present to the network at earlier times is still available through the feedback connections.

Although this type of network has mainly been used so far for time series classification, it is very promising for the classification problems in this work. The major advantage is that an arbitrary number of flaws (feature vectors) can be presented to the network consecutively, which makes specifying a maximum size of the input vector unnecessary.

A problem that can occur with recurrent neural networks is that information which lies back a long time becomes irrelevant, as it is propagated over the weighted connections. One possible way to deal with this problem is to use long shortterm memory.

Another interesting question is how the order in which the feature vectors are presented to the network plays a role for classification. One approach would be to sort the input vectors somehow, but firstly, there is no natural ordering among vectors, and secondly, if a reasonable way is found how to sort the vectors (according to relevance, e.g.), all feature vectors would have to be known before classification begins, which is not always the case, especially in production processes. Future research will focus on how to construct a network structure that is order invariant by itself.

## 2.4 Comparison of the Applied Methods

Classification mistake rates are near zero for the linearly separable classification problems and typically range from 20 to 40 percent for the linearly inseparable ones when using linear classifiers.

With the use of neural networks, the mistake rates can be dropped to 30 down to 1 percent for the linearly inseparable problems. It is clear that their classification capabilities are almost always better as those of linear classifiers because the decision boundary need not be linear any more. Yet, it can be seen that even neural networks have problems with those criteria which require considering the relative positions of the flaws.

Especially for this case, the use of recurrent neural networks is very promising as they are able to consider several flaws in all. A considerable amount of theoretical research has to be done to find out which structure of network is best suitable and how many time steps are necessary to use all the information which is relevant for classification.

## Acknowledgement

This work was partially funded by the European Commission under grant #016429. The contents reflect only the author's view.

## References

- [1] C. Borgelt, F. Klawonn, R. Kruse, D. Nauck: *Neuro-Fuzzy-Systeme*. Friedr. Vieweg & Sohn Verlagsgesellschaft, Wiesbaden, 3rd edition, 2003.
- [2] R. O. Duda, P. E. Hart, D. G. Stork: *Pattern Classification*. John Wiley & Sons, Inc., New York, 2nd edition, 2001.
- [3] J. L. Elman: *Finding Structure in Time*. Cognitive Science, Vol. 14(2), pp. 179-211, 1990.
- [4] S. Hochreiter, J. Schmidhuber: *Long short-term memory*. Neural Computation, vol. 9(8), pp. 1735-1780, 1997.
- [5] S. D. Seichter: *Model-based Learning Classifiers for Surface Inspection Problems*. Diploma Thesis. Vienna University of Technology, 2006.
- [6] R. J. Williams, D. Zipser: *A learning algorithm for continually running fully recurrent neural networks*. Neural Computation, vol. 1, pp. 270-280, 1989.

### Corresponding author: Simon Seichter

Simon Seichter, Felix Breiteneker, Institute for Analysis and Scientific Computation, Vienna Univ. of Technology  
Wiedner Hauptstr. 8-10, A-1040 Vienna, Austria  
[simon.seichter@gmail.com](mailto:simon.seichter@gmail.com),  
[felix.breiteneker@tuwien.ac.at](mailto:felix.breiteneker@tuwien.ac.at)

Christian Eitzinger, Image Processing Group,  
Profactor Research Ltd.  
Im Stadtgut A2, A-4407 Steyr, Austria  
[christian.eitzinger@profactor.at](mailto:christian.eitzinger@profactor.at)

Received: June 10, 2006

Revised : September 8, 2006

Accepted: September 20, 2006





## BOOK REVIEWS - JOURNAL REVIEWS

### Book Reviews

#### Modeling Systems Theory of Nonlinear Control – An Introduction

Helmut Schwarz

Shaker, Aachen, 2000

549 + vii pages, ISBN 3-8265-7525-3

A solid translation of 'Einführung in die Systemtheorie nichtlinearer Systemtheorie nichtlinearer Systeme' which appeared one year earlier from the same publisher, although some of the diagrams are only labelled in German. It cover every kind of axiomatically derivable modes, the so called 'white box' systems.

Starting with simple transfer models, emphasising on the changes between time domain and and Laplace domain. It continues with the state space theory, introduces the notion of reachability, and gives a short introduction into stability theory.

The biggest part of the book deals with bilinear state-space systems, their generalisation as quadratic state-space system, (although in this case only those with linear inputs are covered in whole) with emphasis on the observability and the design of controllers those systems.

Only in some cases the examples used are to advanced for the techniques used to describe the model and solve the arising problems, leading to unnecessary technical difficulties with some of the equation. On the other hand, the author thoughtfully provided an index with not only his preferred mathematical notation, which his used in the book, but also covering all important abbreviations and a short introduction into the more advanced topics of linear algebra and vector calculus.

Beginner	Intermediate	Expert
		●
Theory	Mixed	Practice
	●	
Lecture Note	Monograph	Proceedings
	●	

Florian Judex, Vienna Univ. of Technology  
efelo@osiris.tuwien.ac.at

#### Foundations Of Generic Optimization Volume 1: A Combinatorial Approach to Epistasis

M. Iglesias, B. Naudts, A. Verschoren, C. Vidal

Springer, Heidelberg, 2005

296 + xiv pages, ISBN 1-4020-3666-3

The notion of epistasis originates in cell biology, where it refers to a linkage between genes in a senses of information encoding. In genetic algorithms (GAs), it refers to the interaction of genes when determining the fitness function as function of genetic code used in the GA. This notion was one of the hot topics in the early 1990s, and is now consolidated.

The authors focus on the so called 'normalised epistasis', a measure to determine how much a certain fitness function differs from a first order fitness function, where the fitness is a sum of functions which only depend on one of the genes. To construct these measure, it focuses on theoretical results, most of them derived with combinatorics and linear algebra.

But it is impossible to determine the audience the authors had in mind while writing the book. On one hand, the provide a 20 page chapter 0, titled 'GAs for absolute beginners', which suggest that it is aimed at mathematicians, who are not familiar with this topic. On the other hand, there is an appendix of nearly the same size dealing with the basic notions of linear algebra, suggesting practitioners as intended target.

Likewise, the book varies between simple examples, highly theoretical constructs and complex mathematical theory, satisfies none of the groups mentioned above.

Overall, a good idea that got lost halfway in the book, maybe due the multitude of authors.

Beginner	Intermediate	Expert
		●
Theory	Mixed	Practice
●		
Lecture Note	Monograph	Proceedings
	●	

Florian Judex, Vienna Univ. of Technology  
efelo@osiris.tuwien.ac.at

**Theory of Modeling and Simulation**

Second Edition

Bernhard P. Zeigler, Herbert Praehofer, Tag Gon Kim  
Academic Press, 2000

510 + xxi pages, ISBN 0-12-778455-1

This book gives a very good and well structured introduction into the theory of modelling and simulation. It covers all relevant topics of the area and in its second edition it takes into account the changes and more recent developments of modelling and simulation. Continuous and discrete modelling is been taken care of alike and no major topic in the field has been left out.

The first few chapters provide a good overview over the why simulation is used and how modelling is done in theory, creating a framework and defining the necessary terms that will later be used. Then the topic of modelling formalism and simulation algorithms is being taken care of, explaining the basic formalism (DEVS, DTSS and DESS) with their advantages and disadvantages as well as their boundaries.

In the later half the topic of system morphisms (abstraction, representation and approximation) is handled and finally, the problems and possibilities of system design are covered in the last part of this book.

Throughout the book a rather strict mathematical approach is taken to the subject which might scare off some readers, but as this book doesn't aim at people who just use simulation as a tool but at people who use it extensively and those who want to have a deeper understanding of it, this is more of an advantage than a disadvantage.

The key features are

- Provides a comprehensive framework for continuous and discrete event modelling and simulation
- Explores the mathematical foundations of simulation and modelling
- Discusses system morphisms for model abstraction and simplification
- Presents a new approach to discrete event simulation of continuous processes
- Includes parallel and distributed simulation of discrete event models
- Presentation of a concept to achieve simulator interoperability in the form of the DEVS-Bus
- Complete coverage necessary for compliance with High Level Architecture standards

As the whole theory is created and explained from scratch it is not necessary to already possess excessive knowledge in the field for reading this book.

Beginner	Intermediate	Expert
Theory	Mixed	Practice
Lecture Note	Monograph	Proceedings

Stefan Pawlik, Vienna Univ. of Technology  
*KingHenry@gmx.at*

**Advanced Dynamic – System Simulation, Model – Replication Techniques and Monte Carlo Simulation**

Granino A. Korn

John Wiley &amp; Sons, Inc., Hoboken, New Jersey, 2007

221 + xv pages, ISBN 978-0-470-08188-4

'Simulation is experimentation with models' is how this book starts. With many hands-on examples, it is presented, how simulation can be performed with various kinds of models. The simulation software used for presentation is DESIRE, and a significant part of the books treats the proper handling of the software. A CD with a free but powerful version of the software is included.

The spectrum of considered topics ranges from difference equations over partial differential equations to artificial neural networks. This shows the broad applicability of DESIRE. In all these areas, it can show its power: Solving very fast thousands of vectorised equations. DESIRE must not be mistaken for a 'thousand-built-in-functions, hundred toolboxes, multi-purpose' software as for example MATLAB, but treated as what it is: A very specialised tool for dynamic systems, which on its field is hardly to beat.

The author did not use the standard TeX formatting as used in many other text books, but formatted equations in a way looking very similar to the presented example source code snippets. This, in the beginning, takes getting used to, but is helpful when it comes to translating mathematical models into simulation models.

The first chapter of the book gives an introduction to computer simulation and treats some classical dynamical systems and their software representation, for example a prey predator model or a billiard ball simulation.

# Uncertain what lies ahead?

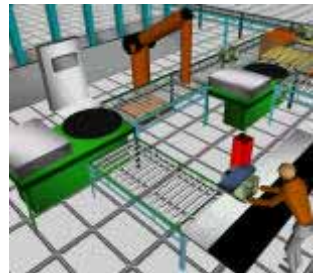
## Don't Speculate... Simulate!



You are operating in a complex and unpredictable business environment where balancing capacity defines your performance and profits. You deal with a lot of uncertainty and have to rework your planning continuously. Then Enterprise Dynamics simulation and control software is the ultimate powerful decision and planning tool to balance your resources. Enterprise Dynamics gives you an accurate image of your business processes and insight into utilization and yield of your resources, effectiveness, and logistical performance.



Enterprise Dynamics®, built by the people that brought Taylor to the market, is today's leading simulation software. Our software combines the powerful Enterprise Dynamics® Engine and many ready-to-use building blocks grouped into Enterprise Dynamic Suites for specific lines of business. Enterprise Dynamics® meets the latest standards in dynamic engineering and can be integrated into your existing system.



Download your free evaluation version at:

**Incontrol Enterprise Dynamics**  
• The Netherlands **+31 346.552500**  
• Germany, **+49 203.3051.1170**  
**info@EnterpriseDynamics.com**

**ENTERPRISE**  
**DYNAMICS**  
Simulation Software®



[www.EnterpriseDynamics.com](http://www.EnterpriseDynamics.com)



In chapter two, discrete models and hybrid models are discussed. In DESIRE a lot of means exist to implement for example digital controllers or continuous systems with discretely changing parameters.

Chapter three is quite technical but very important for the simulationist, as he learns here how to vectorise a model, how to write systems of coupled equations in a single line and how to create submodels.

The chapters four and five belong together as they both treat the multiple simulations of systems to find proper model parameters. This can simply be done as a parameter influence study, as an optimisation with respect to a given set of measurement data or as a Monte Carlo study.

In chapter six the author turns from classical dynamical systems described as differential or difference equations to artificial neural networks. 45 pages are dedicated to the vectorised formulation of neural networks and their application in regression and pattern recognition and classification. The author's intention was obviously not to give a fundamental introduction to neural networks, but to show the applicability of DESIRE in programming them compactly and efficiently.

In the final chapter, some more applications are presented. With DESIRE it is, for instance, possible to implement a fuzzy controller for a dynamical system, as well as a discretisation of a partial differential equation using the method of lines, which yields a system of ordinary differential equations.

Summarising, this book shows some practical problems and their solution using DESIRE, covering a wide variety of application fields of simulation, not forgetting to touch the theory behind. It can be very useful for researchers having already some knowledge on modelling and simulation and searching for a fast and transparent way to solve large problems.

Beginner	Intermediate	Expert
Theory	Mixed	Practice
Lecture Note	Monograph	Proceedings

Gerhard Höfner, Vienna Univ. of Technology  
gerhard.hoefner@tuwien.ac.at

### Stochastic Modeling and Optimization: With Applications in Queues, Finance, and Supply Chains

David D. Yao, Hanqin Zhang, Xun Yu Zhou (Eds.)  
Springer, New York, 2003; 468 pages, ISBN 0-387-95582-8

This book grew out of a workshop of the same name organised by the Academy of Mathematics and Systems Science of the Chinese Academy of Science and the Department of Systems Engineering and Engineering Management of the Chinese University of Hong Kong in May 2001. It contains 14 papers by various authors, most of whom participated in the workshop.

The editors caution that the papers in the book are not always based on the presentations at the workshop.

The volume is organised loosely into four parts. The first five chapters contain a collection of several basic methodologies. They are devoted to singularly perturbed Markov chains, related applications in stochastic optimal control, stochastic approximation, emphasising convergence properties, a performance-potential based approach to Markov decision programming, and interior-point techniques applied to stochastic programming. The three chapters in the second part are concerned with queuing theory.

Chapters 6 and 7 both study processing networks – a general class of queuing networks – focusing, respectively, on limit theorems in the form of strong approximation, and the issue of stability via connections to related fluid models. The third part features several studies in finance engineering like continuous time mean-variance portfolio selection. The last part illustrates three different applications that address the general issue of coordinating supply and demand.

This book covers the broad range of research in stochastic models and optimization. Applications covered include networks, financial engineering, production planning and supply chain management. Each contribution is aimed at graduate students working in operations research, probability, and statistics.

Beginner	Intermediate	Expert
Theory	Mixed	Practice
Lecture Note	Monograph	Proceedings

Thomas Löscher, Vienna Univ. of Technology  
tloescher@osiris.tuwien.ac.at





## BOOK AND JOURNAL ANNOUNCEMENTS

### Journal Announcements

#### SNE Special Issue *Parallel and Distributed Simulation Methods and Environments*

The new *SNE Special Issue Series* has been introduced as an extension of the regular SNE. The aim is to publish high quality scientific and technical papers concentrating on a specific topic. Using this approach the *Special Issues* will present the state of research in specific modelling and simulation oriented topics in Europe, and interesting papers from the world wide modelling and simulation community.

The first Special Issue of SNE was edited by members of the ASIM Working Group *Methods of Modelling and Simulation*. It is devoted to *Parallel and Distributed Simulation Methods and Environments* and includes seven selected papers, and a call for a benchmark in distributed and parallel simulation.

#### Content SNE Special Issue (SNE 16/2)

The development of parallel and distributed simulation methods and software tools has been strongly influenced by High Level Architecture (HLA) in recent years. HLA has its origins in the military simulation community. As a consequence of its openness and generic character it has also had a significant impact on non-military applications and is now an IEEE standard for distributed simulation. The *first paper* by Strassburger (Fraunhofer Institute Magdeburg, Germany) introduces the history of HLA, presents its main concepts and discusses recent developments. It provides enough background information for non-experienced readers in this field for the two further HLA related contributions in this journal.

The *second and third papers* discuss specific parallel and distributed simulation approaches for Discrete Event specified Systems (DEVS) and the associated simulator algorithms. Zacharewicz, Frydman and Giambiasi (University Marseille, France) investigate new lookahead computation methods in the G-DEVS/HLA environment. G-DEVS is a specific extension of the DEVS theory and its simulator algorithms for hybrid dynamic systems.

Continuous and discrete model components and their associated simulators can be located on different computers and integrated in a global simulation model using HLA technology.

The contribution by Wainer and Glinsky (Carleton University, Ottawa, Canada) investigates parallel simulation techniques for DEVS and Cell-DEVS models that combine cellular automata with DEVS theory. In their parallel simulation environment, CD++, the DEVS simulation algorithms are modified and combined with conservative and optimistic synchronization algorithms.

*Scientific and Technical Computing Environments* (SCEs) such as MATLAB, Scilab or Octave are essential tools in today's computational engineering and science. Especially optimization and simulation are well supported by integrated algorithms and subsystems like Simulink, Scicos or Stateflow. The *fourth paper* by Fink, Pawletta and Lampe (Wismar University of Technology, Germany) gives a detailed overview about SCE based parallel processing. In this paper, a new taxonomy on SCE based parallel processing is presented, followed by the identification and assignment of more than 30 existing projects. Furthermore, simulation and optimisation applications which have been parallelised under usage of SCEs are discussed. Parallel runtime results as well as general application characteristics are presented.

The *fifth, sixth and seventh papers* have been motivated by engineering applications. Stenzel, Pawletta, Ems and Bünning (University Wismar and MTG Marinetechnik GmbH, Hamburg; Germany) describe an application, where existing real-world software components, mainly written in FORTRAN, have to be integrated into an HLA compliant federation. FORTRAN/HLA integration approaches are examined in detail, whereas experiences in the field of MATLAB/HLA connectivity serve as design pattern.

The contribution by Eichler, Knöchel, Altmann, Hartung and Hartung (Fraunhofer Institute Dresden and Cadence Design Systems GmbH, Feldkirchen; Germany) describes the coupling of different simulators via TCP/IP network socket connection. The implementation and application of such a co-simulation is described in detail for the simulators MATLAB/Simulink and AMS Designer.



The contribution by Leitner, Wassertheurer, Breitenacker, Hessinger and Holzinger (ARC Seibersdorf research GmbH, Vienna; Vienna University of Technology; Medical University Graz; Austria) presents a Lattice-Boltzmann model (LBM) for solving fluid mechanical problems in engineering and biomedical applications. The investigated model is relevant for blood flow simulation because it uses Reynolds and Womersley numbers found in haemodynamics with a realistic time dependent pressure gradient as a boundary condition. A big advantage of LBM is the possibility of easy parallelization. Therefore different approaches of implementations are discussed with respect to parallelisation.

Finally, this SNE Special Issue publishes a call for a benchmark on parallel and distributed simulation tasks. This new *ARGESIM Benchmark on Parallel and Distributed Simulation* (ARGESIM Comparison CP2) extends the ARGESIM Comparison on *Parallel Simulation Techniques* from 1994 (ARGESIM Comparison CP1). The three tasks of this new benchmark are more general, so that also different algorithms for solving the tasks can be used, so that different strategies for parallelisation or distribution of the tasks can be set up and compared, and so that not only simulation software is addressed. The tasks are: Monte-Carlo - study for parameters in a damped mass-spring system, a Lattice-Boltzmann simulation for a cavity flow problem, and solutions of a partial differential equation with different approaches.

Guest Editors of this SNE Special Issue on *Parallel and Distributed Simulation Methods and Environments* (SNE 16/2) are Thorsten and Sven Pawletta from University Wismar. The issue was sent to all ASIM members - together with the regular SNE 16/1 (SNE 46), and sample copies were sent to other European Simulation Societies (with ordering offer). In 2007, a new system for individual SNE subscriptions will be set up.

Thorsten Pawletta & Sven Pawletta  
Research Group Computational Engineering  
and Automation, Wismar University of  
Technology, Business and Design  
PF 1210, 23952 Wismar, Germany  
[pawel@mb.hs-wismar.de](mailto:pawel@mb.hs-wismar.de), [s.pawletta@et.hs-wismar.de](mailto:s.pawletta@et.hs-wismar.de)  
[WWW.MB.HS-WISMAR.DE/cea](http://WWW.MB.HS-WISMAR.DE/cea)

Felix Breitenacker  
Vienna University of Technology  
Inst. f. Analysis and Scientific Computing  
Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria  
[Felix.Breitenacker@tuwien.ac.at](mailto:Felix.Breitenacker@tuwien.ac.at)

## SNE Special Issue 2007 *Verification and Validation* - Call for Contributions

Simulation is an important method which helps to take right decisions in system planning and operation. Building high-quality simulation models and using the right input data are pre-conditions for achieving significant and usable simulation results. For this purpose, a simulation model has to be well-defined, consistent, accurate, comprehensive and applicable.

The quality criteria can be proved by verification (*building a model in the right way*) and validation (*building the right model*).

The ASIM-Working Group *Simulation in Production and Logistics* which has worked on this topic since three years accommodates the increased significance of verification and validation and will publish the forthcoming *Special Issue SNE 17/2* of Simulation News Europe.

Papers on one or more of the following topics will be welcome:

- Procedure Models for Verification and Validation
- Methods for Verification and Validation
- Certification and Accreditation
- Information and Data Acquisition for Simulation Models and their Verification and Validation
- Verification and Validation - Documentation Aspects
- Credibility
- Automatic Verification and Validation
- Case Studies and Practical Experiences

The guest editors of this SNE Special Issue (SNE 17/2), Sigrid Wenzel (University Kassel), Markus Rabe (Fraunhofer Institut IPK, Berlin), und Sven Spieckermann (SimPlan AG, Maintal) invite for submitting a contribution. Contributions should not exceed eight pages (template) and should be mailed directly to the editor not later than August 30, 2007; contributions will be peer reviewed.

Templates for preparing a contribution are available at the ASIM website, [WWW.ASIM-GI.ORG](http://WWW.ASIM-GI.ORG), menu *International - SNE - Templates*, choose the template for Technical Notes.

Sigrid Wenzel  
Department of Mechanical Engineering  
University of Kassel, Kurt-Wolters-Strasse 3  
D-34125 Kassel, Germany  
[s.wenzel@uni-kassel.de](mailto:s.wenzel@uni-kassel.de)  
[WWW.UNI-KASSEL.DE/fb15/ipk/pfp/](http://WWW.UNI-KASSEL.DE/fb15/ipk/pfp/)



## ARGESIM BENCHMARKS ON MODELLING AND SIMULATION

### ARGESIM Benchmarks on Modelling and Simulation: Revised Definitions, Extended Solutions, and Supplemental Information

Felix Breitenecker, Vienna University of Technology; [Felix.Breitenecker@tuwien.ac.at](mailto:Felix.Breitenecker@tuwien.ac.at)

ARGESIM started in 1990 the series *Comparison of Simulation Software* in the journal *Simulation News Europe* (SNE). The ARGESIM Comparisons are based on relatively simple, easily comprehensible processes. Different modelling techniques and their implementation as well as features of modelling and experimentation within simulators, also with respect to application area, are compared.

#### Development of Comparisons

Since start of the comparison series, there have taken place new developments in software, algorithms, and modelling. Consequently also the comparisons developed further on, from comparisons of simulation software towards comparisons of modelling and simulation techniques and tools. This development can be seen in definitions and solutions published from 1990 to 2006 in 44 SNE issues: 20 definitions, and 298 comparison solutions (being 7 per SNE issue). The following list of comparisons and benchmarks shows also the broad variety of the applications.

- C1 Lithium-Cluster Dynamics, SNE 0 (1990)
- C2 Flexible Assembly System, SNE 2 (1991)
- C3 Generalised Class-E Amplifier, SNE 2 (1991)
- C4 Dining Philosophers I, SNE 3 (1991)
- C5 Two State Model, SNE 4 (1992)
- C6 Emergency Department SNE 6 (1992)
- C7 Constrained Pendulum, SNE 7 (1993)
- CP1 Parallel Simulation Techniques, SNE 10 (1994)
- C8 Canal-and-Lock System, SNE 16 (1996),
- C9 Fuzzy Control of a Two Tank System, SNE 17 (1996)
- C10 Dining Philosophers II, SNE 18 (1996)
- C11 SCARA Robot, SNE 22 (1998)
- C12 Collision of Spheres, SNE 27 (1999),
- C13 Crane Crab and Embedded Control, SNE 35/36 (2002)
- C14 Supply Chain, SNE 34 (7/2002)
- C15 Clearance Identification, SNE 35/36 (2002)
- C16 Restaurant Business Dynamics, SNE 40 (2004)
- C17 Spatial Dynamics of SIR Epidemics, SNE 41/42 (2004)
- C18 Neural Networks vs. Transfer Functions, SNE 43 (2005)
- C19 Pollution in Groundwater Flow, SNE 44/45 (2005)

#### Re- organisation towards Benchmarks

In 2006, a re-organisation of the comparisons has been started, based on aspects of content, and on aspects of documentation. First, the comparisons developed more towards benchmarks for modelling approaches, underlined by more content on modelling

in solutions submitted, and by more emphasis on modelling aspects in the definitions of the ‘newer’ comparisons / benchmarks, especially in case of C16, C17, C18, and C19 (which as first consequence are now called benchmarks).

It turned out, that some comparisons should be updated, because of changing challenges and misleading definitions. There has been a discussion, whether it makes sense to continue with the ‘older’ comparisons. On the one hand, it makes sense to continue with new solutions to ‘older’ comparisons which make use of new modelling approaches, or which at least use new versions of a specific simulator (a recent version of a simulator clearly offers much more features as a version from ten years ago). On the other hand, some of the tasks which are to be solved for a certain comparison, have become trivial, obsolete, or ‘strange’, because the task is too specific and linked to time-dependent development of simulator features.

Furthermore, for some newer benchmark definitions insufficient and misleading information has been given, or tasks have been formulated in a too special manner. Additionally, readers wanted to get additional information on the background for a certain benchmark, related to modelling procedure, real system existing, motivation for tasks, etc. Main reason for these request is the fact, that the benchmarks are widely used for education purposes, where additional information would be helpful. And last, but not least, the stronger emphasis on modelling made it almost too difficult to document a solution within one page of SNE.

As consequence, it was decided to re-organise the comparison structure and comparison handling in the following way:

- **From Comparisons to Benchmarks:** Definitions of new comparisons will put more emphasis on modelling and on different modelling approaches.
- **Revised Definitions:** SNE will publish revised definitions for ‘older’ comparisons in order to continue them as benchmark, and will update unclear definitions of the ‘newer’ benchmarks.
- **Supplemental Information:** SNE will provide supplemental information to background, modelling approach, etc. by further contributions (Technical Notes, Benchmark Notes).

- **Extended Solution Documentation:** SNE will allow two pages for each benchmarks solution.
- **Overview Benchmarks:** SNE will also introduce *Overview Benchmarks*, which concentrate on specific modelling and simulation paradigms - to be investigated by at least three different models.

#### Revised Benchmark Definitions

The SNE issue SNE 16/3 starts with publication of two revised definitions:

- *Extended Fuzzy Control of a Two-Tank - System - ARGESIM Benchmark C9R*  
extension of control model, update of tasks
- *Pollution in Groundwater Flow – ARGESIM Benchmark C19R with Spatially Distributed Modelling*  
extends and clarifies first definition in SNE 44/45

The SNE issue SNE 17/1 will publish the next two revised benchmark definitions:

- *Crane with Complex Embedded Control ARGESIM Benchmark C13R with Implicit Modelling, Digital Control and Sensor Action*  
improvement of models and control structure, parameters for (more) stable control, modernised sensor actions, and updated tasks
- *ARGESIM Benchmark on Parallel and Distributed Simulation*  
ARGESIM Benchmark CP2 - Overview Benchmark; almost totally new redesign of CP1 *Parallel Simulation Techniques*, addressing not only classic time domain simulation

For the next issues, the following revised benchmark definitions, new benchmarks definitions, and definitions of overview benchmarks are planned:

- C17R Spatial Dynamics of SIR-type Epidemic (clarification of first definition)
- C18R Neural Networks vs. Transfer Functions (extending and generalisation of model and tasks)
- C1 Lithium-Cluster Dynamics (general update)
- C7 Constrained Pendulum (general update or overview benchmark)
- C20 Flexible Manufacturing System with Scheduling Optimisation (new benchmark)
- CS1 Structural Dynamic Systems (overview benchmark)

#### Two-page Layout for Solution

SNE issue SNE 16/3 introduces the new two-page layout of for documentation of solutions, used as well for 'old' comparisons and 'new' benchmarks, providing the following structure with sections:

**Simulator.** Features and advantages of the simulator used should be sketched within 1/4 page SNE.

**Modelling.** Strong emphasis should be put on the modelling approach, etc., so that about 1 page SNE is reserved for this important subject.

**A-C -Tasks.** For experiment documentation and results of the three tasks (A-C) also about 1 page SNE is reserved.

**Résumé.** And finally a resume should summarise the important facts of the solution.

This new layout can be seen in the following solutions published in SNE 16/3:

- C9 Fuzzy Control of a Two Tank System with Dymola
- C9 Fuzzy Control of a Two Tank System with MATLAB/Simulink
- C9R Extended Fuzzy Control of a Two-Tank using MATLAB/Simulink
- C17 Spatial Dynamics of SIR Epidemics with MAPLE
- C19R Pollution in Groundwater Flow using COMSOL Multiphysics

For the overview benchmarks double space - four pages SNE - will be provided, scaling the sections appropriately.

#### Benchmark - related Information and Publications

As announced in previous issues, SNE starts a series on technical notes with detailed information on each benchmark and with additional information e.g. on alternative modelling techniques. These series is introduced in SNE 16/3 by the contribution *Cellular Automata Models for SIR-type Epidemics* dealing with various aspects of benchmark C17 'Temporal and Spatial Evolution of an SIR-type Epidemic'.

#### Solutions

We invite readers to send in solutions for all type of comparisons and benchmarks, whereby also solutions to 'old' (non-revised) comparisons are accepted. For a template, please visit

- ARGESIM website [WWW.ARGESIM.ORG](http://WWW.ARGESIM.ORG), menu *SNE*
- or the ASIM website [WWW.ASIM-GL.ORG](http://WWW.ASIM-GL.ORG), menu *International, SNE*.

**Author:** Felix Breitenecker  
Vienna University of Technology  
ARGESIM / Inst. f. Analysis and Scientific Computing  
Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria  
[Felix.Breitenecker@tuwien.ac.at](mailto:Felix.Breitenecker@tuwien.ac.at)





## Extended Fuzzy Control of a Two-Tank System - ARGESIM Benchmark C9R

Igor Škrjanc, University of Ljubljana, Slovenia; [igor.skrjanc@fe.uni-lj.si](mailto:igor.skrjanc@fe.uni-lj.si)

ARGESIM Benchmark C9R 'Extended Fuzzy Control of a Two-Tank System' is based on a nonlinear model for two coupled tanks, the first with inflow, the second with constant outflow. The inflow is to be controlled properly, so that the water level of the second tank holds a constant level. For control, the benchmark investigates fuzzy control within an overall digital control of the system. First, two fuzzy controllers with integral action are defined, to be analysed and tested with the tank system, both having sufficient compensation features. As alternative, a proportional fuzzy control is investigated, having a faster transient response, but no compensation. On modelling level, the benchmark investigates modelling techniques for the nonlinear system and methods and features for modelling discrete control and especially fuzzy control. On experiment level, control setup and transient responses are investigated. There, the first task requires calculation and display of the fuzzy control surfaces, independent on the model to be controlled. The second and third task deal with time domain analysis of the transient behaviour of the different fuzzy controllers under certain scenarios (changing setpoints for desired water level and disturbances). This benchmark extends and revises the ARGESIM Comparison C9 'Fuzzy Control of a Two-Tank System'.

### Introduction

The number of applications containing fuzzy components is still increasing. Modern simulation systems provide enhancements to implement fuzzy components in a convenient way. In SNE 17, July 1996, J. Goldynia has defined the first comparison, which should test the features of simulators with respect to fuzzy control. For this ARGESIM Comparison C9 'Fuzzy Control of a Two-Tank System', up to now 16 solutions have been sent in. Seven solutions make use of MATLAB/ Simulink, presenting different implementations. The other solutions showed implementations of fuzzy control in continuous and hybrid simulators, where the fuzzy module had to be programmed.

This revised extended definition improves the design of the fuzzy control and updates the tasks to be performed. The previous control design was not optimal. Main reason is a missing integral action in the control loop; consequently the revised definition designs the fuzzy control with integral action, to be compared with the a modified version of a proportional fuzzy control. The tasks to be performed are updated: instead of investigations with respect to weighted rules - being now a standard feature, a comparison of different types of fuzzy controllers is investigated. On experiment level, this new benchmark deals with control setup and transient responses are investigated. Task A requires calculation and display of the fuzzy control surfaces, independent on the two-tank - model. Task B and task C deal with time domain analysis of the transient response behaviour of the different fuzzy controllers under certain scenarios (changing setpoints for desired water level, and an isolated disturbance in the control).

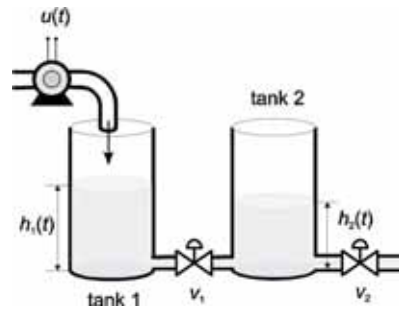


Figure 1: Schematic model of the two-tank system.

## 2 Dynamic Model for Two-Tank System

A two tank system (see Figure 1) in a specific configuration is characterised by the following nonlinear ODE set:

$$\begin{aligned}\dot{h}_1 &= c_1 u - f \\ \dot{h}_2 &= f - c_2 v_2 |h_2|^{0.5} \\ f &= c_3 v_1 \sqrt{|h_1 - h_2|} \text{sign}(h_1 - h_2)\end{aligned}$$

There  $h_1$  stands for the level in the first tank,  $h_2$  is the level in the second tank, the positions of the valves  $v_1$  and  $v_2$  are defined between 0 and 1. In our case the valve positions are  $v_1 = 0.4$ ,  $v_2 = 0.3$ . The process model includes characteristics of the liquid (laminar, turbulent, friction). The constants  $c_1$ ,  $c_2$  and  $c_3$  are:

$$\begin{aligned}c_1 &= 0.067 \text{ m/sV} \\ c_2 &= \begin{cases} 1.2 \cdot 0.0605 \text{ m}^{1/2} / \text{sV}, & h_2 < 0.16 \text{ m} \\ 1.0 \cdot 0.0605 \text{ m}^{1/2} / \text{sV}, & h_2 \geq 0.16 \text{ m} \end{cases} \\ c_3 &= 0.06624 \text{ m}^{1/2} / \text{sV}\end{aligned}$$

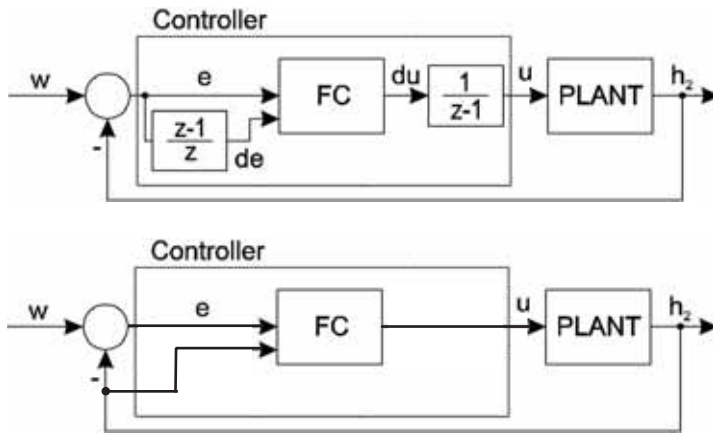


Figure 2: Closed-loop fuzzy control of liquid level  $h_2$ , IFC with integral action (upper picture), and PFC with pure proportional control (lower picture).

### 3 Fuzzy Control Design

For the two-tank system, two types of fuzzy control are designed. The task is to control the liquid level  $h_2$ . Both controls work with two inputs and one output. The first fuzzy control I-FC works with inputs  $e(k) = w - h_2(t_k) = w - h_2(k)$ ,  $w$  being the desired set-point for  $h_2$ , and  $de(k) = e(k) - e(k-1)$ , and with integral action at the output; the second fuzzy control P-FC uses inputs  $e(k) = w - h_2(k)$  and  $h_2(k)$  without integral action at output. Both types are shown in Figure 2.

#### 3.1 Fuzzy Control with Integral Action - I-FC

The IFC fuzzy controller should be designed to have the capability to eliminate the steady-state control error and a suitable transient response. This means that the controller should have the integral control action. The controller should in general represent the following relation:

$$u(k+1) = u(k) + du(k)$$

$$du(k) = f(e(k), de(k))$$

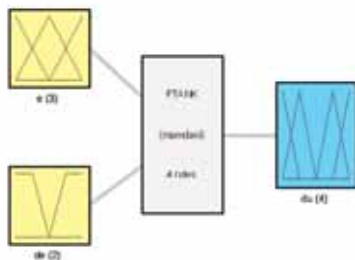


Figure 3: Structure of fuzzy controller.

The structure of the fuzzy controller should be the following: two input variables, divided in two and three membership functions and one output variable which is divided into four membership functions. The structure is shown in Figure 3. Two fuzzy controllers, I-FC1 and I-FC2 are designed. They should be implemented as discrete systems operating at one second sampling time.

The type of the fuzzy controllers should be *Mamdani*, what means that the output should be defined by membership function, the operator of the *intersection* between fuzzy sets is defined as the *product* of membership values and for the *union* operator between fuzzy sets the *sum* of membership values is used. The *defuzzification* method

which has to be used to calculate the crisp output is defined as *center of gravity* and for *implication* the *product* is used. Membership functions for  $e$  and  $de$  are defined in Figure 4a and 4b, membership functions of the output variable  $du$  in Figure 4c.

The membership functions for variables  $e$  and  $de$  are the same in the case of controller I-FC1 and I-FC2, the difference between both fuzzy controllers is in segmentation and form of the output fuzzy variable, which should be in I-FC1 in a form of *triangle membership functions* as shown in Figure 4c and in the case of I-FC2 in the form of *singleton membership functions* which are shown in Figure 4d (approximation of singleton functions). The controller should have integral action (I-FC) to eliminate the control error in steady state and to have the transient response as fast as possible (short settling time).

The *if-then* rules of both controllers should be the same and they should be given as four rules base in the following way, where the number in brackets gives the weighting of the rules:

1. **If** (e is low) **then** (du is open-fast) (1)
2. **If** (e is high) **then** (du is close-fast) (1)
3. **If** (e is good) **and** (de is falling) **then** (du is open-slow) (1)
4. **If** (e is good) **and** (de is rising) **then** (du is close-slow) (1)

#### 3.2 Proportional Fuzzy Controller - P-FC

Alternatively a fuzzy controller with a proportional fuzzy control P-FC is designed. This controller should be in general represented by the following relation:

$$u(k) = f(e(k), h_2(k))$$



This functional dependency between control error  $e(k)$ , the output variable  $h_2(k)$  and the control variable  $u(k)$  enables compensation of nonlinearity because the value of the control signal depends on the current operating point of the system. The control is suitable in the case of small, neglectable uncertainties and disturbances. The same operators for the basic fuzzy operations should be used as in the case of I-FC1 and I-FC2. The *if-then* rules of the P-FC controller are given as five rules base in the following way:

1. **If** (e is low) **then** (u is P5) (1)
2. **If** (e is high) **then** (u is P1) (1)
3. **If** (e is good) **and** (h<sub>2</sub> is high) **then** (u is P4) (1)
4. **If** (e is good) **and** (h<sub>2</sub> is low) **then** (u is P2) (1)
5. **If** (e is good) **and** (h<sub>2</sub> is middle) **then** (u is P3) (1)

The membership functions of the input variables for the P-FC controller are defined in Figure 5a for variable  $e$  and in Figure 5b for variable  $h_2$ . The output membership functions are defined in Figure 5c.

### 3 Tasks - Experiments

The ‘new’ benchmarks require for a solution a short description of the simulator (**S**), a detailed description of modelling (**M**), description and results of experiments (tasks **A**, **B**, **C**), and a short résumé (**R**) for classification.

**Modelling.** For documentation of modelling and implementation, please:

- describe the features for fuzzy control in your simulator or the interface to an interfaced fuzzy tool,
- model the controllers I-FC1 and I-FC2 (and P-FC) by means of features of the simulator or an appropriate additional tool linked to the simulator,
- and give a rough model description of the I-FC controllers and of the overall model, and indicate model changes for replacing the I-FC controllers by a proportional fuzzy controller P-FC.

Furthermore, describe, how the two-tank system is modelled (textually, graphically, library-based), and how system model, fuzzy control model, and additional discrete control blocks are composed to the overall model.

**A - Task: Fuzzy Controller Surfaces.** Compute and visualise the 3-dimensional characteristic surface of the fuzzy controllers I-FC1 and I-FC2. Define  $e$  in interval  $[-30, 30]$  on  $x$ -axis,  $de$  in the interval  $[-0.05, 0.05]$  on  $y$ -axis, and  $du$  on the  $z$ -axis.

If your system does not support singletons directly, you may use any kind of emulation. Indicate, whether your system evaluates directly the fuzzy control algorithm, or whether it generates a control surface, which is interpolated.

**B - Task: Transient Response.** Simulate the whole fuzzy control system using I-FC1 and I-FC2 alternatively for changing offset values and control disturbance. The reference signal and the disturbance should have the following profile:

- $w = 0.30$  m in the interval of first 20000 seconds,
- in the next 20000 seconds the reference equals  $w = 0.60$  m,
- and in the interval of the last 20.000 seconds it should be  $w = 0.40$  m;
- additionally, at time instant 50.000 seconds a process input disturbance occurs:  $u_z = 0.1$  V.

Plot  $h_2$ ,  $h_1$  and  $u$  versus time, for I-FC1 and I-FC2. The simulation time is equal 60.000 seconds. Compare IFC1 and IFC2 by indicating differences in quality of control and behaviour caused by disturbance.

**C - Task: Comparison with Proportional Fuzzy Control.** This task should compare the the integral-type fuzzy controllers I-FC1 and I-FC2 with the proportional fuzzy controller P-FC.

The following tasks should be performed:

- Comment the necessary changes of the model.
- Compute and visualise the 3-dimensional characteristic (surface) of the fuzzy controller P-FC ( $e$  in interval  $[-30, 30]$  on the  $x$ -axis,  $h_2$  in the interval  $[0, 100]$  on the  $y$ -axis, and  $u$  on the  $z$ -axis).
- Show time domain simulation results for the P-FC controller with same scenario as in case of I-FC controllers and discuss qualitative control behaviour and reaction on disturbance.

### 4 Solutions - Results

The expected results show certain advantages and disadvantages of controllers I-FC and P-FC. The transient response of P-FC can be very fast, but the control error can not be eliminated. On the other hand, I-FC can reject the control error, but its action is relatively slow. These fuzzy control designs can be combined to a controller of type PI-FC, having advantages of both types.

Solutions should fit into two pages SNE. Templates for text (.doc or .rtf) may be downloaded from [WW.ARGESIM.ORG](http://WW.ARGESIM.ORG), menu *SNE* or at [WWW.ASIM-GL.ORG](http://WWW.ASIM-GL.ORG), menu *International - SNE*.

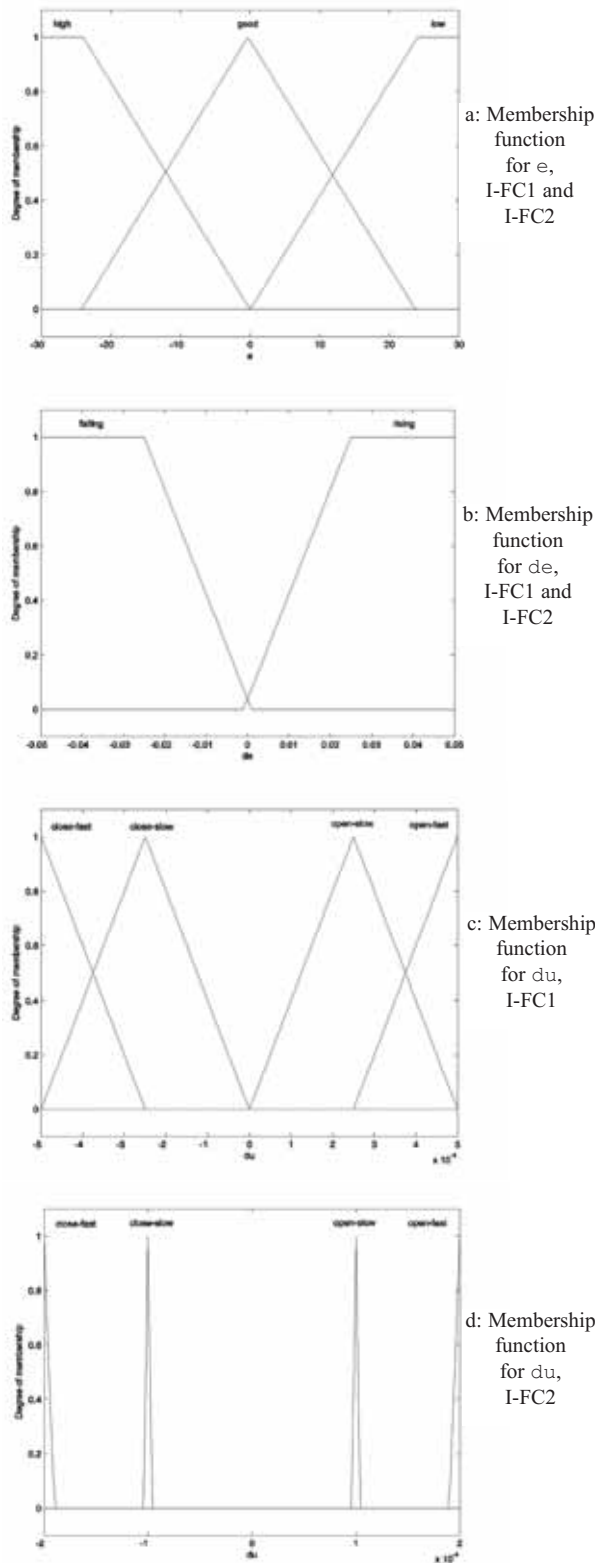


Figure 4: Membership functions for I-FC1 and I-FC2; a, b: membership functions for  $e$  and  $de$  (I-FC1, I-FC2); c, d: output membership functions for I-FC1 and I-FC2 ( $du$ ).

Figures must be also provided separately as file, any format, also formulas in case authors are not using the .doc or .rtf - template (e.g. .pdf). Furthermore, all model files (and batch files, etc., if any) must be sent in properly documented style - they will be provided at the web, so that readers can download and can make use of them.

#### Corresponding Author:

Igor Škrjanc, Laboratory of Modelling, Simulation and Control (LMSC), Faculty of Electrical Engineering Univ. of Ljubljana, Tržaška 25, 1000 Ljubljana, Slovenia  
igor.skrjanc@fe.uni-lj.si

Received: September 10, 2006

Revised: November 25, 2006

Accepted: November 30, 2006

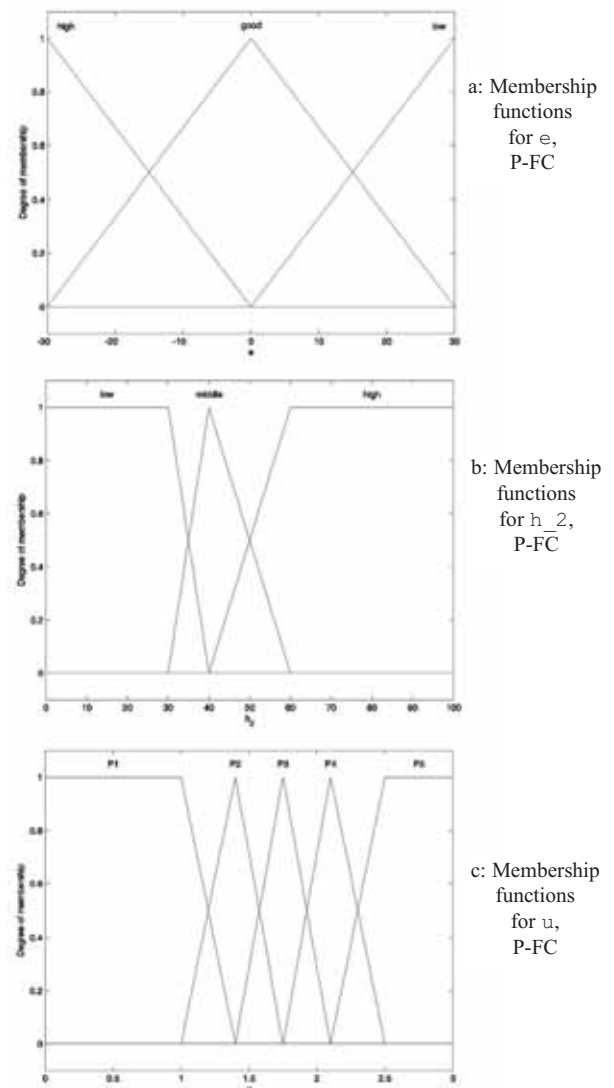


Figure 5: Membership functions for P-FC; a, b: membership functions for  $e$  and  $de$ ; c: output membership functions ( $du$ ).





## ‘Pollution in Groundwater Flow’ – ARGESIM Benchmark C19R with Spatially Distributed Modelling

Florian Judex, Felix Breitenecker, Gerhard Höfner, Vienna University of Technology  
{*Florian.Judex, Felix.Breitenecker, Gerhard.Höfner*}@tuwien.ac.at

The ARGESIM Benchmark C19R ‘Pollution in Groundwater Flow’ is based on a case study: in a homogeneous ground water body a singular pollution source contaminates the ground water stream; for decontamination, downstream two facilities are set up, which should reduce the contamination. Basis for modelling is the two-dimensional transport equation with degradation term for the pollution concentration, an analytical approximation for the solution in case of homogeneous flow, and an analytical approximation for the steady state. The benchmark first investigates the spread of pollution without counteraction, comparing numerical and analytical solutions. The more complex tasks of this benchmark deal with modelling and implementation of the facilities for decontamination and with calculating simulation results for continuous or schedule-controlled action of the facilities. The benchmark addresses quite different modelling approaches and solution techniques, from classical discretisation methods via FEM to alternatives techniques like cellular automata, Monte-Carlo methods and Random Walk.

### Introduction

Since many years the demand for pure water is increasing, as well for human consumption as well as an ingredient in industrial processes. In many regions, the surface water available does not suffice, so more and more ground water has to be used. Exploring existing ground water bodies uncovers unfortunately many polluted areas, sometimes with unknown pollutant sources. In this exploration, data can only be gathered via wells, which is expensive and sometimes not possible. Therefore, modelling and simulation of a polluted groundwater body can help in various cases: determination of the pollution plume, localisation of the pollution source, planning of facilities for decrease of pollution, etc.

This benchmark is based on the following case study: in a homogeneous ground water body, flowing in  $x$ -direction, a singular pollution source contaminates the ground water stream. As the source is not known or situated in an inaccessible area, the groundwater must be decontaminated somewhere else in flow direction.

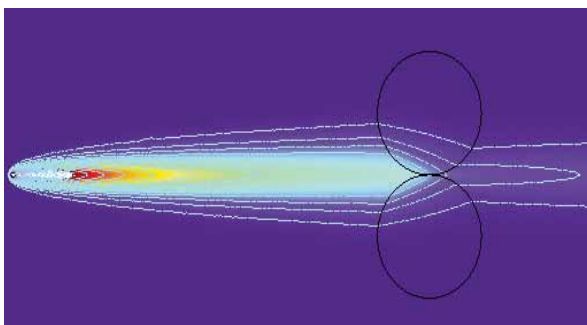


Figure 1: Polluted flow and effect of treatment facilities (FEM simulation).

A possible solution is to set up two treatment facilities – for example air spargers, which force oxidation – symmetrically to estimated maximal flow in  $x$ -direction. Figure 1 shows this situation, whereby the polluting source and the effect of the facilities can be seen.

Basis for modelling of the groundwater flow is the transport equation, describing the pollution concentration, a PDE with constant or state-dependent parameters and more or less complex boundary conditions. Of importance are furthermore analytical approximations for the pollution concentration in the homogeneous case, which may be compared with numerically calculated solutions. For modelling and simulating decontamination by degradation terms in the PDE, investigation start in an appropriate approximation for the steady state solution.

In principle, quite different modelling approaches and solution techniques can be applied, from classical discretisation methods via FEM to alternatives techniques like cellular automata, Monte-Carlo methods and Random Walk. In simple cases also approximating analytical solutions may exist. But in any case, or any chosen approach, there must be the possibility to embed analytical approximations, and to model or calculate a steady state solution. In reality, the choice of a modelling method or solution technique, may also depend on the data available, and on the aim of the simulation.

This comparison investigates different modelling methods and solution techniques with increasing degree of difficulty. First the spread of the pollution is without any counteraction is considered, whereby numerical solutions are to be compared with the analytical approximation.



In the following the pollution spread influenced by the counteraction with two treatment facilities investigated; for this inhomogeneous flow, boundary conditions for subregions must be noticed, and as initial condition a steady state must be found. And finally, the action of the treatment facilities should be controlled by a time schedule.

## 1 PDE Model for Pollution Concentration

Basis for modelling is the transport equation, describing the concentration  $c(t, x, y)$  of a pollutant in the saturated zone of a homogeneous two-dimensional ground water body with respect to both convection and dispersion. A simplified version of the transport equation is:

$$\begin{aligned} \frac{\partial c}{\partial t} + \frac{u_x}{R} \cdot \frac{\partial c}{\partial x} + \frac{u_y}{R} \cdot \frac{\partial c}{\partial y} = \\ = \frac{\alpha_L \cdot |\vec{u}|}{R} \cdot \frac{\partial^2 c}{\partial x^2} + \frac{\alpha_T \cdot |\vec{u}|}{R} \cdot \frac{\partial^2 c}{\partial y^2} - \lambda \cdot c \end{aligned}$$

which lacks the general terms for sources and sinks, as they will be kept simple in this example, but includes a degradation term which will be needed.

Table 1 shows parameter values being typical for the slow flows under investigation. Note that the porous velocity  $u_x$  is equal to about 1.7 meters by day, which is quite fast for groundwater but was chosen to ease modelling and simulation. It should also be mentioned that the porous velocity is only the average speed of water and the contained pollutant, averaging over every possible path in the porous strata forming the aquifer.

Description	Name	Value
pore velocity	$\vec{u} = \begin{pmatrix} u_x \\ u_y \end{pmatrix}$	$\vec{u} = \begin{pmatrix} 10^{-5} \\ 0 \end{pmatrix} \text{ m/s}$
dispersivity	$\alpha_T = \alpha_L$	0.05 m
retardation factor	$R$	1
degradation	$\lambda$	0 1/s
thickness of the saturated flow	$m$	10 m
effective porous volume	$n_e$	0.25
input rate of pollutant mass	$M$	2 mg/s

Table 1: Parameter values for pollution spread.

The effective porous volume  $n_e$  is the fraction of the water bearing stratum (aquifer) which is used by the groundwater flow, and is derived with experiments. In this comparison 1 m<sup>3</sup> of material with an effective porous volume  $n_e = 0.25$  can contain up to 250 litres of water. This maximum is actually reached in the saturated zone, which is the zone considered in this investigation. The  $h = 10$  meters of soil therefore represent 2.5 meters of water.

## 2 Analytical Approximation in Steady State

Assuming a steady source of pollutant  $M$  in (0, 0) on an infinite area allows to derive an approximating solution for the concentration  $c(t, x, y)$  for the parameters given in Table 1 by a product of exponential function and complimentary error function:

$$\begin{aligned} c(x, y, t) &= \frac{c_0}{4\sqrt{\pi\alpha}\sqrt{r}} e^{\frac{x-r}{2\alpha}} \operatorname{erfc}\left(\frac{r-|u|t}{\sqrt{2\alpha|u|t}}\right) \\ c_0 &= \frac{M}{m n_e |u|}, \quad r = \sqrt{x^2 + y^2}, \\ \operatorname{erfc}(x) &= 1 - \operatorname{erf}(x) \end{aligned}$$

This approximation is a slightly simplified form, taking into account the isotropy of the aquifer and the simple form of the ground water flow. It also incorporates the assumption that the concentration does not differ in  $z$ -direction, which is accomplished by the term  $m \cdot n_e$  in the denominator of the formula for  $c_0$ , just dividing the pollution by the 2.5 meters of water. Especially important is the retardation factor 1, which stands for no retardation – the pollutant does neither react nor compound with the soil, and is instantly transported.

By means of the analytical approximation, e. g. in case of homogeneous spread of the pollution, the 'pollution wave' can be calculated with reasonable accuracy (Figure 2,  $c(x, y, t)$  for  $t = 40$  days,  $t = 60$  days,  $t = 80$  days).

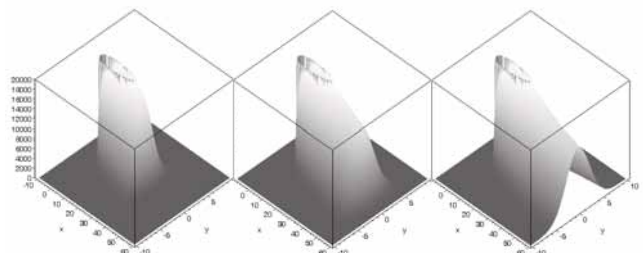


Figure 2: Evolution of pollution wave for  $t = 40$  days,  $t = 60$  days, and  $t = 80$  days.



### 3 Experiments - Tasks

The classical ARGESIM Comparisons require three tasks to be performed with the defined dynamic system, mostly addressing investigations and analysis in the time domain; furthermore information on the simulator used and a short description of the model implementation should be given - all to be presented within one page SNE. The new or revised ARGESIM Benchmarks extend the three tasks - Task **A**, Task **B**, Task **C** - and the simulator description - Task **Simulator** - by requesting a detailed description of the model implementation, whereby also different modelling approaches may be presented - Task **Modelling**, and by a short resume of the benchmark solution - Task **Resume**, trying also a classification of the approach. For presentation of all tasks two pages SNE may be used (task **Modelling** min.  $\frac{3}{4}$  page SNE.) Furthermore, model source files should be sent in. More details at [WWW.ARGESIM.ORG](http://WWW.ARGESIM.ORG), menu *SNE*.

**Modelling.** This benchmark can be tackled by very different approaches, from FEM via classical PDE discretisations to alternative methods like Cellular Automata and Random Walk. So we ask for a presentation of the approach used - in case of alternative approaches the 'mapping' of the PDE onto the chosen algorithm should be sketched. In case of graphical modelling tools, please provide snapshot from the modelling procedure. Furthermore, the model implementation needs significant model extensions especially for Task **B** and Task **C**, which should also be documented.

**A - Task: Simulation of Pollution Spread.** Under simplified conditions, the concentration of pollution spreads from the source into  $x$  - direction looks like a plume (Figure 3). There exist a lot of approaches and numerical techniques for solving the transport equation. Aim of this task is to compare a numerical solution based on any technique with the approximate analytical solution given before for the homogeneous case under investigation.

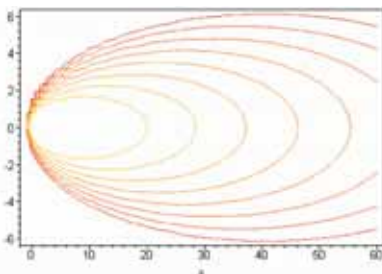


Figure 3: Pollution - isolines, plum-like spread.

Have in mind that the analytical solution is derived by using an infinite plane, and accommodate this in your model. When using the Finite Element Method or Finite Differences, you will almost certainly do this by using a flux boundary conditions of appropriate types.

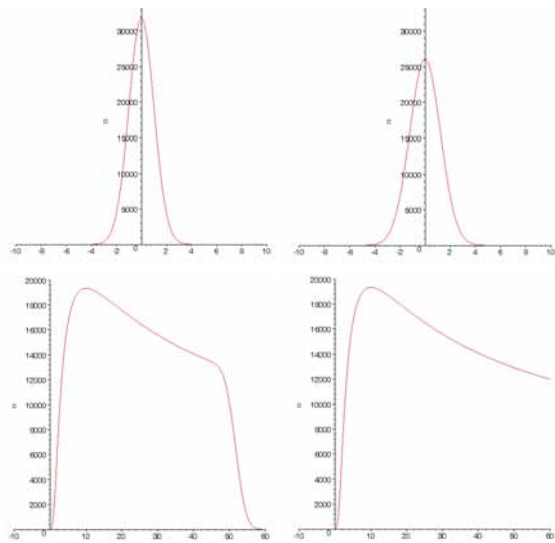


Figure 4: Pollution concentration,  $x$ - and  $y$ - sections.

During the direction of flow, the mass flow will be mostly convection driven at the right boundary, diffusion driven on the lower and upper boundary, and non-existing at the left boundary if chosen not too close to the source of the pollution, as upstream spread is only driven by dispersion.

Sections of any kind give information on the pollution spread. Figure 4 shows sections for  $x$  and  $y$ ; Figure 5 is the so-called break-through curve, showing the pollution wave passing a certain position.

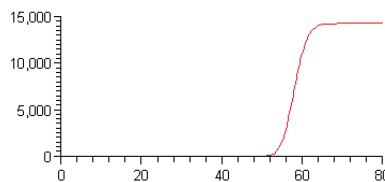


Figure 5: Pollution break-through curve for position ( $x=0$ ,  $y=40$ ).

For comparing numerical and analytical simulations, a rectangular area with  $-10 \leq x \leq 60$ ,  $-20 \leq y \leq 20$ , is chosen, with constant pollution source  $M=2.0$  mg/s in place  $(0,0)$  - other parameters see Table 1, with observation period of 150 days. Results should be compared with the analytical approximation at the line  $(50,y)$  at  $t=50$ ,  $t=100$ , and  $t=150$  days (absolute values and differences).

**B - Task: Pollution Reduction by Facilities.** Main goal is to reduce or to eliminate the pollution. As the pollution source cannot be influenced directly, facilities can be set at certain locations reducing the pollution locally (wells with chemical substances, pumps blowing in oxygen for precipitation, etc.). In the surrounding of such facilities locally elimination of the pollution takes place, reflected by an increase of the degradation parameter in the transport equation in a neighbourhood of the location.

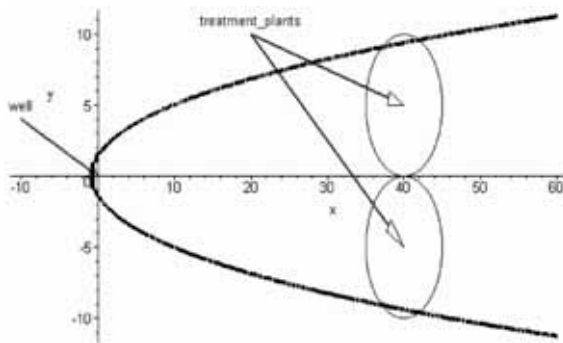


Figure 6: Boundary of pollution (thick line), influence areas of plants (thin circle lines).

The task is now, to investigate the influence of a facility with two plants. In order to get reasonable results, investigations should start from a steady state solution  $c(x, y, \infty)$ . With  $K_0$  being the modified Bessel function of second kind, and with  $c_0$  and  $r$  as before, the steady state solution is approximately given by

$$c(x, y, \infty) = \frac{c_0}{2\pi\alpha} e^{\frac{x}{2\alpha}} K_0\left(\frac{r}{2\alpha}\right), K_0 = \int_0^\infty \frac{\cos xt}{\sqrt{t^2 + 1}} dt$$

From engineering viewpoint, this formula is a very good approximation for the steady state solution. For arguments  $(r/2\alpha) > 1$  the approximation shows an error of about 10 %, which drops down to less than 1 % for arguments  $(r/2\alpha) > 10$ .

The facility consists of two plants situated at (40, 5) and (40, -5). The effect of the plants on the contamination is modelled by a degradation parameter  $\lambda$  being non zero in a surrounding of the plants. A reasonable choice is a value  $\lambda = -10^{-6} \cdot \ln 10$  in a circle neighbourhood of each plant with a radius of  $d/2 = 5$  m centred on the coordinates of each plant.

Figure 6 shows this scenario: place and action radius of the plants allow an degradation across the full width of contamination. The degradation lets drop down the pollutant concentration  $c$  to 10 % for a given control volume spending exactly  $10^6$  seconds in one of those areas.  $10^6$  seconds is that time span, the control volume would need to cross the circles right across the diameter  $d$  of 10 meters, the average remaining concentration will be much higher then that 10 %.

The task is now, to model this scenario appropriately, starting from the given steady state solution (approximation) and to investigate the degradation of pollution in time and space. We ask for documenting the implementation or numerical calculation of the steady state solution, and for display of simulation results.

Results for pollution and degradation should be documented as plot of the lines  $(30, y)$ ,  $(40, y)$ , and  $(50, y)$ ,  $20 \leq y \leq 20$ , for  $t = 100$  days.

**C-Task: Controlled Pollution Reduction.** To minimize costs for operating the plants and to allow for maintenance, the hours of operation must be limited. A reasonable strategy lets the plants operate only during night and at weekend, so that maintenance can be done at regular working hours, and so that the cheaper electric energy during the night hours can be used.

This strategy can be modelled by a periodical change of the degradation parameter  $\lambda$  from  $\lambda = -10^{-6} \cdot \ln 10$  (plants on) to  $\lambda = 0$  (plants off).

Task is now, to model this strategy appropriately (please give implementation details) and to simulate the system starting from the steady state solution with the following strategy:

- plants are active Monday to Friday from 0 to 8am and from 8pm to 12 pm,
- plants are active weekends around the clock, and
- plants are switched off else

As result, plots against time are now appropriate: plot the concentration at (50, 0), i.e.  $c(50, 0, t)$  for  $0 \leq t \leq 150$  (days) for switched operation given above, together with concentration for continuous operation (results from Task B).

## References

- [1] W. Kinzelbach: *Groundwater Modelling - An Introduction with Sample Programs in BASIC*. Elsevier Science Publ., 1986.

**Corresponding author:** Florian Judex

Florian Judex, Felix Breiteneker  
Institute for Analysis and Scientific Computing,  
Vienna University of Technology,  
Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria  
Gerhard Höfinger, Inst. for Mechanics of Materials and  
Structures, Vienna University of Technology,  
Karlsplatz 13, 1040, Vienna  
{Florian.Judex, Felix.Breiteneker,  
Gerhard.Hoefinger}@tuwien.ac.at

Received: November 20, 2005

Revised: December 5, 2005; September 10, 2006

Accepted: October 25, 2006





## Three Structural Different Modelling Approaches to ARGESIM Comparison C7 ‘Constrained Pendulum’ using the Modelica-Simulator MOSILAB

Günther Zauner, ‘Die Drahtwarenhandlung’ - Simulation Services, Vienna;  
Felix Breitenecker, Vienna Univ. of Technology; [Guenther.Zauner@drahtwarenhandlung.at](mailto:Guenther.Zauner@drahtwarenhandlung.at)

**Simulator:** *MOSILAB* (*MO*deling and *SI*mulation *LAB*oratory) is a simulator developed by Fraunhofer-Institutes FIRST, IIS/EAS, ISE, IBP, IWU and IPK within the research project GENSIM. It is a generic simulation tool for modeling and simulation of complex multidisciplinary technical systems. The simulation environment supports the procedures modeling, simulation and post processing. The model description in MOSILAB is done in the MODELICA standard. Additional features to assure high flexibility during modeling the concept of structural dynamics is implemented. This is done by extending the Modelica standard with state charts, controlling dynamic models. The model description language resulting is called MOSILA. Moreover, simulator coupling with standard tools (e.g. MATLAB / Simulink, FEMLAB) is realised.

**Modelling.** The motion of the constrained pendulum is usually defined with  $\varphi$  and  $\dot{\varphi}$  as states. But using the tangential velocity  $v = l\dot{\varphi}$  instead of angular velocity, has the benefit, that only the length of the pendulum has a discrete change in case of hitting or leaving the pin:

$$\dot{\varphi} = \frac{v}{l}, \quad \dot{v} = -g \sin \varphi - \frac{d}{m} v$$

**Standard Modelica approach.** In this approach only standard MODELICA code is used. It is defined in the MOSILAB equation layer as implicit law (it is non necessary to transform to an explicit state space):

```
equation /*pendulum*/
  v = ll*der(phi); vdot = der(v);
  mass*vdot/ll + mass*g*sin(phi) + damping*v = 0;
```

The state event, which appears every time when the rope of the pendulum hits or ‘leaves’ the pin, is modelled in an algorithm section with if (or when) - conditions:

```
algorithm
  if (phi<=phipin) then length:=ls; end if;
  if (phi>phipin) then length:=ll; end if;
```

This section defines length allocation of the constrained pendulum for all tasks. MOSILAB handles the if-clause (when-clause) by means of an state event finder.

**MOSILAB state chart approach.** This approach makes use of an additional feature of MOSILAB, modelling of discrete elements by state charts, which may be used instead of if- or when- clauses, with much higher flexibility and readability in case of complex conditions. Boolean variables define the status of the system and are managed by the statechart:

```
event Boolean lengthen(start=false),
      shorten(start = false);

equation
  lengthen=(phi>phipin); shorten=(phi<=phipin);
  .. here /*pendulum*/ -equations .....

statechart
  state LengthSwitch extends State;
  State Short,Long,Initial(isInitial=true);
  transition Initial -> Long end transition;
  transition Long -> Short event shorten action
    length := ls;
  end transition;
  transition Short -> Long event lengthen action
    length := ll;
  end transition; end LengthSwitch;
```

From the modelling point of view, this description is equivalent to the description with if-clauses. The MOSILAB translator clearly generates there an implementation with different internal equations. MOSILAB’s simulator performs simulation by handling the state event within the integration over the simulation horizon.

**Hybrid model decomposition approach.** MOSILAB’s state chart construct is not only a good alternative to if- or when - clauses within one model, it offers also the possibility to switch between structural different models. This very powerful feature allows any kind of hybrid composition of models with different state spaces and also of different type (from ODEs to PDEs, etc.). In case of the constrained pendulum, we decompose the system into two different models: Short pendulum model, and Long pendulum model, controlled by a state chart (Fig. 1).

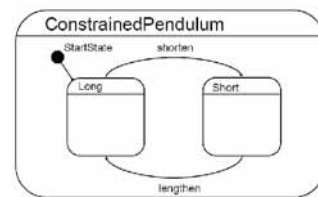


Figure 1: MOSILAB implementation with two different models controlled by a state chart

The model description defines now first the two pendulum models, and then the event as before:

```
model Long
equation
  mass*vdot/ll + mass*g*sin(phi) + damping*v = 0;
end Long;
model Short
equation
  mass*vdot/ls + mass*g*sin(phi) + damping*v = 0;
end Short;
event discrete Boolean lengthen(start=true),
      shorten(start = false);

equation
  lengthen = (phi>phipin); shorten=(phi<=phipin);
```



The following state chart creates first instances of both pendulum models during the initial state (*new*). The transitions organise the switching between the pendulums (*remove*, *add*). The *connect* statements are used for mapping local states to global state variables:

#### statechart

```
state ChangePendulum extends State;
State Short, Long, startState(isInitial=true);
transition startState -> Long action
  L:=new Long(); K:=new Short(); add(L);
end transition;
transition Long->Short event shorten action
  disconnect ...; remove(L); add(K); connect ...
end transition;
transition Short -> Long event lengthen action
  disconnect ...; remove(K); add(L); connect .....
end transition; end ChangePendulum;
```

**A - Task: Simulation of the System:** Simulations were performed with all three modelling approaches, giving same results. The approach with state charts allow an easier handling of the different initial conditions for this task, because no additional if-clauses are necessary. Furthermore, simple extensions of the state chart would allow also arbitrary initial conditions. For simulation, MOSILAB's IDA-DASSL solver was used, Figure 2 and Figure 3 show results.

**B - Task: Comparison of Linear and Nonlinear Model.** The linearised model is implemented in the same way as the the nonlinear model, just substituting  $\sin\varphi$  with  $\varphi$ . To calculate the difference of the results both state equations can be put into one model, or one can use state chart to couple nonlinear and linear model in parallel (results shown in Figure 3):

**model nonlinear**

**equation**

```
mass*vdot/l + mass*g*sin(phi) + damping*v = 0;
```

**model linear**

**equation**

```
mass*vdot/l + mass*g*phi + damping*v = 0;
```

**equation**

```
difference = phi - phiLin
```

**statechart**

```
state combine extends State
State run, init(isInitial=true)
transition init -> run action
  L:= new linear(phiLin=L.phi);
  NL:= new nonlinear(phi=NL.phi);
  add(L); add(NL);
end transition; end combine;
```

**C - Task: Boundary Value Problem:** One way to solve this problem is optimisation, using simulator coupling with MATLAB/Simulink. A simpler way is to transform the problem to an initial value problem by integrating equation backwards in time. The simulation is stopped event-controlled, when the desired angle is reached. The solution is the angular velocity at  $t = 0$ , which is approximately 2.185.

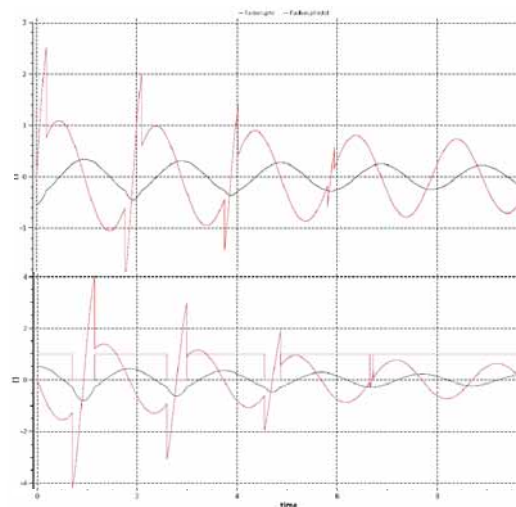


Figure 5: Angle  $\varphi$ , angle velocity and switching variable for different initial values of the nonlinear pendulum.

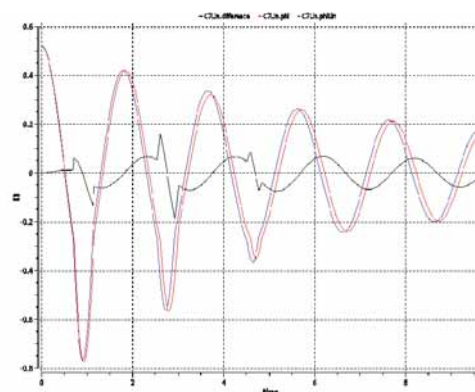


Figure 6: Angles  $\varphi$  for nonlinear and linear model with angle difference

**Résumé.** For system modelling and modelling of the state events, classical constructs from Modelica were used in a first approach; two other approaches model the state space changes by state charts (available in MOSILAB 2.0) controlling submodels (Tasks **M**). Scenarios with arbitrary different initial values are easily modelled by state charts for the initial phase; standard Modelica if-clauses may become here complex (Task **A**). Model comparison is done by simulating the models in parallel, for comparison modelling also state chart control is used (Task **B**). The boundary value problem (Task **C**) is simply solved by reversing time.

**Corresponding Author:** Günther Zauner

Günther Zauner, 'Die Drahtwarenhandlung' - Simulation Services, Neustiftgasse 57-59, 1070 Vienna, Austria; [Guenther.Zauner@drahtwarenhandlung.at](mailto:Guenther.Zauner@drahtwarenhandlung.at)  
Felix Breitenecker, Inst. f. Analysis and Scientific Computation, Vienna University of Technology, Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria

Received: October 20, 2006

Revised: November 18, 2006; December 10, 2006

Accepted: November 26, 2007



## Directly Programmed Fuzzy Control in ARGESIM Benchmark C9 'Fuzzy Control of a Two-Tank System' using Dymola

Anto Sodja, University of Ljubljana, Slovenia; [asodja@gmail.com](mailto:asodja@gmail.com)

**Simulator:** *Dymola 5.3d*, Dynamic Modeling Laboratory, is an object oriented simulation environment for acausal modeling (textual and graphical), simulation and visualisation of continuous, hybrid and discrete models. Dymola is able to understand Modelica, a unified textual and graphical modelling language offering many libraries for applications. Simulation can be carried out either in Dymola's own simulation environment or included in Simulink (only under Windows). For this solutions, Dymola version 5.3d for Linux was used ([WWW.DYNASIM.COM](http://WWW.DYNASIM.COM)).

**Modelling.** The model is composed of the Fuzzy Controller Model, the Plant model, and standard blocks from the Modelica Block Library (sum block, source block) - Figure 1. The plant model is described textually by the system governing ODEs using Dymola notation, put into a graphical block.

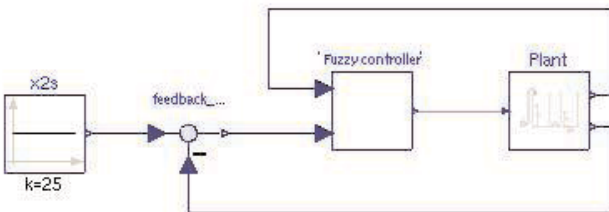


Figure 1: Top-level model: fuzzy controller and plant in closed loop.

There is no fuzzy control module or block as a part of Modelica Standard Library available inside the Dymola environment. The model was implemented on top of components from `Modelica.Blocks.Interfaces` library. The fuzzy controller is comprised of hierarchical connections of simple blocks. Input fuzzification blocks are made up of blocks for every fuzzy set. Those blocks are derived from Modelica's SISO block and linearly interpolate input between points in table that are given as a parameter to the every instance of a block:

```
parameter Real values[ 2 ];
parameter FuzzySetMembership membership[ 2 ];
algorithm
  if u <= values[ 1 ] then y := membership[ 1 ];
  elseif u <= values[ 2 ] then
    y := interpolate(values[ 1:2 ],
                    membership[ 1:2 ], u);
  elseif u <= values[ 3 ] then
    y := interpolate(values[ 2:3 ],
                    membership[ 2:3 ], u);
  else y := membership[ 3 ];
  end if;
```

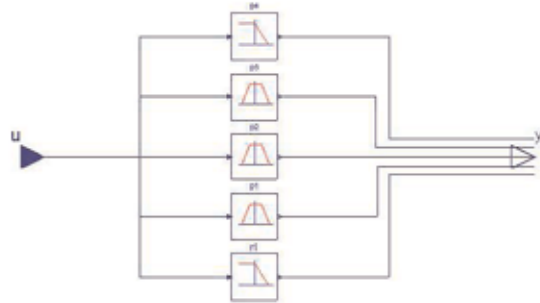


Figure 2: Input fuzzification structure - connected input fuzzy set blocks.

For different membership set's shapes different blocks have to be written. Structure of input fuzzification block can be seen on Figure 2. The same approach is used for the interface engine, it is looped through a 2D rules table and a *max-prod* operation for every rule is performed:

```
parameter Integer IFTable[ rows,cols ];
algorithm
  y := zeros(ny);
  for i in 1:rows loop
    for j in 1:cols loop
      y[ IFTable[ i,j ] ] :=
        max( y[ IFTable[ i,j ] ], (set_v[ i ] * set_h[ j ] ) )
    end for;
  end for;
```

For defuzzification every output set membership block provides two outputs: deviation from center and surface under degree of membership or degree of membership itself, resp. Those outputs are then simply routed to a block which calculates an output with selected defuzzification formula - center of gravity:

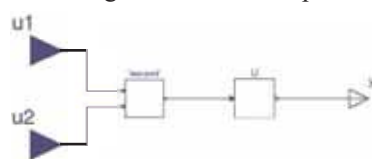


Figure 3: Fuzzy controller - two input fuzzification blocks, interface engine, output defuzzification block.

```
Interfaces.RealInput vars[ nin ];
Interfaces.RealInput weights[ nin ];
Interfaces.RealOutput y;
equation
  y = sum(vars*weights) / sum(weights);
```

**A - Task: Computation of Controller Surfaces.** The controller model is used as stand-alone Dymola model, with a discrete ramp signal on both inputs, with  $x_1$  starting at 0 and going up to 70 and  $e_{x_2}$  running from -70 to 70 for every  $x_1$  change. For surface plotting MATLAB was used due to lack of 2D-plotting abilities in Dymola.

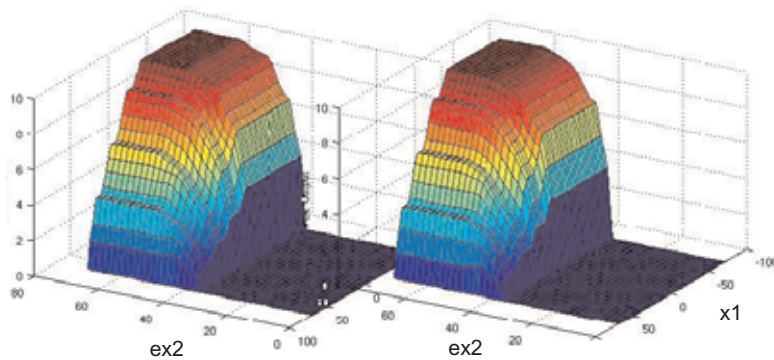


Figure 4: Control surface of FC1 (left) and FC2 (right).

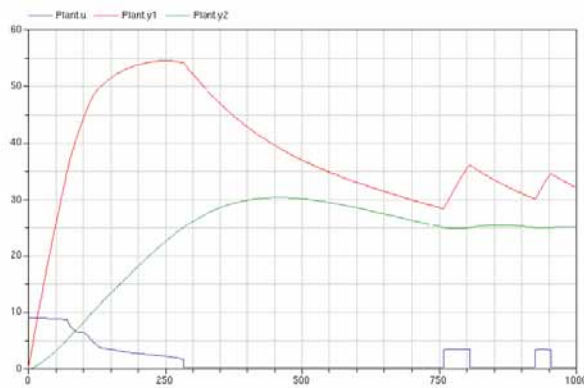


Figure 5: States and control for step reference - FC1 (triangular membership functions).

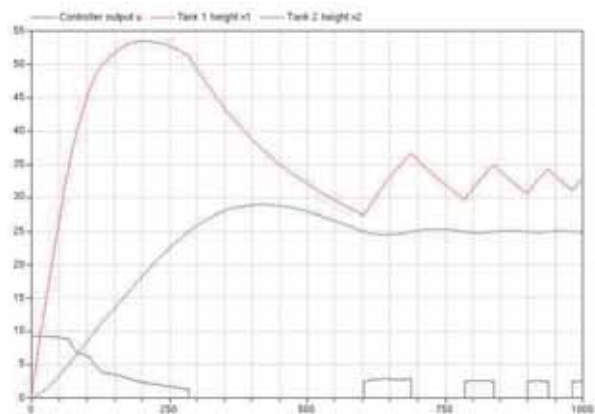


Figure 6: States and control for step reference - FC2 (singleton membership functions).

Data were extracted using `dymload.m` and `dymget.m` (MATLAB scripts provided together with Dymola). The result vectors need to be re-shaped into 2-D representation. Both surfaces (Figure 4, at left FC1 with triangular functions, at right FC2 with singletons) look very similar, differences can be only seen for high values of  $x_1$ , at  $x_2 \sim 45$ , and at top of the surfaces.

Duration of calculation was measured by standard Unix utility `time` command, since model is compiled. No significant difference between calculation of FC1 and FC2 exist, with  $ta_{FC1} = 0.872$  s,  $ta_{FC2} = 0.884$  s, and with ratio  $ta_{FC1} / ta_{FC2} = 0.9864$  s.

### B - Task: Simulation of the System.

Simulation of the whole system (Athlon XP-M 2500+) was performed in the Dymola environment and took 108 ms for a timespan of 1000 s in case of FC1, and 105 ms in case of FC2 (ratio is  $tb_{FC1} / tb_{FC2} = 1.0286$ ).

Also not much difference can be seen in performance of both controllers (results for transient behaviour in Figure 5 and Figure 6). As expected, fuzzy control with singletons switches more often.

**C - Task: Weighted Fuzzy Control.** Weighting is implemented by extending the `Interface-Engine` block with an additional table of weights. The calculated output set membership values are then multiplied by those weights.

```
parameter Real Weights[ rows,cols];
...
y[ IFTable[ i,j]] := max(y[ IFTable[ i,j]],
Weights[ i,j] * (set_v[ i] * set_h[ j]));
```

Calculation time for FC3 surface plot is  $tc_{FC3} = 0.932$  s, again very close to calculation times for FC1 and FC2.

**Résumé:** In this Dymola/Modelica solution, for plant model and for the discrete control structure standard Modelica blocks are used. Fuzzy control has to be programmed directly in textual Modelica code, making use of table function features and of very similar structure for all controller types, with and without weighting (singletons also programmed directly). The Dymola fuzzy control model can be used standalone, computing control values for arbitrary input values directly. For this solutions, Dymola version 5.3d for Linux was used.

**Corresponding Author:** Anton Sodja,  
Laboratory of Modelling, Simulation and Control (LMSC),  
Faculty of Electrical Engineering  
Univ. of Ljubljana, Tržaška 25, 1000 Ljubljana, Slovenia  
[asodja@gmail.com](mailto:asodja@gmail.com)

Received: October 20, 2006

Revised: November 18, 2006; December 10, 2006

Accepted: December 15, 2006





## A Discrete Model Approach to ARGESIM Benchmark C9 'Fuzzy Control of a Two-Tank System' with MATLAB, Simulink and Fuzzy Control Toolbox

L. Wallentin, R. Hausleitner, M. Paier, G. Zauner,  
Vienna University of Technology, [GZauner@osiris.tuwien.ac.at](mailto:GZauner@osiris.tuwien.ac.at)

**Simulator:** This comparison solution has been accomplished with MATLAB, Simulink, and with MATLAB's Fuzzy Toolbox. MATLAB is an environment for numerical computations and a high level programming language. Simulink is a tool, integrated into MATLAB, which allows to model a dynamic system by graphical blocks and to simulate its real world behaviour. The Fuzzy Toolbox supports design of fuzzy control to be used in MATLAB and Simulink

([WWW.MATHWORKS.COM](http://WWW.MATHWORKS.COM)).

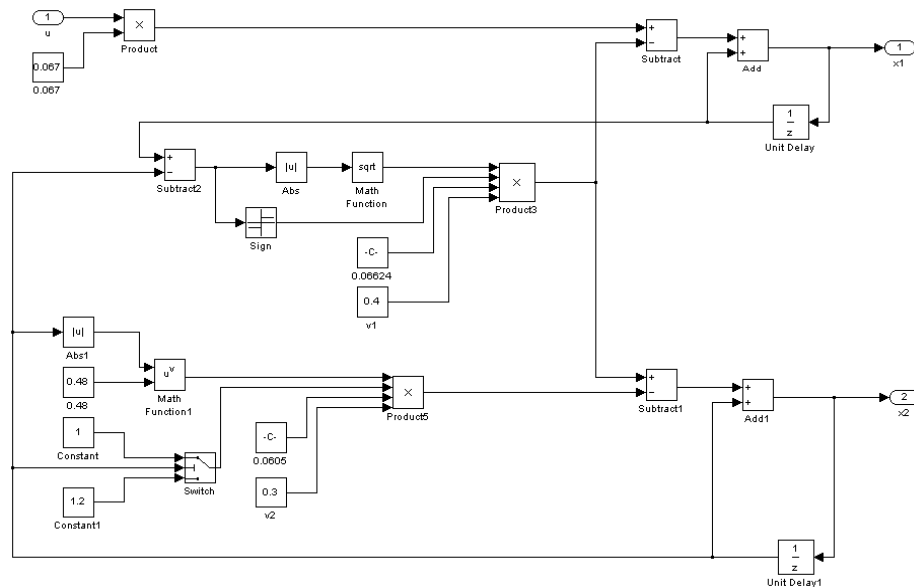


Figure 2: Submodel of the two tank system - explicit discrete model.

**Modelling.** The model is composed in Simulink in a modular manner (Figure 1): submodels for the fuzzy controller (input  $x_1$ ,  $ex_2$ ; output  $u$ ) and for the plant (input  $u$ ; output  $x_1$ ,  $x_2$ ), reference as constant input (output  $x_{2s}$ ), and sum block for deviation  $ex_2$  (input  $x_{2s}$ ,  $x_2$ ; output  $ex_2$ ).

The two-tank system is implemented with standard Simulink blocks (Figure 2) as explicit discrete system: instead of integrator blocks for  $x_1$  and  $x_2$ , discrete equations are implemented with unit delay blocks directly fed back. In principle, the ODEs are solved by Euler algorithm with stepsize 1 (equals control cycle time).

The *Fuzzy Logic Toolbox* allows to define fuzzy controllers in a convenient way. Fuzzy control of Mamdani type and Sugeno type are supported. FC1 uses the Mamdani fuzzy model and FC2 the Sugeno model, which provides support for singletons. Also very useful in this toolbox is the included rule base editor, where the rule base of this example is defined.

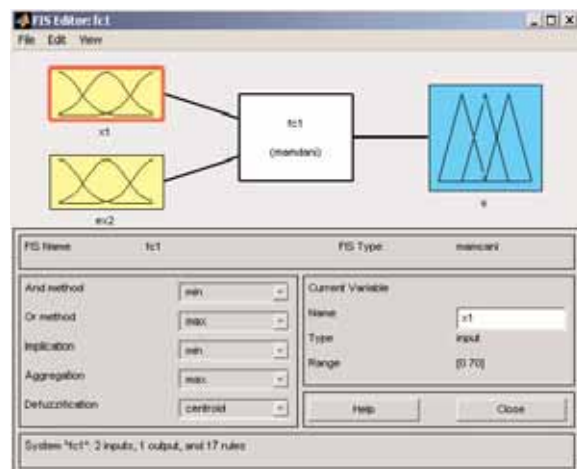


Figure 3: GUI for defining the fuzzy control (FC1).

The controller itself is defined in comfortable graphical user interface (Figure 3). For a better overview the fuzzy controller is put into a submodel with sampling block (Figure 4).

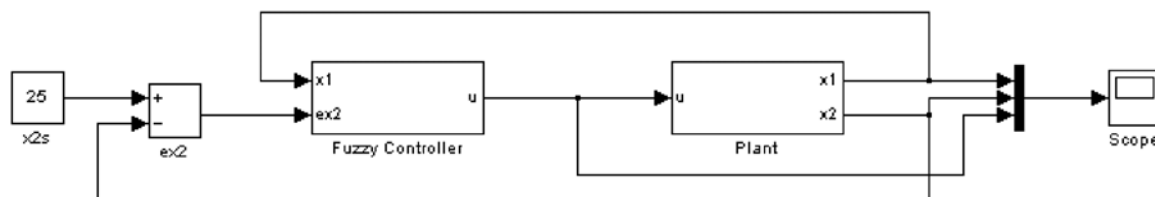


Figure 1: Modular model of the two tank system with fuzzy control.

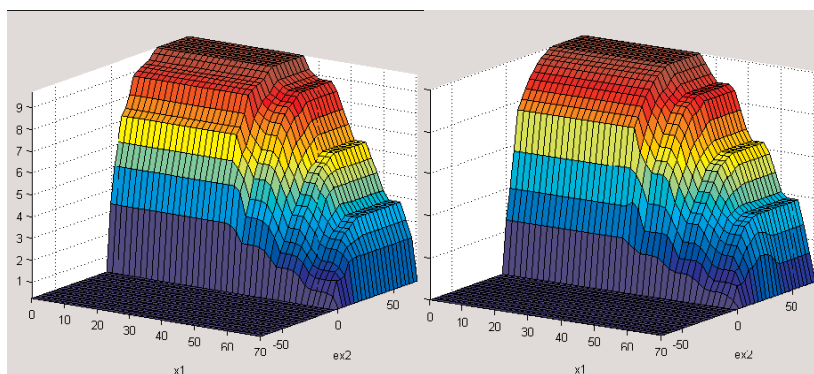


Figure 5: Control surface of FC1 (left) and FC2 (right).

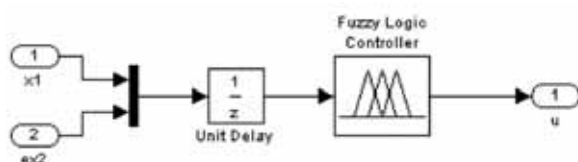


Figure 4: Submodel Fuzzy Controller with discretisation and fuzzy logic module from Fuzzy Control Toolbox.

### A - Task: Computation of Controller Surfaces.

The Fuzzy Control Toolbox generates directly data for the surfaces, with a discrete ramp signal on both inputs. For FC1 and FC2 resp., simply different control types (Mamdani or Sugeno) are chosen in the GUI. The surfaces are plotted by standard MATLAB plot features (Figure 5). Small differences only can be seen for high level of  $x_1$ . The calculation times for these surfaces on a Pentium 4 2.4 GHz machine are  $ta_{FC1} = 0.235$  s and  $ta_{FC2} = 0.078$  s; the ratio is  $ta_{FC1} / ta_{FC2} = 3$ .

### B - Task: Simulation of the System.

Simulation was performed in the Simulink environment menu-driven. For a simulation horizon of 1000 s the computation times are  $tb_{FC1} = 0.469$  s,  $tb_{FC2} = 0.219$ ,

and ratio  $tb_{FC1} / tb_{FC2} = 2.14$ . Simulation results (Figure 5 and Figure 6) again show differences only for high level of  $x_1$ .

### C - Task: Weighted fuzzy control.

Weighting factors in the rules are a standard feature in the rule editor. The rule base editor allows to specify the weight for each rule in the GUI. The calculation time for the FC3 surface is  $tc_{FC3} = 0.079$  s.

**Résumé:** In this MATLAB/Simulink (Rel. 2006a) solution, the Fuzzy Control Toolbox was used, so that modelling of different fuzzy controls was a standard task. Interestingly, singletons seem to be implemented different to standard membership functions, so that computation times are significantly faster. A specialty in this solution is the discrete model used for the plant: instead of ODEs, difference equations (with stepsize equal to control cycle time) have been used - equivalent to Euler integration of the ODEs with unit stepsize. The results do not differ from results with ODE solvers of higher order and smaller or controlled stepsize.

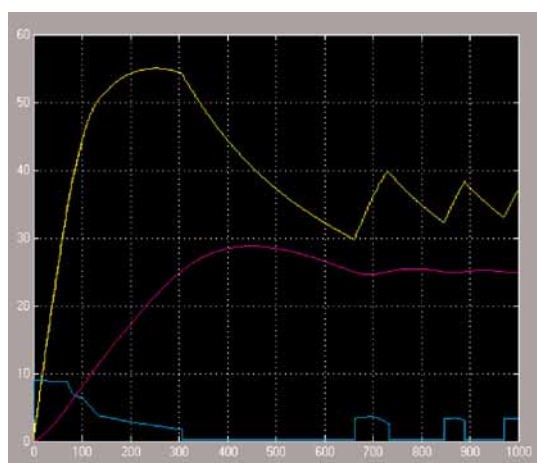


Figure 6: States and control for Mamdani control (FC1).

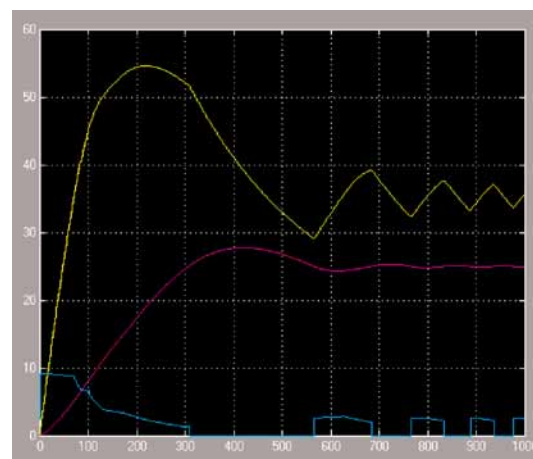


Figure 7: States and control for Sugeno control (FC2).

**Corresponding Author:** G. Zauner,  
L. Wallentin, R. Hausleitner, M. Paier,  
Institute for Analysis and Scientific Computing  
Vienna University of Technology  
Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria  
[GZauner@osiris.tuwien.ac.at](mailto:GZauner@osiris.tuwien.ac.at)

Received: June 30, 2006

Revised: October 20, 2006

Accepted: November 26, 2006



## A Java-supported Approach to ARGESIM Benchmark C9 'Fuzzy Control of a Two-Tank System' with Anylogic

Thomas Mair, Robert Leidenfrost, Andreas Platschek, Shabnam Tauböck  
Vienna University of Technology, *Shabnam.Tauboeck@tuwien.ac.at*

**Simulator:** Anylogic is an object oriented simulation software, supporting Agent Based, Discrete Events and Systems Dynamics based simulations. This comparison primarily uses the Systems Dynamics - based approach, because it enables a very smooth way to model the waterflow in the system. For fuzzy control design, Java was used in the Anylogic environment.

**Modelling.** In our approach, the System is split up into two basic blocks, the model of the two tank system (the plant) and the Fuzzy controller. The plant is implemented by System Dynamics approach, as shown in Figure 1, using stock variables. Every tank in the system is mapped as a stock variable in the model. Therefore we have a stock variable  $x_1$ , representing the water level of the first tank,  $x_2$  for the second tank, in represents the water pipe, and out the sink. The flow in the system is described by two differential equations; here they both have been split up in two parts, the part defining the water flowing into the tank and the part flowing out of the tank. These parts of the differential equation are distributed into the three flow variables  $u$ ,  $v_1$  and  $v_2$  in the plant model:

$$\begin{aligned} u &= 0.067 * u\_var \\ v_1 &= 0.06624 * 0.4 * \sqrt{|x_1 - x_2|} * \text{sign}(x_1 - x_2) \\ v_2 &= 0.0605 * 0.3 * r * |x_2|^{0.48} \end{aligned}$$

So the the resulting formulas for the water levels in the tanks are pretty simple:

$$x_1 = u - v_1, \quad x_2 = v_1 - v_2$$

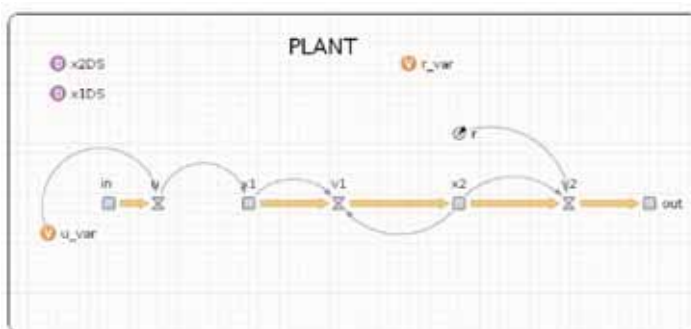


Figure 1: C9 - System Dynamics model in Anylogic.

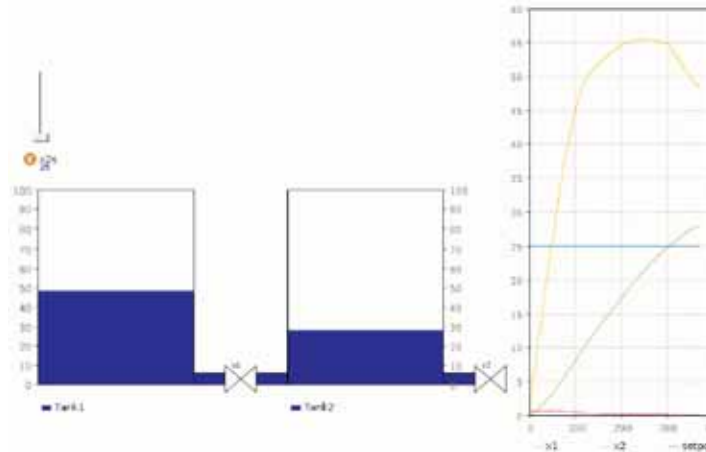


Figure 2: Direct visualization of tank dynamics in AnyLogic.

**Controller Model.** The second big part of the model is the fuzzy controller. We implemented the fuzzy rules into a Java function, which then is called periodically. The fact that Anylogic is based on Java enables the users to lookup mathematical functions (e.g. max, min, sqrt, ...) in the Java documentation. This makes live really easy. Anylogic also allows the user to import Java classes, making it possible to call any functions implemented in Java. In the following a code snippet for the control algorithm FC1 (the code has a total of 140 lines):

```
if (ex2 < 0)
{ ex2n1 = 1; }
else if (ex2 >= 0 && ex2 <= 10)
{ ex2n1 = (ex2 / (0 - 10) - 10 / (0 - 10)); }
else
{ ex2n1 = 0; }
if (ex2 >= 0 && ex2 <= 20)
{ ex2n1 = (ex2 / (0 - 20) - 20 / (0 - 20)); }
else
{ ex2n1 = 0; }
if (ex2 >= 0 && ex2 <= 10)
{ ex2p1 = (ex2 / (10)); }
else if (ex2 >= 10 && ex2 <= 30)
{ ex2p1 = (ex2 / (10 - 30) - 30 / (10 - 30)); }

* *** putting it all together *** */
p1 = min(ex2n1, x1p3);
p2 = max(min(ex2n1, x1p2), min(ex2p1, x1p3));
p3a = max(min(ex2n1, x1p1), min(ex2p1, x1p2));

p7 = max(p7a, p7b);
p8 = min(ex2p3, x1n1);
n1a = max(min(ex2n1, x1n1), min(ex2n1, x1p1));
n1b = max(min(ex2n1, x1p2), min(ex2n1, x1p3));

n1h = max(n1f, n1e);
u_var = ((1/3) * n1 * 0.5 + 1.25 * p1 + 2.5 * p2 + 3.75 * p3
+ 5 * p4 + 6.25 * p5 + 7.5 * p6 + 8.75 * p7 +
(10 - 0.5) * p8 * 1/3) / (n1 / 2 + p1 + p2 + p3 + p4 +
p5 + p6 + p7 + p8 / 2);
```

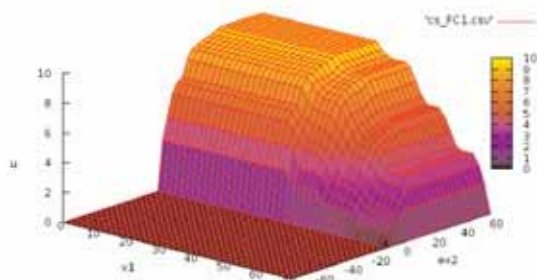


Figure 3: Control surface of FC1.

Anylogic provides a big set of different 2D diagram types, to visualize the results. For this comparison, we abused a stack chart diagram, to make a simplified visualization of the two tank system. A simulation, showing this visualization is shown in Figure 2.

#### A - Task: Computation of Controller Surfaces.

The control surface of the controller represents the output of the controller at all possible input combinations. To get these, two counting variables (because of two inputs -  $x_1$  and  $x_2$ ) are used to iterate over the ranges of the two inputs, as specified. These variables  $i$  and  $j$  range from 0 to 40 and from -20 to 20, resp. To get the same amount of points within the different ranges of  $x_1$  and  $x_2$ ,  $j$  is multiplied with an appropriate factor, so we get the ranges specified:  $x_1$  - [0 ... 70] and  $x_2$  - [-70 ... 70],  $41 \times 41$  points = 1681 data points. Unfortunately, Anylogic does not provide the 3D surface plots, so the results of the simulation were exported and the surface plot was done by gnuplot. The result for Task A2, the control surface plot of FC1, is shown in Figure 3.

Specifying the simulation time is non-trivial. First it takes time to build the model: 5 seconds - first build after starting Anylogic, and 0.4 seconds - rebuilding. The simulation time itself very much depends on the scale of the model time to the real time. We decided to use the 512x speed for our simulations. This led us to a Simulation time  $ta_{FC1} = 3.8$  s and a  $ta_{FC2} = 3.7$  s, ratio is  $ta_{FC1} / ta_{FC2} = 1.027$  - almost no difference.

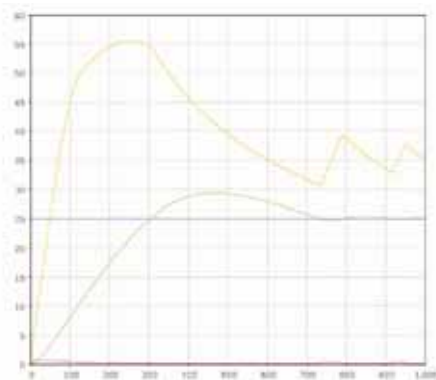


Figure 4: States and control, FC1.

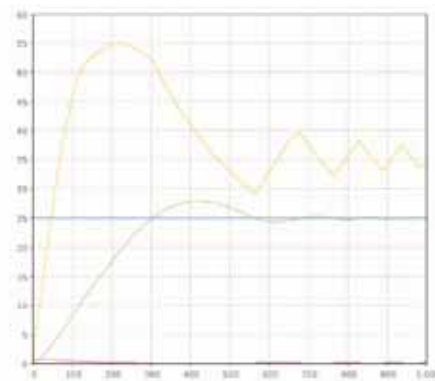


Figure 5: States and control, FC3.

**B - Task: Simulation of the System.** Simulation was performed in the 'standard' AnyLogic environment. Figure 4 shows the output of the system using fuzzy controller FC1. As can be seen the waterlevel  $x_2$  of the second tank first overshoots the setpoint of 25 and then swings into the setpoint after 700 s. As mentioned before, measuring simulation time works only on a relative basis in AnyLogic. With same speed factor as in Task A1, simulation time was about 1.3 s, with no difference for FC1 and FC2.

**C - Task: Weighted fuzzy control.** The last task included a new Fuzzy controller (FC3), using FAM interference. The rule base is the same as with FC2, but some of the functions are weighted with a factor - simple changes in the Java - programmed control algorithm. Calculation time for controller surface and simulation time does not really change, compared with Task A and Task B, resp. Results of simulation with FC3 (Figure 5) show that the system swings into the setpoint much faster.

**Résumé:** The SD modelling approach in Anylogic allows for easy modelling of the plant in C9, but AnyLogic does not offer any control model library, neither classic control nor fuzzy control. So all algorithms must be programmed in Java. On the other side, AnyLogic can be seen as Java programming and development environment, so that programming tasks are easy - for Java people.

**Corresponding Author:** Shabnam Tauböck,  
Thomas Mair, Robert Leidenfrost, Andreas Platschek  
Institute for Analysis and Scientific Computing  
Vienna University of Technology  
Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria  
[Shabnam.Tauboeck@tuwien.ac.at](mailto:Shabnam.Tauboeck@tuwien.ac.at)

Received: June 27, 2006

Revised: October 5, 2006

Accepted: October 12, 2006





## A Classic Solution to ARGESIM Benchmark C9R 'Extended Fuzzy Control' using MATLAB / Simulink and Fuzzy Control Toolbox

Igor Škrjanc, University of Ljubljana, Slovenia; [igor.skrjanc@fe.uni-lj.si](mailto:igor.skrjanc@fe.uni-lj.si)

Felix Breiteneker, Vienna University of Technology, Austria

**Simulator:** MATLAB is an environment for Numerical vector and matrix computations with a high level programming language. Simulink is MATLAB's simulator, which allows to model a dynamic system by graphical blocks and to simulate it in the time domain. MATLAB and Simulink come with additional tools, Toolboxes and Blocksets, for special purposes/applications. The *Fuzzy Toolbox* supports design, modelling and simulation of fuzzy control in MATLAB and in Simulink.

(WWW.MATHWORKS.COM).

**Modelling.** The model of the overall system is composed in Simulink in a modular manner (Figure 1): submodels for the fuzzy control (Fuzzy Logic Controller) and for the plant (model2tank), reference as table input (Setup h2), disturbance as step function input (Disturbance), and z-transformation blocks for discrete control action. The two-tank system is implemented with standard Simulink blocks (Figure 2); there for the blocks of type Abs, Sign, Switch zero crossing detection is enabled, so that discontinuous changes are synchronised with the ODE solver (ODE23).

The *Fuzzy Logic Toolbox* provides all features to design the fuzzy control. Starting with the basic *FIS editor* (Figure 3) to construct the structure of fuzzy inference system, on command line or within the GUI the fuzzy control can be parametrised and views: *mfedit* to edit membership functions, *ruleedit* is the rule editor and parser, *ruleview* to view the rules and fuzzy inference diagram, and *surfview* which is the output surface viewer.

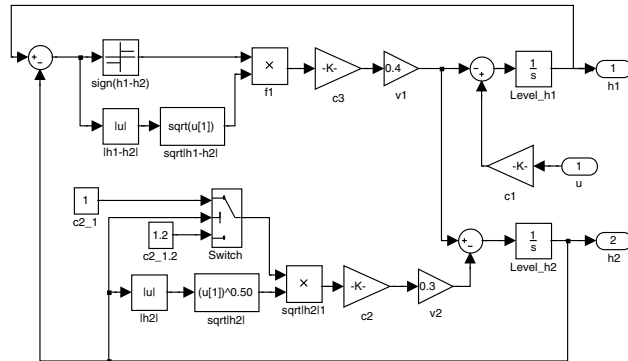


Figure 2: Simulink submodel for two-tank system.

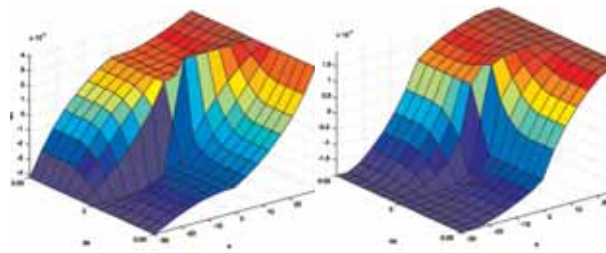


Figure 3: Control surfaces I-FC1 (left) and I-FC2 (right).

**A - Task: Controller Surfaces.** The fuzzy controller can be used directly in MATLAB. It is not necessary to generate a surface plot by a MATLAB program which calls the defined fuzzy controller, because the FIS editor offers *surfview* for this purpose. Figure 5 shows the control surfaces for I-FC1 and I-FC2: they seem very similar, but they have different scales ( $u_{I-FC1} \sim 2 \cdot u_{I-FC2}$ ); qualitative differences occur on the 'plateau' left above and at the slope at right (different flexion). The fuzzy controller is stored as structure, and all control parameters are stored in a .fis file.

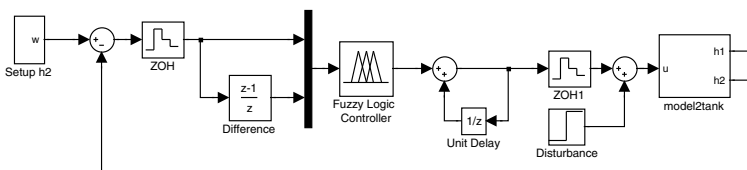


Figure 1: Simulink model of the overall system.

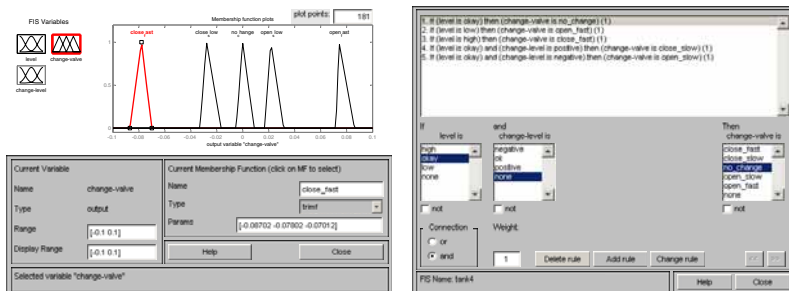


Figure 4: Editor for membership functions and for rule base.

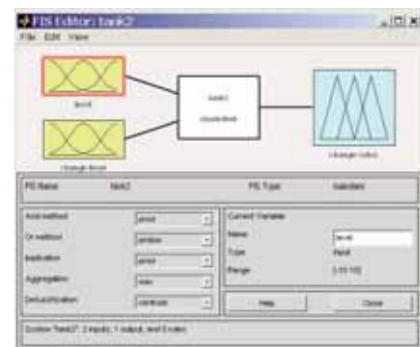


Figure 3: Basic editor for fuzzy control.

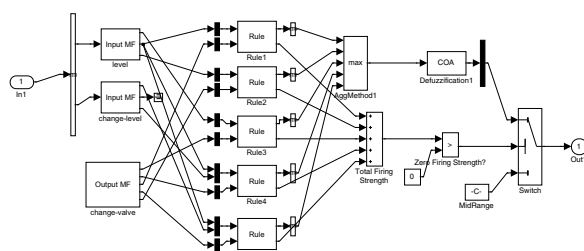


Figure 5: Simulink submodel fuzzy controller, automatically generated from .fis file.

**B-Task: Transient Response.** In Simulink, the block Fuzzy Controller reads the .fis file and sets up internally a Simulink submodel (Figure 5), again with submodel blocks consisting of basic Simulink blocks and function blocks, so that fuzzification of inputs, inference, and defuzzification for output is done completely at Simulink level (necessary for real-time).

Comparing the results for controllers I-FC1 and I-FC2 (Figure 6 and Figure 7), it can be seen that smooth division of output space in case of I-FC1 results in better control (no overshoot, faster settling time). Both controllers are of integral type and can reject the disturbance, which appear at the input of the process or at the output of the process.

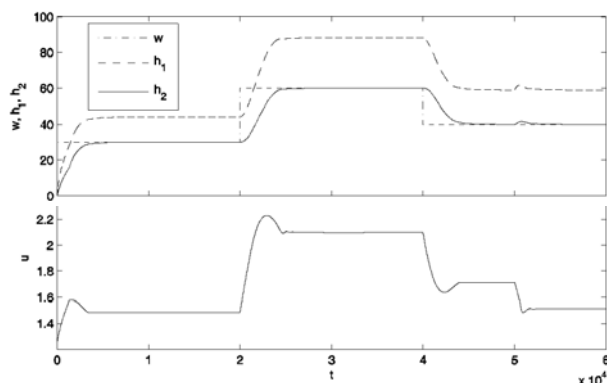


Figure 6: State variables  $h_1$ ,  $h_2$  and reference signal (upper) and control signal (lower) over time for I-FC1.

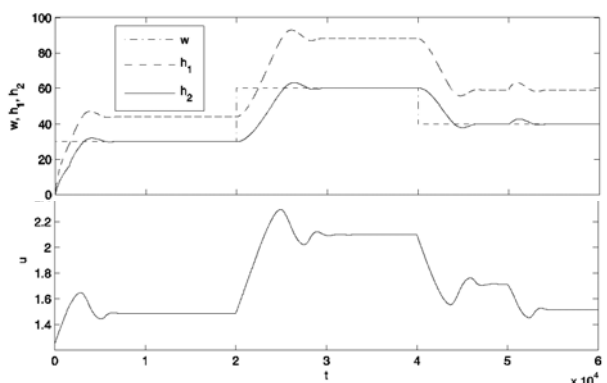


Figure 7: State variables  $h_1$ ,  $h_2$  and reference signal (upper) and control signal (lower) over time for I-FC2.

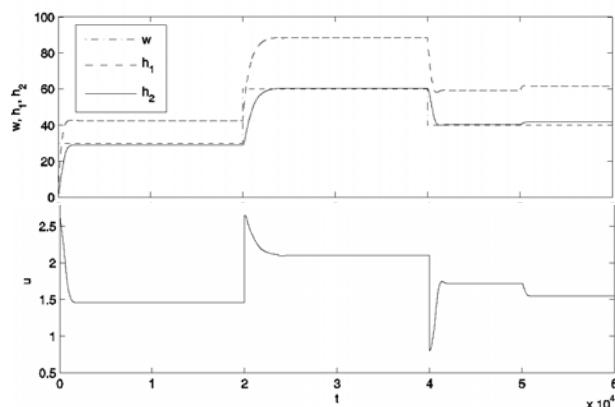


Figure 9: State variables  $h_1$ ,  $h_2$  and reference signal (upper) and control signal (lower) over time for P-FC.

**C-Task: Comparison with Proportional Fuzzy Controller.** The proportional fuzzy controller P-FC is designed as before with the MATLAB *FIS editor*, creating also the control surface for P-FC: Figure 8 shows a much smoother control surface for P-FC. The Simulink model for this control is simpler, because no integral action has to be modelled.

Time response for the P-FC controller is quite different (Figure 8). The control can neither reject the input disturbance nor it can compensate the disturbance at  $t=50.000$  s (control error remains). But P-FC control reacts much faster than I-FC control.

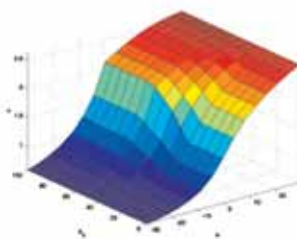


Figure 8: Control surface for P-FC

**Résumé:** This classic MATLAB / Simulink solution (Rel. 2006a) benefits from the comfortable features of the Fuzzy Control Toolbox. At MATLAB level, the toolbox allows to design all fuzzy controllers and to generate the control surfaces within a GUI. For modelling the plant, and for embedding the discrete control, standard simulink blocks are used, supplemented by the fuzzy control block, which automatically generates the Simulink model for the fuzzy control. Results are documented with MATLAB plot features.

**Corresponding Author:** I. Škrjanc, [igor.skrjanc@fe.uni-lj.si](mailto:igor.skrjanc@fe.uni-lj.si)  
Laboratory of Modelling, Simulation and Control (LMSC)  
Faculty of Electrical Engineering, Univ. of Ljubljana  
Tržaška 25, 1000 Ljubljana, Slovenia  
Felix Breiteneker, Inst. f. Analysis and Scientific Computing, Vienna University of Technology,  
Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria

Received: June 25, 2006

Revised: October 23, 2006; November 30, 2006

Accepted: December 15, 2006



## A Solution to ARGESIM Benchmark C17 'SIR-type Epidemic' using Numerical Programming Features in MAPLE

Štefan Emrich, Hannes Glavanovits, Tanaz Khorzad,  
Vienna University of Technology, Austria; *Stefan.Emrich@tuwien.ac.at*

**Simulator:** MAPLE has primarily been designed as a computer algebra system for analytical/symbolical computation. Recent versions support also numerical vector and matrix manipulations on a high level, so that also numerical tasks of any kind can be solved. Among several numerical algorithms, MAPLE offers solvers for ODEs and DAEs in a convenient way (additionally with symbolic solutions, e.g. by series). Usually, users work in a Java-based GUI, with command window, display window, etc., where commands are put in a straightforward way. Programming in terms of complex programs - as used in this solution - is possible although not very convenient.

**Modelling.** Three modelling approaches for SIR-type epidemics have to be implemented: ODEs, difference equations (DEs) and cellular automata (CA). ODE modelling and solving is a standard task of MAPLE: ODEs are defined, then the solver is called:

```
a:=0.2; r:=0.6/10^4;
ics:=K(0)=100, S(0)=16000, R(0)=0 :
ode:=diff(S(t),t)=-r*S(t)*I(t),
      diff(I(t),t)=r*S(t)*I(t)-a*I(t),
      diff(R(t),t)=a*I(t);
solution:=dsolve({ode,ics},numeric);
```

Difference equations must be implemented in nested loops. The Euler discretisation with unit stepsize for the ODEs makes use of arrays and loops:

```
for i from 1 to 50 do
  Sus[i] := Sus[i-1]*(1-r)^(Inf[i-1]/16000);
  Inf[i] := Inf[i-1]+Sus[i-1]*
    (1-(1-r)^(Inf[i-1]/16000))-a*Inf[i-1];
  Rec[i] := Rec[i-1]+a*Inf[i-1];
end do;
```

A much more elaborate task is the implementation of a cellular automata model. As MAPLE is not capable of handling subroutines in terms of external functions/programs all the routines need to be written in a single file in nested loops. In principle, the CA is implemented by arrays representing the grid of cells of the automaton, which are updated in unit steps. Two reasons suggest to structure the update of the cells: first, the one-file implementation in MAPLE needs to be structured for better coding, debugging, and reading; and second, codes for CA update become standardised using evolution operators like *update*, *propagation*, *transition*, etc. Consequently the MAPLE implementation follows this suggestion: manipulation of the CA is realised by evolution functions like *Set-Particles*, *ParticleMovementHPP*, *InfectionTotal*, etc.

These functions are then called repeatedly within a loop and so resemble the CA. The implementation has been kept very flexible to allow changes in the general structure (automata dimension, lattice structure, etc.)

The main loop for the cell state update moves the particles = individuals (*MovementParticle*), checks for collision with or without infection (*Collision*), checks for recovery (*Recovery*), and summarises the new amount of susceptible, infected and recovered particles (*RecollectQuantity*) for comparison with ODE and DE solutions :

```
CellularAutomaton:=proc(quant)
  local counter;
  counter:=0;
  while counter<quant do
    MovementParticleFHP();
    CollisionFHP();
    Recovery();
    RecollectQuantity();
    counter:=counter+1;
  end do;
```

Appendices characterise the special structure of the lattice gas cellular automata used (HHP: Hardy - de Pazzis - Pomeau automaton; FHP: Frisch - Hasslacher - Pomeau automaton). As example, the FHP movement on a square grid is implemented by:

```
MovementParticleFHP:=proc()
  global Cells, Cellcopy, ... : local i,j,k;
  for k from 1 by 1 to ParticlePerCell do
    for i from 1 by 1 to Cell_width do
      for j from 1 by 1 to Cell_length do
        Cellcopy[MoveFHP[k](i,j,k)] :=
          Cells[i,j,k];
      end do; end do; end do;
```

**A - Task: CA and ODE Simulation.** MAPLE works on basis of procedures: for ODE solution, first the ODE is defined and parametrised (see before); then a solver procedure is set up, to be used for calculating the solution *solution* at time instants *solution(i/100000000)* (results in Figure 1):

```
solution(0);
for i from 1 to 10 do
  solution(i/100000000): end;
odeplot(lsg,[[t,S(t)],[t,K(t)],
             [t,R(t)]],0..100);
```

For CA simulation, first the grids with the cell states are initialised with random distribution of the particles. Cellular automaton simulation is started by calling *CellularAutomaton:=proc(quant)* for the desired number of updates (*quant*), using a specified CA type.

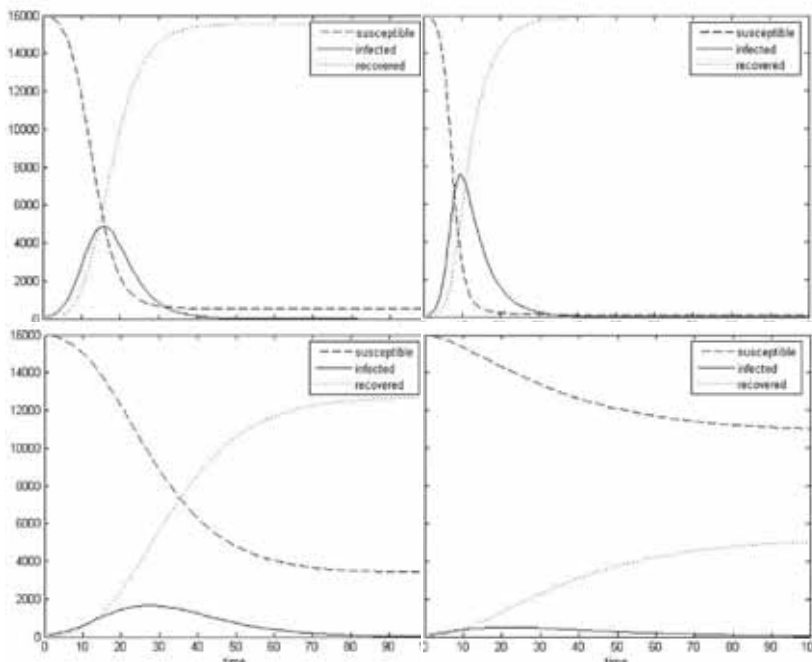


Figure 1: Results of different SIR models - a) ODE solution, b) FHP-CA solution, c) HPP-CA solution, d) random HPP-CA.

The types investigated are FHP-CA, HPP-CA, and HPP-CA with random deflection. The results (Figure 1) show qualitatively similar behaviour, but the CA dynamics is significantly slower.

**B-Task: Vaccination Strategies in CAs.** For the FHP-CA model, different vaccination strategies are modelled by different initial distribution of recovered individuals (particles) on the cell grid, because recovered people behave like vaccinated - they cannot get infected. An initial distribution of recovered individuals (4.000) can be implemented as follows:

```
CellNumber:=proc(status,number,xUpL,...)
while j<number do
row:=Generate(integer(distribution=
uniform[yUpL-1, ylowR],projection=ceil));
column:=Generate(integer(distribution=
uniform[xUpL-1, xLowR],projection=ceil));
end do;end proc;
```

Simulation results (Figure 3; infected individuals) for the different vaccination strategies show, that the solutions do not differ significantly. But interestingly, partial area vaccination results in less infected individuals than full area vaccination (all with 4.000 vaccinated).

**C-Task: ODE vs. CA Solutions.** ODE solution and DE solution are references for the summed up dynamics of susceptible, infected and recovered individuals of CA models, in case of spatial equal distribution of all individuals in each time step. For comparison, the ODE solution is calculated as before, and the DE solution is calculated by the simple loop shown in Task M.

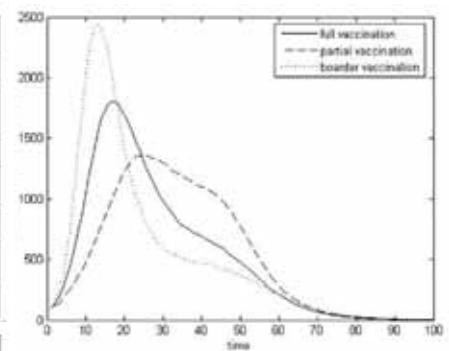


Figure 2: Infected for different vaccination strategies, FHP-CA.

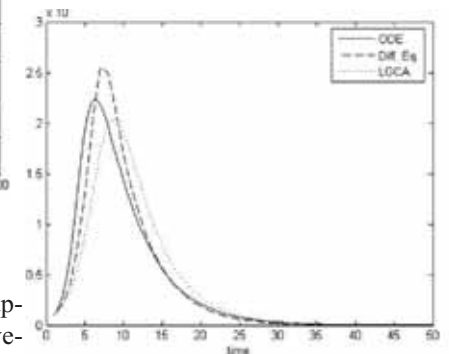


Figure 3: Infected for different models: ODEs, DEs, 'uniform' FHP-CA.

In FHP-CA update after movement, collision, and recovery, all individuals must

be equally distributed on the grid, as given in the code snippet below. Results in Figure 3 show indeed a good coincidence of all three solutions.

```
MovementParticleFHP(): CollisionFHP():...
R:=Eval(3); I:=Eval(2); S:=Eval(1);
NumberRandom(3,R,1,1,C_width,C_length):
NumberRandom(2,I,1,1,C_width,C_length):...
RandomSetParticles():
```

**Résumé:** While ODE solution is a standard task for MAPLE 10 in this benchmark solution, modelling of a CA and programming of a CA update algorithm is a nontrivial task. The chosen implementation works with arrays for the CAs, which are updated by evolution functions programmed as MAPLE procedure. The implementation follows suggestions for standardised evolution operators for CA modelling and requires deep knowledge of MAPLE programming.

**Corresponding Author:** Štefan Emrich,  
H. Glavanovits, T. Khorzad, [Stefan.Emrich@tuwien.ac.at](mailto:Stefan.Emrich@tuwien.ac.at)  
Vienna University of Technology, Austria;  
Inst. f. Analysis and Scientific Computing,  
Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria

Received: September 20, 2006

Revised: October 25, 2006; November 16, 2006

Accepted: December 10, 2006





## A FEM - based Approach to ARGESIM Benchmark C19R 'Pollution in Groundwater Flow' using COMSOL Multiphysics

Harald Teufelsbauer, University of Natural Resources and Applied Life Sciences, Vienna  
*harald.teufelsbauer@boku.ac.at*

**Simulator:** *COMSOL Multiphysics* is a modelling package for the simulation of a wide range of physical processes one can describe with partial differential equations (PDEs). COMSOL offers the opportunity to interact with *MATLAB*. Functions which are defined in *MATLAB* can directly be called in COMSOL if *MATLAB* is working in the background. The modelling of most phenomena can be done easily through predefined templates. Modifying these to specific applications is possible through equation-based modelling capabilities.

**Modelling.** The groundwater flow can be modelled with a transport equation, describing convection and diffusion processes. COMSOL Multiphysics offers a predefined template which is suitable for modelling this task. The equation offered is

$$\delta_u \frac{\partial c}{\partial t} + \nabla \cdot (-D \nabla c) = R - u \cdot \nabla c$$

COMSOL allows solving the transport equation for one- up to three-dimensional geometries. To solve the task of groundwater flow rectangle geometry is used. COMSOL offers a very user-friendly GUI, where also more complex two-dimensional geometries can be drawn by combination of circles, ellipses, rectangles, lines and 2<sup>nd</sup> and 3<sup>rd</sup> degree Bezier curves.

Figure 1 shows the meshed rectangle geometry used for the simulations. The steady source of pollutant *M* is placed at the point (0, 0). As COMSOL does not support the definition of concentrations or fluxes on singular points, the pollution source is realised by a little hole with an arbitrary chosen radius, e.g.  $r_M = 0.2$  m. A triangular mesh can be generated by one simple mouse click. If necessary, a user controlled mesh generation can be defined in a mesh setting menu. Local mesh refinements can be done very simple, by marking the desired area with the mouse and clicking a refinement button. The domain consists of one big rectangle rigged by the pollution source (subdomain). Two cycles define two further subdomains, describing the facilities used for task B and task C. The properties of these subdomains can be treated separately. The subdomain setting mask (Figure 5) allows defining the PDE's coefficients and the initial values of each subdomain.

The boundaries can be treated by dint of the boundary settings mask, where different kinds of boundary conditions can be selected.

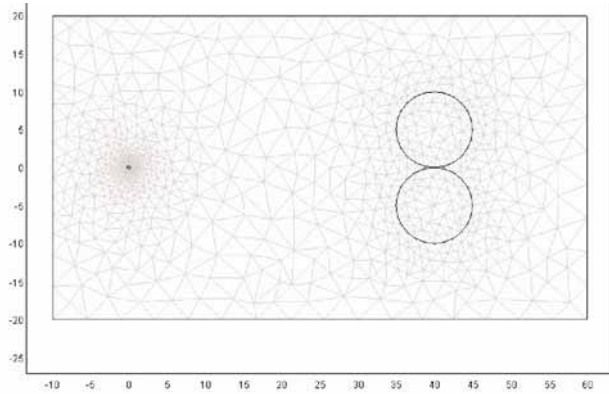


Figure 1: Model geometry and finite element mesh with refinements at pollutant source *M* and around facilities for pollution depletion.

On each selected margin the boundary conditions fulfilling the transport equation can define a forced concentration or a flux through the boundary. The right boundary of the rectangle is allocated to a flux condition, whereas the inward flux is zero.

The other boundaries of the rectangle use convective flux conditions, which assume that all mass passing through these boundaries is convection – dominated. This is useful in convection-dominated mass balances where the outlet concentration is unknown. The source of pollutant *M* is defined by dint of a flux boundary condition, provided by COMSOL where the right hand side describes the user defined incoming flux, respectively the pollution:

$$-\vec{n} \cdot (-D \nabla c + c \vec{u}) = \frac{M}{h \cdot n_e \cdot 2r_M \pi}$$

**A - Task: Simulation of Unaffected Pollution Spread.** COMSOL provides a number of different solvers and solver settings. All tasks of this comparison are solved using an UMFPACK solver with a relative tolerance of  $10^{-5}$ .

Figure 2 and Figure 3 show results for time steps  $t = 50$  and  $t = 100$ . The solution of time step  $t = 150$  fits exactly to the solution at  $t = 100$ . The accuracy of the FEM solution can be increased especially by the mesh fineness and its quality. The analytical approximation is calculated in *MATLAB*, called directly from COMSOL Multiphysics in a subdomain definition window.

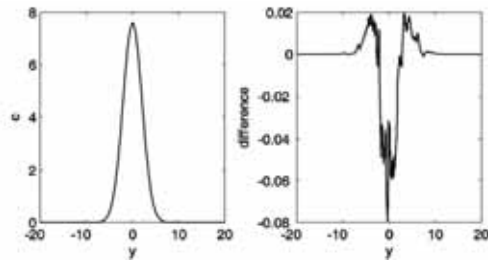


Figure 2: FEM solution at  $(50, y)$ ,  $t = 50$  (plot at left) and difference analytical / FEM solution (plot at right).

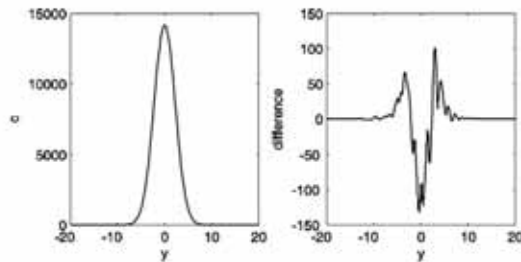


Figure 2: FEM solution at  $(50, y)$ ,  $t = 100$  (plot at left) and difference analytical / FEM solution (plot at right).

## B - Task: Pollution Reduction by Facilities.

COMSOL is able to derive the steady state solution of the transport equation numerically. This solution can be used as initial value of the following time dependent solving process, by using the restart-button instead of the solve-button. Before the time dependent solution can be started with switched-on plants, the reaction rate  $R$  in the subdomain settings of the two circles have to be changed from zero to  $-\lambda \cdot c$  (Figure 5; numerical solutions given in Figure 4).

**C - Task: Controlled Pollution Reduction.** If the facilities are periodically active, Monday to Friday from 20.00 - 08.00, and at weekends around the clock, the reaction rate can be controlled by a MATLAB script function. COMSOL is able to handle this function by calling it in the subdomain settings. The MATLAB function `reaction.m` is called in the reaction rate input line of the subdomain setting mask (Figure 5). Numerical results are given in Figure 6.

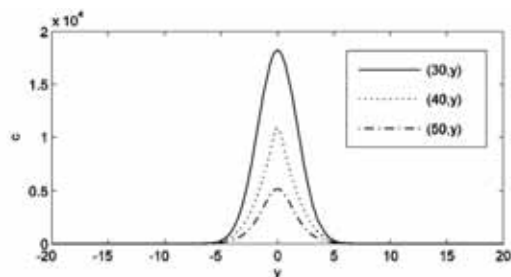


Figure 4: Concentration at the lines  $(30, y)$ ,  $(40, y)$ ,  $(50, y)$  at  $t = 100$  days, with continuously working pollution reduction facilities

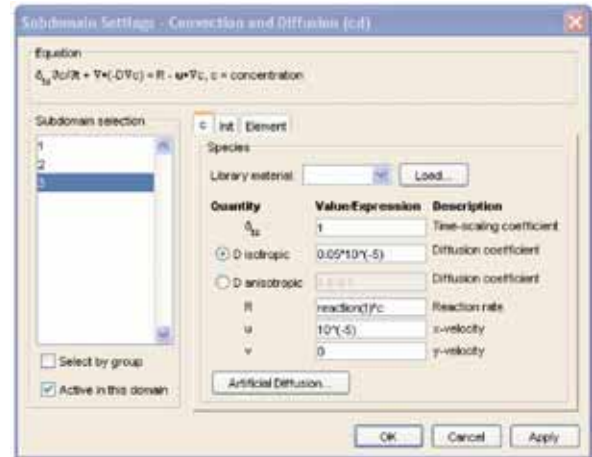


Figure 5: Definition mask for subdomain dynamics.

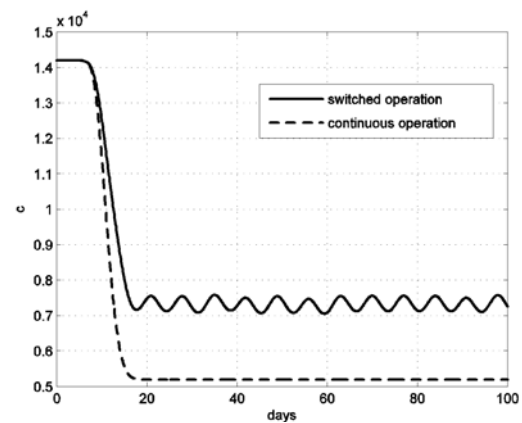


Figure 6: Concentration at  $(50, 0)$  for  $0 \leq t \leq 100$  days with pollution reduction facilities, continuous and switched operation.

**Résumé:** This solution with COMSOL Multiphysics is based on the Finite-Element - Method (FEM). The groundwater stream is modelled as (finite) rectangle, with subdomains for the source and for the two facilities. On each subdomain, refined grids and suitable boundary conditions are used, in order to calculate a solution with a suitable solver (UMFPACK). The approximation for the solution, the continuous degradation for the plant activities, and time control for the plant operation are modelled by MATLAB functions, to be called directly from COMSOL Multiphysics. For this solution, COMSOL Multiphysics 3.1 has been used.

## Corresponding Author:

Harald Teufelsbauer, [harald.teufelsbauer@boku.ac.at](mailto:harald.teufelsbauer@boku.ac.at)  
Department of Structural Engineering & Natural Hazard,  
Institute of Mountain Risk Engineering,  
University of Natural Resources and Applied Life Sciences  
Peter Jordan - Strasse 82, 1190, Vienna, Austria

Received: September 15, 2006

Revised: October 20, 2007

Accepted: October 25, 2007

# SNE NEWS SECTION

## DATA & QUICK INFO



### CONTENT

Info EUROSIM	page 2
Info EUROSIM Societies	page 3 - 6
Info ASIM	page 3
Info CROSSIM, CSSS, DBSS, FRANCOSIM	page 4
Info HSS, ISCS, PSCS, SIMS	page 5
Info SLOSIM, UKSIM, LSS CAE-SMSG, ROMSIM,	page 6
In Memoriam Len Dekker	page 7
EUROSIM Congress 2007	page 7

**SNE - Simulation News Europe** is the official journal of EUROSIM and sent to most members of the EUROSIM Societies as part of the membership benefits. Furthermore **SNE** is distributed to other societies and to individuals active in the area of modelling and simulation. **SNE** is registered with ISSN 1015-8685. Circulation of printed version is 3000 copies. **SNE at Web** – recent issues of **SNE** are also available via internet at [WWW.ARGESIM.ORG](http://WWW.ARGESIM.ORG). Members of EUROSIM Societies may have access to the **SNE** Archive.

This special *News Section* compiles data from EUROSIM and EUROSIM societies: addresses, weblinks, officers of societies with function and email. It is planned to publish this *EUROSIM Data & Quick Info* annually, from 2007 on in the *SNE Special Issues*. Furthermore, this News Section has the sad duty to announce the death of Len Dekker, one of the fathers of EUROSIM.

### SNE REPORTS EDITORIAL BOARD

#### EUROSIM

Borut Zupančič, [borut.zupancic@fe.uni-lj.si](mailto:borut.zupancic@fe.uni-lj.si)  
Felix Breiteneker, [Felix.Breiteneker@tuwien.ac.at](mailto:Felix.Breiteneker@tuwien.ac.at)

ASIM: Thorsten Pawletta, [pawel@mb.hs-wismar.de](mailto:pawel@mb.hs-wismar.de)  
CROSSIM: Jadranka Božikov, [jbozikov@snz.hr](mailto:jbozikov@snz.hr)  
CSSS: Mikuláš Alexík, [alexik@frtk.utc.sk](mailto:alexik@frtk.utc.sk)  
DBSS: A. Heemink, [a.w.heemink@its.tudelft.nl](mailto:a.w.heemink@its.tudelft.nl)  
FRANCOSIM: Y. Hamam, [y.hamam@esiee.fr](mailto:y.hamam@esiee.fr)  
HSS: András Jávör, [javor@eik.bme.hu](mailto:javor@eik.bme.hu)  
ISCS: M. Savastano, [mario.savastano@unina.it](mailto:mario.savastano@unina.it)  
PSCS: Zenon Sosnowski, [zenon@ii.pb.bialystok.pl](mailto:zenon@ii.pb.bialystok.pl)  
SIMS: Esko Juuso, [esko.juuso@oulu.fi](mailto:esko.juuso@oulu.fi)  
SLOSIM: Borut Zupančič, [zupancic@fe.uni-lj.si](mailto:zupancic@fe.uni-lj.si)  
UKSIM: Alessandra Orsoni, [A.Orsoni@kingston.ac.uk](mailto:A.Orsoni@kingston.ac.uk)  
CAE-SMSG: María J. la Fuente, [maria@autom.uva.es](mailto:maria@autom.uva.es)  
LSS: Yuri Merkuryev, [merkur@itl.rtu.lv](mailto:merkur@itl.rtu.lv)  
ROMSIM: Florin Stanciulescu, [sflorin@ici.ro](mailto:sflorin@ici.ro)

#### ARGESIM

Felix Breiteneker, [Felix.Breiteneker@tuwien.ac.at](mailto:Felix.Breiteneker@tuwien.ac.at)  
Anna Breiteneker, [Anna.Breiteneker@liwest.at](mailto:Anna.Breiteneker@liwest.at)  
Nikolas Popper, [Niki.Popper@drahtwarenhandlung.at](mailto:Niki.Popper@drahtwarenhandlung.at)

**INFO:** [WWW.ARGESIM.ORG](http://WWW.ARGESIM.ORG); [sne@argesim.org](mailto:sne@argesim.org)

If you have any information, announcement, etc. you want to see published, please contact a member of the editorial board in your country or [sne@argesim.org](mailto:sne@argesim.org).

*Editorial Information / Impressum - see front cover, inside*



## Information EUROSIM



### EUROSIM Federation of European Simulation Societies

**General Information.** *EUROSIM*, the Federation of European Simulation Societies, was set up in 1989. The purpose of *EUROSIM* is to provide a European forum for regional and national simulation societies to promote the advancement of modelling and simulation in industry, research, and development.

- WWW info *EUROSIM*: [WWW.EUROSIM.INFO](http://WWW.EUROSIM.INFO)

**Member Societies.** *EUROSIM* members may be national simulation societies and regional or international societies and groups dealing with modelling and simulation. At present *EUROSIM* has eleven full members and three observer members:

- ASIM - Arbeitsgemeinschaft Simulation (Austria, Germany, Switzerland)
- CROSSIM - Croatian Society for Simulation Modelling (Croatia)
- CSSS - Czech & Slovak Simulation Society (Czech Republic, Slovak Republic)
- DBSS - Dutch Benelux Simulation Society (Belgium, The Netherlands)
- FRANCOSIM - Société Francophone de Simulation (Belgium, France)
- HSS - Hungarian Simulation Society (Hungary)
- ISCS - Italian Society for Computer Simulation (Italy)
- PSCS - Polish Society for Computer Simulation (Poland)
- SIMS - Simulation Society of Scandinavia (Denmark, Finland, Norway, Sweden)
- SLOSIM - Slovenian Simulation Society (Slovenia)
- UKSIM - United Kingdom Simulation Society (UK, Ireland)
- CEA-SMGS - Spanish Modelling and Simulation Group (Spain; Observer Member)
- LSS - Latvian Simulation Society (Latvia; Observer Member)
- ROMSIM - Romanian Society for Modelling and Simulation (Romania; Observer Member)

Contact addresses, weblinks and officers of the societies may be found in the information part of the societies.

**EUROSIM Board / EUROSIM Officers.** *EUROSIM* is governed by a board consisting of one representative of each member society, president and past president, and representatives for SNE and SIMPRA. The President is nominated by the society organising the next *EUROSIM* Congress. Secretary and Treasurer are elected out of members of the Board.

- President: Borut Zupančič (SLOSIM); [borut.zupancic@fe.uni-lj.si](mailto:borut.zupancic@fe.uni-lj.si)
- Past President: Yskander Hamam (FRANCOSIM); [y.hamam@esiee.fr](mailto:y.hamam@esiee.fr)
- Secretary: Peter Fritzson (SIMS); [petfr@ida.liu.se](mailto:petfr@ida.liu.se)
- Treasurer: Felix Breitenecker (ASIM); [Felix.Breitenecker@tuwien.ac.at](mailto:Felix.Breitenecker@tuwien.ac.at)
- SIMPRA Representative: Jürgen Halin; [halin@jet.mavt.ethz.ch](mailto:halin@jet.mavt.ethz.ch)
- SNE Representative: Felix Breitenecker; [Felix.Breitenecker@tuwien.ac.at](mailto:Felix.Breitenecker@tuwien.ac.at)

**SNE - Simulation News Europe.** *EUROSIM* societies are offered to distribute to their members the journal *Simulation News Europe* (SNE) as official membership journal. SNE is a scientific journal with reviewed contributions in the *Notes Section* as well as a membership newsletter for *EUROSIM* with information from the societies in the *News Section*. Publisher are *EUROSIM*, ARGESIM and ASIM

- Editor-in-Chief SNE: Felix Breitenecker; [Felix.Breitenecker@tuwien.ac.at](mailto:Felix.Breitenecker@tuwien.ac.at)
- WWW SNE: [WWW.ARGESIM.ORG](http://WWW.ARGESIM.ORG), menu *SNE*;  
[WWW.ASIM-GI.ORG](http://WWW.ASIM-GI.ORG), menu *International*

**SIMPRA.** Members of *EUROSIM* societies can subscribe the official *EUROSIM* scientific journal *Simulation Modelling Practice and Theory* (SIMPRA) at a significantly reduced price.

- Editor-in-Chief SIMPRA: Jürgen Halin; [halin@jet.mavt.ethz.ch](mailto:halin@jet.mavt.ethz.ch)
- WWW info SIMPRA: [EES.ELSEVIER.COM/simpat/](http://EES.ELSEVIER.COM/simpat/)

**EUROSIM Congress.** *EUROSIM* is running the triennial conference series *EUROSIM Congress*. The congress is organised by one of the *EUROSIM* societies. *EUROSIM* 2007 will be organised by SLOSIM in Ljubljana, September 9-13, 2007; *EUROSIM* 2010 will be organised by CSSS in Prague, September 2010.

- WWW info *EUROSIM* 2007: [WWW.EUROSIM2007.ORG](http://WWW.EUROSIM2007.ORG)
- Chair OC *EUROSIM* 2007: Borut Zupančič (SLOSIM); [borut.zupancic@fe.uni-lj.si](mailto:borut.zupancic@fe.uni-lj.si)
- Chair IPC *EUROSIM* 2007: Rihard Karba (SLOSIM); [rihard.karba@fe.uni-lj.si](mailto:rihard.karba@fe.uni-lj.si)
- Info *EUROSIM* 2010: Mikuláš Alexík (CSSS); [alexik@frtk.utc.sk](mailto:alexik@frtk.utc.sk)





**ASIM**  
**German Simulation Society**  
**Arbeitsgemeinschaft Simulation**  
 WWW.ASIM-GI.ORG

ASIM (Arbeitsgemeinschaft Simulation) is the association for simulation in the German speaking area, servicing mainly Germany, Switzerland and Austria. ASIM was founded in 1981 and has now about 700 individual members, and 30 institutional or industrial members. Furthermore, ASIM counts about 300 affiliated members.

- WWW info EUROSIM: WWW.ASIM-GI.ORG with members' area (downloads, etc.)
- Email contacts: [info@asim-gi.org](mailto:info@asim-gi.org), [admin@asim-gi.org](mailto:admin@asim-gi.org)
- Address: ASIM - Inst.f. Analysis and Scientific Computing  
Vienna University of Technology  
Wiedner Hauptstrasse 8-10, 1040 Vienna, AUSTRIA

**ASIM Working Groups.** ASIM, part of GI - Gesellschaft für Informatik, is organised in Working Groups, dealing with applications and comprehensive subjects:

- **GMMS** Methods in Modelling and Simulation  
Info: Peter Schwarz, [schwarz@eas.iis.fhg.de](mailto:schwarz@eas.iis.fhg.de)
- **SUGMBB** Simulation in Environmental Systems  
Info: J. Wittmann, [wittmann@informatik.uni-hamburg.de](mailto:wittmann@informatik.uni-hamburg.de)
- **STS** Simulation of Technical Systems  
Info: H.-T. Mammen, [Heinz-Theo.Mammen@hella.com](mailto:Heinz-Theo.Mammen@hella.com)
- **SPL** Simulation in Production and Logistics  
Info: Sigrid Wenzel, [s.wenzel@uni-kassel.de](mailto:s.wenzel@uni-kassel.de)
- **SVS** Simulation of Transport Systems  
Info: U. Brannolte, [Brannolte@bauing.uni-weimar.de](mailto:Brannolte@bauing.uni-weimar.de)
- **SBW** Simulation in OR  
Info: C. Böhnlein, [boehnlein@wiinf.uni-wuerzburg.de](mailto:boehnlein@wiinf.uni-wuerzburg.de)
- **EDU** Simulation in Education/Education in Simulation  
Info: W. Wiechert, [wiechert@simtec.mb.uni-siegen.de](mailto:wiechert@simtec.mb.uni-siegen.de)

### ASIM Publications

**SNE - Simulation News Europe.** ASIM is publishing (co-publishing) SNE, which is regularly published and sent to all ASIM members (as part of their membership; 900 issues) and for promotion purposes (300 issues). Since 2006, the ASIM Working Groups publish *SNE Special Issues* with state-of-the-art reports on modelling and simulation in their workspace.

**ASIM News.** In December 2005, the *ASIM Nachrichten* has been replaced by an electronic news-letter - *ASIM Newsletter*. Editors are Th. Pawletta and C. Deatcu, Univ. Wismar, [pawel@mb.hs-wismar.de](mailto:pawel@mb.hs-wismar.de).

**ASIM Notes - ASIM Mitteilungen.** The trademark *ASIM Mitteilungen* (*ASIM Note*) stands for all publications of ASIM and of the the ASIM Working Groups.

Each publication gets an identification as *ASIM Notes*, independent of the publisher, and independent of the publication medium (printed books, CD, Web). *ASIM Notes* range from printed books (with CDs) published by Springer, via workshop publication published in SNE or ARGESIM, to compiled abstracts published at the ASIM webserver.

**ASIM Books.** ASIM co-operates with the SCS Publishing House e.V., with ARGESIM (Vienna University of Technology), and with Shaker Verlag Aachen in publication of two book series (*Fortschritte in der Simulationstechnik - Frontiers in Simulation* and *Fortschrittsberichte Simulation - Advances in Simulation*) and in publication of Proceedings. Publications in these series range from monographs via proceedings to PhD theses.

**ASIM Board and Officers:** The ASIM board consists of elected officers (elected all three years), of the chairpersons of the ASIM Working Groups (independently elected all three years), and of co-opted specialists.

Function	Name and Email
President	Felix Breiteneker, <a href="mailto:Felix.Breiteneker@tuwien.ac.at">Felix.Breiteneker@tuwien.ac.at</a>
Vice-President	Sigrid Wenzel, <a href="mailto:s.wenzel@uni-kassel.de">s.wenzel@uni-kassel.de</a>
Vice-President	Thorsten Pawletta, <a href="mailto:pawel@mb.hs-wismar.de">pawel@mb.hs-wismar.de</a>
Secretary	Claus-Burkhard Böhnlein, <a href="mailto:boehnlein@wiinf.uni-wuerzburg.de">boehnlein@wiinf.uni-wuerzburg.de</a>
Treasurer	Ingrid Bausch-Gall, <a href="mailto:Ingrid@Bausch-Gall.de">Ingrid@Bausch-Gall.de</a>
Membership Affairs	Sigrid Wenzel, <a href="mailto:s.wenzel@uni-kassel.de">s.wenzel@uni-kassel.de</a> W. Maurer, <a href="mailto:werner.maurer@zhwin.ch">werner.maurer@zhwin.ch</a> I. Bausch-Gall, <a href="mailto:Ingrid@Bausch-Gall.de">Ingrid@Bausch-Gall.de</a> Felix Breiteneker, <a href="mailto:Felix.Breiteneker@tuwien.ac.at">Felix.Breiteneker@tuwien.ac.at</a>
Universities	W. Wiechert, <a href="mailto:wiechert@simtec.mb.uni-siegen.de">wiechert@simtec.mb.uni-siegen.de</a>
Industry	Sigrid Wenzel, <a href="mailto:s.wenzel@uni-kassel.de">s.wenzel@uni-kassel.de</a> Klaus Panreck, <a href="mailto:Klaus.Panreck@hella.com">Klaus.Panreck@hella.com</a>
Conferences	Klaus Panreck, <a href="mailto:Klaus.Panreck@hella.com">Klaus.Panreck@hella.com</a> A. Gnauck, <a href="mailto:albrecht.gnauck@tu-cottbus.de">albrecht.gnauck@tu-cottbus.de</a>
Publications	Th. Pawletta, <a href="mailto:pawel@mb.hs-wismar.de">pawel@mb.hs-wismar.de</a> Felix Breiteneker, <a href="mailto:Felix.Breiteneker@tuwien.ac.at">Felix.Breiteneker@tuwien.ac.at</a>
Repr. EUROSIM	Felix Breiteneker, <a href="mailto:Felix.Breiteneker@tuwien.ac.at">Felix.Breiteneker@tuwien.ac.at</a>
Deputy	W. Wiechert, <a href="mailto:wiechert@simtec.mb.uni-siegen.de">wiechert@simtec.mb.uni-siegen.de</a>
Edit. Board SNE	Th. Pawletta, <a href="mailto:pawel@mb.hs-wismar.de">pawel@mb.hs-wismar.de</a> Heads of ASIM Working Groups
Web EUROSIM	Johannes Kropf, <a href="mailto:jkropf@osiis.tuwien.ac.at">jkropf@osiis.tuwien.ac.at</a>



## CROSSIM - Croatian Society for Simulation Modelling

CROSSIM - *CROatian Society for SIMulation Model-ling* was founded in 1992 as a non-profit society with the goal to promote knowledge and use of simulation methods and techniques and development of education. CROSSIM is a full member of EUROSIM since 1997.

- Web info: [WWW.EUROSIM.INFO](http://WWW.EUROSIM.INFO)
- Address: CROSSIM / Jadranka Božikov  
Andrija Stampar School of Public Health,  
Medical School, University of Zagreb  
Rockefeller St. 4, HR-10000 Zagreb, CROATIA

Function	Name and Email
President	Jadranka Božikov, <a href="mailto:jbozikov@snz.hr">jbozikov@snz.hr</a>
Vice-president	Vesna Dušak, <a href="mailto:vdusak@foi.hr">vdusak@foi.hr</a>
Secretary	Vesna Bosilj-Vukšić, <a href="mailto:vbosilj@efzg.hr">vbosilj@efzg.hr</a>
Executive	Vlatko Čerić, <a href="mailto:vceric@efzg.hr">vceric@efzg.hr</a>
Board Members	Tarzan Legović, <a href="mailto:legovic@irb.hr">legovic@irb.hr</a>
Repr. EUROSIM	Jadranka Božikov, <a href="mailto:jbozikov@snz.hr">jbozikov@snz.hr</a>
Edit. Board SNE	Jadranka Božikov, <a href="mailto:jbozikov@snz.hr">jbozikov@snz.hr</a>
Web EUROSIM	Jadranka Božikov, <a href="mailto:jbozikov@snz.hr">jbozikov@snz.hr</a>



## CSSS - Czech and Slovak Simulation Society

CSSS - *The Czech and Slovak Simulation Society* has about 150 members working in Czech and Slovak national scientific and technical societies (*Czech Society for Applied Cybernetics and Informatics*, *Slovak Society for Applied Cybernetics and Informatics* -SSAKI). The main objectives of the society are: development of education and training in the field of modelling and simulation, organising professional workshops and conferences, disseminating information about modelling and simulation activities in Europe. Since 1992 CSSS is full member of EUROSIM.

- Web info: [WWW.FIT.VUTBR.CZ/CSSS](http://WWW.FIT.VUTBR.CZ/CSSS)
- Address: CSSS / Mikuláš Alexík, University of Zilina  
dept. Technical Cybernetics, Univerzitná 8215/1,  
010 26 Zilina, SLOVAC REPUBLIC

Function	Name and Email
President	Jan Štefan, <a href="mailto:jan.stefan@marq.cz">jan.stefan@marq.cz</a>
Vice-president	Mikuláš Alexík, <a href="mailto:alexik@frtk.fri.utc.sk">alexik@frtk.fri.utc.sk</a>
Treasurer	Miroslav Šnorek, <a href="mailto:snorek@cslab.felk.cvut.cz">snorek@cslab.felk.cvut.cz</a>
Executive	Vlatko Čerić, <a href="mailto:vceric@efzg.hr">vceric@efzg.hr</a>
Board Members	Tarzan Legović, <a href="mailto:legovic@irb.hr">legovic@irb.hr</a>
Repr. EUROSIM	Mikuláš Alexík, <a href="mailto:alexik@frtk.fri.utc.sk">alexik@frtk.fri.utc.sk</a>
Deputy	Miroslav Šnorek, <a href="mailto:snorek@cslab.felk.cvut.cz">snorek@cslab.felk.cvut.cz</a>
Edit. Board SNE	Mikuláš Alexík, <a href="mailto:alexik@frtk.fri.utc.sk">alexik@frtk.fri.utc.sk</a>
Web EUROSIM	Adam Jaros, <a href="mailto:adam.jaros@fri.utc.sk">adam.jaros@fri.utc.sk</a>

## DBSS - Dutch Benelux Simulation Society

The Dutch Benelux Simulation Society (DBSS) was founded in July 1986 in order to create an organisation of simulation professionals within the Dutch language area. DBSS has actively promoted creation of similar organisations in other language areas. DBSS is a member of EUROSIM and works in close cooperation with its members and is further affiliated with SCS International, IMACS, and the Chinese Association for System Simulation and the Japanese Society for Simulation Technology.

- Web info: [WWW.EUROSIM.INFO](http://WWW.EUROSIM.INFO)
- Address: DBSS / A. W. Heemink, Delft University of Technology, ITS - twi, Mekelweg 4,  
2628 CD Delft, THE NETHERLANDS

Function	Name and Email
President	A.W. Heemink, <a href="mailto:a.w.heemink@its.tudelft.nl">a.w.heemink@its.tudelft.nl</a>
Vice-president	W. Smit, <a href="mailto:smitnet@wxs.nl">smitnet@wxs.nl</a>
Treasurer	W. Smit, <a href="mailto:smitnet@wxs.nl">smitnet@wxs.nl</a>
Secretary	W. Smit, <a href="mailto:smitnet@wxs.nl">smitnet@wxs.nl</a>
Repr. EUROSIM	A.W. Heemink, <a href="mailto:a.w.heemink@its.tudelft.nl">a.w.heemink@its.tudelft.nl</a>
- Deputy	W. Smit, <a href="mailto:smitnet@wxs.nl">smitnet@wxs.nl</a>
Edit. Board SNE	A.W. Heemink, <a href="mailto:a.w.heemink@its.tudelft.nl">a.w.heemink@its.tudelft.nl</a>

## FRANCOSIM - Société Francophone de Simulation

FRANCOSIM was founded in 1991 and aims to the promotion of simulation and research, in industry and academic fields. Francosim operates two poles.

- Pole Modelling and simulation of discrete event systems; Pole contact: Henri Pierreval,  
[pierreval@ifma.fr](mailto:pierreval@ifma.fr)
- Pole Modelling and simulation of continuous systems; Pole contact: Yskandar Hamam,  
[y.hamam@esiee.fr](mailto:y.hamam@esiee.fr), [WWW.ESIEE.FR/~HAMAMY](http://WWW.ESIEE.FR/~HAMAMY)

- Web info: [WWW.EUROSIM.INFO](http://WWW.EUROSIM.INFO)
- Address: FRANCOSIM / Yskandar Hamam  
Groupe ESIEE, Cité Descartes,  
BP 99, 2 Bd. Blaise Pascal,  
93162 Noisy le Grand CEDEX, FRANCE

Function	Name and Email
President	Yskandar Hamam, <a href="mailto:y.hamam@esiee.fr">y.hamam@esiee.fr</a>
Vice president	-
Treasurer	François Rocaries, <a href="mailto:f.rocaries@esiee.fr">f.rocaries@esiee.fr</a>
Repr. EUROSIM	Yskandar Hamam, <a href="mailto:y.hamam@esiee.fr">y.hamam@esiee.fr</a>
Deputy	-
Edit. Board SNE	Yskandar Hamam, <a href="mailto:y.hamam@esiee.fr">y.hamam@esiee.fr</a>



## HSS - Hungarian Simulation Society

The Hungarian Member Society of EUROSIM was established in 1981 as an association promoting the exchange of information within the community of people involved in research, development, application and education of simulation in Hungary and also contributing to the enhancement of exchanging information between the Hungarian simulation community and the simulation communities abroad. HSS deals with the organization of lectures, exhibitions, demonstrations, and conferences.

- Web info: [WWW.EUROSIM.INFO](http://WWW.EUROSIM.INFO)
- Address: HSS / András Jávör, Dept. Information & Knowledge Management, Budapest Univ. of Technology and Economics, Sztoczek u. 4, 1111 Budapest, HUNGARY

Function	Name and Email
President	András Jávör, <a href="mailto:javor@eik.bme.hu">javor@eik.bme.hu</a>
Vice-president	Gábor Szűcs, <a href="mailto:szucs@itm.bme.hu">szucs@itm.bme.hu</a>
Secretary	Ágnes Vigh, <a href="mailto:vigh@itm.bme.hu">vigh@itm.bme.hu</a>
Repr. EUROSIM	András Jávör, <a href="mailto:javor@eik.bme.hu">javor@eik.bme.hu</a>
Deputy	Gábor Szűcs, <a href="mailto:szucs@itm.bme.hu">szucs@itm.bme.hu</a>
Edit. Board SNE	András Jávör, <a href="mailto:javor@eik.bme.hu">javor@eik.bme.hu</a>
Web EUROSIM	Gábor Szűcs, <a href="mailto:szucs@itm.bme.hu">szucs@itm.bme.hu</a>

## ISCS - Italian Society for Computer Simulation

The Italian Society for Computer Simulation (ISCS) is a scientific non-profit association of members from industry, university, education and several public and research institutions with common interest in all fields of computer simulation.

- Web info: [WWW.EUROSIM.INFO](http://WWW.EUROSIM.INFO)
- Address: ISCS / Mario Savastano, c/o CNR - IRSIP, Via Claudio 21, 80125 Napoli, ITALY

Function	Name and Email
President	Mario Savastano, <a href="mailto:mario.savastano@unina.it">mario.savastano@unina.it</a>
Vice-president	F. Maceri, <a href="mailto:Franco.Maceri@uniroma2.it">Franco.Maceri@uniroma2.it</a>
Secretary	Paola Provenzano, <a href="mailto:paola.provenzano@uniroma2.it">paola.provenzano@uniroma2.it</a>
Treasurer	Pasquale Arpaia
Repr. EUROSIM	F. Maceri, <a href="mailto:Franco.Maceri@uniroma2.it">Franco.Maceri@uniroma2.it</a>
Deputy	-
Edit. Board SNE	Mario Savastano, <a href="mailto:mario.savastano@unina.it">mario.savastano@unina.it</a>

## PSCS - Polish Society for Computer Simulation

PSCS was founded in 1993 in Warsaw. PSCS is a scientific, non-profit association of members from universities, research institutes and industry in Poland with common interests in variety of methods of computer simulations and its applications. At present PSCS counts 264 members.

- Web info: [WWW.PTSK.MAN.BIALYSTOK.PL](http://WWW.PTSK.MAN.BIALYSTOK.PL)
- Address: PSCS / Leon Bobrowski, c/o IBIB PAN, ul. Trojdena 4 (p.416), 02-109 Warszawa, POLAND

Function	Name and Email
President	Leon Bobrowski, <a href="mailto:leon@ibib.waw.pl">leon@ibib.waw.pl</a>
Vice-president	Andrzej Chudzikiewicz, <a href="mailto:ach@it.pw.edu.pl">ach@it.pw.edu.pl</a>
Treasurer	Zenon Sosnowski, <a href="mailto:zenon@ii.pb.bialystok.pl">zenon@ii.pb.bialystok.pl</a>
Secretary	Zdzisław Galkowski, <a href="mailto:Zdzislaw.Galkowski@simr.pw.edu.pl">Zdzislaw.Galkowski@simr.pw.edu.pl</a>
Repr. EUROSIM	Leon Bobrowski, <a href="mailto:leon@ibib.waw.pl">leon@ibib.waw.pl</a>
Deputy	Andrzej Chudzikiewicz, <a href="mailto:ach@it.pw.edu.pl">ach@it.pw.edu.pl</a>
Edit. Board SNE	Zenon Sosnowski, <a href="mailto:zenon@ii.pb.bialystok.pl">zenon@ii.pb.bialystok.pl</a>

## SIMS - Scandinavian Simulation Society

SIMS is the *Scandinavian Simulation Society* with members from the four Nordic countries Denmark, Finland, Norway and Sweden. The SIMS history goes back to 1959. SIMS practical matters are taken care of by the SIMS board consisting of two representatives from each Nordic country. Iceland will be represented by one board member.

**SIMS Structure.** SIMS is organised as federation of regional societies. There are **FinSim** (*Finnish Simulation Forum*), **DKSIM** (*Dansk Simuleringsforening*) and **NFA** (*Norsk Forening for Automatisering*).

- Web info: [WWW.SCANSIMS.ORG](http://WWW.SCANSIMS.ORG)
- Address: SIMS / Peter Fritzson, IDA, Linköping University, 58183, Linköping, SWEDEN

Function	Name and Email
President	Peter Fritzson, <a href="mailto:petfr@ida.liu.se">petfr@ida.liu.se</a>
Treasurer	Vadim Engelson, <a href="mailto:vaden@ida.liu.se">vaden@ida.liu.se</a>
Repr. EUROSIM	Peter Fritzson, <a href="mailto:petfr@ida.liu.se">petfr@ida.liu.se</a>
Deputy	-
Edit. Board SNE	Esko Juuso, <a href="mailto:esko.juuso@oulu.fi">esko.juuso@oulu.fi</a>
Web EUROSIM	Vadim Engelson, <a href="mailto:vaden@ida.liu.se">vaden@ida.liu.se</a>



### SLOSIM - Slovenian Society for Simulation and Modelling

SLOSIM - *Slovenian Society for Simulation and Modelling* - was established in 1994 and became the full member of EUROSIM in 1996. Currently it has 69 members from both slovenian universities, institutes, and industry. It promotes modelling and simulation approach to problem solving in industrial as well as in academic environments by establishing communication and cooperation among the corresponding teams.

- Web info: [MSC.FE.UNI-LJ.SI/SLOSIM](http://MSC.FE.UNI-LJ.SI/SLOSIM)
- Address: SLOSIM / Rihard Karba, Faculty of Electrical Engineering, University of Ljubljana, Tržaška 25, 1000 Ljubljana, SLOVENIA

Function	Name and Email
President	Rihard Karba, <a href="mailto:rihard.karba@fe.uni-lj.si">rihard.karba@fe.uni-lj.si</a>
Vice-president	Leon Žlajpah, <a href="mailto:leon.zlajpah@ijs.si">leon.zlajpah@ijs.si</a>
Secretary	Aleš Belič, <a href="mailto:ales.belic@fe.uni-lj.si">ales.belic@fe.uni-lj.si</a>
Treasurer	Milan Simčič, <a href="mailto:milan.simcic@fe.uni-lj.si">milan.simcic@fe.uni-lj.si</a>
Repr. EUROSIM	Rihard Karba, <a href="mailto:rihard.karba@fe.uni-lj.si">rihard.karba@fe.uni-lj.si</a>
Deputy	Borut Zupančič, <a href="mailto:borut.zupancic@fe.uni-lj.si">borut.zupancic@fe.uni-lj.si</a>
Edit. Board SNE	Rihard Karba, <a href="mailto:rihard.karba@fe.uni-lj.si">rihard.karba@fe.uni-lj.si</a>
Web EUROSIM	Aleš Belič, <a href="mailto:ales.belic@fe.uni-lj.si">ales.belic@fe.uni-lj.si</a>

### UKSim - United Kingdom Simulation Society

UKSim has more than 100 members throughout the UK from universities and industry. It is active in all areas of simulation and it holds a biennial conference as well as regular meetings and workshops.

- Web info: [WWW.UKSIM.ORG.UK](http://WWW.UKSIM.ORG.UK)
- Address: UKSIM / Alessandra Orsoni, Kingston Business School, Kingston Hill, Kingston-Upon-Thames, Surrey, KT2 7LB, UNITED KINGDOM

Function	Name and Email
President	David Al-Dabass, <a href="mailto:david.al-dabass@ntu.ac.uk">david.al-dabass@ntu.ac.uk</a>
Secretary	Alessandra Orsoni, <a href="mailto:A.Orsoni@kingston.ac.uk">A.Orsoni@kingston.ac.uk</a>
Treasurer	B. Thompson, <a href="mailto:barry@bjtcon.ndo.co.uk">barry@bjtcon.ndo.co.uk</a>
Membership Chair	K. Al-Begain, <a href="mailto:kbegain@glam.ac.uk">kbegain@glam.ac.uk</a>
Univ. Liaison Chair	R. Cheng, <a href="mailto:rhc@maths.soton.ac.uk">rhc@maths.soton.ac.uk</a>
Ind. Liaison Chair	Richard Zobel, <a href="mailto:r.zobel@ntworld.com">r.zobel@ntworld.com</a>
Conf. Venue Chair	John Pollard, <a href="mailto:j.pollard@ee.ucl.ac.uk">j.pollard@ee.ucl.ac.uk</a>
Repr. EUROSIM	A. Orsoni, <a href="mailto:A.Orsoni@kingston.ac.uk">A.Orsoni@kingston.ac.uk</a>
Edit. Board SNE	A. Orsoni, <a href="mailto:A.Orsoni@kingston.ac.uk">A.Orsoni@kingston.ac.uk</a>

### CEA-SMSG - Spanish Modelling and Simulation Group

CEA is the Spanish Society on Automation and Control. In order to improve the efficiency and to deep into the different fields of automation, the association is divided into thematic groups, one of them is named 'Modelling and Simulation', constituting then the CEA-SMSG.

- Web info: [WWW.CEA-IFAC.ES/wwwgrupos/simulacion](http://WWW.CEA-IFAC.ES/wwwgrupos/simulacion)
- Address: CEA-SMSG / María Jesús de la Fuente, System Engineering and Automatic Control department, University of Valladolid, Real de Burgos s/n., 47011 Valladolid, SPAIN

Function	Name and Email
President	María J. la Fuente, <a href="mailto:maria@autom.uva.es">maria@autom.uva.es</a>
Repr. EUROSIM	María J. la Fuente, <a href="mailto:maria@autom.uva.es">maria@autom.uva.es</a>

### LSS - Latvian Simulation Society

The Latvian Simulation Society (LSS) has been founded in 1990 as the first professional simulation organisation in the field of Modelling and simulation in the post-Soviet area. Its members represent the main simulation centres in Latvia, including both academic and industrial sectors.

- Web info: [BRIEDIS.ITL.RTU.LV/imb/](http://BRIEDIS.ITL.RTU.LV/imb/)
- Address: LSS / Yuri Merkuryev, Dept. of Modelling and Simulation Riga Technical University, Kalku street 1, Riga, LV-1658, LATVIA

Function	Name and Email
President	Yuri Merkuryev, <a href="mailto:merkur@itl.rtu.lv">merkur@itl.rtu.lv</a>
Repr. EUROSIM	Yuri Merkuryev, <a href="mailto:merkur@itl.rtu.lv">merkur@itl.rtu.lv</a>

### ROMSIM - Romanian Modelling and Simulation Society

ROMSIM has been founded in 1990 as a non-profit society, devoted to both theoretical and applied aspects of modelling and simulation of systems. ROMSIM currently has about 100 members from both Romania and Republic of Moldova.

- Web info: [INFODOC.ICL.RO/romsim](http://INFODOC.ICL.RO/romsim)
- Address: ROMSIM / Florin Stanculescu, National Inst. for Research in Informatics, Averscu Avenue 8-10, 71316 Bucharest, ROMANIA

Function	Name and Email
President	Florin Stanculescu, <a href="mailto:sflorin@ici.ro">sflorin@ici.ro</a>
Vice-president	Florin Hartescu, <a href="mailto:flory@ici.ro">flory@ici.ro</a>
Secretary	Zoe Radulescu, <a href="mailto:radulescu@ici.ro">radulescu@ici.ro</a>
Repr. EUROSIM	Florin Stanculescu, <a href="mailto:sflorin@ici.ro">sflorin@ici.ro</a>
Deputy	Florin Hartescu, <a href="mailto:flory@ici.ro">flory@ici.ro</a>
Edit. Board SNE	Florin Stanculescu, <a href="mailto:sflorin@ici.ro">sflorin@ici.ro</a>





## In memoriam prof.dr.ir. Len Dekker (1932-2006)

In October 2006 the very sad news arrived that Prof. Len Dekker has passed away. Len Dekker is the founder of the Dutch Benelux Simulation Society (DBSS) and has served as the chairman of DBSS for a long time, and he is a co-founder and a former president of EUROSIM.



Prof. Len Dekker was full professor at the Faculty of Applied Mathematics and the Faculty of Applied Physics at the Delft University of Technology. All through his life he had devoted his research to advancing the computing systems for system simulation. His interest in computer simulation began at the end of 1960s, by that time computer was still in its infancy, the most common applications were doing some calculations and counting in administration. Computers by then had very limited memory (a few Kbytes) and were very slow compare to today's standard. Prof. Len Dekker and his research group at Delft University were fascinated by the challenge of increasing the processing speed of the computing systems. In the earlier 1970s Len Dekker has designed and built a hybrid (analog-digital) computer for simulation of dynamic systems modelled by a set of differential equations. The analog system comprises a number of 'integrators' realised with analog amplifiers. This hybrid computer system has been installed at the computing centre in the early 70s and has served as a powerful simulation tool.

The hybrid computer system is in fact one of the earliest parallel computers. 'Parallel processing' has been the research focus of Len Dekker for more than three decades. After the completion of the hybrid computer system, he and his research group started to design and to implement the Delft Parallel Processor (DPP). The first DPP with 8 processors become operational in the mid-1970, unlike the hybrid system the DPP comprises only digital processing units. It was one of the first parallel computers by that time.

Len Dekker was a research pioneer in parallel processing. In the 80s and 90s computer technology has made a tremendous progress both in processor speed and memory storage that commercial computer manufacturers flood the market with parallel (super)computers. Len Dekker has turned his focus to parallel algorithms and parallelisation of large scale systems simulations. Len Dekker retired from Delft University of Technology in 1997.

Prof. Len Dekker has also devoted much of his time to the DBSS and has served for a long time as the chairman. He is also one of driving forces in setting up the Federation of European Simulation Societies (EUROSIM). Furthermore he has set up the journal Simulation Practice and Theory and served as the first editor-in-chief of the journal for many years. His efforts and devotion in these different areas have helped many people in the fields of simulation. We are indebted to his many contributions to the scientific community.

*Arnold W. Heemink*

## EUROSIM Congress.

The EUROSIM Congress is arranged every three years by a member society of EUROSIM. EUROSIM'04, the *5th EUROSIM Congress*, took place in Noisy-le-Grand, near Paris, France in Sept. 2004. The *6th EUROSIM Congress* will be organised by the Slovene Society for Simulation and Modelling SLOSIM in close cooperation with German simulation society ASIM and other simulation societies.

### EUROSIM 2007

#### 6th EUROSIM Congress

September 9-14, 2007, Ljubljana, Slovenia

[WWW.EUROSIM2007.ORG](http://WWW.EUROSIM2007.ORG)

*Ljubljana* - Your Host City. Ljubljana, the capital of Slovenia which is the member of the European Union, is the heart of the political, economic, cultural and scientific life of Slovene nation. It was build on the place of a Roman city Emona. Numerous churches, theatres, museums, galleries, the Medieval castle, give Ljubljana a reputation of being a modern and one of the



most beautiful towns in Europe. Especially impressive are some works of the famous architect Jože Plecnik.

For more information about EUROSIM 2007, please contact:

Prof. Borut Zupancic, chair of the congress

[borut.zupancic@fe.uni-lj.si](mailto:borut.zupancic@fe.uni-lj.si)

Prof. Rihard Karba, chair of the IPC

[rihard.karba@fe.uni-lj.si](mailto:rihard.karba@fe.uni-lj.si)



# ASIM



# ASIM



## ASIM - Buchreihen / ASIM Book Series

**Fortschritte in der Simulationstechnik (FS) / Series Frontiers in Simulation (FS)**

**- Monographs, Proceedings:**

W. Borutzky: *Bond Graphs Methodology for Modelling Multidisciplinary Dynamic Systems*. FS 14, ISBN 3-936150-33-8, 2005.

M. Becker, H. Szczerbicka (eds.): *19th Symposium Simulation Techniques*. Proceedings Tagung ASIM 2006, Hannover; FS 16, ISBN 3-936150-49-4, 2006.

S. Wenzel (Hrsg.): *12. Fachtagung Simulation in Produktion und Logistik*. Proceedings Tagung ASIM SPL 2006; ISBN 3-936150-48-6, 2006.

F. Hülsemann, M. Kowarschik; U. Rüdte: *18th Symposium Simulation Techniques*. Proceedings Tagung ASIM 2005 Erlangen; FS 15, ISBN 3-936150-41-, 2005.

*Available / Verfügbar:* SCS Publishing House e.V., Erlangen, [WWW.SCS-PUBLISHINGHOUSE.DE](http://WWW.SCS-PUBLISHINGHOUSE.DE)  
Download ASIM Website [WWW.ASIM-GI.ORG](http://WWW.ASIM-GI.ORG) (partly; for ASIM members)

**Fortschrittsberichte Simulation (FB) / Advances Simulation (AS) / ASIM Mitteilung (AM)**

**ARGESIM Reports (AR) - Special Monographs, PhD Theses, Workshop Proceedings**

C. Deatcu, S. Pawletta, T. Pawletta (eds.): *Modelling, Control and Simulation in Automotive and Process Automation*. Proceedings ASIM Workshop Wismar 2006, ARGESIM Report 31, AM 101; ISBN 3-901-608-31-1, 2006.

H. Ecker: *Suppression of Self-excited Vibrations in Mechanical Systems by Parametric Stiffness Excitation*. ARGESIM Report FB 11, ISBN 3-901-608-61-3, 2006.

M. Gyimesi: *Simulation Service Providing als Webservice zur Simulation Diskreter Prozesse*. ARGESIM Report FB 13, ISBN 3-901-608-63-X, 2006.

J. Wöckl: *Hybrider Modellbildungszugang für biologische Abwasserreinigungsprozesse*. ARGESIM Report FB 14, ISBN 3-901608-64-8, 2006.

Th. Löscher: *Optimisation of Scheduling Problems Based on Timed Petri Nets*. ARGESIM Report Vol. 15, ASIM / ARGESIM Vienna, 2007; ISBN 978-3-901608-65-0.

*Available / Verfügbar:* ARGESIM/ASIM Publisher, TU Vienna, [WWW.ARGESIM.ORG](http://WWW.ARGESIM.ORG)  
Download / Bestellung zum Mitgliederpreis € 10.- ASIM Website [WWW.ASIM-GI.ORG](http://WWW.ASIM-GI.ORG)

**Reihen der ASIM-Fachgruppen / Series of ASIM Working Groups**

S. Collisi-Böhmer, O. Rose, K. Weiß, S. Wenzel (Hrsg.): *Qualitätskriterien für die Simulation in Produktion und Logistik*. AMB 102, Springer, Heidelberg, 2006; ISBN 3-540-35272-4.

M. Rabe, S. Spiekermann, S. Wenzel (Hrsg.): *Verifikation und Validierung für die Simulation in Produktion und Logistik*. AMB 103, Springer, Heidelberg, 2006; ISBN 3-540-35281-3.

J. Wittmann, M. Müller (Hrsg.): *Simulation in Umwelt- und Geowissenschaften - Workshop Leipzig 2006*. Shaker Verlag, Aachen 2006, AM 106; ISBN 978-3-8322-5132-1.

A. Gnauck (Hrsg.): *Modellierung und Simulation von Ökosystemen - Workshop Kölpinsee 2006*. Shaker Verlag, Aachen 2007, AM 107; ISBN 978-3-8322-6058-3.

*Available / Verfügbar:* Bookstore / Buchhandlung, ermäßigter Bezug für ASIM Mitglieder  
Info at ASIM website [WWW.ASIM-GI.ORG](http://WWW.ASIM-GI.ORG)



REPORTS



REPORTS



SCS  
Publishing  
House



SCS  
Publishing  
House





*515.000.000 KM, 380.000 SIMULATIONEN  
UND KEIN EINZIGER TESTFLUG.*

*DAS IST MODEL-BASED DESIGN.*

*Nachdem der Endabstieg der beiden Mars Rover unter Tausenden von atmosphärischen Bedingungen simuliert wurde, entwickelte und testete das Ingenieur-Team ein ausfallsicheres Bremsraketen-System, um eine zuverlässige Landung zu garantieren. Das Resultat – zwei erfolgreiche autonome Landungen, die exakt gemäß der Simulation erfolgten. Mehr hierzu erfahren Sie unter: [www.mathworks.de/mbd](http://www.mathworks.de/mbd)*

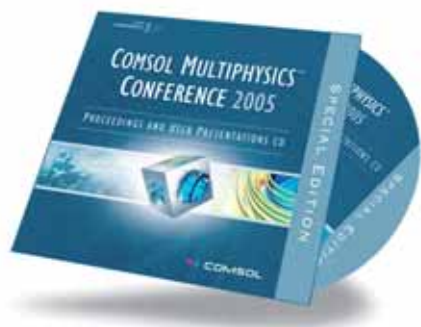
**MATLAB<sup>®</sup>  
& SIMULINK<sup>®</sup>**

# Proceedings CD der Konferenz zur Multiphysik-Simulation

## ANWENDUNGSBEREICHE:

- Akustik und Fluid-Struktur-Interaktion
- Brennstoffzellen
- Chemietechnologie und Biotechnologie
- COMSOL Multiphysics™ in der Lehre
- Elektromagnetische Wellen
- Geowissenschaften
- Grundlegende Analysen, Optimierung, numerische Methoden
- Halbleiter
- Mikrosystemtechnik
- Statische und quasi-statische Elektromagnetik
- Strömungssimulation
- Strukturmechanik
- Wärmetransport

Bestellen Sie hier Ihre kostenlose Proceedings CD mit Vorträgen, Präsentationen und Beispielmolellen zur Multiphysik-Simulation:



TITEL, NACHNAME \_\_\_\_\_

VORNAME \_\_\_\_\_

FIRMA / UNIVERSITÄT \_\_\_\_\_

ABTEILUNG \_\_\_\_\_

ADRESSE \_\_\_\_\_

PLZ, ORT \_\_\_\_\_

TELEFON, FAX \_\_\_\_\_

EMAIL \_\_\_\_\_