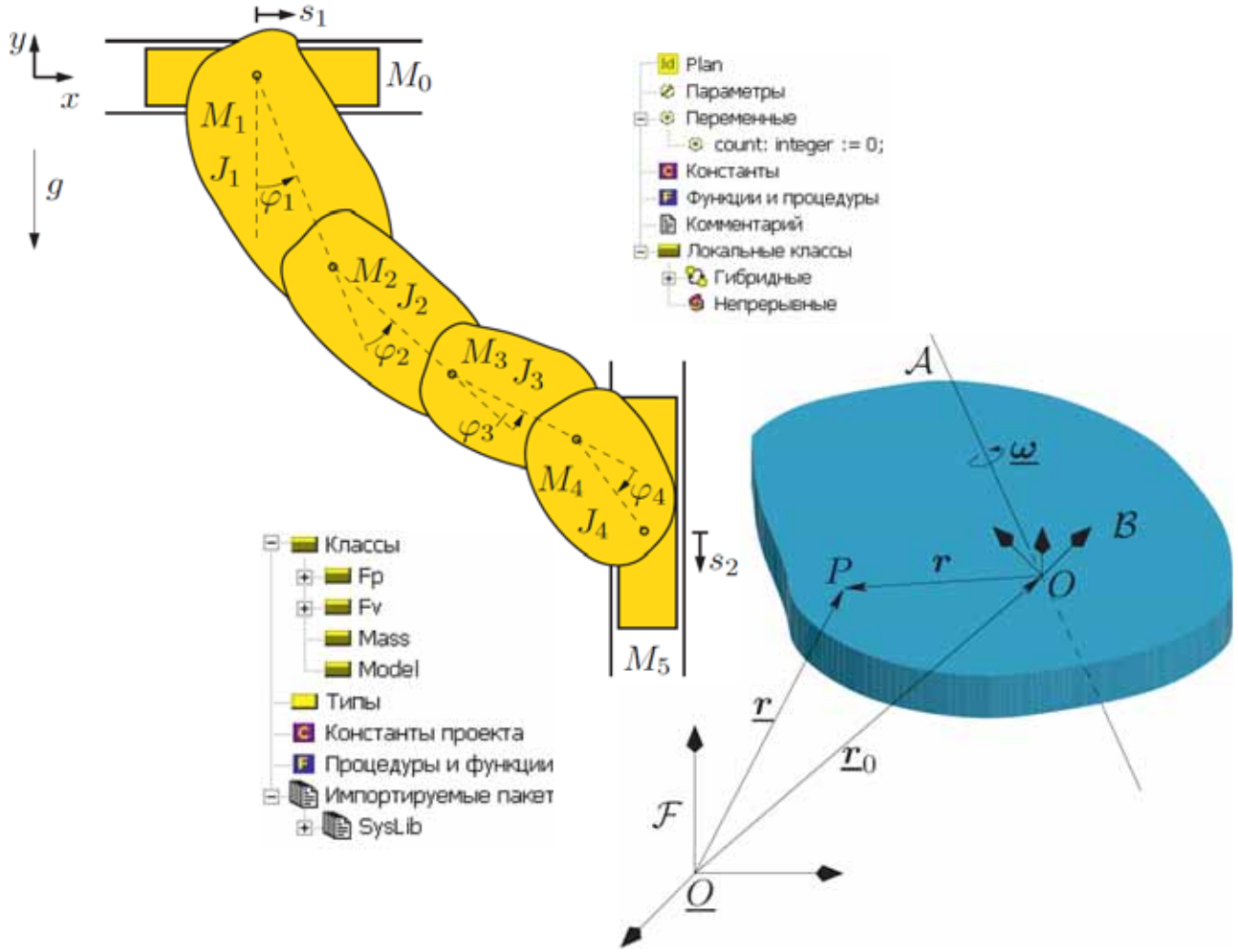
SNE SIMULATION NEWS EUROPE



Volume 20 Number 1
April 2010

doi: 10.11128/sne.20.1.0996
DOI Reprint ISSN 0929-2268

Print ISSN 2305-9974
Online ISSN 2306-0271

Journal on Developments and Trends in Modelling and Simulation

Membership Journal for Simulation Societies in EUROSIM

ARGESIM

Dear Readers,

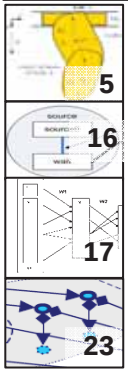
This SNE issue SNE 20/1 starts the 20th anniversary of SNE – an occasion to look back and to look in the future. The look back shows 'constants' (in the sense of 'invariables' or 'standard'), continuous developments, and short-living developments. The frame with a review of SNE title pages reflects SNE's development, from SNE 0 on in 1990 with the first layout – as EUROSIM newsletter with some technical notes. Soon SNE was faced with an almost immortal case study – Dining Philosophers – SNE 3, 1991 – and in further issues – a basis for SNE's successful series 'Comparisons / Benchmarks for Simulation Technique' - a real constant. In 1993, SNE got a new layout and was printed together with SIMPRA – and in SNE 11 we were proud to announce a Gopher Server with SNE content – both short-living developments. The EUROSIM Congress 1195 in Vienna – announced at title page of SNE 12 – encouraged SNE to plan a development towards a scientific journal, followed by SNE at the WWW – introduced in SNE 13, 1995 - both continuous developments. In SNE 14, also in 1995, we could publish the 100th comparison note, and now in 2010 we count 340 benchmark note – a continuous development, already a constant. SNE 29/30 – with new layout - introduced SNE's new structure with Technical Notes, Short Notes, Comparison Notes, etc, and with separation of the News Section. In SNE 34 readers met at the title page another almost 'immortal' case study – Bouncing Ball – dealt with in several issues in notes and comparisons and benchmarks – another real constant. In 2006, with SNE 46 – SNE 16/1, we completed the SNE re-arrangement towards a scientific journal, with increasing editorial board, peer review, and structure for SNE contributions, called notes; and also in 2006, we introduced SNE Special Issues, the first SNE 16/2 on 'Parallel and Distributed Simulation Methods', followed by special issues SNE 17/2 and SNE 18/2 on 'Structural Dynamic Modelling', and SNE 19/2 on 'Quality Aspects', and as next special issue SNE 20/2 on 'Simulation & Education' – continuous development and almost a new constant. The title page of the current issue SNE 20/1 – physical modelling and tools with Cyrillic labels - should underline the increasing internationality of SNE contributions – continuous development for the future.

But there is still another important constant: SNE is the ribbon, which unites and ties EUROSIM together – in past and in future. EiC and Editorial Staff would like to thank gratefully all, who have helped SNE to become a story of success – thank you !

Felix Breitenecker, editor-in-chief, eic@sne-journal.org

SNE 20/1 in Three Minutes

Impressum, Table of Contents: page 4

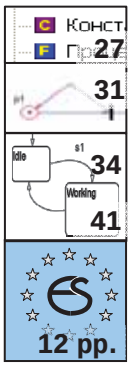


Towards an Object-oriented Implementation of von Mises' Motor Calculus - evaluates Modelica Modelling

Concepts and Architecture of the IBKSim Network Simulator - XML meets Simulation

A New Technique for Interactive Simulation of Recurrent Neural Networks – presents an efficient method for simulating neural nets.

Development of Sustainable Modelling and Simulation Tools – Experiences and Challenges



Modelling Hybrid Systems in MVStudium – introduces a simulator from St. Petersburg

QS-PN Simulation Tool – Petri Nets for Queuing Systems

A Comparative Case Study of Flexsim, Dymola and Matlab/Stateflow evaluates C2 'Flexible Assembly System'

C6 'Emergency Dept./ SimEvents

EUROSIM short info - overview on EUROSIM and EUROSIM societies; *Reports* from ASIM, SLOSIM, SIMS, PSC and from simulation activities in former Yugoslavia and in Kosovo

ASIM - Buchreihen / ASIM Book Series

Fortschritte in der Simulationstechnik (FS) / Frontiers in Simulation (FS) Proceedings Conferences - Monographs, Proceedings:

I. Troch, F. Breitenecker (eds): *Proceedings MATHMOD 09 - Abstract Volume / Full Papers CD Volume*. Proc. 6th Conference Mathematical Modelling Vienna, February 2009, Vienna; ARGESIM Reports no. 34 & no. 35, ASIM / ARGESIM Vienna, 2009; ISBN 978-3-901608-34-6, ISBN 978-3-901608-35-3

M. Rabe (ed.): *Advances in Simulation for Production and Logistics Applications*. Proc. 13. ASIM-Fachtagung Simulation in Produktion und Logistik, October 2008, Berlin; Fraunhofer IRB-Verlag, Stuttgart, 2008, ISBN 978-3-8167-7798-4.

B. Zupančič, R. Karba, S. Blažič (eds.): *Proceedings EUROSIM 2007 - Abstract Volume / CD Volume*. Proc. 6th EUROSIM Congress on Modelling and Simulation, Sept. 2007, Ljubljana, Slovenia. ARGESIM Report no. 32, ASIM / ARGESIM Vienna, 2007; ISBN 978-3-901608-32-2.

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.

W. Borutzky: *Bond Graphs Methodology for Modelling Multidisciplinary Dynamic Systems*. FS 14, ISBN 3-936150-33-8, 2005.

Fortschrittsberichte Simulation (FB) - ARGESIM Reports (AR) - Special Monographs, PhD Theses, ASIM Workshop Proceedings

D. Leitner: *Simulation of Arterial Blood Flow with the Lattice Boltzmann Method* ARGESIM Report 16, ASIM / ARGESIM Vienna, 2009; ISBN 978-3-901608-66-7.

Th. Löscher: *Optimisation of Scheduling Problems Based on Timed Petri Nets*. ARGESIM Report 15, ASIM / ARGESIM Vienna, 2009; ISBN 978-3-901608-65-0.

R. Fink: *Untersuchungen zur Parallelverarbeitung mit wissenschaftlich-technischen Berechnungsumgebungen*. ARGESIM Report 12, ASIM / ARGESIM Vienna, 2008; ISBN 978-3-901608-62-9.

M. Gyimesi: *Simulation Service Providing als Webservice zur Simulation Diskreter Prozesse*. ARGESIM Report 13, ASIM / ARGESIM Vienna, ISBN 3-901-608-63-X, 2006.

J. Wöckl: *Hybrider Modellbildungszugang für biologische Abwasserreinigungsprozesse*. ARGESIM Report 14, ASIM / ARGESIM Vienna, ISBN 3-901608-64-8, 2006.

H. Ecker: *Suppression of Self-excited Vibrations in Mechanical Systems by Parametric Stiffness Excitation*. ARGESIM Report 11, ISBN 3-901-608-61-3, 2006.

C. Deatcu, P. Dünow, T. Pawletta, S. Pawletta (eds.): Proceedings 4. ASIM-Workshop Wismar 2008 – Modellierung, Regelung und Simulation in Automotive und Prozess-automation. ARGESIM Report 31, ASIM / ARGESIM Vienna, ISBN 978-3-901608-31-5, 2008.

J. Wittmann, H.-P. Bader (Hrsg.): *Simulation in Umwelt- und Geowissenschaften - Workshop Dübendorf 2008*. Shaker Verlag, Aachen 2008, ISBN 978-3-8322-7252-4.

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: ASIM / ARGESIM Publisher Vienna - WWW.ASIM-GI.ORG
SCS Publishing House e.V., Erlangen, WWW.SCS-PUBLISHINGHOUSE.DE
Download ASIM Website WWW.ASIM-GI.ORG (partly; for ASIM members),
Bookstores / Buchhandlung, tw. ermäßigter Bezug für ASIM Mitglieder WWW.ASIM-GI.ORG

Table of Contents

SNE 20(1) Reprint doi: 10.11128/sne.20.1.0996

SNE Editorial Board

SNE - *Simulation News Europe* (SNE) is advised and supervised by an international editorial board. This board is taking care on peer reviewing and handling of *Technical Notes*, *Education Notes*, *Short Notes*, *Software Notes*, and of *Benchmark Notes* (definitions and solutions). Work of the board is supported by a new SNE Contribution-, Management-, and Reviewing System via the new SNE website. At present, the board is increasing:

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
office@sne-journal.org; www.sne-journal.org

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 bechmarks for modelling and simulation, membership information for EUROSIM and Simulation Societies.

Editor-in-Chief: Felix Breitenecker, Inst. f. Analysis and Scientific Computing, Vienna University of Technology, Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria; Felix.Breitenecker@tuwien.ac.at

Layout: Markus Wallerberger, markus.wallerberger@gmx.at

Administration, Web: Anna Mathe, anna.mathe@tuwien.ac.at

Printed by: Grafisches Zentrum, TU Vienna, Wiedner Hauptstrasse 8-10, 1040, Vienna, Austria

Publisher: ARGESIM/ASIM; c/o Inst. for Scientific Computing, TU Vienna, Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria, and ASIM (German Simulation Society), c/o Wohlfartstr. 21b, 80939 Munchen;

Towards an Object-oriented Implementation of von Mises' Motor Calculus Using Modelica

Tobias Zaiczek, Olaf Enge-Rosenblatt, Fraunhofer Institute for Integrated Circuits, Dresden, Germany

{Tobias.Zaiczek, Olaf.Enge}@eas.iis.fraunhofer.de

SNE Simulation Notes Europe SNE 20(1), 2010, 5-15, doi: 10.11128/sne.20.tn.09961

This paper deals with a first implementation of the socalled motor calculus within Modelica. The motor calculus can be used to describe the dynamical behaviour of spatial multibody systems in an efficient way. This method represents an alternative approach to modelling of multibody systems. In the paper, some fundamentals of motor calculus are summarized. Furthermore, a simple implementation of motor algebra by special additional Modelica code within some components of the Modelica Multibody Standard Library is presented. This approach fully corresponds with the paradigm of object-oriented modelling. However, the present realisation is not equation-based in its full sense because of the missing possibility of operator overloading (at least in the available Modelica simulator environment). Instead of this, some functions are used carrying out the necessary calculations. Using this implementation, some examples are given to prove the applicability and correctness of the implemented approach.

Introduction

The notion of motor, composed of the words *mo*ment and ro*tor*, was coined by Clifford in 1873 in his algebra of biquaternions [4]. But Clifford did apply his concept neither to the modelling of motion of a single rigid body nor to the modelling of spatial multibody systems. The approach of motor calculus to 3D mechanics was suggested by von Mises in 1924 [11, 12]. In the first part [11], von Mises introduces the dual motor product. He indicates the role of the dual motor product as a measure of the instantaneous change of a motor associated to a rigid body by the action of a second motor. In the second part [12], von Mises applied the motor calculus in the derivation of a general form of the equations of motion of a rigid body. Due to this work, translations and rotations, velocities and angular velocities, forces and torques, etc. can be described by motor calculus (or motor algebra). Hence, this approach is well suited to investigate the behaviour of spatial multibody systems.

One of the authors studied motor calculus in his Diploma thesis [22] initiated and supervised by Prof. K. Reinschke from the Technical University Dresden (one of the former institutes of R. von Mises). Recent publications dealing with this subject can rarely be found (exept e.g. for [8, 18]). In the context of the modelling language for heterogenous systems Modelica (see e.g. [5, 13, 19]), the motor calculus has not been taken into account up to now.

Within the Modelica community, spatial multibody systems are usually modelled using the Modelica Multibody Standard Library (see [14] or [15]). Meanwhile, many researchers apply this library to model different kinds of – partially very complex – multibody systems [2, 9, 10, 16, 20]. This library has proven to be a well suited resource to modelling such systems. However, applying the motor calculus, the equations of motion for a rigid body become more concise and clearer, e.g.

$$\dot{\mathfrak{p}} = \mathfrak{f} \tag{1}$$

($\mathfrak{p}$ – momentum motor, $\mathfrak{f}$ – force motor). Despite the formal equivalence to Newton's Second Law for a point mass, this equation fully describes the three-dimensional mechanics of a rigid body.

The motivation to follow up the motor calculus in the Modelica context is to investigate the possible simplification of handling spatial mechanical systems. A test realisation within the Modelica Multibody Standard Library has been carried out by implementing special additional Modelica code within some components of this library. These modifications take advantage of the built-in feature of inheritance. Hence, it is possible to compare both approaches e. g. with respect to numerical correctness.

In the following section, some fundamentals of motor calculus are shortly sketched. Some of the most important mathematical operations are defined. The test

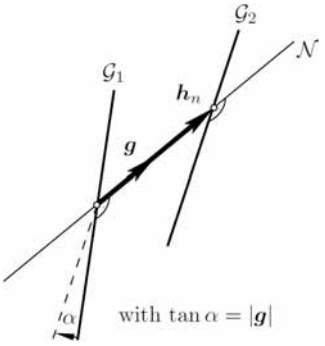


Figure 1. Geometrical interpretation of motors.

implementation is presented in Section 2. It fully corresponds with the paradigm of object-oriented modelling (see e.g. [3]). In modelling and simulation, one usually distinguishes between equations and assignments. In this context, the test implementation is not completely equation-based because some special mathematical operations had to be realised by functions. Some examples in section 3 show the principal applicability of the motor calculus approach.

1 Fundamentals of motor calculus

A motor

$$\mathfrak{h} = \begin{pmatrix} \boldsymbol{g} \\ \boldsymbol{h}_o \end{pmatrix} \tag{2}$$

is an ordered pair of vectors, $\boldsymbol{h}_o$ and $\boldsymbol{g}$, that define a vector field

$$\boldsymbol{h}(\boldsymbol{r}) = \boldsymbol{h}_o + \boldsymbol{g} \times \boldsymbol{r} \tag{3}$$

in the three-dimensional Euclidean space. In this definition, $\boldsymbol{r}$ is the position vector of any point in space, while the vectors $\boldsymbol{h}$ and $\boldsymbol{g}$ are called the *moment* and the *resultant vector* of the motor, respectively. Accordingly, $\boldsymbol{h}_o$ stands for the *moment* of the motor at the origin O of the reference coordinate system.

For every motor, an infinite number of points exists, for which the moment of the motor $\boldsymbol{h}$ is parallel to the resultant vector $\boldsymbol{g}$. All these points exhibit the same moment $\boldsymbol{h}_n$ and lie on a straight line $\mathcal{N}$ given by:

$$\boldsymbol{r}_n(\lambda) = \frac{\boldsymbol{g} \times \boldsymbol{h}_o}{|\boldsymbol{g}|^2} + \lambda \boldsymbol{g} \tag{4}$$

Geometrical interpretation A very strong goal of the motor calculus is the fact that motors and all operations with motors (that will be defined later on) can be interpreted as geometrical objects or constructions. Hence, all motors can be seen as abstract objects that do not depend on the choice of a reference frame. R. von Mises emphasises this fact by giving the definition of motors in terms of geometrical objects describing them. Here, just an interpretation of the foregoing definition is given.

For every pair of straight lines ($\mathcal{G}_1$ and $\mathcal{G}_2$) defined in Euclidean space, there exists a straight line $\mathcal{N}$ connecting them and being orthogonal to both of them (see Figure 1). For a pair of non-parallel lines, $\mathcal{N}$ is uniquely defined. Otherwise, there exists an infinite number of such connecting lines that are parallel to each other. Now, every ordered pair of straight lines ($\mathcal{G}_1$, $\mathcal{G}_2$) can be mapped to a motor (see Figure 1). In this case, $\mathcal{N}$ is denoted as *motor axis*, according to von Mises. The oriented segment of the axis $\mathcal{N}$ between the intersection with $\mathcal{G}_1$ and the intersection with $\mathcal{G}_2$ can be interpreted as the moment $\boldsymbol{h}_n$ of the motor on its axis. The smaller one of both angles included by the lines $\mathcal{G}_1$ and $\mathcal{G}_2$ is understood as a measure for the orientation and is simultaneously interpreted as the length of the resultant vector g. The tangent of this angle α is equal to the length of the resultant vector, while the direction of the resultant vector is defined in such a manner that $\mathcal{G}_1$ can be transferred into $\mathcal{G}_2$ by a mathematically positive screw motion across the resultant vector. The mapping from an ordered pair of straight lines to a motor is not a one-to-one mapping because all ordered pairs of straight lines that can be transferred into each other by a screw motion across $\mathcal{N}$ define the same motor.

1.1 Motor calculus

In the following, some computational rules of motor calculus are recalled.

Let $\mathfrak{h}$, $\mathfrak{h}_1$ and $\mathfrak{h}_2$ be three motors given by

$$\mathfrak{h} = \begin{pmatrix} \boldsymbol{g} \\ \boldsymbol{h}_o \end{pmatrix}, \quad \mathfrak{h}_1 = \begin{pmatrix} \boldsymbol{g}_1 \\ \boldsymbol{h}_{o1} \end{pmatrix}, \quad \mathfrak{h}_2 = \begin{pmatrix} \boldsymbol{g}_2 \\ \boldsymbol{h}_{o2} \end{pmatrix} \tag{5}$$

Then, according to von Mises, the following mathematical operations can be defined:

Addition

The addition of motors is performed component-wise according to:

$$\mathfrak{h}_1 + \mathfrak{h}_2 = \begin{pmatrix} \boldsymbol{g}_1 + \boldsymbol{g}_2 \\ \boldsymbol{h}_{o1} + \boldsymbol{h}_{o2} \end{pmatrix} \tag{6}$$

The neutral element of the addition is the zero motor

$$\mathfrak{o} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{0} \end{pmatrix} \tag{7}$$

Multiplication

For the multiplication of motors the following three cases can be distinguished:

Multiplication with a scalar The scalar multiplication is defined component-wise

$$\alpha\mathfrak{h} = \begin{pmatrix} \alpha\boldsymbol{g} \\ \alpha\boldsymbol{h}_o \end{pmatrix} \tag{8}$$

Inner product The result of the inner product of two motors is a scalar. Thus, the product corresponds to the scalar product of the vector calculus. The definition is

$$(\mathfrak{h}_1, \mathfrak{h}_2) = (\boldsymbol{g}_1, \boldsymbol{h}_{o2}) + (\boldsymbol{g}_2, \boldsymbol{h}_{o1}) \tag{9}$$

while $(\boldsymbol{g},\boldsymbol{h})$ is the scalar product of two vectors. Using matrix notation, the equation

$$(\mathfrak{h}_1, \mathfrak{h}_2) = \begin{pmatrix} \boldsymbol{g}_1^T & \boldsymbol{h}_{o1}^T \end{pmatrix} \Gamma \begin{pmatrix} \boldsymbol{g}_1 \\ \boldsymbol{h}_{o2} \end{pmatrix}$$

holds, where Γ is a well-chosen matrix according to:

$$\Gamma = \begin{pmatrix} \boldsymbol{0} & \boldsymbol{I}_3 \\ \boldsymbol{I}_3 & \boldsymbol{0} \end{pmatrix}$$

and $\boldsymbol{I}_3$ denotes the (3×3) identity matrix.

Outer product The outer product of two motors results in another motor, which is composed as follows:

$$\mathfrak{h}_1 \times \mathfrak{h}_2 = \begin{pmatrix} \boldsymbol{g}_1 \times \boldsymbol{g}_2 \\ \boldsymbol{g}_1 \times \boldsymbol{h}_{o1} + \boldsymbol{h}_{o2} \times \boldsymbol{g}_1 \end{pmatrix} \tag{10}$$

The outer product is also referred to as motorial product or as dual motor product [6]. In terms of vectors and vector dyads, the product can be written as

$$\mathfrak{h}_1 \times \mathfrak{h}_2 = \begin{pmatrix} \boldsymbol{0} & \boldsymbol{G}_1 \\ \boldsymbol{G}_1 & \boldsymbol{H}_{o1} \end{pmatrix} \Gamma \begin{pmatrix} \boldsymbol{g}_2 \\ \boldsymbol{h}_{o2} \end{pmatrix} \tag{11}$$

where $\boldsymbol{G}_1$ and $\boldsymbol{H}_{o1}$ are the cross product matrices of the vectors $\boldsymbol{g}_1$ and $\boldsymbol{h}_{o1}$, respectively.

Motor dyads

In analogy to the vector calculus, von Mises declared dyads for the motor calculus by linear vector functions mapping motors to motors. Referred to a concrete coordinate system, such a dyad can be represented as a (6×6) matrix.

The mapping can be described in the following manner:

$$\mathfrak{T} \circ \mathfrak{h}_1 = \begin{pmatrix} \boldsymbol{T}_{11} & \boldsymbol{T}_{12} \\ \boldsymbol{T}_{21} & \boldsymbol{T}_{22} \end{pmatrix} \Gamma \begin{pmatrix} \boldsymbol{g}_1 \\ \boldsymbol{h}_{o1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{T}_{11}\boldsymbol{h}_{o1} + \boldsymbol{T}_{12}\boldsymbol{g}_1 \\ \boldsymbol{T}_{21}\boldsymbol{h}_{o1} + \boldsymbol{T}_{22}\boldsymbol{g}_1 \end{pmatrix}. \tag{12}$$

The neutral element of the dyadic multiplication in motor calculus is the identity motor dyad $\mathfrak{G}$ that can be represented in every frame as

$$\mathfrak{G} = \begin{pmatrix} \boldsymbol{0} & \boldsymbol{I}_3 \\ \boldsymbol{I}_3 & \boldsymbol{0} \end{pmatrix} \tag{13}$$

Now, all calculation rules for the motor calculus can be derived readily, some of which are presented here for any arbitrarily chosen motors $\mathfrak{h}_1$, $\mathfrak{h}_2$, $\mathfrak{h}_3$ .and $\alpha \in \mathbb{R}$:

$$\begin{aligned}
(\mathfrak{h}_1, \mathfrak{h}_2) &= (\mathfrak{h}_2, \mathfrak{h}_1) \\
\mathfrak{h}_1 \times \mathfrak{h}_2 &= -\mathfrak{h}_2 \times \mathfrak{h}_1 \\
(\mathfrak{h}_1, (\mathfrak{h}_2 + \mathfrak{h}_3)) &= (\mathfrak{h}_1, \mathfrak{h}_2) + (\mathfrak{h}_1, \mathfrak{h}_3) \\
\mathfrak{h}_1 \times (\mathfrak{h}_2 + \mathfrak{h}_3) &= \mathfrak{h}_1 \times \mathfrak{h}_2 + \mathfrak{h}_1 \times \mathfrak{h}_3 \\
(\alpha\mathfrak{h}_1, \mathfrak{h}_2) &= \alpha(\mathfrak{h}_1, \mathfrak{h}_2) \\
\alpha\mathfrak{h}_1 \times \mathfrak{h}_2 &= \alpha(\mathfrak{h}_1 \times \mathfrak{h}_2) \\
(\mathfrak{h}_1, \mathfrak{h}_2 \times \mathfrak{h}_3) &= (\mathfrak{h}_2, \mathfrak{h}_3 \times \mathfrak{h}_1) = (\mathfrak{h}_3, \mathfrak{h}_1 \times \mathfrak{h}_2)
\end{aligned} \tag{14}$$

Remark: Due to the definition of addition and scalar multiplication, motors span a vector space over the field of real numbers. Moreover, by the introduction of the outer product, motors form a Lie-Algebra (Named after the mathematician Sophus Lie *1842, †1899) since all the following conditions are fulfilled:

1. The bilinearity of the motorial product is given, i. e., for all real α and β, the motors $\mathfrak{h}_1$, $\mathfrak{h}_2$, and $\mathfrak{h}_3$ satisfy the equations
   $$(\alpha\mathfrak{h}_1 + \beta\mathfrak{h}_2) \times \mathfrak{h}_3 = \alpha\mathfrak{h}_1 \times \mathfrak{h}_3 + \beta\mathfrak{h}_2 \times \mathfrak{h}_3$$
   and
   $$\mathfrak{h}_1 \times (\alpha\mathfrak{h}_2 + \beta\mathfrak{h}_3) = \alpha\mathfrak{h}_1 \times \mathfrak{h}_2 + \beta\mathfrak{h}_1 \times \mathfrak{h}_3$$
2. The motorial multiplication is skew commutative, i. e.,
   $$\mathfrak{h}_1 \times \mathfrak{h}_2 = -\mathfrak{h}_2 \times \mathfrak{h}_1$$
3. The Jacobian Identity holds, i. e. for arbitrarily chosen motors $\mathfrak{h}_1$, $\mathfrak{h}_2$, and $\mathfrak{h}_3$, the equation:
   $$\mathfrak{h}_1 \times (\mathfrak{h}_2 \times \mathfrak{h}_3) + \mathfrak{h}_2 \times (\mathfrak{h}_3 \times \mathfrak{h}_1) + \mathfrak{h}_3 \times (\mathfrak{h}_1 \times \mathfrak{h}_2) = 0$$
   is true.

Coordinate Transformations

For concrete calculations with motors, it is necessary to introduce a coordinate system, also called frame, in which the components of the motor are given. Considering two different frames $\mathcal{F}_1$ and $\mathcal{F}_2$, it may be of interest how to transform the components of a motor

$\mathfrak{h}$ given in frame $\mathcal{F}_1$ into the components referred to frame $\mathcal{F}_2$ and vice versa. So let vector r_{12} denote the position vector of the origin of $\mathcal{F}_2$ declared in frame $\mathcal{F}_1$. Furthermore, let the rotation from frame $\mathcal{F}_1$ to frame $\mathcal{F}_2$ be given by the direction cosine matrix A. Then, the transformation is performed by the equation

$$[\mathfrak{h}]_{\mathcal{F}_2} = \left[\begin{pmatrix} \boldsymbol{0} & \boldsymbol{A} \\ \boldsymbol{A} & -\boldsymbol{A}\boldsymbol{R}_{12} \end{pmatrix} \circ \mathfrak{h} \right]_{\mathcal{F}_1} \tag{15}$$

Here, the matrix $\boldsymbol{R}_{12}$ is the cross product matrix of the vector r_{12}.

Differentiation with respect to real-valued parameters

Consider a motor $\mathfrak{h}$ that depends on a real parameter t (e. g. the time). Then, the first derivative of this motor with respect to t can be computed componentwise:

$$\frac{d\mathfrak{h}}{dt} = \begin{pmatrix} \frac{d\boldsymbol{g}}{dt} & \frac{d\boldsymbol{h}_o}{dt} \end{pmatrix}^T \tag{16}$$

Differentiation in moving frames

The temporal change of a motor seen from two different frames will, in general, lead to differing results if one frame, say $\mathcal{F}_1$, moves relatively to the other frame, say $\mathcal{F}_0$. The relative motion of the origin of frame $\mathcal{F}_1$ measured in frame $\mathcal{F}_0$ shall be given by the velocity vector $\underline{\boldsymbol{v}}_0$, while the angular velocity vector of frame $\mathcal{F}_1$ with respect to frame $\mathcal{F}_0$ is denoted by $\underline{\boldsymbol{\omega}}$. Then, the equation

$$\dot{\mathfrak{h}} = \mathring{\mathfrak{h}} + \begin{pmatrix} \underline{\boldsymbol{\omega}} \\ \underline{\boldsymbol{v}}_0 \end{pmatrix} \times \mathfrak{h} \tag{17}$$

holds for the derivation with respect to time observed in frame $\mathcal{F}_0$. In Eq. (6), $\mathring{\mathfrak{h}}$ denotes the derivation w. r. t. time of the motor $\mathfrak{h}$ observed in frame $\mathcal{F}_1$.

1.2 Applications of motor calculus

The most important application of motor calculus is the description and analysis of the static and dynamic behaviour of rigid bodies subject to external forces and torques. Following the ideas of von Mises, the next paragraphs will give an overview, how to describe the rigid body movements in the three-dimensional space in a very effective way using the motor calculus.

Before that, some definitions have to be explained that are essential for the succeeding subsections. To describe the motion of a rigid body in three-dimensional space, one chooses a reference point O of the body. The motion of point O can be expressed w. r. t. a reference frame $\mathcal{F}$ by the position vector $\underline{\boldsymbol{r}}_0$ (see Figure 2). The origin of frame $\mathcal{F}$ is denoted by $\underline{O}$.

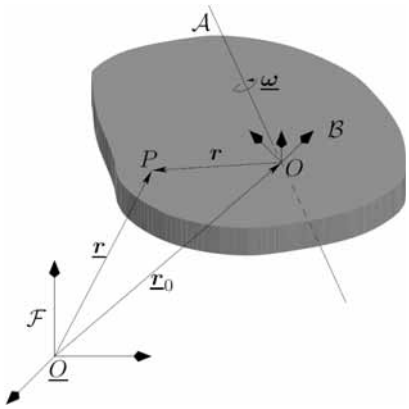


Figure 2. Definition of vectors at the rigid body.

For the description of all other points of the rigid body, it is suitable to introduce a body fixed frame, called body frame $\mathcal{B}$, with the origin located in the reference point O. To distinguish the position vectors of both frames, the position vectors of the inertial frame are underlined. The position of an arbitrarily chosen point P of the body is therefore given by

$$\underline{\boldsymbol{r}} = \underline{\boldsymbol{r}}_0 + \boldsymbol{r} \tag{18}$$

The motion of the rigid body is fully described by the velocity of the reference point $\underline{\boldsymbol{v}}_0 = \dot{\underline{\boldsymbol{r}}}_0$ and the angular velocity $\underline{\boldsymbol{\omega}}$ the body frame $\mathcal{B}$ is rotating w. r. t. $\mathcal{I}$.

Definition of physically motivated motors

The introduction of motor calculus is justified by the comfortable applicability to mechanical rigid body issues in three-dimensional space. As already described before, some physical quantities for the description of rigid body movements can be composed to motors. Hence, the motion laws of rigid body mechanics can be written in a very compact and clear form. This will be shown in the subsequent paragraphs.

We introduce some motors that are able to describe the motion sequence of a rigid body as well as the acting torques and forces in a physically meaningful manner.

The first motor is called the force motor f combining the resulting force $\boldsymbol{f}$ and torque $\boldsymbol{d}_o$ (referred to the reference point O) acting on the rigid body, i. e.

$$\mathfrak{f} = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{d}_o \end{pmatrix} \tag{19}$$

For any rigid body, every single force $\boldsymbol{f}_i$ and torque $\boldsymbol{d}_j$ can be assigned to a force motor according to

$$\mathfrak{f}_{f,i} = \begin{pmatrix} \boldsymbol{f}_i \\ \boldsymbol{r}_i \times \boldsymbol{f}_i \end{pmatrix} \quad \text{and} \quad \mathfrak{f}_{d,j} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{d}_j \end{pmatrix} \tag{20}$$

respectively. The resulting force motor can then be simply calculated as the sum of all single force motors

$$\mathfrak{f} = \sum_{(i)} \mathfrak{f}_{f,i} + \sum_{(j)} \mathfrak{f}_{d,j} \tag{21}$$

Please note that the overall torque do as well as the representation of the motor depend upon the chosen reference point O. Hence, the torque referred to any other point with the position vector r is calculated by

$$\boldsymbol{d}(\boldsymbol{r}) = \boldsymbol{d}_o + \boldsymbol{f} \times \boldsymbol{r} \tag{22}$$

That is exactly the relationship stated in Eq. (1). This characteristic can be interpreted as a force screw (see e.g. [1]), since there always exists an instantaneous line on which the force and the torque vectors act parallel. A second motor, the so-called velocity motor, is able to describe the whole motion of a rigid body. It consists of the velocity vector $\underline{\boldsymbol{v}}_0$ of the chosen reference point O and the angular velocity vector $\underline{\boldsymbol{\omega}}$ representing the rotation of the body w. r. t. an inertial frame:

$$\mathfrak{v} = \begin{pmatrix} \underline{\boldsymbol{\omega}} \\ \underline{\boldsymbol{v}}_0 \end{pmatrix} \tag{23}$$

This motor is able to describe the velocity $\underline{\boldsymbol{v}}$ of any point $\boldsymbol{r}$ of the rigid body by the equation

$$\boldsymbol{v}(\boldsymbol{r}) = \boldsymbol{v}_o + \boldsymbol{\omega} \times \boldsymbol{r} \tag{24}$$

Two other important vectors in the description of dynamic mechanical systems are the momentum vector $\boldsymbol{p}$ and the angular momentum vector $\boldsymbol{l}_o$. Both are combined in the momentum motor $\mathfrak{p}$ with

$$\mathfrak{p} = \begin{pmatrix} \boldsymbol{p} \\ \boldsymbol{l}_o \end{pmatrix} \tag{25}$$

Similar to the force motor, the representation of momentum motor depends upon the chosen reference point. Between the angular momentum $\boldsymbol{l}_o$ referred to O and the angular momentum vector $\boldsymbol{l}(\boldsymbol{r})$ referred to any other point at position $\boldsymbol{r}$, the relationship

$$\boldsymbol{l}(\boldsymbol{r}) = \boldsymbol{l}_o + \boldsymbol{p} \times \boldsymbol{r} \tag{26}$$

holds. This statement can be proven by using the definition of the vectors $\boldsymbol{p}$ and $\boldsymbol{l}_o$ according to

$$\begin{aligned} \boldsymbol{p} &= \int \underline{\dot{\boldsymbol{r}}}\, dm = \dot{\boldsymbol{r}}_o \int dm + \underline{\boldsymbol{\omega}} \times \int \boldsymbol{r}\, dm \\ &= m\dot{\boldsymbol{r}}_o - m\boldsymbol{r}_s \times \underline{\boldsymbol{\omega}} = m\boldsymbol{v}_o - m\boldsymbol{r}_s \times \underline{\boldsymbol{\omega}} \end{aligned} \tag{27}$$

$$\begin{aligned} \boldsymbol{l}_o &= \int \boldsymbol{r} \times \underline{\dot{\boldsymbol{r}}}\, dm = \int \boldsymbol{r}\, dm \times \underline{\dot{\boldsymbol{r}}}_o + \int \boldsymbol{r} \times (\underline{\boldsymbol{\omega}} \times \boldsymbol{r})\, dm \\ &= m\boldsymbol{r}_s \times \underline{\boldsymbol{v}}_o + \boldsymbol{\Theta}_o\, \underline{\boldsymbol{\omega}} \end{aligned} \tag{28}$$

where m denotes the mass of the body and $\boldsymbol{\Theta}_o$ the inertia tensor w. r. t. the reference point O. The vector $\boldsymbol{r}_s$ is the position vector of the centre of mass referred to the body frame given by $\boldsymbol{r}_s = \int \boldsymbol{r}\, dm / \int dm$.

Some fundamental laws of mechanics in terms of motor calculus

With the definitions above, a relationship between the velocity motor $\mathfrak{v}$ and the momentum motor $\mathfrak{p}$ can be derived by introducing the inertia dyad $\mathfrak{M}$ for the motor calculus:

$$\mathfrak{p} = \mathfrak{M} \circ \mathfrak{v} = \begin{pmatrix} m\boldsymbol{I} & -m\boldsymbol{R}_s \\ m\boldsymbol{R}_s & \boldsymbol{\Theta}_o \end{pmatrix} \circ \mathfrak{v} \tag{29}$$

The new symbol $\boldsymbol{R}_s$ describes the cross product dyad of the vector $\boldsymbol{r}_s$.

Referred to a concrete frame in u, v and w, the dyad can be written as a (6×6) matrix of the following form:

$$\mathfrak{M} = \begin{pmatrix} m & 0 & 0 & 0 & mw_s & -mv_s \\ 0 & m & 0 & -mw_s & 0 & mu_s \\ 0 & 0 & m & mv_s & -mu_s & 0 \\ 0 & -mw_s & mv_s & \Theta_{uu} & \Theta_{uv} & \Theta_{uw} \\ mw_s & 0 & -mu_s & \Theta_{vu} & \Theta_{vv} & \Theta_{vw} \\ -mv_s & mu_s & 0 & \Theta_{wu} & \Theta_{wv} & \Theta_{ww} \end{pmatrix}$$

where u_s, v_s, and w_s are the coordinates of centre of mass. Choosing the body frame parallel to the body's principal axes of inertia and selecting the centre of mass as the reference point, $\mathfrak{M}$ becomes a diagonal matrix. With the help of the foregoing motor relations, the main mechanical laws can be rewritten in terms of motors.

The first law describes the change of momentum and angular momentum in the presence of external forces and torques in a very efficient and short way, namely

$$\dot{\mathfrak{p}} = \mathfrak{f}$$

Here, $\dot{\mathfrak{p}}$ denotes the time derivative of the momentum motor $\mathfrak{p}$ observed in an *inertially fixed* reference frame.

The unique simplicity and shortness of this equation is doubtless a goal of this calculus, even more considering that it formally takes exactly the form of Newton's Second Law for mass points. Unfortunately, this formula is not very practical, since the derivation has to be done w. r. t. the inertial frame. However, the

momentum motor is much easier to determine in a body fixed frame, because the inertia dyad $\mathfrak{M}$ is therein constant. So, a much more applicable form for concrete calculations can be derived using (6) to express the time derivation w. r. t. the body frame

$$\mathring{\mathfrak{p}} + \mathfrak{v} \times \mathfrak{p} = \mathfrak{f} \tag{30}$$

where $\mathfrak{p}$, $\mathfrak{v}$, and $\mathfrak{f}$ are referred to the origin of the body frame.

Replacement of the momentum motor using Eq. (7) yields the following relationship:

$$\mathfrak{M} \circ \mathring{\mathfrak{v}} + \mathfrak{v} \times (\mathfrak{M} \circ \mathfrak{v}) = \mathfrak{f} \tag{31}$$

The kinetic energy of a rigid body can be expressed by means of motor calculus as follows:

$$T = \frac{1}{2}(\mathfrak{v}, \mathfrak{p}) \quad \text{with} \quad \mathfrak{p} = \mathfrak{M} \circ \mathfrak{v} \tag{32}$$

Again, this expression agrees formally with the equation of the kinetic energy of a mass point, if therein the mass is substituted by the inertia dyad $\mathfrak{M}$ and the vectors are substituted by their corresponding motors.

Similarly, the equation for the power performed by the applied forces and torques is given by

$$P = (\mathfrak{f}, \mathfrak{v}) \tag{33}$$

so that the energy law for a rigid body results in

$$\frac{dT}{dt} = \frac{1}{2}\frac{d}{dt}(\mathfrak{v}, \mathfrak{M} \circ \mathfrak{v}) = (\mathfrak{f}, \mathfrak{v}) \tag{34}$$

2 Object-oriented implementation

The test implementation presented here is based on the Modelica Multibody Standard Library. Hence, it fully corresponds with the paradigm of object-oriented modelling. Due to some limitations of the Modelica language, compromises had to be made during implementation of the motor calculus. Because of the necessarily used functions, the realisation is not a completely equation-based formulation.

2.1 Motor library

The first step of the implementation towards a description of rigid body motion by means of motor calculus is the realisation of a general motor class. From the view of data structure, motors are nothing more than a combination of six scalars. According to the definition of motors provided above, the first idea of arranging these scalars within the motor class was to group them into two vectors of type `Real`. The first vector would represent the resultant vector and the second vector would be the moment vector of the motor at the reference point:

```
record Motor "Motor"
  Real[3] res "resultant vector";
  Real[3] mom "moment vector at O";
end Motor;
```

Unfortunately, this approach, similar to the implementation of the complex numbers in [7], prohibits the use of basic mathematical operators on the newly defined data types. A solution would be the overloading of these operators as it is possible in C++ [17]. However, Modelica does still not support this feature. Thus, an alternative implementation has been chosen, where all six scalars are stored within one vector:

```
type Motor = Real[6]
       "Motor: [Resultant;Moment at r0]";
```

The reason for the chosen implementation was the ability to keep at least the operators "+" and "–" as well as the multiplication with scalars for the motor calculus in its original sense. Within the context of inheritance, no real specialization concerning the physical units of the quantities can be made. Hence, the child classes of velocity motor, force motor, and momentum motor have also a quite simple definition, namely:

```
type VelocityMotor = Motor  "Velocity motor";
type ForceMotor    = Motor  "Force motor";
type MomentumMotor = Motor  "Momentum motor";
```

All the other calculation rules introduced in section 1.1 had to be implemented using Modelica functions. The first function has been written to perform the inner product between two motors according to Eq. (2):

```
function dot "Inner product of motor calculus"
  input Motor m1 "First motor";
  input Motor m2 "Second motor";
  output Real r3 "Resulting scalar";
  algorithm
     r3 := m1[1:3]*m2[4:6] + m1[4:6]*m2[1:3];
end dot;
```

Similarly, the outer product has been implemented as stated in Eq. (3):

```
function 'x' "Outer product of motor calculus"
  input  Motor m1 "First motor";
  input  Motor m2 "Second motor";
  output Motor m3 "Resulting motor";
  algorithm
     m3 := vector([cross(m1[1:3],m2[1:3]);
                   cross(m1[1:3],m2[4:6])
                  +cross(m1[4:6],m2[1:3])]);
end 'x';
```

A function that returns the moment of the motor for any position vector $\boldsymbol{r}$ has also been realised to simplify the motor handling:

```
function mom "Moment of the motor referred to position r"
  input Motor m       "Motor";
  input Modelica.SIunits.Position[3] r
                      "Position vector";
  output Real[3] mom "Moment of the motor";
  algorithm
     mom := m[4:6] + cross(m[1:3],r);
end mom;
```

The foregoing reasons for the simple implementation of the motor class apply for the implementation of the motor dyads, too. Hence, a motor dyad given w. r. t. a given frame can be expressed as a (6×6) matrix:

```
type MotorDyad = Real[6,6] "Motor Dyad";
```

To apply a motor dyad to a motor, another function has been created. Referring to Eq. (4), the function has been defined by:

```
function times "Application of a Motor Dyad on Motor"
  input  MotorDyad m1 "Motor dyad to be applied";
  input  Motor m2     "Input motor";
  output Motor m3     "Output motor";
algorithm
  m3 := m1[:,1:3]*m2[4:6] + m1[:,4:6]*m2[1:3];
end times;
```

Finally, there exist two functions that are able to transform the components of a motor from one frame to another and vice versa (refer to section 2.1.4):

```
function coordChange1
           "Transforms motor from frame a to frame b"
import F= Modelica.Mechanics.MultiBody.Frames;
  input Modelica.SIunits.Position[3] r_0
           "Vector pointing from origin of frame a to
            origin of frame b, resolved in frame a";
  input F.Orientation R
        "Orientation object of frame b resolved in frame a";
  input Motor m1    "Motor resolved in frame a";
  output Motor m2   "Motor resolved in frame b";
algorithm
  m2 := vector([R.T*m1[1:3];R.T*mom(m1,r_0)]);
end coordChange1;

function coordChange2
           "Transforms motor from frame b to frame a"
import F= Modelica.Mechanics.MultiBody.Frames;
  input Modelica.SIunits.Position[3] r_0
           "Vector pointing from origin of frame a to
            origin of frame b, resolved in frame a";
  input F.Orientation R
        "Orientation object of frame b resolved in frame a";
  input Motor m1    "Motor resolved in frame a";
  output Motor m2   "Motor resolved in frame b";
algorithm
  m2 := vector([transpose(R.T)*m1[1:3];
          transpose(R.T)*m1[4:6]
        + cross(r_0,transpose(R.T)*m1[1:3])]);
end coordChange2;
```

2.2 Multibody implementation

After implementing the most important operations of the motor calculus, we were able to take advantage of the efficient description of the rigid body motion. Therefore, as a first step, the existing implementation of a rigid body object from the Modelica Multibody Standard Library was adapted to the motor algebra. To simplify the implementation, all interfaces and all existing variables were kept. Only some small changes had to be made within the so-called `Body` class. The first changes were the declaration of the following physically motivated `Motor` and `MotorDyad` objects:

```
// Motor Dyads
Real[3,3]   I0       "Inertia dyad wrt. B";
MotorDyad   I_mot    "Motorial inertia dyad wrt. B";

// Motors
VelocityMotor vel_B "Velocity motor wrt. B";
MomentumMotor mom   "Momentum motor wrt. B";
ForceMotor    f_g   "Gravity force motor wrt B";
ForceMotor    f_a   "Cut force motor wrt. B";
```

Afterwards, all declared motors and motor dyads had to be defined using the following statements:

```
// force motors
f_g = vector([ m*frame_a.R.T*g_0;
             cross(r_CM, m*frame_a.R.T*g_0)]);
f_a = vector([frame_a.f; frame_a.t]);

// velocity motor
vel_B = vector([ frame_a.R.w;
                 frame_a.R.T*der(frame_a.r_0)]);

// inertia matrices
I0 = I + m*(diagonal(r_CM*r_CM*ones(3))
               - [r_CM]*transpose([r_CM]));
I_mot = [diagonal({m, m, m}), -skew(m*r_CM);
               skew(m*r_CM) , I0];

// momentum motor
mom = vector(times(I_mot, vel_B));
```

Finally, the equations of motion originally implemented according to

```
frame_a.f = m*(Frames.resolve2(frame_a.R,
                                a_0 - g_0)
     + cross(z_a, r_CM)
     + cross(w_a, cross(w_a, r_CM)));
frame_a.t = I*z_a + cross(w_a, I*w_a)
     + cross(r_CM, frame_a.f);
```

have been replaced by the very clear and short Eq. (8):

```
f_a = der(mom) + 'x'(vel_B,mom) - f_g;
```

Because of the object-oriented structure of the Modelica Standard Library, the changes had to be imple-

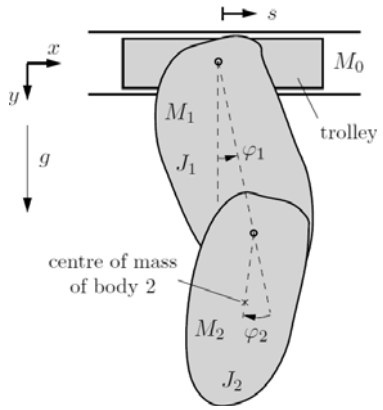


Figure 3. Sketch of double pendulum.

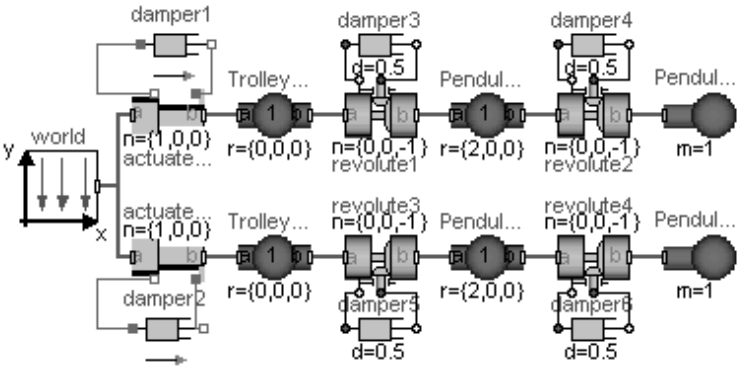


Figure 5. Implementation of the double pendulum.

mented only once. All subclasses of the `Body` class, like `BodyShape`, `BodyBox`, or `BodyCylinder` inherit the changes automatically.

3 Examples and verification

As a first example, the movable double pendulum (Fig. 3) was chosen to show the correctness of the implemented body classes based on motor calculus. The pendulum consists of a trolley with the mass M_0 and two rigid bodies with masses M_1 and M_2. The trolley is able to move horizontally. The first body is suspended on the trolley by a revolute joint. The second body is suspended on the first body via a revolute joint, too. Both axes of rotation are parallel to the z-axis which lies perpendicular to the xy-plane (see Figure 3). The moments of inertia of both bodies around the axis of rotation w. r. t. their particular centre of mass are given by J_1 and J_2.

The pendulum moves from an initial deflection of $\varphi_1(0) = 90°$ and $\varphi_2(0) = 0°$ due to the earth's gravity field. A viscous friction, acting in every joint, damps the motion of the pendulum. As a reference, the same pendulum system has been implemented using the Modelica Standard Library. A sketch of the structure is shown in the lower part of Figure 5. The upper part of this figure shows the pendulum using the modified `Body` objects adapted to the motor calculus.

Figure 4 shows the trajectory for the positions of the trolley. Figures 6 and 7 depict the time histories of the revolute joint angles φ_1 and φ_2. In every diagram, the trajectory of both systems, the double pendulum using the motor calculus and the double pendulum using the Modelica Standard Library, were plotted together.

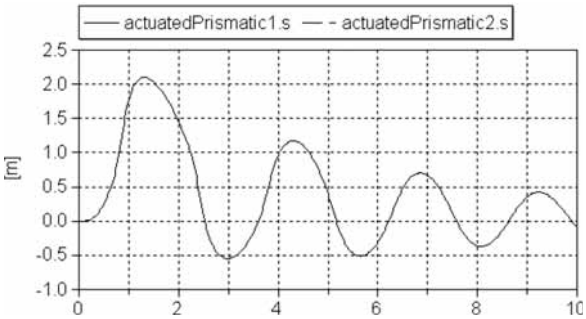


Figure 4. Trajectory of the trolley position s.

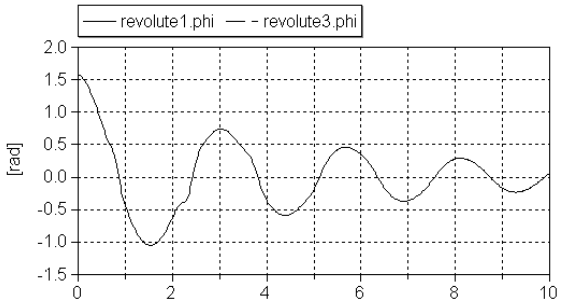


Figure 6. Trajectory of the first pendulum angle φ_1.

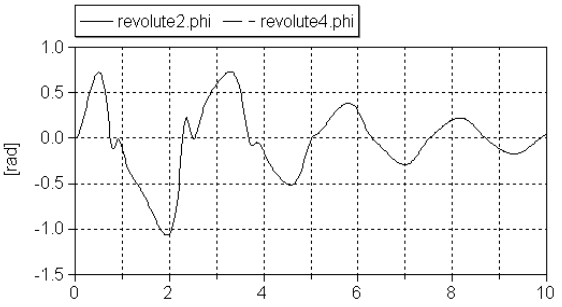


Figure 7. Trajectory of the first pendulum angle φ_2.

Figure 8 presents the deviation for all corresponding variables (`residue_s`, `residue_phi1/2`). Apparently, the deviation of the position stays smaller than $1.2 \cdot 10^{-11}\,\mathrm{m}$ for the given simulation time of 10s. The deviations of both pendulum angles are also very small. They do not exceed $10^{-11}\,\mathrm{rad}$. Hence, these differences can be interpreted as numerical errors of the simulator, because for simulations with a lower error tolerance, the deviations decrease. For a comparison even a third implementation within the simulation system Matlab (refer to [21]) was consulted that led to very similar results.

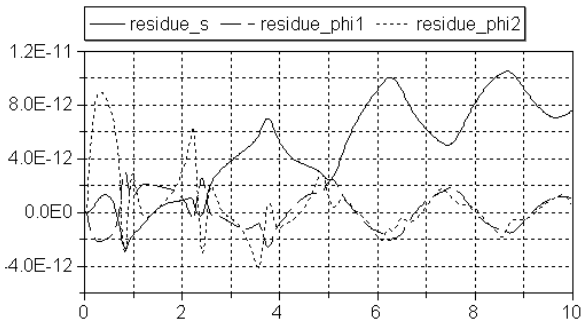


Figure 8. Deviation of the most interesting coordinates between the motor calculus and the Modelica Standard Library implementation.

3.1 Fourfold pendulum on two movable sliders

The second example is a fourfold pendulum. It consists of two trolleys and a chain of four rigid bodies between them. Both trolleys are guided along straight tracks (see Figure 10). Hence, this example contains a *closed kinematic loop*. Similar to the foregoing example, the pendulum moves from an initial deflection due to the gravity field of the earth and is damped by a viscous friction in every joint. The initial values for the pendulum angles are

$$\begin{aligned} \varphi_1(0) &= 45\,\mathrm{deg} \quad \varphi_2(0) = -15\,\mathrm{deg} \\ \varphi_3(0) &= 30\,\mathrm{deg} \quad \varphi_4(0) = -37.5\,\mathrm{deg} \end{aligned} \tag{35}$$

As before, the pendulum system was implemented twice. The first pendulum system works on the basis of the modified Mechanical Multibody Library, while the second one uses the Multibody Standard Library and serves as a reference. Hence, the deviations to the modified model can be calculated. They have the same order of magnitude as in the example before and can thus be explained by numerical errors.

For the rough illustration of the simulation results, Figure 9 shows the configuration of the pendulum at ten different time instances (the time interval is 1 s). The dashed lines show the tracks of both trolleys. The

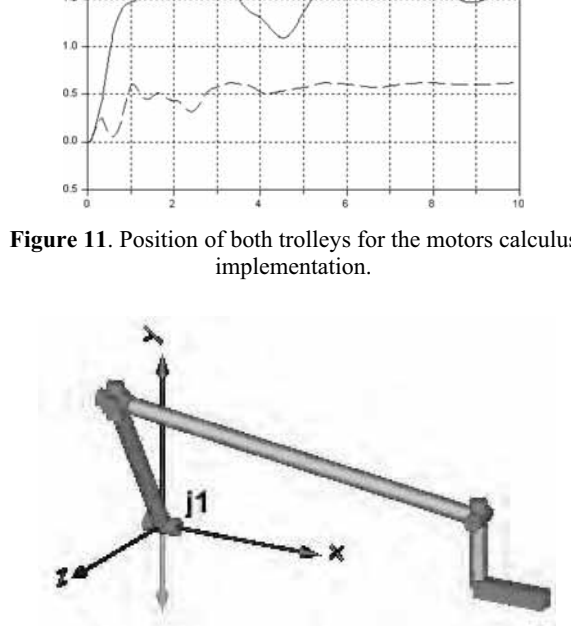


Figure 11. Position of both trolleys for the motors calculus implementation.

Figure 12. Sketch of the fourbar mechanism.

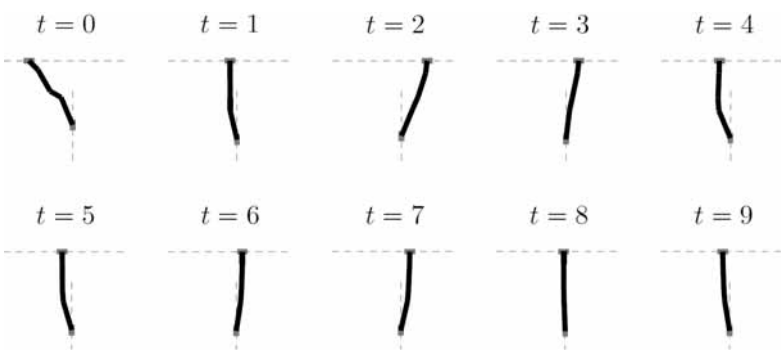


Figure 9. Sequence of configurations of the fourfold pendulum.

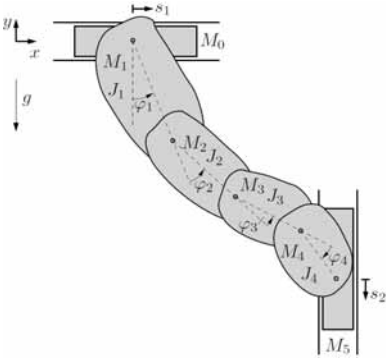


Figure 10. Sketch of the fourfold pendulum.

bold plotted polygon consists of four segments. It represents the idealized shape of the chain. In Figure 11, the position of both trolleys are plotted against the time.

3.2 Fourbar mechanism

The last example is a so-called fourbar mechanism from the Modelica Standard Library, that, again, sets up a closed kinematic loop (see Figure 12). However, in this example, the rigid bodies do not perform planar motions any more and, hence, the whole complexity of the three-dimensional mechanics is necessary.

The fourbar mechanism moves under the influence of the earth's gravitational field. The initial condition of the angular velocity of the first revolute joint (j1) is set to $300\,\mathrm{deg/s}$. In opposite to the foregoing examples, this system is completely undamped. As in the paragraphs before, the example was implemented twice in one model. One system has just been kept in its original form while in the second system, all `BodyCylinder` objects have been replaced by the modified `BodyCylinder` objects. The difference between both implementations is shown in Figure 13. The numerical results of the simulation show an increasing deviation with advancing time. The reason for this fact may be the absence of any damping elements. Indicated by this result, further investigations on numerical accuracy seem to be necessary for the future.

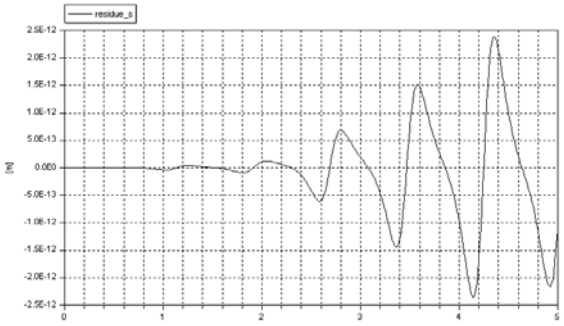


Figure 13. Deviation of the slider position between the motor calculus and the Modelica Standard Library implementation.

4 Summary and outlook

The paper traces the idea of applying the so-called motor calculus within Modelica modelling language to handle models of spatial multibody systems in an efficient way. This method represents an alternative approach to modelling such systems. This approach is characterized by a clear and concise formulation of the equations of motion.

To get some experiences with possibilities and limits of this approach, a first test implementation was carried out. The Modelica Multibody Standard Library was used to implement appropriate extensions within some selected submodels. This implementation allows a comparison of the standard library implementation and the motor calculus implementation by means of simple simulation tasks. Appropriate results are presented in the paper.

These results seem to encourage the idea of motor calculus usage within Modelica. Nevertheless, there are open challenges to be solved in the future. Operand overloading would be a very helpful feature in this context. Furthermore, efficient methods have to be adapted to compute actual position and orientation of a rigid body from its velocity motor. That's why further investigations as well as implementation work will still have to be carried out for a full support of rigid-body motion equation by means of motor calculus in Modelica.

References

[1] R.S. Ball. *A Treatise on the Theory of Skrews. Cambridge University Press*, 1900.

[2] F. Casella, M. Lovera. *High-accuracy orbital dynamics simulation through Keplerian and equinoctial parameters*. In Proc. 6th Int. Modelica Conference, Bielefeld, Germany, March 3–4, 2008, pages 505–514. The Modelica Association, 2008.

[3] F.E. Cellier. *Continuous System Modeling*. Springer, 1991.

[4] W.K. Clifford. *Preliminary sketch of bi-quaternions*. Proc. London Math. Soc., 4:381–395, 1873.

[5] P. Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Wiley-IEEE Press, 2003.

[6] J. Gallardo-Alvarado. *Kinematics of a hybrid manipulator by means of screw theory*. Journal Multibody System Dynamics, 14(3–4):345–366, 2005

[7] A. Haumer, C. Kral, J.V. Gragger, H. Kapeller. *Quasistationary modeling and simulation of electrical circuits using complex phasors*. In Proc. 6th Int. Modelica Conference, Bielefeld, Germany, March 3–4, 2008, pages 229–236. The Modelica Association, 2008.

[8] C. Heinz. Motorrechnung im X_{1+3+3}. *Zeitschrift für Angewandte Mathematik und Mechanik (ZAMM)*, 67(11):537–544, 1987.

[9] C. Knobel, G. Janin, A. Woodruff. *Development and verification of a series car Modelica/Dymola multibody model to investigate vehicle dynamics systems*. In Proc. 5th Int. Modelica Conference, Vienna, Aus-

tria, September 4–5, 2006, pages 167–173. The Modelica Association, 2006.

[10] I.I. Kosenko, M.S. Loginova, YA.P. Obraztsov, and M.S. Stavrovskaya. *Multibody systems dynamics: Modelica implementation and Bond Graph representation*. In Proc. 5th Int. Modelica Conference, Vienna, Austria, September 4–5, 2006, pages 213–223. The Modelica Association, 2006.

[11] R. von Mises. *Motorrechnung, ein neues Hilfsmittel der Mechanik*. Zeitschrift für Angewandte Mathematik und Mechanik (ZAMM), 4(2):155–181, 1924.

[12] R. von Mises. *Anwendungen der Motorrechnung*. Zeitschrift für Angewandte Mathematik und Mechanik (ZAMM), 4(3):193–213, 1924.

[13] www.modelica.org/documents. 2008-04-22

[14] www.modelica.org/libraries/Modelica/, 2008-04-22

[15] M. Otter, H. Elmqvist, and S. E. Mattsson. *The New Modelica MultiBody Library*. In Proc. 3rd Int. Modelica Conference, Linköping, Sweden, November 3–4, 2003, pages 311–330. The Modelica Association, 2003.

[16] T. Pulecchi, M. Lovera. *Object-oriented modelling of the dynamics of a satellite equipped with single gimbal control moment gyros*. In Proc. 4th Int. Modelica Conference, Hamburg, Germany, March 7–8, 2005, Proc., pages 35–44. The Modelica Association, 2005.

[17] B. Stroustrup. *The C++ Programming Language – Special Edition*. Addison-Wesley, 2007.

[18] H. Stumpf, J. Badur. *On the non-abelian motor calculus. Zeitschrift für Angewandte Mathematik und Mechanik (ZAMM)*, 70(12):551–555, 1990.

[19] M.M. Tiller. *Introduction to Physical Modeling with Modelica*. Springer, 2001.

[20] L. Viganò, G. Magnani. *Acausal modelling of helicopter dynamics for automatic flight control applications*. In Proc. 5th Int. Modelica Conference, Vienna, Austria, September 4–5, 2006, pages 377–384. The Modelica Association, 2006.

[21] T. Zaiczek. *Modellbildung für mechanische Systeme mit einer endlichen Anzahl von Freiheitsgraden und Steuerungsentwurf mithilfe von* $m \geq 1$ *Aktuatoren*. Technical report, 2006.

[22] T. Zaiczek. *Modellierung, Regelung und Simulation mechanischer Starrkörpersysteme im dreidimensionalen Raum*. Diploma thesis, TU Dresden, Germany, 2007.

Corresponding author: Tobias Zaiczek
Fraunhofer Institute for Integrated Circuits,
Design Automation Division, Dresden, Germany
Tobias.Zaiczek@eas.iis.fraunhofer.de

Accepted EOOLT 2007, June 2007
Received: August 10, 2007
Accepted: August 20, 2007

XML Meets Simulation: Concepts and Architecture of the IBKSim Network Simulator

Lukas Wallentin, Marco Happenhofer, Christoph Egger, Joachim Fabini

Vienna University of Technology, Austria

SNE Simulation Notes Europe SNE 20(1), 2010, 16-20, doi: 10.11128/sne.20.tn.09962

One option to evaluate various aspects concerning the performance of large computer networks is simulation. In order to understand the influence of single parameters onto the overall performance of a network, large series of simulations must be processed. Therefore, automation of simulation series is an important issue in the development of network simulators. In this paper we present the second version of the discrete event based simulator IBKSim. This simulator incorporates the experience of IKNSim which has been developed in 2005 at the same institute. Using XML (Extensible Markup Language) for the configuration and logging function, the new version bridges the gap between usability and automation. In addition to the simulator description we outline our experience with XML as configuration and logging language.

Introduction

Performance analysis and identification of design problems in communication systems can be very challenging due to the size of communication networks, the inherent complexity of network protocols and the influence of the environment. Detailed simulation of communication systems offers an excellent opportunity to perform experiments to gain a deeper insight into their behavior under different conditions.

In this paper we present IBKSim, the second simulator which has been developed at the Institute of Broadband Communications of the Vienna University of Technology. Based on the experience with IKNSim and other simulation environments, IBKSim was developed to meet the requirements of a state of the art network simulator. The motivation was to produce not only a simulator which supports large simulation series but also to facilitate single experiments on networks. As a result XML [1] was introduced as format to describe both, simulation scenarios as well as the simulator's output. The usage of XML differentiates IBKSim from other network simulators i.e. NS2[2] and OMNET++[3]. Another important target was the extensibility of the simulator to allow other developers to use IBKSim as an development and testing platform for own network protocols and algorithms.

The remainder of this paper is structured as follows: Section 1 provides a closer description of the architecture of IBKSim. Section 2 explains the benefits of XML as configuration and logging language, whereas section 3 presents the usage of IBKSim to solve typical scenarios. The paper concludes with a summary.

1 Basic IBKSim Concept

IBKSim is a discrete event based simulator written in C++. The simulator kernel is based on the simulator described in [4] and is pictured in Figure 2. It consists of a basic `simEntity` class for the representation of simulation objects. Simulation objects exchange objects of the type `simEvent` which represents current or future events. Future events are stored in the order of their occurrence in an queue-object of the class `simQueue`. As long as there is at least one event in the simulation queue the simulator controller (`simControl`) removes the next event and executes it on the associated simulation object. The object reacts on this event and eventually submits new events to the simulation queue. The simulation ends if there is either no event left in the simulation queue or if the moment of the occurrence of the next event lies behind a preconfigured point of time.

By using this kernel it is possible to build own simulations by implementing new simulation objects which are derived from `simEntity`. Since the purpose of IBKSim is to simulate networks, many additional modules and functions have been implemented.

The IBKSim clear differentiates between user and object developer which is particularly important for the usability and automatic simulation series. In this context a user is a person or a program which generates a simulation scenario using the implemented simulation objects. The implementation of simulation object is the object developer's main task.

1.1 User perspective

From the user's point of view, IBKSim takes a configuration file as input and generates a logging file. If

the simulator runs in the so called debugging mode it additionally generates output to the terminal. In the configuration file, which is written in XML, the user can describe the whole simulation scenario. Listing 1 depicts an example of such a XML file. After configuring the maximum simulation duration and the initial random function seed, the user models the network. A network consists of nodes which are containers for components. Every node and every component within this node has a unique name. Additionally each component has a specific type and an optional debugging parameter. The type describes to which simulation object class the component belongs to. The debugging parameter is just used in the debugging mode. Depending on the value of the parameter, the component generates trace information on the terminal.

Since components are the actual simulation objects, every node needs at least one component. The concept that a node is a group of simulation objects, as shown in Figure 1, allows to simulate different network layers as different objects. Each component can have different additional parameters which are specific to it. The component description follows the configuration of the connections.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<simulation>
  <paramlist>
    <duration value="100"/>
    <initialseed value="1"/>
  </paramlist>
  <network name="net1">
    <node name="source">
      <component name="source0" type="source" debug="0">
        <source prio="0" rate="10">
        </source>
      </component>
      <component name="wait" type="delay" debug="1">
      </component>
      <clink src="source0" dst="wait" direction="both"/>
    </node>
    <node name="sink">
      <component name="sink0" type="sink" debug="0">
      </component>
    </node>
  </network>
  <linklist>
    <link name="src-sink" direction="both">
      <src net="net1" node="source" cmp="wait"/>
      <dst net="net1" node="sink" cmp="sink0"/>
    </link>
  </linklist>
  <loglist>
    <logging type="file" interval="10">
     <config filename="logfile.xml"/>
     <log net="net1" node="source" cmp="source0" v="1"/>
     <log net="net1" node="sink" cmp="sink0" v="1"/>
    </logging>
  </loglist>
</simulation>
```

Listing 1. Example description of a simulation scenario.

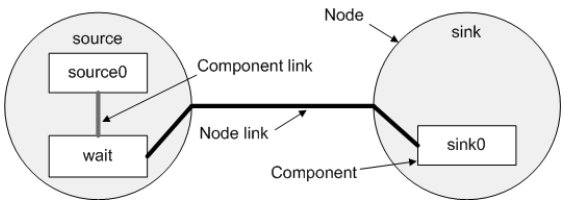


Figure 1. Visualization of the network described in Listing 1.

After the description of all network nodes, the nodes can be linked using their unique names. In contrast to the intra node links within nodes the inter node links represent physical links and therefore they support the additional parameters delay, jitter and loss-rate.

The last configuration concerns the logging function. The logging function collects cyclical logging information from the selected components. This allows to monitor certain user defined component parts over the time. The logging information which is again in XML format is saved in the specified log file as pictured in Figure 2.

The user must know which components are implemented in the simulator and which parameters they have, without possessing any knowledge about the code. The benefit of using XML for configuration and cyclical logging is described in Section 3.

1.2 Developer perspective

To implement a new simulation object the developer has to create a new class which is derived from `simEntity`. Furthermore he can use additional functions which have been implemented into the simulator.

Component factory Following start-up, the simulator parses the input file and generates a network of simulation objects which are commonly connected by links. This is realized using a component factory. For every component specified in the configuration file, the factory calls the static build function of the class which is associated with the type of the component. This function generates a simulation object and configures it depending on the additional parameters in the XML file. Afterwards it returns the new object to.

The usage of this so called factory pattern has some benefits for the developers. Since every class of simulation object has its own build function, and the build function gets the complete XML-content between the start and the end tag of the component, the developer

can easily specify new XML-tags for the component configuration. Additionally he can use the build function to trigger init functions in the newly generated objects.

The developer does not need to use the component factory if it is not required. This is useful, e.g., when a simulation object is just started by another simulation object. One specific example is a server object which dynamically starts new simulation objects to handle requests. If the developer wants to use the component factory, he must implement an own build function and register the class with the factory.

Static connections As mentioned previously, it is possible to describe connections between components in the configuration file. To implement this functionality, the basic simEntity class implements the function connect. The component factory calls this function automatically to perform the linking between components which is basically an exchange of pointers. Using the pointers, a simulation object can transmit an event to another object. To simulate a physical link between two nodes the simulator places an additional simulation object between them which simulates the delay and the loss rate of the physical link.

Layered architecture IBKSim supports layered architectures by providing a class which has a special implementation of the already mentioned connect function. It automatically connects the components within a node derived from this class.

Messages and timer The reception of messages in the form of packets or frames are common events in communication networks. By including objects into events, it is possible to combine the transmission of events and messages between the simulation objects in an intuitive way. It is also an efficient way to implement preemptive timers. One option to implement a timer in an event based simulator is to send an event from a simulation object to itself. In the case that the timer must be deleted, the simulator searches for the event in the event list to remove it. To avoid the search, this is solved in a different way in IBKSim. Every timer includes a small object with a flag which states if the timer is active. To delete a timer the flag must be disabled. When the event occurs, the simulation object can ignore the event when the flag is not set. Using this concept, the search in the event list can be avoided.

Logging. The simulator supports two ways of logging. If the simulator runs in the debug mode and the debug level for a component is set, the component can output traces to the terminal. This event triggered logging function is intended mainly for evaluating single simulations and for development. For simulation series the cyclic logging function has been developed.

```
<ibkSimLog>
  <logentry time="10">
    <log net="net1" node="source" cmp="source0">
      <source sentPackets="102"/>
    </log>
    <log net="net1" node="sink" cmp="sink0">
      <sink receivedPackets="89" averageDelay="0.452"/>
    </log>
  </logentry>
  <logentry time="20">
    <log net="net1" node="source" cmp="source0">
      <source sentPackets="98"/>
    </log>
    <log net="net1" node="sink" cmp="sink0">
      <sink receivedPackets="92" averageDelay="0.483"/>
    </log>
  </logentry>
  <logentry time="30">
    <log net="net1" node="source" cmp="source0">
      <source  sentPackets="105"/>
    </log>
    <log net="net1" node="sink" cmp="sink0">
      <sink receivedPackets="102" averageDelay="0.464"/>
    </log>
  </logentry>
  ...
</ibkSimLog>
```

Listing 2. Example of a log file.

For the cyclic logging function each simulation object implements a function called `getLogs()`. This function returns an XML string sharing statistical information. A special simulation object called `logger` calls this function on a regular basis, collects all the strings and writes them to a log file. This time triggered logging function is useful to monitor the change of different simulation object variables over time.

2 The benefit of XML as configuration and logging language

XML is a markup language to structure data in text files. This system and vendor independent language is a royalty free open standard [1]. IBKSim uses XML files for configuration and for logging due to the benefits it offers to the user of the simulator as well as for the development of additional simulator tools.

By using XML for configuration a user is not bound to a specific tool or editor to generate simulation scenarios. There are a number of XML editors, i.e. XPontus [5] and Altova XMLSpy [6], specifically designed to generate and validate xml files. Alterna-

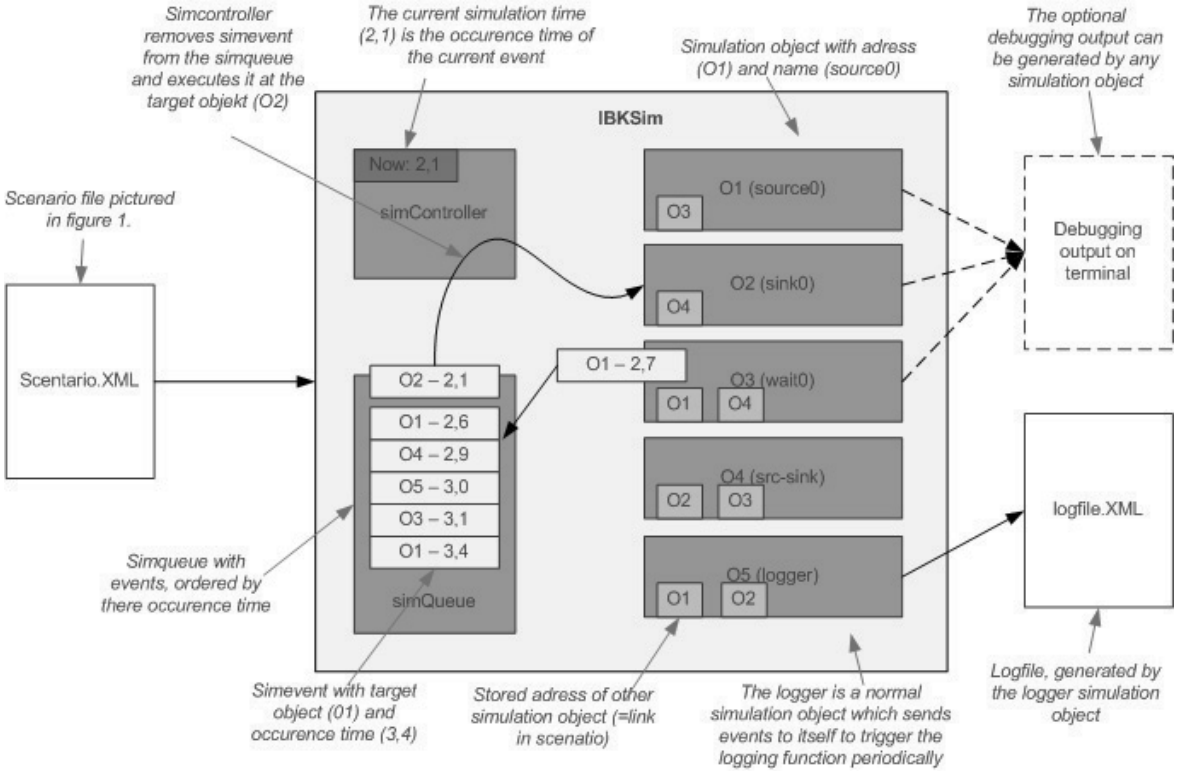


Figure 2. Overview over IBKSim.

tively, one can use a text editor to write a simulation scenario.

Since XML does not use a binary file format, simple scripts can be used to generate series of simulation scenarios. Additionally there exist a number of libraries for different programming languages to handle XML files in programs, i.e. Xerces [7] for C++, Java and Perl which simplifies the development of tools for the generation of complex simulation scenarios.

The usage of XML as log file format has some additional advantages. Microsoft Excel 2007 can directly handle XML files, which eases post-simulation processing and analysis.

Additionally XML processors can convert XML-files into other XML or human readable file types. One type of XML file is SVG [8] (Scalable Vector Graphics), a file format specifying vector graphics and animations using XML. By converting the log-file into a SVG file, a XML processor can generate an animation for presentation. With the same technique the log file can be converted into a comma separated values file (csv). This is useful if the user wants to analyze the data with statistic tools like *R* [9] which expects this data format as input.

Summarizing, we consider XML to be the most appropriate and flexible choice, both for scenario generation and for logging.

3 Work experience using IBKSim

The workflow for single simulations is straight forward. A user generates a simulation scenario in XML by using a text or XML editor. He starts the simulation and afterwards analyses the log file using e.g. Excel. Additionally he can analyze the simulator's debugging output if the components are configured accordingly.

For simulation series we are using a Beowulf cluster with 20 CPUs running Debian GNU/Linux 4.0 [10]. To manage the work load the sun grid engine [11] is used which accepts jobs in the form of shell scripts. To perform a simulation series the cluster uses three files: a template of the simulation scenario, a template of a job script which should be processed by the grid engine and a generation script.

A template is a XML configuration file where parameter values have been replaced by keywords. For instance the file in Listing 1 could be transformed to a template by replacing the initial seed value in line

five from *1* to *--seed--*. The generation script can use this template to generate e.g. 100 simulation scenarios by copying the template and replacing the keyword *--seed--* with numbers from 1 to 100. The inserted value also becomes part of the name of the configuration and log file.

Additionally the generation script produces for each generated configuration file a job script using the job template. To perform solely the simulation a call of the simulator with the according configuration file in the job script is sufficient. Since in most cases a statistical analysis follows the simulation series, the job script is frequently used to pre-process logging data. A typical example is the calculation of the median and the quartiles of a logged parameter. For such operations *r* ("little R"), the command line version of *R,* is used. After generation of job scripts and configuration files, the generation script submits all jobs to the grid engine. When all simulations are processed, the log files or the files containing the preprocessed data are analyzed.

4 Summary

In this paper we presented the network simulator IBKSim. We outlined the architecture of this discrete event based simulator and described selected functions in detail. Particularly the advantages of XML as configuration and logging file format have been described in detail. Additionally typical examples of the simulation workflow using IBKSim were given. Concluding, the integration of XML has shown to be efficient and rewarding both for the user and the developers of IBKSim.

Acknowledgement

The authors would like to thank Jon Schuringa for the development of the IKNSim and all the other members of the Institute of Broadband Communications who were involved in the IKNSim and IBKSim projects.

References

[1] T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, F. Yergeau: *Extensible markup language (XML) 1.0*., W3C recommendation, vol. 5, 2008.

[2] S. McCanne, et al.: *The Network Simulator - ns-2*, online: http://www.isi.edu/nsnam/ns/, visited: April 2009

[3] A. Varga, et al.: *The OMNeT++ discrete event simulation system*, Proceedings of the European Simulation Multiconference (ESM'2001), 319-324, 2001

[4] O. Spaniol, S. Hoff: *Ereignisorientierte Simulation*, Thomson Publishing, 37, 1995

[5] Y. Zoundi.: *XPontus XML Editor*, online: http://xpontus.sourceforge.net/, visited: April 2009

[6] Altova Inc.: *XMLSpy - XML Editor for Modeling, Editing, Transforming, & Debugging XML Technologies*, online: http://www.altova.com/xml-editor/, visited: April 2009

[7] The Apache Software Foundation.: *Xerces-C++ XML Parser*, online: http://xerces.apache.org/, visited: April 2009

[8] J. Ferraiolo, F.Jun, D. Jackson: *Scalable Vector Graphics (SVG) 1.1 Specification*, W3C recommendation, 2003

[9] R.Ihaka, R. Gentleman: *R: a language for data analysis and graphics*, Journal of Computational and graphical statistics, American Statistical Association, Institute of Mathematical Statistics, and Interface Foundation of North America,299-314, 1996

[10] Debian-Project*: Debian – The Universal Operating System*, online: http://www.debian.org/, visited: April 2009

[11] W. Gentzsch: *Sun grid engine: Towards creating a compute power grid*, First IEEE/ACM International Symposium on Cluster Computing and the Grid, 2001. Proceedings, 35-36, 2001

Corresponding author: Lukas Wallentin
Vienna University of Technology
Institute of Broadband Communications
Wiedner Hauptstraße 8 – 10, 1040 Vienna, Austria
Lukas.Wallentin@tuwien.ac.at

Received: March 10, 2009

Revised: January 20, 2010

Accepted: February 20, 2010

A New Technique for Interactive Simulation of Recurrent Neural Networks

Granino A. Korn, University of Arizona, USA, *gatmkorn@aol.com*

SNE Simulation Notes Europe SNE 20(1), 2010, 21-26, doi: 10.11128/sne.20.tn.09963

We present new techniques for modeling the feedback loops of recurrent neural networks, including networks that incorporate tapped delay lines or gamma delay lines. Very fast simplified programs result. Examples of applications include signal prediction and dynamic-model matching. We also suggest interesting future research on improved programs for time-series recognition and classification.

Introduction

This article describes much-simplified computer programs for interactive simulation of recurrent neural networks. Sections 1 to 4 briefly review dynamic-system simulation and our open-source software for Windows and Linux [1, 2, 3]. We employ a compact, human- and machine-readable vector notation, including very powerful vector index-shift operations for modeling delay lines and filters. The remainder of this report applies these techniques to neural-network simulation.

Section 5 presents a simple backpropagation model representing each neuron layer by a one-line vector assignment. Section 6 then describes a significant innovation: a technique for programming the time-delayed feedback in recurrent networks without the complication of special context layers. Sections 7 to 9 next apply our simple vector index-shift notation to neural networks with input and feedback delay lines or gamma delay lines.

Finally, Sections 10 and 11 discuss applications to model matching and time-history prediction and suggest other applications for future research.

1 A Simulation Language for Interactive Dynamic-system Modeling

Desire simulation programs[1,2] model dynamic systems using a natural mathematical notation for successive difference-equation assignments like

```
x = x + a * sin(c*t)
y = x
...
```

Listing 1. Difference-equation notation in Desire

and/or differential-equation-system assignments like

```
u         = alpha * sin(w * t + beta) + c
d/dt x    = xdot
d/dt xdot = -a * x - b * xdot
```

Listing 2. Differential equation system in Desire

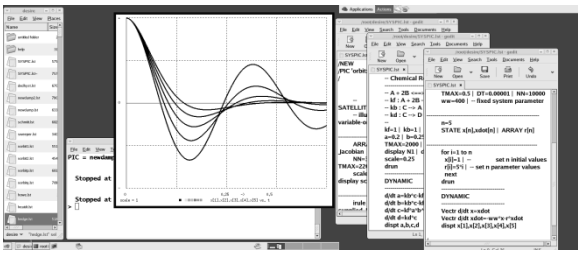


Figure 1. Desire with a command window, file manager, and 3 screen-editor windows. Programs in multiple editor windows can be run to compare models (based on [3]).

Such model definitions are screen-edited into a `DYNAMIC` *program segment* (Figure 1). Simulation studies are controlled by typed interactive commands and/or by an *experiment-protocol script*. Experiment-control commands set or change parameters and initial conditions and then call *simulation runs* that produce time-history displays. Each simulation run exercises the model by calling the `DYNAMIC` program segment for `NN` successive time steps, as in

```
t0 = 0    | t = t0  | NN = 2000 | TMAX = 100
a = -5.00 | x = 17.1
drun
```

| is a statement delimiter. When the experiment protocol encounters the first `drun` statement the `DYNAMIC` segment is compiled with a fast runtime compiler and runs immediately to produce time-history displays (Figure 1). More elaborate experiment protocols can call multiple simulation runs with modified parameters and different `DYNAMIC` segments [2, 3].

2 Fast, Human- and Machine-readable Vector Operations

Desire experiment-control scripts can declare *vectors* like `x ≡ (x[1], x[2], …, x[n])` and *matrices* like `W ≡ (W[1,1], W[1,2], …, W[n, m])` with single or multiple `ARRAY` statements such as

```
ARRAY x[n], a[m], b[n], c[n], y[m],
      W[m, n], u[n], v[n], …
```

DYNAMIC program segments can then use the vectors and matrices in *vector assignments*, and *vector differential equation*, say

```
Vector x     = a + alpha * b * c
Vector y     = tanh(W * x)
Vectr d/dt x = beta * cos(t + c)
```

which automatically compile into multiple scalar operations

```
x[i] = a[i] + alpha * b[i] * c[i]     (i=1,…,n)
y[i] = tanh(∑_{k=1}^{m} W[i,k] * x[i])  (i=1,…,n)
d/dt x[i] = alpha * cos(t + c[i])     (i=1,…,n)
```

MATRIX assignments similarly compile into multiple assignments to matrix elements W[i,k] [2]. All these compiler operations unroll program loops, so that the resulting binary code is fast.

We can also compute vector-component sums and inner products like

$p = \sum_{k=1}^{n} u[k]$

$p = \sum_{k=1}^{n} u[k] * v[k]$

with *inner-product assignments* DOT p = u*1 and DOT p = u*v, again without program-loop overhead.

Desire vector operations permit very fast vectorized Monte Carlo simulation of engineering and biological systems and can model fuzzy-logic controllers and partial differential equations as well as the neural-networks we shall discuss here [3].

3 Vector Index-shifting, Delay Lines, and Filters

Given an *n*-dimensional vector x ≡ (x[1], x[2], …, x[n]) and an integer *k*, the *index-shifted vector* x{k} is the *n*-dimensional vector (x[1+k], x[2+k], …, x[n+k]), with components referring to indices less than 1 or greater than *n* set to 0. Significantly, the assignments

```
Vector x = x{-1}  |  x[1] = input
```

Listing 3. Vector assignment using index-shift notation

compile into

```
x[i] = x[i - 1]      (i = 1,…,n)
x[1] = input
```

This neatly models shifting successive samples of a function u(t) into a *tapped delay line* with tap outputs x[1] = input, x[2], …, x[n]. Note that the assignment x[1] = input overwrites the Vector operation's assignment x[1] = 0 at each step.

Assignments like Listing 3 can, for instance, model a complete nth-order digital filter *with only two program lines* – a very efficient notation and a very efficient implementation. Sections 7 to 9 will describe neural networks incorporating tapped delay lines and also gamma delay lines[5] modeled with a similar index-shift operation.

4 Neural-network Models

DYNAMIC program segments (Listing 1) that include differential equations compute state-variable derivatives. An integration routine selected by the experiment-control script then combines derivative values from successive time steps to update differential-equation state variables [2].

Desire can model biological neurons with differential equations (e.g. pulsed inegrate-and-fire neurons) [3], but the neural-network models we discuss here are much simpler. For DYNAMIC program segments without differential equations, the simulation time t automatically steps through t = 0, 1, 2, …, NN by default (users can, if desired, specify different starting times and/or time increments). Neuron activations and connection weights are represented by real numbers that roughly model neuron pulse rates and synapse chemistry. Both are updated with simple difference equations in successive time steps. Thus, we can handle problems that combine differential-equation models and neural networks, as in sampled-data control systems.

5 A Simple Backpropagation Network

Figure 2 shows a simple three-layer neural network. Desire's interpreted experiment-protocol script declares the three neuron layers in turn with

```
ARRAY x[nx] + x0[1] = xx v[nv], y[n]
x0[1] = 1
```

and two connection-weight matrices W1 and W2 with

```
ARRAY W1[nv, nx + 1], W2[ny, nv]
```

Desire array declarations like ARRAY x[nx] + x0[1] = xx act like Fortran equivalence statements: xx[3] is identical with x[3], and xx[nx + 1] is identical with x0[1]. As is customary, the input layer xx adjoins a one-dimensional bias vector x0 to the normal nx-dimensional network input x. With x0[1] set to 1, we can then conveniently represent input biases as nv extra connection weights W1[i, 1].

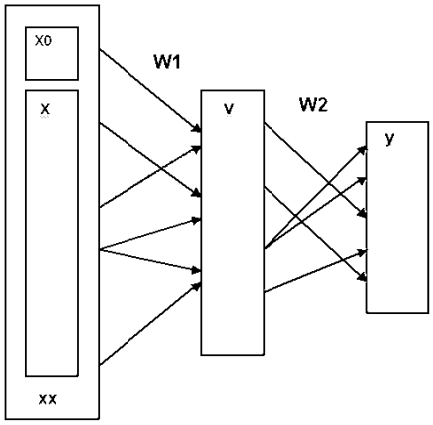


Figure 2. A simple backpropagation network.

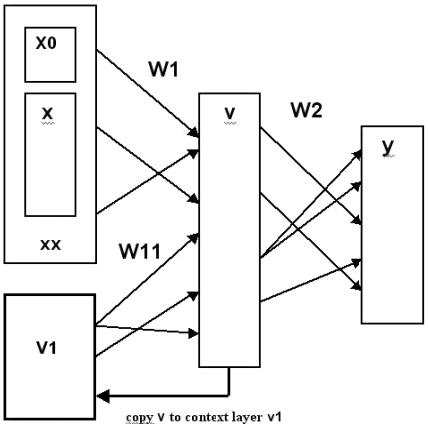


Figure 3. A simple Elman recurrent network.

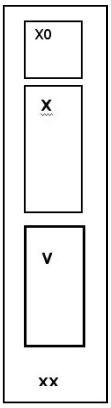


Figure 4. Modified input layer.

The runtime-compiled DYNAMIC program segment defines the network dynamics with

```
Vector v = tanh(W1 * xx)
Vector y = W2 * v
```

if we use a tanh activation function for the nonlinear hidden layer. To produce simple backpropagation updating, we declare target, error, and error-propagation vectors with

```
ARRAY target[ny], error[ny], vdelta[nv]
```

and program

```
Vector error  = target - y
Vector vdelta = W2% * error * (1 - v^2)
DELTA  W1     = lrate1 * vdelta * xx
DELTA  W2     = lrate2 * error * v
```

Listing 4. Simple backpropagation updating

Here W2% denotes the transpose of the connection-weight matrix W2, and

```
DELTA  W = matrix expression       is equivalent to
MATRIX W = W + matrix expression
```

These assignments update vectors and matrices with data computed earlier, starting with given initial values. Desire ARRAY declarations initialize all subscripted variables to the default value zero. That is fine for the vectors; but the experiment-protocol script must initialize the connection weights W1[i,k] and W2[i,k] with small random values.

In addition to declaring and initializing neuron-layer arrays, the experiment-protocol script for a neural-network experiment must set parameters and initial values of scalar state variables (if any) and then schedule training and test simulation runs with drun statements. The script also selects integration rules (if any) and the display scale and colors. For simplicity, our text omits these housekeeping operations.

6 Simplified Recurrent-network Programming

An Elman recurrent network (Figure 3) [5, 6] copies all or some of the hidden network layer v to a *context layer* v1 which is fed back to v together with the input xx. The experiment-protocol script declares the original 3 neuron layers xx, v, and y and the connection weight matrices W1 and W2 as before,

```
ARRAY x[nx] + x[0[1] = xx,
      v[nv], y[ny], W1[nv, nx + 1], W2[ny, nv]
x0[1] = 1
```

and adds the context layer v1 and a new connection-weight matrix W11:

```
ARRAY v1[nv], W11[nv, nv]
```

The network dynamics in the DYNAMIC program segment become

```
Vector v1 = v
Vector v = tanh(W1 * xx) + tanh(W11 * v1)
Vector y = W2 * v
```

Listing 5. Implementation of network dynamics

To update W11 as well as W1 and W2 by backpropagation now requires two error-propagation vectors v1delta and v2delta, and the updating program becomes more complicated. But there is a much better way!

Just as we concatenated the input layer x and its bias layer x0, we can declare a single new input layer xx that combines our hidden layer v with x and x0 (Fig.4):

```
ARRAY x[nx] + x0[1] + v[nv] = xx  |  x0[1] = 1
```

Listing 6. Declaration of a new input layer

The two connection-weight matrices W1 and W11 of the Elman network in Figure 3 can now be replaced with a single connection-weight matrix W1,

```
ARRAY W1[nv, nx + 1 + nv]
```

W1 feeds xx to the hidden layer v just as in Figure 2 – but xx now includes the hidden-layer activations v computed in the preceding iteration. The simple backpropagation-updating assignments (Listing 4) for the static network of Figure 2 then work without change for the recurrent neural network in Figure 3. Only the array dimensions have changed.

It is just as easy to implement time-delayed feedback from the output layer y (Jordan recurrent network), or from both v and y. Backpropagation updating remains exactly the same. This simplified implementation of recurrent-network feedback is by no means restricted to backpropagation networks. This technique serves equally well for two-layer linear and nonlinear networks, for softmax pattern recognizers, and for radial-basis-function networks, which are all easy to program in the Desire language [3]. In each case we simply reuse the unchanged program for a static neural network.

7 Networks with Input Delay Lines

The earliest neural network with time-history memory was Widrow's adaptive filter [5]. In Figure 5, successive values of a single time-series input input enter a delay line whose taps feed a static neural network trained to filter, recognize, or predict time-series patterns. Desire's compact index-shift operation (Listing 3) is exactly what is needed for modeling such networks.

Widrow's original network, for example, combined a delay line and a simple linear network layer

```
Vector x = x{-1}  |  x[1] = input
Vector y = W * x
```

Widrow's network had a single output y[1] and thus implemented a linear filter that could be trained with his new LMS algorithm to match a target time series.

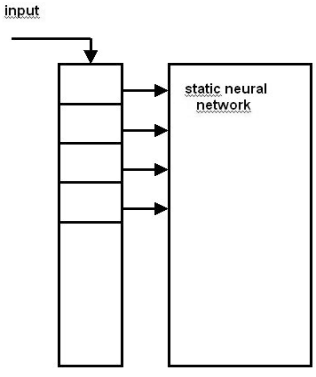


Figure 5. A static neural network fed by an input delay line

In our notation this successsive-approximation rule would be

```
DELTA W = lrate * (target - y) * x
```

Improved designs incorporate a nonlinear multilayer network, say the backpropagation network of Section 5:

```
Vector x = x{-1}  |  x[1] = input
Vector v = tanh(W1 * x)
```

or other types of static networks [5]. All need only ordinary static-network training.

8 NARMAX Networks use Delay-line Feedback

The recurrent network in Figure 6 has a single input input to a delay-line layer x of length nx as before. The output layer y has only a single output output. y[1]. The (scalar) error in the network output is

```
ERROR = target - y[1]
```

where target is a desired output time series. Successively delayed samples of ERROR enter a second delay-line layer error of length ne.

The delayed error samples are fed back to the neural network.

Referring to Figure 6, we again concatenate all input-layer vectors, in this case the two delay lines x and error, into a single input layer xx:

```
ARRAY x[nx] + x0[1] + error[ne] = xx
```

xx feeds the hidden layer v of an ordinary backpropagation network.

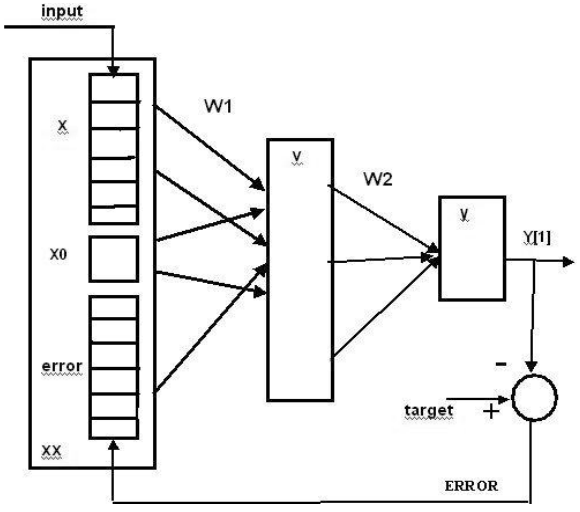


Figure 6. A NARMAX (Nonlinear Auto-Regressive Moving Average with eXogenous inputs) network.

```
ARRAY x1[nx] + x0[1] + error1[ne] = xx
x0[1] = 1
ARRAY v[nv], y[1], error[ne], vdelta[nv]
ARRAY W1[nv, nx + ne + 1], W2[1, nv]
  ...
DYNAMIC
Vector x1 = x1{-1} |
x1[1] = input              // input delay line
Vector v=tanh(W1 * xx)     // hidden layer
Vector y = W2 * v          // output layer
ERROR = target - y         // output error
Vector error = error{1} |
error[1] = ERROR           // feedback delay line
Vector vdelta=W2%*error*(1-v^2) // backpropagation
DELTA W1= lrate1 * vdelta * xx
DELTA W2= late2* error * v
```

Programmers must specify the input and target time series for different applications.

Once again the backpropagation program is exactly the same as in Section 5. One can also substitute different types of neural networks for the backpropagation layers in Figure 6.

9 Networks with Gamma Delay Lines

A simple tapped delay line of length n "remembers" its input for only *n* time steps. Principe's *gamma delay line*[5] replaces each delay-line element with a simple first-order filter. That effecttively gives neural-network input and feedback delay lines a much longer memory, so that the networks tend to perform better or use fewer neurons. Our vector index-shift notation models a gamma delay line with

```
Vector x = x + beta*(x{-1} - x)
x[1] = input
```

which automatically compiles into

```
x[i] = x[i] + beta*(x[i - 1] - x[i])   i=1,…,n
x[1] = input
```

`beta` is a scalar filter parameter set by the experiment-protocol script; we have compactly programmed n difference equations for *n* identical first-order filters (It is convenient to program `Vector x = x + beta*(x{-1} - x)` as `Vectr delta x = mu*(x{-1} - x)`). We normally prefer such gamma delay lines for NARMAX networks.

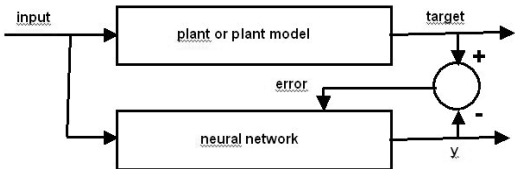


Figure 7. Matching a neural network to a plant or plant model. `input`, `target`, `y`, and `error` can be scalars or vector functions of the time `t`.

10 Applications

The most common applications of recurrent networks are

- model matching (e.g. plant models for control-system design)
- time-series prediction
- recognition or classification of time-series patterns

Figure 8 demonstrates a model-matching experiment. The program can be screen-edited and rerun immediately for truly interactive modeling. We programmed Elman networks with 2 and 3 hidden layers and a NARMAX network to match one of Narendra's difference-equation plant models[7] described by

```
f  = [Y(k)*Y(k-1)*Y(k-2)*input(k-1)*(Y(k-2)-1)
      + input(k)]/[1 + Y(k-1)^2 + Y(k-2)^2)]
target(k) = Y(k)           (k=0,1,2,...)
```

Listing 6. Narendra's difference-equation plant models

The networks were trained with random-noise input and tested with Narendra's test function.

```
s = 0.5 * ( (1 - 0.2*swtch(t-500)) * sin(w*t)
           + 0.2*swtch(t-500) * sin(ww * t) )
```

Listing 7. Narenda's test function

Training typically converged in 8 out of 10 simulation runs. All three recurrent networks then matched the plant equally well (Figure 8).

For modeling a predictor the "present" neural-network input is a delayed version of a specified "future" time series target:

```
ARRAY buffer[m]
Vector buffer = buffer{-1}
buffer[1] = target
input = buffer[m]
```

The neural network output y is then trained to match target. We programmed a textbook problem[5] predicting the chaotic Lorenz[1] and Mackey-Glass[5] time series (Desire models Mackey-Glass with only two program lines

```
tdelay S = D(signal, tau)
d/dt signal = a * S/(1 + Sd^c) - b * signal
```

where `tdelay` is a time-delay operator, and `a`, `b`, and `tau` are specified constants). Our Elman and networks predicted this time series within a few percent for 50 time steps ahead (Figure 9). As expected, gamma delay lines worked better than simple delay lines of the same length. Prediction was still successful when we removed the feedback delay line from the NARMAX network, resulting in the simpler model of Figure 5.

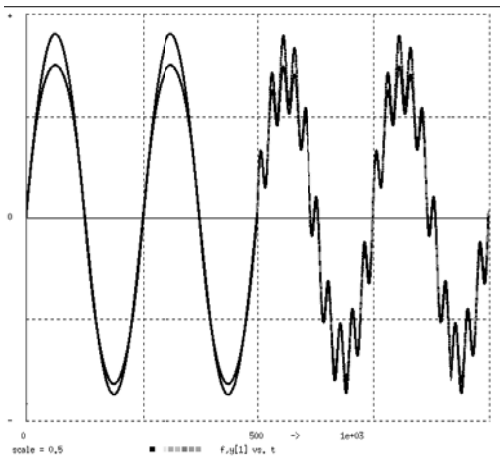


Figure 8. Simple Elman network matching Narndra's plant model (Listing 6). Plant and network were fed Narendra's test function (Listing 7). The graphs of `target=f` and `y` essentially reproduce Narendra's results obtained with his four-layer NARNAX network [7]..

Readers interested in the details of these studies – or in repeating our experiments – will find the compact Desire programs for 20 model-matching and prediction experiments included in the open-source Desire distribution file.

11 Conclusions and Future Research

The essential contribution of this article is the novel application of the Desire language's array declaration (6) in Sec. 7. Acting much like a Fortran equivalence statement, this programming trick eliminates entire neuron layers and greatly simplifies recurrent-network updating algorithms. The resulting neural-network models are smaller, faster, and easier to understand.

On a 3.15 GHz 2-CPU Penryn-class personal computer, the screen-edited, runtime-compiled programs exhibited in this report all compiled and produced time-history displays within 30 msec. This compilation delay is not noticeable, so that truly interactive modeling is possible. The recurrent-network programs in Figures 8 and 9 converged within 1 to 3 seconds.

We demonstrated simple applications to Elman, Widrow, and NARMAX networks to model matching and time-series prediction. Time-series pattern recognition (pattern classification) will be the first interesting topic for future work. Neuron layers implementing various softmax classifiers[3] will replace the backpropagation network in Figures 4 and 6. The required training procedure is again simply that for a static network.

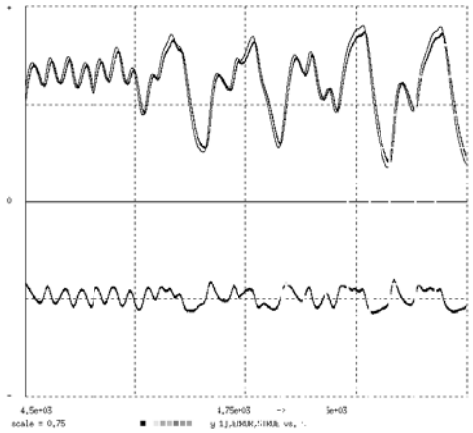


Figure 9. This display shows `target`, `y`, and `ERROR` for an Elman network predicting the Mackey-Glass chaotic time series. The original graphs were in color.

Our new trick of concatenating neuron-layer arrays would work equally well in most computer languages. But Desire's combination of an interpreted experiment protocol and fast runtime-compiled simulation runs makes interactive modeling – which can involve hundreds of program changes in one day – especially convenient.

12 References

[1] G.A. Korn. *Neural Networks and Fuzzy-logic Control on Personal Computers and Workstations*. MIT Press, Cambridge, 1995.

[2] G.A. Korn. *Interactive Dynamic-system Simulation under Windows*. Gordon and Breach, London, 1998.

[3] G.A. Korn. *Advanced Dynamic-system Simulation: Model-replication Techniques and Monte Carlo Simulation*. Wiley, Hoboken, NJ, 2007.

[4] G.A. Korn. *Fast Simulation of Digital and Analog Filters Using Vectorized State Equations*. Simulation News Europe 18-1, April 2008.

[5] J. Principe, et al. *Neural and Adaptive Systems*. Wiley, Hoboken, NJ, 2000.

[6] J.L. Elman. *Finding Structure in Time*. Cognitive Science, 14:179-211, 1990.

[7] K.S. Narendra, K. Parthasarathy. *Identification and Control of Dynamic Systems Using Neural Networks*. IEEE Trans. on Neural Networks, 1:4-27, 1990.

Corresponding author: Granino A. Korn
ECE Department, University of Arizona, USA
gatmkorn@aol.com

Received: May 14, 2009
Revised: October 23, 2009
Accepted: January 10, 2010

SHORT NOTE

Experiences and Challenges in Development of Sustainable Modelling and Simulation Tools

K. Juslin, VTT Technical Research Centre of Finland, *kaj.juslin@vtt.fi*

SNE Simulation Notes Europe SNE 20(1), 2010, 27-30, doi: 10.11128/sne.20.sn.09965

Modelling and simulation software tools should be sustainable even if it is not definitely required for temporary research projects with a publication as outcome. Especially, when using modelling in simulation as an integrated tool supporting both engineering and operation of industrial processes—the models should sustain for the whole life cycle of the target process—from initial design to decommission, at least 60 years. The solution relates, first, to apply such formal specifications of the physical model that easily can be carried as simple text files between generations of computer and operation system platforms. Second, reliable simulation engines are needed that are programmed in high level language enabling for transportation to the new platforms and. Third, standardised procedures for re-verification of the computational functionality are required. Fourth, easy connectivity to concurrently developing engineering databases and graphics interfaces is needed, as well. The advance of multi core processors, even on laptop computers, calls for reconsideration of the mathematical methods used so far in order to make use of the inherent computing power. Companion model graphs are supposed to have a significant mission in automated computer model generation from semantic plant specifications, as well as in even distributing to computation load on available cores.

1 Specification of simulation models with interconnects to engineering tools

My experiences relate to the Apros simulation environment developed more than 20 years ago [1]. Some models then developed are still in use! First computer platform was VAX/VMS, then several UNIX workstations, and now Windows based PCs. Now the third generation of graphics modelling and simulation interface is in use. Of course, the simulation engine has developed all the time, new functionalities and model libraries have been included, but keeping intact the formal model specifications [2]. The Apros Specification Language (ASL) supports introduction of new object types made up from predicates, subjects and required number of attributes. During application modelling these objects are instantiated by naming the subjects and assigning values to the attributes. The attribute values may also include references to other objects and attributes. The predicates include concepts for adding and deleting structures and connections in the model as well as for running the model. Apros is supplied with a Communication Library (ACL) enabling development of connections to third party software. Also, a modern OPC interface is available for standardised connections to e.g. control system software. ASL was developed before XML but the Apros specification triples are easy to wrap to XML formats. Evolving interoperability standards, such as ISO 15926, promote uniform utilisation of branch-specific specifications of equipments, industrial processes and even buildings, as well as of configuration files for automation systems.

Digital catalogues are already available over Internet for various kinds of construction materials and production plant components. New challenges are set for the simulation engines, on the other hand on increased computational speed, much faster than real-time, but also on more detailed description of the processes and on replication of more complicated physical phenomena.

2 Detailed spatial discretisation and distribution of computer power needs

Whence focusing on modelling and simulation of power plants and industrial processes, special attention has to be paid on highly parallel computation of an applicable class of non-linear but piece-vice monotonically continuous differential equations arising from spatial discretisation based on the geometry of the specific industrial processes to control volumes taking into consideration the relevant mechanistic first principles partial differential equations with regard to conservation of momentum, energy and mass substances and the related non-linear empiric material properties.

Until now it has been sufficient to model the material properties of water and steam with sufficient accuracy in conditions encountered in nuclear and conventional power plants. New plants designs cover the super-critical region of the steam tables to increase thermal efficiency. Accordingly the steam tables have been extended. In new designs of clean combustion plants also the production processes of oxygen as well as the flue gas cleaning processes need to be modelled [3]. In new designs of nuclear power plants new coolants need to be modelled, such as sodium or lead in liquid metal cooled reactors as well as helium or carbon dioxide in gas cooled reactors [4]. Especially when boiling of mixtures of several substances need to be considered the additional workload is considerable and problematic for real time simulations on single core processors. Multi-core processors and multi-node clusters may help, subject that the computing loads relevant to the spatial discretisation of the process model can be evenly distributed to the available computing nodes. The variables updated during the iterations needed for the solution at the end of each time-step should be as far as possible kept within relevant node. That means that all material properties needed on a node should be calculated on that node.

An interesting issue is, where and how to calculate the integrated pressure solution to minimize communication over main memory between the computing nodes. Specific descriptive graphs of the simulated process may provide the information needed for automated distribution of the computation tasks [5]. Both iterative and direct methods to solve the integrated pressure flow network should be considered. One possibility is communicate all readily calculated spatially distributed companion model branch values to a dedicated node which could provide for an efficient sparse matrix solution of all the pressure values to be delivered back to the spatial nodes. The other possibility is to apply parallel point Jacobi type of iterative solution of the pressure, whereas the sparse matrix not even would be necessary to solve [6].

3 Direct and iterative sparse solution of companion model based equations

Tearing and lumping procedures require knowledge of the interaction intensity between the variables of heterogeneous real world systems. The dependences f_{ij} between two local variables x_i and x_j and the transition variable w_{ij} are described by the Mechanistic Transition Equation (MTE),

$$f_{ij}\left(x_i, x_j, w_{ij}, \frac{d}{dt}w_{ij}, \int w_{ij}dt, t\right) = 0$$

After suitable spatial discretisation, temporal discretisation, and instant linearization the dependencies can be written as

$$x_i g_{ij} - x_j g_{ji} + s_{ij} - w_{ij} = 0$$

In this Versatile Companion Model (VCM) equation g_{ij} and g_{ji} represent the forward and backward coefficients and s_{ij} the source term. Directed graphs have been considered well suited to describe physical system interactions. The tearing and lumping may allow for simulation of different islands of a homogeneous with different time-steps.

3.1 Graphs may be used to illustrate mechanistic dependencies

Isomorphisms between oriented linear graphs and lumped physical systems were already discussed more than 50 years ago [7]. In Figure 1, a continuously stirred tank with in flows and out flows is shown. The tank is considered as a lumped control volume with the pressure p, specific entalphy of the mixture h and the mass fractions of three included substances c. The horizontal edges denote accumulation of mass, energy and respective concentrations. The edges between the concentration vertices denote chemical reactions. Graphs may be the solution whence developing intelligent modelling tools making use of semantic plant structural specifications and design databases.

3.2 The mechanistic edge and relevant companion model graph

We may consider a specific nonlinear mechanistic dependency f_{ij} between the two nodes x_i and x_j with a simple edge as set forth in Figure 2. The versatile companion model has been developed provide for depicting the instantly, at the end of each iteration step, linearized dependencies. The internal structure

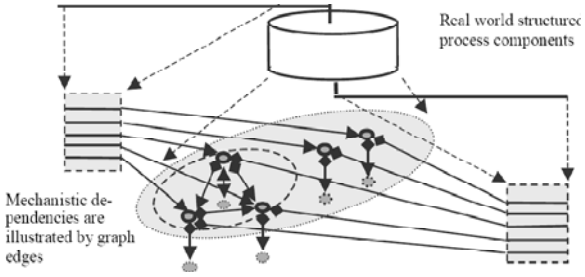


Figure 1. Illustration of real world dependencies by a mechanistic graph.

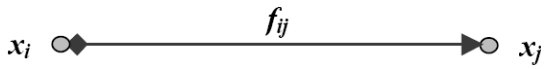


Figure 2. Mechanistic dependency edge.

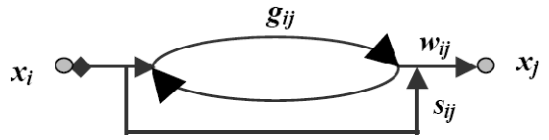


Figure 3. Companion model graph.

of the versatile companion model can be illustrated by three parallel sub-edges representing separate dependencies whereas g_{ij} denotes the forward influence edge, g_{ji} the backward influence edge and s_{ij} the forced source influence edge (Figure 3). The causality of the companion model graph is specified by the relevant coefficients. For instance, the forward and backward coefficients may be different.

3.3 Direct sparse matrix solution

Separate implicit islands are identified from the graph network for the solution of each related set of equations. Whilst no accumulation is considered in a simple vertex we may write for all simple edges that are implicitly connected to node *j* as follows:

$$\sum_i w_{ij} = 0$$

Applying this scheme for all implicitly interconnected simple vertices i the relevant through variables w_{ij} are eliminated and we get the linear matrix equation for solving the internal local state vector $\underline{x_e}$ for the implicit island:

$$A\,\underline{x_e} = \underline{b}$$

It is to be noted that also parallelised sparse matrix solution schemes are available. Fine grain parallelisation on available cores on the solver node may be used in first case. Ehen the state vector is solved we may proceed by solving the through variables w_e from the relevant companion model equations.

3.4 Iterative sparse matrix solution

If the matrix is less sparse and strongly diagonally dominant, it may be suited for iterative solution schemes such as Point Jacobi that are much easier to parallel than sparse direct methods. Temporary elements b_i and d_i are built by looping through the relevant edges connected to the internal node i, whereas j denotes both internal and external nodes, except for node i. The new iteration step value is denoted by k and the old by $k-1$.

$$\mathrm{d_i} = \sum_{\mathrm{j,out}} \mathrm{g_{ij}} + \sum_{\mathrm{j,in}} \mathrm{g_{ji}}$$

$$b_i = \sum_{j,out}(-s_{ij} + g_{ij}x_{j,k-1} + \sum_{j,in}(s_{ji} + g_{ji}x_{j,k-1})$$

The new internal node i values x_i, k are then solved from

$$x_{i,k} = b_i/d_i$$

Note that in this iterative case the matrix equation is not needed to be constructed at all. Neither is there any need for solution order optimisation to avoid fill in of new matrix elements. The success of the iteration loop may be checked by solving the values v_{ij} of each edge from companion model equation and subsequently calculating the mass errors in each node.

4 Multicore processors are now available in ordinary laptop computers

Apros simulation engine has been optimised for running in single core processors with vector processing extensions. A complete power plant model can already be run on a modern PC with 100 ms time-step connected to a DCS system for testing of its functionality before installation at real plant. Of course, such a system may be used for optimising operational procedures, checking of emergency procedures and for training of plant operators, as well. New promising applications include predictive simulators for enhanced control systems or to provide 'what if' evaluation possibilities for the operators. Moore's law has been supposed to continue at least for 10 years with regard to number of transistors that can be placed on an integrated circuit. The obviously slower advances in increasing the single processor's speed will be compensated with multiple cores on the chip. Now we have quad core chips in our laptop computers. In a few years we will have hundreds or even thousands of cores at our disposal. We will face embedded highly parallel processing devices at our ubiquitous service, not only residing in supercomputer centres.

The new challenge is how to benefit from highly parallel computing. There are several bottlenecks. The speed of the main memory has not been increasing. Previously a floating point calculation on a single core computer consumed tens of memory cycles. The on-chip cash memories of modern multicore processors are, however, very fast. Now tens of floating point calculations is to be made on the chip to keep the processor node busy awaiting for next data from common main memory. Affordable compute cluster servers can be equipped with tens of such nodes to gain more virtual computing power. Compiler and

operating system developers have implemented tools to support the take-up of multi-threaded processing: Concurrent threads on each core, threads cores in parallel on the processor nodes, and nodes in parallel on cluster installations. Separate programs run already nicely in parallel, but only using one core each, and with no own speed up. The programs need to be recompiled with parallelisation options to get any benefit from the new hardware at all. Specific comment lines can be included in existing source code to instruct the compiler on such portions of the code that can be computed in parallel. OpenMP standard has been implemented on many platforms [8].

However, to grab the possibilities now soon available, the source codes need to be revised. The program data organisation needs to be re-developed to minimise required cash operations to main memory. The computing algorithms need to be re-developed to promote on chip operations and minimize communication between computer nodes. At the same time the mathematical solver algorithms need to be developed to support multi scale modelling of heterogeneous processes requiring that separate parts of the integrated model are simulated with different time-steps [9]. It seems that Niclas Wirths law: "Software gets slower faster than hardware gets faster", will come true even with regard to scientific computing subject that the software architectures and mathematical methods not are strongly adapted to the hardware developments. New laptops on the shelves of the computer stores are already supplied with multicore processor chips, 64 bit memory access, and 64 bit operating systems. Do not panic, referring to the millennium syndrome, your 32 bit single threaded applications will most likely still run, because traditional 32 bit single thread instruction sets are still directly supported, or at least emulated with some performance degradation. However, if you would take advance of the new platform developments you should act. The scientific problem is, that inherently serial algorithms are not very easy to parallelise. This seems to be a generic challenge. Orchestrated research efforts are definitely required, also at European level.

References

[1] E. Silvennoinen, I. Raiskinen, K. Juslin. *Introduction to APROS Simulation Environment*. ESM'88, European Simulation Multiconference, Nice, France, June 1–3, 1988, 4 pp.

[2] K. Juslin, *A companion Model Approach to Modelling and Simulation of Industrial Processes*. Espoo 2005, VTT Publications 574, ISBN 951-38-6660-2, URL: *http://www.vtt.fi/inf/pdf/*, 155+15 pp.

[3] K. Juslin, R. Lilja, A. Tourunen. *Challenges in Dynamic Simulation of Clean Combustion Power Plants*. PowerPlantSim 2008, Austin, Texas, SCS, March 25-27, 2 pp.

[4] K. Juslin, E. Puska, K. Porkholm. *Are You Ready for New Generation of Nuclear Power Plants? APROS is!* PowerPlantSim 2008 congress, Austin, Texas, SCS, March 25-27, 2 pp.

[5] K. Juslin, *Advanced Graph Based Industrial Process Modelling*. 7th Int. Conference on System Simulation and Scientific Computing, ICSC.2008, October 10-12, Beijing, China, 8p.

[6] K. Juslin, *Simulated Highly Parallel Solution of Versatile Companion Model Graph Equations*. PARA 2008, May 13-16, NTNU, Trondheim, Norway, 4 pp.

[7] H.M. Trent. *Isomorphisms between Oriented linear graphs and Lumped Physical Systems*, Journal Acoustic Society of America, Vol. 27, 1955, pp. 500 – 527.

[8] OpenMP Architecture Review Board. *OpenMP Application Program Interface*. Version 3.0 May 2008. Online: http://www.openmp.org/mp-documents/spec30.pdf , January, 7, 2009.

[9] Intel Corporation. *From a Few Cores to Many: A Tera-scale Computing Research Overview*. Online: http://download.intel.com/research/platform/terascale/terascale_overview_paper.pdf , January 7, 2008.

Corresponding author: Kaj Juslin
VTT Technical Research Centre of Finland
Information and Communication Technologies
P.O.Box 1000, Vuorimiehentie 3, Espoo
FI-02044 VTT, Finland, *kaj.juslin@vtt.fi*

Accepted: MATHMOD 2009
Revised: June 15, 2009
Accepted: July 20, 2009

SOFTWARE NOTES

Modeling Hybrid Systems in MvStudium

D. B. Inichov, Y. B. Kolesov, Yu. B. Senichenkov, Saint Petersburg State Polytechnic Univ., Russia

SNE Simulation Notes Europe SNE 20(1), 2010, 31-34, doi: 10.11128/sne.20.sw.09966

A new version of graphical environment MvStudium 6.0 for modeling and simulation of complex dynamical systems enables to design hierarchical event-driven systems using blocks with oriented (Inputs and Outputs) and non-oriented (Contacts and Flows) connections (oriented and non-oriented blocks), whose internal activity may be described by Behavior Charts (B-Chart is a state machine with continuous Do-activities and without orthogonal states) MvStudium 6.0– is a graphical environment with universal equation-based and UML–based object-oriented modeling language, which graphical form based on B-Charts and hierarchical functional diagrams. The environment supports technology of designing hierarchical models using oriented and nonoriented blocks, that behavior is may be described by hierarchical B-Charts. The environment consists of Model Editor and Virtual Test-bench.

Model Editor has four user's interfaces that sequentially become more complex for different types of models: (1) an isolated classical dynamical system, (2) an isolated hybrid system, (3) a hierarchical model with components from multiple domains, (4) a model with predefined plan of computer experiment. User can build model using his own original blocks or imported blocks from other projects or libraries. A solved system of differential-algebraic equations for each current model mode is formed and analyzed on compilation stage for models with oriented blocks, and it formed on runtime for models with non-oriented blocks. Designed by user model is checked and compiled. A model may run under Virtual Test-bench, may be a standalone executable program, or may be realized as hidden (un-visual model) in the form of DLL for using as a component of more complex model or for parameter optimization with the help of Virtual Test-bench toolbox.

Virtual Test-bench is used for debugging, simulation and computer experiments with visual model.

Introduction

Nowadays tools for modeling and simulation of complex dynamical systems are graphical environments equipped by modeling, planning and doing computer experiment languages used in graphical and textual forms, toolboxes for analysis and synthesis of models and engineering libraries. For short we will call them visual modeling tools (VMT). VMT support different modelling technology that make it possible designing models with components from multiple domains, control and controlled blocks, working in model and real time. It's more than enough to call them universal tools taking in account additionally that they are used for research, designing and leaning. Developers of VMT consider universality also as ability to design models of different types of complexity. A dynamical system is said to be complex if it has:

- *Structural complexity*: a system has hierarchical block structure with blocks from multiple domains.
- *Behavior complexity*: a system in whole and each block may have different modes, in other words they have event-driven behavior.
- *Changeable structure*: structure of a system and number of its blocks depends on time.
- *Large dimension*: number of variables and blocks is so large that it is impossible to realize model without special technologies for large scale systems.

In this article we will speak only about behavior complexity of models and our goal is to demonstrate MvStudium's [3, 4, 5] capability to model hybrid or event-driven systems.

1 'Block-modeling' in MvStudium

Graphical language for MvStudium should to enable users easy designing of:

- hybrid systems (mathematical model – differential or algebraic-differential equations with discontinuous right-hand sides),
- multi-component systems with oriented blocks, that are open hybrid system with Inputs and Outputs,
- multi-component systems with non-oriented blocks, that are open hybrid system with Contacts and Flows,
- multi-component systems with both types blocks.

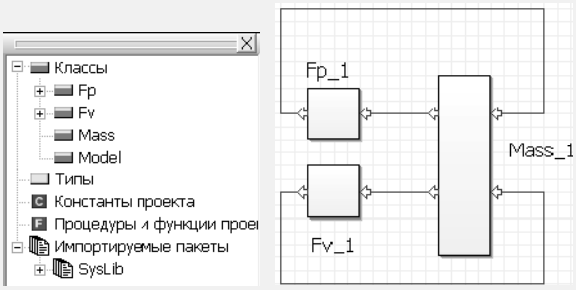


Figure 1a,b. In the project manager (left), there are four classes: Model, Mass, Fp, and Fv. Classes Fp, Fv have two local classes pL, pQube, vL, vQuad thereafter. The class model (right) has type 'Functional diagram'. External variables are of 'Input' and 'Output' types.

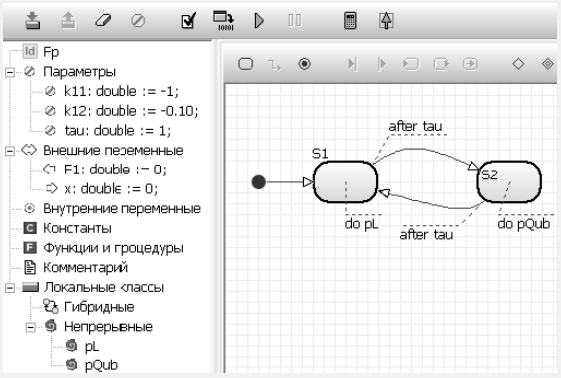


Figure 1c. Fp class anf its B-Chart.

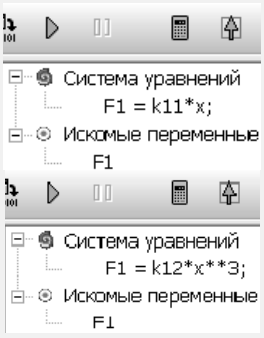


Figure 1d. Equations of local classes pL, pQube.

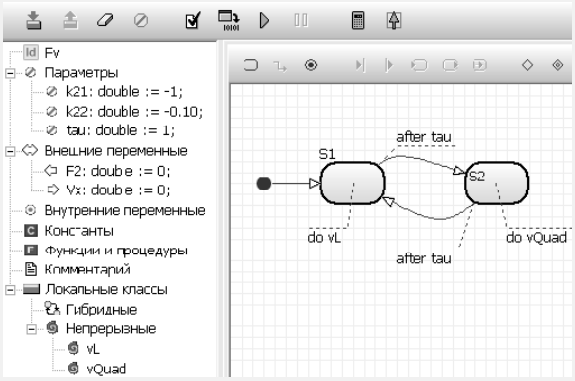


Figure 1e. Fv class and B-Chart.

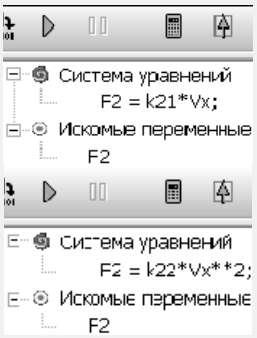


Figure 1f. Equations of local classes vL, vQuad

Hybrid system in MvStudium is tuple HS={**G, Pr, Inv, F**} where

- **G** is state graph (state machine) with usual states and pseudo states ('initial', 'final', 'decision'). Usual state may be simple or composite one. Simple state is a state with behavior defined by system of algebraic-differential equations. Composite state or submachine state is defined by its own state graph. A state may have entry and exit actions. Actions are written in Model Vision Language (MVL) and considered as momentary actions. Orthogonal states are forbidden.
- **Pr** is a set of predicates $\Pr B\colon \mathbb{R}^n \to \{\text{True, False}\} \wedge \Pr R\colon \mathbb{R}^n \to \mathbb{R}^1$, governing conditions of leaving current state and choosing a new current state.
- **Inv** is a set of invariants for states of a graph.
- **F** is a set of algebraic-differential equations governing behavior in usual state. A set **F** includes the element NULL.

Graphical notation for hybrid systems we should call *Behavior Charts*. Thereby, a B-Chart is an UML state machine without orthogonal and historical states and with Do-Activates in form of algebraic-differential equations.

First versions of MvStudium allow designing only hybrid systems and block with oriented blocks (HS_IO). The new one deals with non-oriented blocks (HS_CF).

MVL is object-oriented language. For designing any block system you need to define classes with external variables (input, output, contact, flow) and internal variables – state variables. Variables may be scalar or vectormatrix types. Each class may have a structure and behavior. Behavior is defined by state machine. Singular state machine with one usual state is said to be continuous one and may be present only by system of equation. Systems of algebraic-differential equations from **F** are presented as local classes. You may create your own classes or import them from another projects or libraries. Mechanism of inheritance allows forming trees of classes. Modeling system (class 'Model') may be oriented or non-oriented block or closed block (container block without external vari-

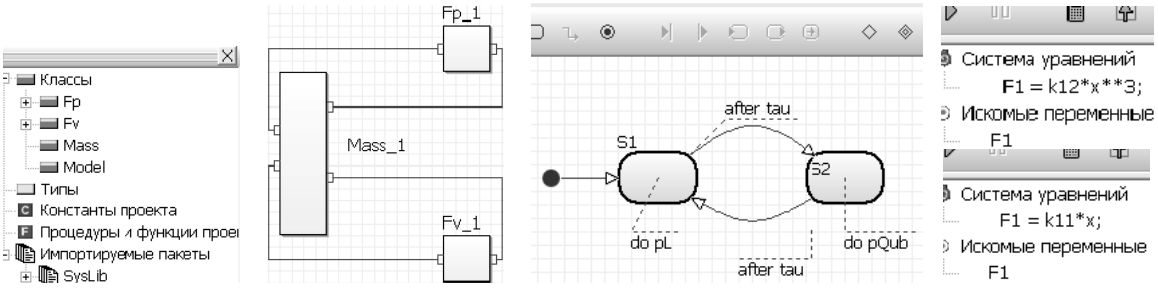


Figure 2. MvStudium project with non-oriented blocks for mechanical system. (a) Project manager with four classes Model, Mass, Fp, and Fv. Classes Fp, Fv have two local classes pL, pQube, vL, vQuad thereafter. (b) Class Model of type 'Functional diagram'. It is used only one type of external variables that is 'contact'. (c) B-Chart of Class Fp. (d) Equations of local classes.

ables). Class 'Model' may have its own B-Chart used for planning computer experiment.

There is no necessity to generate full code for all variants of hybrid behavior for models with oriented blocks. It is enough to have code for all local classes.

Example of a mechanical system. Let us consider mass m, subjects to periodical forces F_1 and F_2 with $2T$ period:

$$m \cdot \frac{d^2x}{dt^2} = F_1 + F_2; \quad x(0) = x_0; \quad x'(0) = x'_0$$

$$F_1 = \begin{cases} k_{11} \cdot x, & [0,\ T] \\ k_{12} \cdot x^3, & [T,2T] \end{cases}; \quad F_2 = \begin{cases} k_{21} \cdot \dfrac{dx}{dt}, & [0,\ T] \\ k_{22} \cdot \left(\dfrac{dx}{dt}\right)^2, & [T,2T] \end{cases}$$

A model with oriented blocks for this case may look like in Figure 1.

2 'Physical Modeling' in MvStudium

MvStudium 6.0 suppots 'physical modeling'. It is possible designing and using non-oriented blocks with Modelica's external variables 'contacts' and 'flows'.

Figure 2 shows fragment of MvStudium project for the examined example of mechanical system realized with the help of non-oriented blocks.

The examined example may be not interesting for users dealing with 'classical' devices of physical modeling such as 'resistor', 'capacity', 'inductance' and other analogous components. Let us consider electrical circuit with an 'ideal' diode that is very simple hybrid system.

In Figure 3 you see Modelica's 'oneport' device that is parent for classes 'resistor', 'capacity', 'ground' and so on. Class 'IdealDiode' is open hybrid system

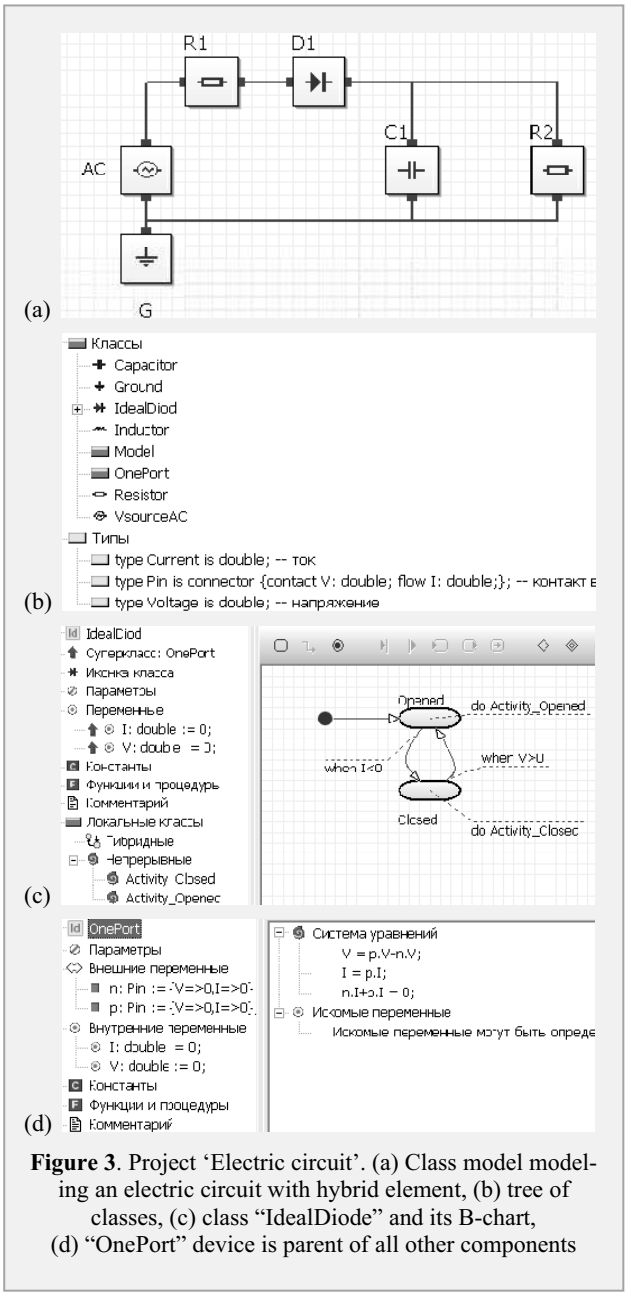


Figure 3. Project 'Electric circuit'. (a) Class model modeling an electric circuit with hybrid element, (b) tree of classes, (c) class "IdealDiode" and its B-chart, (d) "OnePort" device is parent of all other components

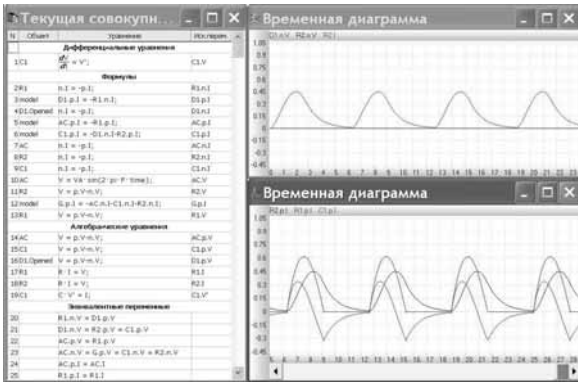


Figure 4. Test-bench windows for project "electric circuit". On the right is the window with solved system of equations divided on formulas, differential and algebraic equations, and equivalent variables.

with external variables 'contact' and 'flow'. Figure 4 demonstrates time diagrams for some components of electric circuit and the special window with final (solved) system of eqution. The solved system depends on time. When current state in B-Chart of "IdealDiode" is changed the solved system is changed too. This window is used while debugging. With the help of debugger it is possible to hold execution in the point of changing current state and check solved system.

3 Computer experiments with models

B-Charts may be used for planning computer experiments. A class Model may have type "functional diagram with its own B-Chart". Imagine that you want to compare two models of mechanical system realized with oriented (InOutModel) and non-oriented (CFModel) blocks (Figure 5). Let us create the class 'Plan' with composite B-Chart for planning experiment. Composite B-Chart has Do-Activity 'Tick-Tack'. 'Tick-Tack' is local class that has B-Chart with two states S1 and S2. Do-Activities of states S1 and S2 are direct instances of classes 'InOutModel' and 'CFModel'.

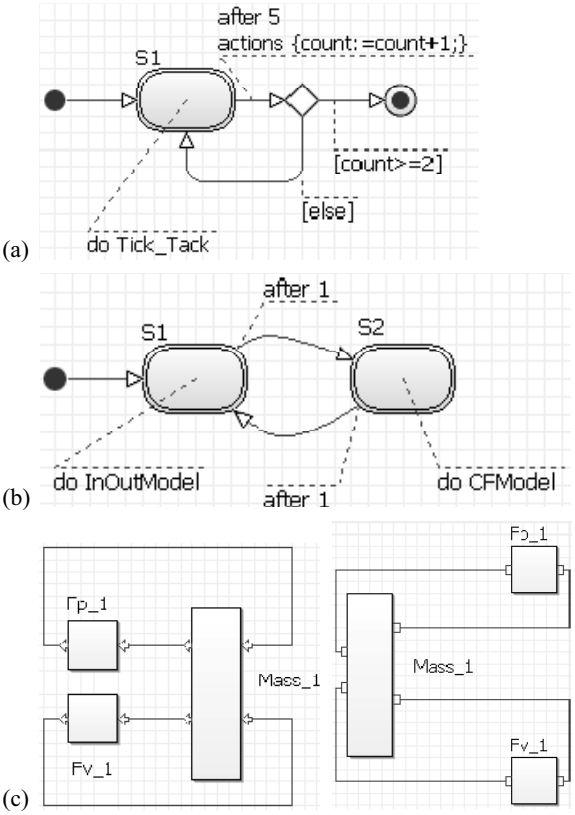


Figure 5. Using B-charts for planning computer experiments. (a) B-chart of highest level which contol number of experiments, (b) B-Chart "Tick-Tack", which periodically switches models under investigation, (c) Do-Activities in form of direct instances of a classes "InOutModel" and "CFModel".

References

[1] Yu.B. Kolesov. *Object-Oriented Modeling of Complex Dynamical Systems*. St. Petersburg: Publishing House of the Polytechnic University, 2004. 239 pp.

[2] Yu.B. Senichenkov. *Numerical Modeling of Hybrid Systems*. St. Petersburg: Publishing House of the Polytechnic University, 2004. 206 pp.

[3] Yu.B. Kolesov, Yu.B. Senichenkov. *Modeling of Systems. Dynamical and Hybrid Systems*. St. Petersburg, BHV, 2006. 224 pp.

[4] Yu.B. Kolesov, Yu.B. Senichenkov. *Modeling of Systems. Object-Oriented Approach*. St. Petersburg, BHV, 2006, 192 pp.

[5] Yu.B. Kolesov, Yu.B. Senichenkov. *Modeling of Systems. Practical Work on Computer Modeling*. St. Petersburg, BHV, 2007, 352 pp.

Corresponding author: Yu. B. Senichenkov,
Polytechnical University Saint Petersburg
Polytechnicheskaj 29, Russia, 195251,
senyb@dcn.infos.ru

Accepted: MATHMOD 2009
Revised: March 20, 2009
Accepted: July 25, 2009

Simulation of Queuing Systems using QS_PN_Simulation Tool

Martina Drozdová, František Zbořil, Faculty of Information Technology, Brno, Czech Republic

{idrozda, zboril}@fit.vutbr.cz SNE

Simulation Notes Europe SNE 20(1), 2010, 35-37, doi: 10.11128/sne.20.sn.09967

The paper deals with a new tool for discrete system simulation called QS_PN_Simulation. This tool allows easy creating of queuing system models as Petri net graphs. Simulation experiments with models can be done and chosen statistic results of simulations can be obtained. The use of this system is demonstrated in the paper: creation of a model of chemical production system, simulation experiments with this model and obtained results are presented here. Supposed extension of proposed tool is then discussed in the conclusion of the paper.

Introduction

This paper deals with one part of a simulation tool that is developing in our department for hybrid/combined (discrete and continuous) simulations. Proposed part of this tool is focused on the discrete simulation only and it allows easy creating of queuing system (QS) models as Petri net (PN) graphs – that is why it was named QS_PN_Simulation tool.

The main idea for QS_PN_Simulation tool design was fact that design using mathematical equations or the other text form is not user friendly. For users a visual design is more appropriate and friendly. Therefore the Petri net was chosen as a main tool for model representation and graphical environment for model building were proposed and implemented. All system has been written in the Java language and it allows a building of the model as Petri net graph, simulation experiments with this model with given parameters and obtained results evaluation.

It can be mentioned proposed QN_PS_Simulation tool has been primarily designed for discrete simulation, but it can be used as a tool for demonstration of Petri net behavior, too.

1 QS_PN_Simulation tool

As was mentioned above the QS-PN-Simulation tool is a new tool for modeling and simulations of discrete systems, it has been implemented in the Java language and it has a user friendly graphical user interface that allows easy creating queuing system models as Petri net graphs.

As is well known Petri net is a discrete mathematic model that allows description of control flows and information dependences inside the modeled system. Modeling of parallelism and manual exclusion is its main positive.

Each model in QS-PN-Simulation tool uses places and transitions as follows:

- *Normal place*: standard place without any special purpose.
- *Queue*: place with requests waiting for service (standard queue).
- *Store*: place with limited capacity.
- *Facility*: place with capacity equal to one.
- *Timed transitions*: there are two types of these transitions: transitions with constant waiting time and transitions with randomly generated waiting time.
- *Immediate transitions*: they always have higher priority than timed transitions. These transitions can be associated with priority or with probability. So, there are three types of immediate transitions:
  - normal immediate transitions,
  - immediate transitions with priority,
  - immediate transitions with probability.

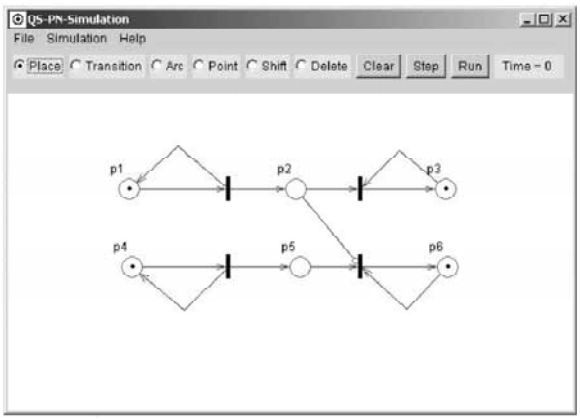
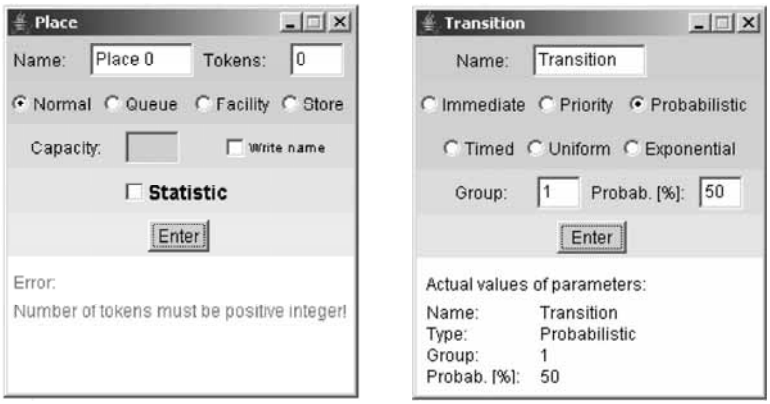


Figure 1. Simple model (main window) in QN_PN_Simulation tool.

SNE Simulation Notes Europe – Print ISSN 2305-9974 | Online ISSN 2306-0271
SNE 20(1), 2010, 1-10 | doi: 10.11128/sne.20.tn.xxxxx

Example of simple model (main window) in the QN_PS_Simulation tool is shown in Figure 1, examples of dialog windows for parameters settings are shown in Figure 2.



(a) (b)

Figure 2. Diaglog windows for parameter settings of (a) place, (b) tranisition.

2 Production system

The (elements of the) system can be in one of four stages:

1. *Filling*: there are four solvents in four tanks that are used to filling reactor. Reactor is filled with solvents in successive steps and time of each filling is 30 minutes. Filling time of tanks are insignificant with respect to time of reaction – that means, that all tanks are full before next filling state.
2. *Reaction*: the reaction begins immediately after the last solvent is poured into reactor, time of reaction is 1080 minutes. After this time, liquid is poured out form the reactor to the stirring tank. Time of discharge of reactor is 60 minutes. When the reactor is empty, it returns to its filling stage immediately.
3. *Stirring*: the process of stirring is done in stirring tank and it takes 360 minutes. Stirrer has breakdown with 3% probability in whichever stage of production process. Time of stirrer repair is 30 - 50 minutes uniformly.
4. *Testing and pouring*: test (60 minutes) follows after stirring and liquid must be stirring again with 10% probability. If the test is successful, liquid is poured out from stirring tank to the storage container. This takes 120 minutes.

Storage container is then put into store. Maximal capacity of store is 20 containers. There is a loader, that immediate loads the container on the deck of cars. Maximal capacity of each car is 3 containers and time of loading is 5 -7 minutes (uniformly). If a car is full, it leaves away. Car goes back to the factory during three till four days.

2.1 Simulation and results

Model of above described production system in QN_PS_Simulation tool (the main window) is shown in Figure 4. The meanings of symbols used are:

S1, …, S4: Solvents 1,…, 4

P1, P2 Start and end of reactor filling

P3 Subnet for filling

P4, P5 Start and end of reaction

P6 Subnet for stirring, testing and breakdown

P7, P14 Start and end of subnet P6

P8, P9 tart and end of stirring

P10 Breakdown of stirrer

P11 Test

P12 Test failed

P13 Test was successful

P14 Liquid is in storage container

P15 Subnet for loading/transport of containers

P16 Start of subnet P15, container is in the store

P17 Car waiting (loading of containers)

P18 Number of containers on the car deck

P19 Container on the deck

P20 Car is ready for departure

The main aim of experiment with the model of chemical production system was a simulation its activity during one year and acquisition new knowledge about real system: number of produced containers, number of stirring tank repairs, number of unsuc-

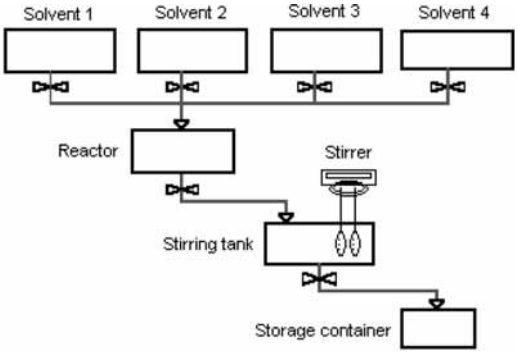


Figure 3. Chemical production system.

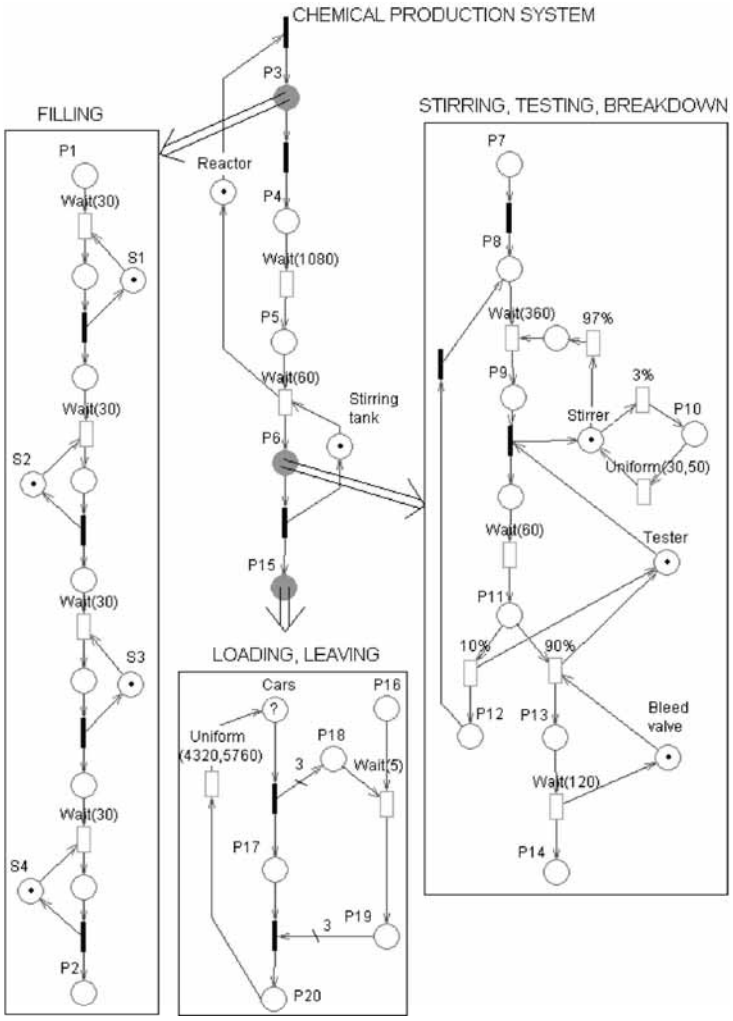


Figure 4. Model of chemical production system in QN_PS simulation tool main window.

cessful stirrings, utilization of individual facilities, optimal number of cars for distribution of chemical liquid, etc. On the basis of simulation results, optimal resolution for acceleration of production could be found.

Obtained simulation results for one year of produce were as follows:

- System produced 415 containers with liquid.
- Stirrer was breakdown two times.
- Number of unsuccessful stirrings was 49.
- Utilization of facilities: reactor: 100 %, stirring tank: 51 %, tester: 5,3 %, bleed valves: 9,5 %.
- Maximal two containers were in store
- Optimal number of cars is two.

If a new reactor was added the results changed:

- System produced 774 containers with liquid.
- Stirrer was breakdown thirty-two times.
- Number of unsuccessful stirring was 96.
- Utilization of facilities: reactor: 100 %, stirring tank: 100 %, tester: 10 %, bleed valves: 18 %.
- Store was overload for two cars (160 containers), the three containers were stored for three cars.
- Optimal number of cars is three.

3 Conclusion

QS-PN-Simulation tool has been tested in several theoretical and practical examples. Results obtained have been compared with those obtained by SIMLIB/C++ that is classical simulation tool. We can state, that correspondence of both results have been very good and that QN_PS_Simulation tool is applicable and usable in practice. Its main advantage is user friendly GUI and we believe that it will be useful for other users, too.

Our next work will be focused on extension of QS-PN-Simulation with analog elements as are integrators, invertors, multipliers etc. Kernel of our tool is designed with aspect of that extension and the only graphical visualization of results has not been solved yet.

References

[1] M. Češka. *Petri nets (in Czech)*, Academic Press CERM Brno, 1994

[2] M. Drozdová. *Discrete simulation system in Java language (in Czech)*, Master thesis, FIT BUT Brno, 2006

Corresponding author: Martina Drozdová,
Faculty of Information Technology,
Božetěchova 2, 61266 Brno, Czech Republic
{idrozda, zboril}@fit.vutbr.cz

Received: October 22, 2008
Accepted: November 21, 2008

ARGESIM BENCHMARKS

Event-based and State-Automata-based Modelling of FMS

A Comparative Case Study of Flexsim, Dymola and Matlab/Stateflow based on the ARGESIM Benchmark C2

Sebastian Schreiber, Mike Barth, René Nicolaus, Markus Schleburg, Alexander Fay
Helmut-Schmidt-University, Institute for Automation Technology, Hamburg, Germany
{sebastian.schreiber; mike.barth; rene.nicolaus; markus.schleburg; alexander.fay}@hsu-hh.de

Simulation Notes Europe SNE 20(1), 2010, 38-44, doi: 10.11128/sne.20.bn02.09968

By implementing the ARGESIM Benchmark C2 "Flexible Assembly System", the simulation system *Flexsim*, the *Matlab/Simulink*-toolbox *Stateflow*, and the equation-based modelling language *Modelica* are compared to each other. Based on the different modelling techniques, the systems will be described and analysed from an automation point of view. Subsequently, the modelling approaches of state-automata (*Stateflow*) as well as object-orientation (*Flexsim*) and equation-based modelling (*Modelica*) are reflected. The analysis includes (1) the time and efforts that are necessary for the modelling process itself, (2) the complexity of the implementation, (3) the possibility to analyse the simulation results, and (4) the possibility to separate control algorithms and controlled system in the implementation.

Introduction

A wide choice of commercial software tools for the simulation of material flow is available on the market (see e.g. [6]). Within this work, the authors analyse how software tools which are commonly used within the automation community can be used for the modelling and simulation of material flow problems. In this context, modelling and simulation methods are important for testing control algorithms without the need of having access to real systems. Therefore, the simulation environment *Flexsim* [1], the *Matlab/Simulink*-Toolbox *Stateflow* [2], and the modelling language *Modelica* [3] implemented in *Dymola* [4] have been selected and applied to the ARGESIM C2 Benchmark "Flexible Assembly System" [5]. There has not been a publication of C2 Benchmark results concerning these tools before.

The article is structured as follows. First, the Benchmark C2 will be introduced by presenting the structure of the target system and its modelling tasks. Subsequently, the different modelling techniques as well as the respective implementations of the C2 Benchmark are explained for Flexsim (Section 2), Modelica (Section 3) and Matlab/Stateflow (section 4). Finally, all three approaches are compared with respect to the required time for modelling, the complexity of the implementation, the possibility to analyse the simulation results, and the possibility to separate the control algorithms from the controlled system. The latter requirement is essential for the test of control algorithms on the model.

This article is an extended and reviewed version of [15], which was presented at the ASIM ST/GMMS-Workshop 2010.

1 Benchmark ARGESIM C2 "Flexible Assembly System"

The benchmark C2 describes a flexible assembly system and has originally been proposed by the ARGESIM in [5]. Its objective is to test different simulation systems with regard to their ability to define and to combine sub-models, as well as to formulate complex control strategies. The VDI guideline 3633 [7] uses this benchmark as an application example for the efficient handling of simulation studies. The average throughput time and the optimal number of pallets are the comparable target results.

The pallets are used to transport single parts through an assembly system, which is shown Figure 1. The number of pallets to be used is one of the parameters that is kept constant during a single simulation run. The system can be separated into eight sub-models placed along a main conveyor belt. Each of these sub-models contains an assembly station.

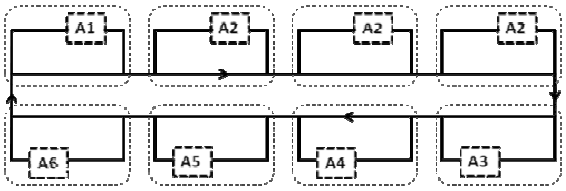


Figure 1. Description of the assembly system.

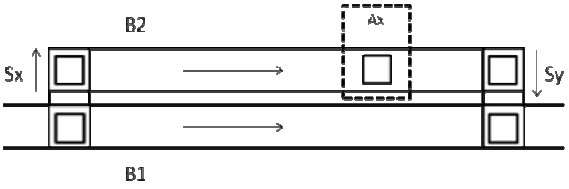


Figure 2. Description of a subsystem.

Station A1 is responsible for unloading finished parts from the pallets and loading new ones. The stations A2 to A6 are each responsible for a predefined processing step. Every part has to pass the processing steps A2 to A5 before it is finished. Station A6 acts as a back-up function and should be used if one of the stations A3 to A5 is busy or down. The processing sequence has to start or end within station A2. The sequence of A3 to A5 is not predefined in this context.

Each subsystem (Figure 2) consists of two conveyor belts `B1` and `B2`, two shifting units `Sx` and `Sy`, and an assembly station `Ax`. If a pallet arriving on `B1` needs to be processed into `Ax`, the shifting unit `Sx` pushes it onto `B2`. If it does not need to access `Ax`, the pallet remains on `B1`. The transportation area between `Sx` and `Ax` on belt `B2` can be used as a buffer. All finished pallets are pushed back to `B1` through `Sy` and have priority to those on `B1`.

2 Simulation environment Flexsim

Flexsim is an object-oriented system for discrete-event simulation and simulation of continuous-flow processes. The available standard library contains more than 40 objects, of which 26 are discrete ones. In May 2010, the current version 5.0 of Flexsim has been released.

All objects refer to one of the so-called "super-classes" `FixedRessources` (e.g. source, sink, or machine) or `TaskExecuter` (e.g. operator, vehicle, or crane). Elements modelling the material flow are called `Flowitems` and form a special class within the system. The objects are connected to each other via input and output ports, where the connection itself represents information and/or material flow.

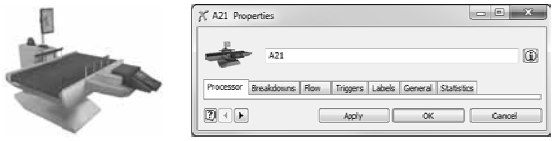


Figure 3. Description (left) and parameterization (right) of a `Processor` object.

Every object allows several possibilities for parameterization. As an example, Figure 3 shows a `Processor` (assembly station A2) and its properties. These properties support all the relevant aspects for modelling a material flow system, such as `Processor`, `Flow`, and `Trigger` (e.g. events `OnEntry` or `OnExit`), and are used to define the element's behaviour. In this context, Flexsim offers pre-defined samples as well as the possibility to define new models within the programming languages C++ or Flexscript. Compared to C++, the use of Flexscript offers the advantage that the simulation model does not have to be compiled before it can be used.

For implementing the benchmark in Flexsim, only standard library elements have been used to allow an efficient traceability. Flexsim itself allows to build own objects and libraries. As an example, Figure 4 shows the implementation of the first assembly station A2. For the sake of visual clarity, the connections between the elements are hidden and the single elements are separated in this Figure. Because material flow is only modelled as an (information) connection in Flexsim, the separation of the elements has no influence on the processing of the simulation model. Figure 4 contains several instances of the objects `Conveyor` and `Processor`. Furthermore, there are some more objects used within the model, e.g. `Combiner`, or `Separator`, which are not displayed here.

For implementing control algorithms, Flexsim supports several possibilities. For example, so-called `Photo Eyes` can be placed on every `Conveyor`, e.g. representing sensor information, or special triggers of an object can be used, e.g. a flow item leaving a con-

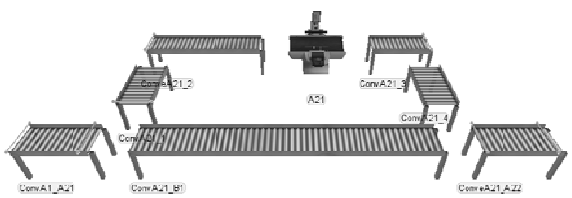


Figure 4. Implementing the first sub-model A2 in Flexsim.

veyor. The listing for the latter possibility is shown in Listing 1. Every `Flowitem` reaching the end of this conveyor triggers the control decision. It represents the decision whether a pallet should be processed in `Ax` (shift to `B2`) or not (stay on `B1`), depending on the pallet's properties, where: (1) is a reference (`treenode`) to the current conveyer `ownerobject(c)` in the global treeview, in which all elements of the simulation model are listed; (2) is a reference to the pallet currently on this conveyor by `parnode(1)`; (3) reads the `target` label on the pallet by `getlablenum(...)`; (4) compares the target information; and (5-6) send the pallet to the defined ports. The example is written in Flexscript, which has a syntax similar to C++.

```
1 treenode current = ownerobject(c);
2 treenode item = parnode(1);
3 int target=getlabelnum(item,"target");
4 if (target==Ax) {
5   return 1;
6 } else return 2;
```

Listing 1. Trigger `Flow-Output-Send to port` of a conveyor in Flexsim.

The parts and pallets are modelled as different `Flowitems`. Each `Flowitem` can carry a number of so-called `Labels`, e.g. information of a RFID tag, which can be analysed and manipulated through every object. The part objects are generated within station A1 and are combined with a pallet for transportation. After being processed on every necessary machine the pallet returns to station A1, in which the part object is separated from the pallet object.

For analyzing the simulation results, Flexsim offers an included module as well as an interface to MS Excel, which was used within this work.

3 Modelica and Dymola

Due to its object-oriented equation-based (OOE) architecture, Modelica is well suited for the modelling of continuous physical systems. By using the module *State-Graph* ([10], [11]) of the Modelica Standard Library (MSL) [9], it is also capable of discrete-event simulation. The Benchmark C2 can be classified as a hybrid model, which represents a combination of both discrete and continuous modelling (e.g. [12], [13]).

Within Modelica, it is possible to define so-called real-world models through known mathematical relationships. By defining interfaces of different variable types (e.g. real, boolean, integer), the combination of

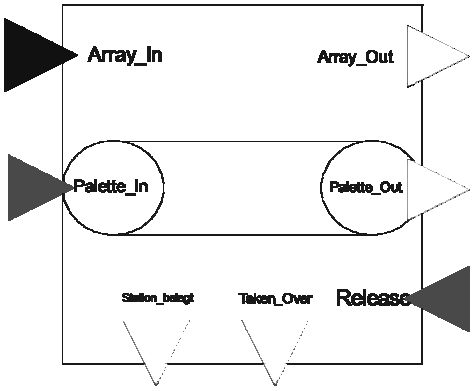


Figure 5. Screenshot of the basic object `Conveyor` in Modelica.

time-based equations and decision routines can be implemented. While applying tests on event-driven models it is often necessary to allow user interaction in parallel to a running simulation. These can be initiated together with the *User-Interaction* (UI) module [9] and a built-in real-time option. The UI module is part of the *MSL*. As an example, a boolean variable (e.g. start of a conveyor) can be changed manually and the user is able to visualize additional feedback variables at the same time (e.g. photo eye on this conveyor).

Following the object-oriented approach of Modelica, the objects of the benchmark can be separated into physical objects (e.g. conveyor, pallets) and decision (control) objects. A combination of physical objects represents the controlled system, while the distributed control decisions are modelled as separated objects. Figure 5 shows the object "conveyor" with its different types of interfaces. There is one array input and one output (SISO) for the exchange of status information of the current pallet. In this context, a capacity of six real variables has been implemented. Furthermore, two interfaces represent boolean type information whether there is a pallet waiting for takeover (input) or is ready for takeover (output). Another input informs the conveyor about the release of a waiting pallet. The output interfaces on the bottom of the conveyor are also of boolean type and represent the status of the object (idle/busy) as well as a trigger signal to the previous object sending a release flag.

The whole modelling of the benchmark is based on the object `Conveyor`. To each object, a range of physical parameters is assigned, e.g. length and conveyor belt speed, processing time, as well as the respective process number in case of representing an assembly station. For modelling the elements `Sx` and `Sy` (see

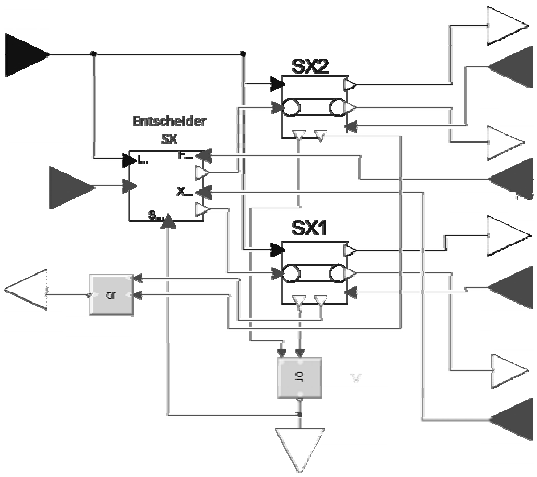


Figure 6. Description of an element `Sx` in Modelica.

Section 1), two `Conveyor` objects and a decision block have been combined to a single object (see Figure 6). For the initialisation phase of each simulation run, a separate module was modelled, which releases a defined number of pallets and later deactivates them.

4 Matlab-toolbox Stateflow

Stateflow is part of the simulation environment *Matlab/Simulink* and supports the modelling and simulation of state automata. A detailed description can be found in [8]. In this context, state-charts are used for modelling state automata whose basic elements are shown in Figure 8. The main modelling elements are states and transitions, which can be grouped as charts or superstates. States can be modelled as exclusive (OR) or parallel (AND) with respect ot their activation within a chart. State transitions are implemented as `event[condition]{condition_action}/transition_action}` and can therefore be triggered by an event and/or a fulfilled condition. In addition, `Truth Tables` can be used for pre-defined control decisions and corresponding actions.

The system behaviour of the benchmark needs to be modelled as enclosed states. For this purpose, the

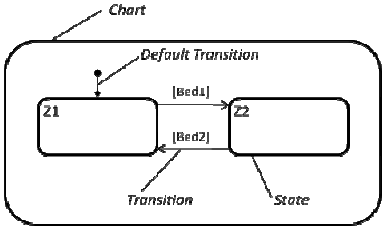


Figure 8: Basic elements in Matlab/Stateflow.

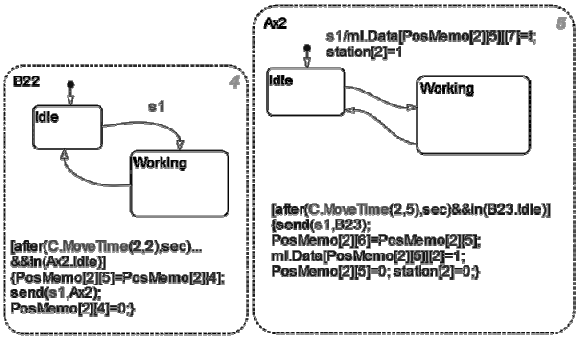


Figure 7. Implementation in *MMatlab/Stateflow.*

conveyors are separated into segments (charts) having the length of a single pallet. Each segment can be empty (`Idle`) or can carry a pallet (`Working`). A pallet receives a unique identifier (ID) which is transported through all segments and is assigned by a global table. Because there is no "physical" pallet object, it is called "virtual" in contrast to the `Flowitems` within Flexsim.

Figure 7 shows the state transition from `Idle` to `Working` within segment `B22` that is triggered trough the event `s1`, which will be activated through the previous segment. The opposite state transition (see Listing 2) consists of: (1) two conditions, where `MoveTime` is a function that generates the processing time for the segment; (2) virtual transport of the pallet, where `PosMemo` is a global 3D-array in Matlab containing the information of all segments in the system (here `[2][4]` describes the segment `B22`), and the status of the current pallet in this segment; (3) release of the event for the next segment `Ax2`; (4) clearing the stored information in `PosMemo` for the segment `B22`.

```
1 [after(MoveTime(2,2),sec)&& in(Ax2.Idle)]
2 {PosMemo[2][5]=PosMemo[2][4];
3 send(s1,Ax2);
4 [PosMemo[2][4]]=0;}
```

Listing 2. Listing of transition `Working` to `Idle` of segment `B22` (see Figure 7).

The structure of a two state segment within one chart can also be used for the assembly station (see Figure 7, right). A segment of an assembly station differs by two aspects from conveyor segments (see Listing 3). First, (4) manipulates the information stored on the current pallet at `PosMemo`, where `[2]=1` indicates that the necessary processing in A2 is done. Second, `station[2]` (6) generates a status information for the assembly station, which is an output variable to *Simulink* and is used for user interaction.

```
1 [after(MoveTime(2,5),sec)&&in(B23.Idle)]
2 {send(s1,B23);
3 PosMemo[2][6]=PosMemo[2][5];
4 ml.Data[PosMemo[2][5]][2]=1;
5 PosMemo[2][5]=0;
6 station[2]=0;}
```

Listing 3. Listing of transition `Working` to `Idle` of segment `Ax2` (see Figure 7).

As already mentioned, Stateflow is embedded into the environment of Matlab/Simulink (see Figure 9). All necessary parameters, e.g. processing times, or the numbers of pallets, are implemented in the form of Simulink inputs. The user interaction in form of displays is shown on the right side. The array `PosMemo` itself is stored on the level of Matlab for later analysis, e.g. by use of Matlab functions.

5 Comparison of modelling approaches

As shown before, the modelling and simulation of the benchmark aspects can be implemented within all three modelling techniques analysed here. The main differences are: (1) the time which was necessary for modelling the system, (2) the complexity of the implementation (3), the possibility to analyse the simulation results, and (4) the possibility to implement the control algorithms and the controlled system separately within the tools. The comparison of these four aspects is done from an automation technology point of view. A brief summary is shown in Table 1 at the end of this section.

There is a considerable difference between the several approaches with respect to the time necessary for modelling (1). Based on the well-suited library for material flow processes, the modelling in Flexsim just comprises the identification of suitable objects, as well as the creation of the necessary connections between them. The library also contains the necessary objects for generating elements (e.g. queue, sink), as well as for combining and separating parts and pallets. The benchmark constraints, e.g. a fraction of pallet capacities for the conveyors, can be implemented correctly. This is different concerning the other approaches. For modelling a conveyor in Modelica or Stateflow, it needs to be split up into an integral number of segments. An object (Modelica) or a state (Stateflow) represents each of these segments, with each of them having the capacity of one pallet. It would be possible to model these segments with a lower capacity, to reach a "continuous-like" behaviour as it is done in Flexsim. Nevertheless, this would

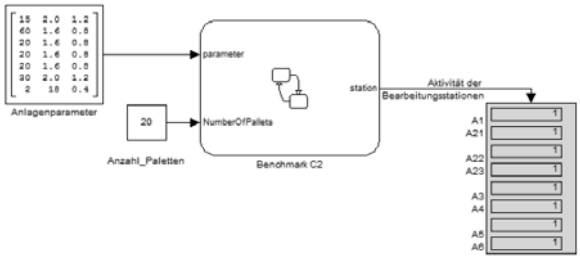


Figure 9. Embedding of *Stateflow* in *Matlab/Stateflow*.

lead to two main problems: First, it would increase the number of objects/states, including more effort for implementation. Second, the model of a pallet would have to be changed in a way that it can be split up and "cover" more than one object/state. This would result in a higher complexity, especially for keeping the inner-coherence of a pallet. Hence, this alternative was not further considered The modelling in Stateflow compromises of using similarly structured charts, which only contain two states and two transitions. When events are sent between the objects, there is no necessity for connecting the charts, because only the recipient needs to be manipulated. This allows a high degree of reuse. For modelling in Modelica, the reuse of objects is achieved by building a basic conveyor object. The effort for modelling this object was higher, compared to the other approaches. Due to the inherited possibility of building up the whole system based on this object, this once-only-effort could be justified.

This modelling has been time consuming for each of the three modelling techniques, but the effort has been well spent: the carefully created objects result in low complexitiy of the C2 Benchmark implementation. The instantiation and parameterization of the objects is the main aspect to be considered. Flexsim, as a commercial simulation software, covers this in an intuitive dialog-based manner. In addition, the already mentioned possibility of describing the control decisions is helpful. Connections between the objects can be implemented through interfaces (ports) on which the control decisions are based on. The implementation in Modelica is mainly based on the combination of previous modelled classes. It is possible to build and reuse a module for representing a whole sub-system, as required in the benchmark description. This can be done by parameterization, e.g. processing and transportation times, or the defined stage in the process. The modules are combined through connecting the pre-defined type-safe inter-

	Flexsim	Modelica/ Dymola	Matlab/ Stateflow
1. necessary time for modelling	object library (+)	basic element (–) combination (+)	state automata (+)
2. implementation complexity	instantiation, connection (+)	instantiation, connection (+)	marking (–) connection (+)
3. ability of analysing	Labels, interface for MS Excel (+)	"virtual" objects fix data structure in early phase (+/–)	„virtual" objects global array (+/–)
4. separation of system and control	combined behaviour-description logic (–)	capsulation of decision modules (+)	separation between Simulink and Stateflow (+)

Table 1: Comparison of the modelling approaches: (+) positive, (+/–) neutral, (–) negative influence.

faces. Before building the connections in Stateflow, all charts have to be placed and clearly marked (ID). Due to the lack of direct connections between the charts and the necessity of a global assignment array, this results in being inflexible, e.g. concerning later revision. The manipulation of the transitions is, in comparison to this, straightforward.

The necessary effort for providing and collecting the relevant information is a very important aspect. Especially the evaluation of the ability to analyze simulation results (3) needs to be considered. Here Flexsim performs well. It is possible to store data within every object (`Label`), esp. on the `Flowitems`. This supports an easy verification of the process variables, as well as a local reasoning and manipulation. In addition, a further development of the models, e.g. extended data collection, can be implemented with few effort. This approach can be realized in Modelica too. However, with respect to information transmission, analyzing, and manipulation, the effort is considerably higher. Furthermore, it is not possible to verify the simulation model without knowing the internal structure of the "virtual" pallet objects. The same problem occurs within Stateflow, where all the relevant information is stored within a global data table. It is important for both approaches, Modelica as well as Stateflow, to define the necessary information before starting the modelling phase. For example, a later extension of the array size in Modelica affects each interface. As already mentioned this is not the case in Flexsim. For analyzing simulations results, each of the three approaches is well suited. A clear preference only depends on the personal aspects and cannot be testified at this point.

Another relevant point is the separation of control algorithms from the controlled system (4), which is especially important in the field of automation. This separation can be implemented in both Modelica and Stateflow. The implementation in Modelica already encapsules the control decisions within the shifting modules. For Stateflow, this could be reached by communicating the relevant information to Matlab/Simulink where the reasoning can be implemented, e.g. using function block diagrams. This is not possible within Flexsim, which is based on the concept of combined behaviour-description logic. This works quite well for acting within this simulation environment but not for testing or rather verifying new control algorithms. In the field of automation technology, these algorithms are usually implemented in different programming languages, e.g. using IEC 61131-3 [14], which are not supported in Flexsim.

6 Summary and Outlook

The modelling approaches as well as the implementation of the benchmark have been described for the three selected simulation environments. The comparison has shown a clear distinction especially in the effort necessary for modelling and for the analysis of the simulation results.

In summary, the three modelling approaches are capable for modelling and simulating the benchmark. The results of the several simulation runs are comparable with those that have already been published.

In further steps, the described benchmark will be extended by a dynamic behaviour with specified stochastic attributes, e.g. breakdowns and changing types of products, which would allow to test new control algorithms regarding to their roubustness. The authors currently develop a new version of the C2 Benchmark and would be thankful for remarks. In addition, the separation of the modelled process and its control should be focused on.

References

[1] Flexsim Software Products: http://www.flexsim.com /products/flexsim/ [last visited: 2010-05-11]

[2] The MathWorks™: http://www.mathworks.de /products/stateflow/ [last visited: 2010-05-11]

[3] Modelica Association: http://www.modelica.org/ [last visited: 2010-05-11]

[4] Dassault Systèmes: http://www.3ds.com/products /catia/portfolio/dymola/ [last visited: 2010-05-11]

[5] ARGESIM: Benchmarks → List of Benchmarks → Flexible Assembly System, http://www.argesim.org/ [last visited: 2010-05-11]

[6] M. Lindemann, S. Schmid. *Simulationswerkzeuge in Produktion und Logistik: Marktübersicht*, PPS Management, Vol. 12 (2), 2007, pp. 48–55. (*in German*)

[7] VDI 3633-1:2000-03: *Simulation von Logistik-, Materialfluß- und Produktionssystemen: Grundlagen*, Beuth Verlag, 2003.

[8] A. Angermann: *Matlab – Simulink – Stateflow. Grundlagen, Toolboxen, Beispiele*, Oldenburg Wissenschaftsverlag GmbH, 2009. (*in German*)

[9] The Modelica Association – Chairman Martin Otter: *Modelica Standard Library 3.1_build5* (released on 2009/12/18). http://www.modelica.org/libraries /Modelica [last visited: 2010-05-11].

[10] Otter M; Arzén K.-E; Dressler, I.: StateGraph - A Modelica Library for Hierarchical State Machines, Proceedings of the 4th Int. Modelica Conference, 2005, pp. 569–578.

[11] J.A. Ferreira, J.P. Estima de Oliveira. *Modelling Hybrid Systems using Statecharts and Modelica*, Proc. 7th IEEE Int. Conference on Emerging Technologies and Factory Automation (ETFA), 1999.

[12] P.J. Mosterman, M. Otter, H. Elmqvist. *Modelling Petri Nets as Local Constraint Equations for Hybrid Systems using Modelica*, Proceedings of the Summer Computer Simulation Conference, 1998, pp. 314–319.

[13] V.S. Prat, A. Urquia, S. Dormido. *ARENALib: A Modelica Library for Discrete-Event System*, Proceedings of the 5th Modelica Conference; Vienna, Austria; September 2006; pp. 539-548.

[14] EN 61131-3:2003: Programmable controllers – Part 3: programming languages (IEC 61131-3:2003). (*or derived national versions*)

[15] S. Schreiber, M. Barth, A. Fay. *Modellierungs- und Simulationsmethoden: Vergleich und Bewertung anhand des Benchmarks ARGESIM C2*. In: Proceedings of "ASIM-Treffen 2010 – STS/GMMS". Ulm, 2010, pp. 290-297. *(in German)*

Corresponding author: Sebastian Schreiber
Helmut-Schmidt-University,
Institute for Automation Technology,
Hamburg, Germany
sebastian.schreiber@hsu-hh.de

Received: March 25, 2010

Accepted: April 5, 2010

Simulation Notes Europe SNE 20(1), 2010, 45-46, doi: 10.11128/sne.20.bn06.09969

A MATLAB-based Solution to ARGESIM Benchmark C6 'Emergency Department' using SimEvents

Gašper Mušič, University of Ljubljana, Slovenia; *gasper.music@fe.uni-lj.si*

Simulator: The popular Simulink time-driven simulation software environment was recently enhanced by SimEvents, an event-driven blockset. The blockset consists of a number of block libraries, supported by a general purpose discrete event simulation engine, but which is embedded in Simulink. SimEvents blocks simulate passing of entities through a network of queues, servers, gates, and switches based on events. Within SimEvents block scheme, entities and events may be generated, and blocks are components that process entities, events, and signals. Communication across blocks is based on both signals and entities. Attributes may be attached to entities, forming a set of entity properties, which may be used to distinguish various types of entities as well as their state during the simulation run. With SimEvents, Simulink grows into a general purpose simulation tool for continuous, discrete-event and hybrid simulation. More information about SimEvents can be found at *http://www.mathworks.com/products/simevents/*.

Model: An emergency department is modelled, where four kinds of patients are admitted. The department comprises a registration, two casualty wards with two doctors each, X-ray room with two X-ray units, and a room where plaster casts are applied or removed. All patients enter the registration, after that their way through the department depends on the severity of their wounds.

Since the model in SimEvents is designed within the graphical user interface, the topological layout of the patient's way through the department is directly mapped into a simulation block scheme. Figure 1 shows a screenshot of the Simulink GUI with SimEvents library. An initial phase of the model development is also shown in the Figure, i.e. the generation of entities that correspond to arriving patients and model of their way through registration facility. The *Start Timer* block is used to attach a stopwatch timer

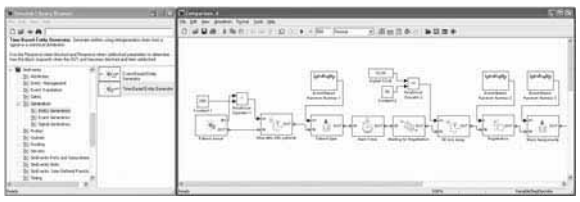


Figure 1. Screenshot of SimEvents.

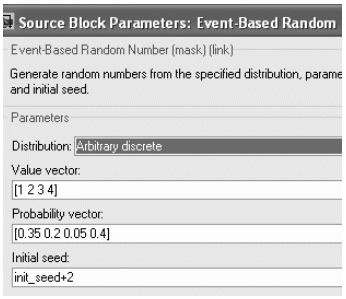


Figure 2. Percent distribution of patients over four types.

to every patient entering the department. The figure shows how the entity generation is stopped after 250 patients and also how the admittance to the registration facility is only allowed after 30 minutes. This way, the requirement about a half hour difference between the time when patients start arriving and the time when doctors start working is modelled. An *Enable Gate* block is employed in both cases.

Statistical parameters are assigned in two ways. Depending on the block and parameter needed, one kind of parameters can be assigned directly via block dialogs, such as the distribution of time between arrivals of patients. Another kind of statistical parameters can be assigned using a random generator block connected to a signal input of an entity processing block. Figure 1 shows several examples of such an assignment, e.g., a patient type is attached as an attribute to the entity representing the patient. The value of the attribute is determined by a random generator, parameters of which are defined through the dialog as shown in Figure 2. In the given case, an arbitrary discrete distribution is used with the values that comply with the problem definition. The `initial_seed` variable is used to enable a number of simulation runs with variable initial conditions of all random generators in the model.

The routing of entities through the model is achieved by the *Output Switch* blocks, where attribute based switching rule is chosen. Path Combiner blocks are used when necessary to merge entities entering a block from different sources. This way the travel path of a specific type of patients through the simulation scheme can be easily followed - illustrated in Fig. 3 showing model part with X-ray Room and Plaster Room.

To improve the readability of the model, a subsystem generation feature of Simulink is used as an abstraction mechanism. This way, individual department facilities are mapped to subsystems, such as X-ray

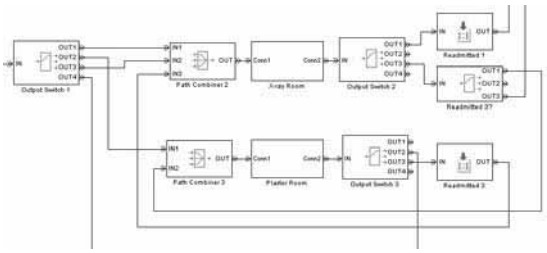


Figure 3. Routing of patients.

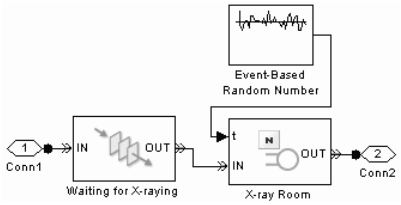


Figure 4. X-ray Room subsystem.

Room and Plaster Room subsystems. The contents of such a subsystem are shown in Figure 4.

To enable statistical evaluation, required quantities are collected in the simulation model by the use of Read Timer and Discrete Event Signal to Workspace blocks. The statistical analysis is then performed by the use of a simple m-script in MATLAB.

A-Task: In SimEvents, a single simulation run is started in a standard Simuling fashion, i.e., within the GUI by pressing the "Start simulation" button, by choosing "Start" in the Simulation menu or by presing Ctrl+T. To obtain results that are representative, however, a number of simulation runs must be initiated with varying initial seed of random generators. This is automated within the MATLAB script that is sketched here:

```
system_name='Comparison_6';
load_system(system_name);
nruns = 100;
w = zeros(nruns,6);
h = waitbar(0,'Please wait...');
seedarray = ceil(rand(nruns,1)*999999)*2+1;
for k = 1:nruns
   waitbar(k/nruns,h);
   init_seed = seedarray(k);
   sim(system_name);
   w(k,1) = mean(...)
            ...
   w(k,5) = std(...);
   w(k,6) = max(...);
end
close(h);
result = mean(w);
```

The average results of 100 simulation runs are shown in Table 1.

Mean Time	Task A	Task B	Task C
Patient 1	223	237	166
Patient 2	146	155	164
Patient 3	242	248	177
Patient 4	135	144	159
Std. Dev.	82	94	75
Ov. treat.time	382	420	381
Close hour	13:52	14:30	13:51

Table 1. Results for tasks a - c: mean of treatment times, standard deviation and closing time.

B-Task: The Casualty Ward model is changed in a way which enables switching between two alternative configurations. Additional *N-Server* and *Output Switch* blocks are inserted to each Casualty Ward subsystem and the switching signal depends on a number of entities waiting in the queue before Casualty Ward 2. A *Discrete Event Subsystem* block implementing simple set/reset signal logic is used to generate the switching signal.

This strategy yields an increase of treatment times for all types of patients, for the standard deviation and also for the overall treatment time (see Table 1).

C-Task: Priority ranking is achieved by inserting separate queues for re-admitted patients before each server in Casualty Ward subsystems. The Path Combiner block, which accepts entities from both queues and outputs them to the server is then configured to accept entities from "re-admitted" queue with higher priority. The Results (Table 1) show a decrease in treatment time for patients of type 1 and 3, and an increase for the others. The standard deviation also decreases while the overall treatment time does not change significantly.

Resumé: The emergency department problem is well suited to simulation with the SimEvents blockset. Patients passing through the department are naturally mapped to flow of entities through the blocks of simulation scheme. Decisions are easily modelled by the use of *Output Switch* blocks where the path of the entity is chosen in dependence of an attribute value. The integration into MATLAB allows for advanced statistical evaluation of the simulation results.

Corresponding Author: Gašper Mušič,
Fac.Electrical Engineering, University of Ljubljana
Tržaška 25, 1000 Ljubljana, Slovenia
gasper.music@fe.uni-lj.si

Received: March 14, 2009 (Rev.Dec.10,2009)
Accepted: January 10, 2010

SNE News Section

Data & Quick Info

Contents

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.

eSNE - SNE at Web SNE issues are also available at `www.argesim.org` as eSNE – *Electronic SNE* . Web-resolution eSNEs are free for download. Subscribers, e.g. members of EUROSIM Societies have access to SNE Archive with high-resolution eSNE copies and and with sources of ARGESIM Benchmarks.

This *EUROSIM Data & Quick Info* compiles data from EUROSIM and EUROSIM societies: addresses, weblinks, officers of societies with function and email, to be published regularly in SNE issues – independent of individual reports of the societies.

INFO:

→ `www.sne-journal.org`
✉ `office@sne-journal.org`
→ `www.eurosim.info`

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`.

Editorial Information/Impressum - see front cover

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.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 thirteen full members and one observer member:

ASIM	Arbeitsgemeinschaft Simulation *Austria, Germany, Switzerland*
CEA-SMSG	Spanish Modelling and Simulation Group *Spain*
CROSSIM	Croatian Society for Simulation Modeling *Croatia*
CSSS	Czech and Slovak Simulation Society *Czech Republic, Slovak Republic*
DBSS	Dutch Benelux Simulation Society *Belgium, Netherlands*
FRANCOSIM	Société Francophone de Simulation *Belgium, France*
HSS	Hungarian Simulation Society *Hungary*
ISCS	Italian Society for Computer Simulation *Italy*
LSS	Latvian Simulation Society *Latvia*
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*
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	Mikuláš Alexík (CSSS), *alexik@frtk.fri.utc.sk*
Past president	Borut Zupančič (SLOSIM) *borut.zupancic@fe.uni-lj.si*
Secretary	Peter Fritzson (SIMS) *petfr@ida.liu.se*
Treasurer	Felix Breitenecker (ASIM) *felix.breitenecker@tuwien.ac.at*
SNE Repres.	Felix Breitenecker *felix.breitenecker@tuwien.ac.at*

SNE – Simulation News Europe. 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*. EUROSIM societies are offered to distribute to their members the journal *Simulation News Europe* (SNE) as official membership journal. SNE Publisher are EUROSIM, ARGESIM and ASIM.

Editor-in-chief	Felix Breitenecker *felix.breitenecker@tuwien.ac.at*

→ www.sne-journal.org, menu SNE

✉ office@sne-journal.org

EUROSIM Congress. EUROSIM is running the triennial conference series EUROSIM Congress. The congress is organised by one of the EUROSIM societies. EUROSIM 2010 will be organised by CSSS in Prague, September 5-10, 2010.

Chairs EUROSIM 2010	Miroslav Šnorek (CSSS) *snorek@fel.cvut.cz* Mikulas Alexik (CSSS) *alexik@frtk.utc.sk*
Organisation EUROSIM 2010	*chairs@eurosim2010.org* *info@eurosim2010.org* *actionm@action-m.com*
Information CSSS	Mikulas Alexik (CSSS) *alexik@frtk.utc.sk*

→ www.eurosim2010.org

ASIM
German Simulation Society
Arbeitsgemeinschaft Simulation

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.asim-gi.org with members' area

info@asim-gi.org, admin@asim-gi.org

ASIM – Inst. f. Analysis and Scientific Computing
Vienna University of Technology
Wiedner Hauptstraße 8-10, 1040 Vienna, Austria

ASIM Officers

President	Felix Breitenecker *felix.breitenecker@tuwien.ac.at*
Vice presidents	Sigrid Wenzel, *s.wenzel@uni-kassel.de*
	T. Pawletta, *pawel@mb.hs-wismar.de*
Secretary	Anna Mathe, *anna.mathe@tuwien.ac.at*
Treasurer	I. Bausch-Gall, *Ingrid@Bausch-Gall.de*
Membership affairs	S. Wenzel, *s.wenzel@uni-kassel.de*
	W. Maurer, *werner.maurer@zhwin.ch*
	I. Bausch-Gall, *Ingrid@Bausch-Gall.de*
	F. Breitenecker, *felix.breitenecker@tuwien.ac.at*
Universities / Research Inst.	S. Wenzel, *s.wenzel@uni-kassel.de*
	W. Wiechert, *W.Wiechert@fz-juelich.de*
	J. Haase, *Joachim.Haase@eas.iis.fraunhofer.de*
	Katharina Nöh, *k.noeh@fz-juelich.de*
Industry	S. Wenzel, *s.wenzel@uni-kassel.de*
	K. Panreck, *Klaus.Panreck@hella.com*
Conferences	Klaus Panreck *Klaus.Panreck@hella.com*
	A. Gnauck, *albrecht.gnauck@tu-cottbus.de*
Publications	Th. Pawletta, *pawel@mb.hs-wismar.de*
	Christina Deatcu, *christina.deatcu@hs-wismar.de*
	F. Breitenecker, *felix.breitenecker@tuwien.ac.at*
Repr. EUROSIM	F. Breitenecker, *felix.breitenecker@tuwien.ac.at*
	N. Popper, *niki.popper@drahtwarenhandlung.at*
Education / Teaching	Ch. Deatcu, *christina.deatcu@hs-wismar.de*
	N. Popper, *niki.popper@drahtwarenhandlung.at*
	Katharina Nöh, *k.noeh@fz-juelich.de*
International Affairs	H. Szczerbicka, *hsz@sim.uni-hannover.de*
	O. Rose, *Oliver.Rose@tu-dresden.de*
Editorial Board SNE	T. Pawletta, *pawel@mb.hs-wismar.de*
	Ch. Deatcu, *christina.deatcu@hs-wismar.de*
Web EUROSIM	Anna Mathe, *anna.mathe@tuwien.ac.at*

Last data update December 2009

ASIM Working Groups. ASIM, part of GI - Gesellschaft für Informatik, is organised in Working Groups, dealing with applications and comprehensive subjects:

ASIM Working Groups

GMMS	Methods in Modelling and Simulation Peter Schwarz, *schwarz@eas.iis.fhg.de*
SUG	Simulation in Environmental Systems Wittmann, *wittmann@informatik.uni-hamburg.de*
STS	Simulation of Technical Systems H.T.Mammen, *Heinz-Theo.Mammen@hella.com*
SPL	Simulation in Production and Logistics Sigrid Wenzel, *s.wenzel@uni-kassel.de*
SVS	Simulation of Transport Systems U. Brannolte, *Brannolte@bauing.uni-weimar.de*
SBW	Simulation in OR C. Böhnlein, *boehnlein@wiinf.uni-wuerzburg.de*
EDU	Simulation in Education/Education in Simulation Katharina Nöh, *k.noeh@fz-juelich.de*

CROSSIM – Croatian Society for Simulation Modelling

CROSSIM-*Croatian Society for Simulation Modelling* 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.

→ www.eurosim.info

vdusak@foi.hr

CROSSIM / Vesna Dušak
Faculty of Organization and
Informatics Varaždin, University of Zagreb
Pavlinska 2, HR-42000 Varaždin, Croatia

CROSSIM Officers

President	Vesna Dušak, *vdusak@foi.hr*
Vice president	Jadranka Božikov, *jbozikov@snz.hr*
Secretary	Vesna Bosilj-Vukšić, *vbosilj@efzg.hr*
Executive board members	Vlatko Čerić, *vceric@efzg.hr*
	Tarzan Legović, *legovic@irb.hr*
Repr. EUROSIM	Vesna Dušak, *vdusak@foi.hr*
Edit. Board SNE	Vesna Dušak, *vdusak@foi.hr*
Web EUROSIM	Jadranka Bozikov, *jbozikov@snz.hr*

Last data update March 2009

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*). 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.

→ www.fit.vutbr.cz/CSSS

snorek@fel.cvut.cz

✉ CSSS / Miroslav Šnorek, CTU Prague
FEE, Dept. Computer Science and Engineering,
Karlovo nam. 13, 121 35 Praha 2, Czech Republic

CSSS Officers

President	Miroslav Šnorek, *snorek@fel.cvut.cz*
Vice president	Mikuláš Alexík, *alexik@frtk.fri.utc.sk*
Treasurer	Evžen Kindler, *ekindler@centrum.cz*
Scientific Secr.	A. Kavička, *Antonin.Kavicka@upce.cz*
Repr. EUROSIM	Miroslav Šnorek, *snorek@fel.cvut.cz*
Deputy	Mikuláš Alexík, *alexik@frtk.fri.utc.sk*
Edit. Board SNE	Mikuláš Alexík, *alexik@frtk.fri.utc.sk*
Web EUROSIM	Petr Peringer, *peringer@fit.vutbr.cz*

Last data update December 2008

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, pierreva@imfa.fr*
- Pole Modelling and simulation of continuous systems. Pole Contact: *Yskandar Hamam, y.hamam@esiee.fr*

→ www.eurosim.info

y.hamam@esiee.fr

✉ FRANCOSIM / Yskandar Hamam
Groupe ESIEE, Cité Descartes,
BP 99, 2 Bd. Blaise Pascal,
93162 Noisy le Grand CEDEX, France

FRANCOSIM Officers

President	Yskandar Hamam, *y.hamam@esiee.fr*
Treasurer	François Rocaries, *f.rocaries@esiee.fr*
Repr. EUROSIM	Yskandar Hamam, *y.hamam@esiee.fr*
Edit. Board SNE	Yskandar Hamam, *y.hamam@esiee.fr*

Last data update April 2006

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 with affiliated societies.

→ www.eurosim.info

a.w.heemink@its.tudelft.nl

✉ DBSS / A. W. Heemink
Delft University of Technology, ITS - twi,
Mekelweg 4, 2628 CD Delft, The Netherlands

DBSS Officers

President	A. Heemink, *a.w.heemink@its.tudelft.nl*
Vice president	W. Smit, *smitnet@wxs.nl*
Treasurer	W. Smit, *smitnet@wxs.nl*
Secretary	W. Smit, *smitnet@wxs.nl*
Repr. EUROSIM	A. Heemink, *a.w.heemink@its.tudelft.nl*
Deputy	W. Smit, *smitnet@wxs.nl*
Edit. Board SNE	A. Heemink, *a.w.heemink@its.tudelft.nl*

Last data update April 2006

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.

→ www.eurosim.info

javor@eik.bme.hu

✉ HSS / András Jávor,
Budapest Univ. of Technology and Economics,
Sztoczek u. 4, 1111 Budapest, Hungary

HSS Officers

President	András Jávor, *javor@eik.bme.hu*
Vice president	Gábor Szűcs, *szucs@itm.bme.hu*
Secretary	Ágnes Vigh, *vigh@itm.bme.hu*
Repr. EUROSIM	András Jávor, *javor@eik.bme.hu*
Deputy	Gábor Szűcs, *szucs@itm.bme.hu*
Edit. Board SNE	András Jávor, *javor@eik.bme.hu*
Web EUROSIM	Gábor Szűcs, *szucs@itm.bme.hu*

Last data update March 2008

PSCS – Polish Society for Computer Simulation - update

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 257 members.

→ www.ptsk.man.bialystok.pl

leon@ibib.waw.pl

✉ PSCS / Leon Bobrowski, c/o IBIB PAN, ul. Trojdena 4 (p.416), 02-109 Warszawa, Poland

PSCS Officers

President	Leon Bobrowski, *leon@ibib.waw.pl*
Vice president	Andrzej Grzyb, Tadeusz Nowicki
Treasurer	Z. Sosnowski, *zenon@ii.pb.bialystok.pl*
Secretary	Zdzislaw Galkowski, *Zdzislaw.Galkowski@simr.pw.edu.pl*
Repr. EUROSIM	Leon Bobrowski, *leon@ibib.waw.pl*
Deputy	A.Chudzikiewicz, *ach@it.pw.edu.pl*
Edit. Board SNE	Z.Sosnowski, *zenon@ii.pb.bialystok.pl*
PSCS Board Members	R. Bogacz , Z. Strzyzakowski Andrzej Tylikowski

Last data update March 2009

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.

→ www.eurosim.info

Mario.savastano@uniina.at

✉ ISCS / Mario Savastano, c/o CNR - IRSIP, Via Claudio 21, 80125 Napoli, Italy

ISCS Officers

President	M. Savastano, *mario.savastano@unina.it*
Vice president	F. Maceri, *Franco.Maceri@uniroma2.it*
Repr. EUROSIM	F. Maceri, *Franco.Maceri@uniroma2.it*
Edit. Board SNE	M. Savastano, *mario.savastano@unina.it*

Last data update April 2005

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).

→ www.scansims.org

esko.juuso@oulu.fi

✉ SIMS / SIMS/Esko Juuso, Department of Process and Environmental Engineering, 90014 Univ.Oulu, Finland

SIMS Officers

President	Esko Juuso, *esko.juuso@oulu.fi*
Treasurer	Vadim Engelson, *vaden@ida.liu.se*
Repr. EUROSIM	Esko Juuso, *esko.juuso@oulu.fi* Erik Dahlquist *erik.dahlquist@mdh.se*
Edit. Board SNE	Esko Juuso, *esko.juuso@oulu.fi*
Web EUROSIM	Vadim Engelson, *vaden@ida.liu.se*

Last data update December 2009

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 approaches to problem solving in industrial as well as in academic environments by establishing communication and cooperation among corresponding teams.

→ www.slosim.si

slosim@fe.uni-lj.si

✉ SLOSIM / Rihard Karba, Faculty of Electrical Engineering, University of Ljubljana, Tržaška 25, 1000 Ljubljana, Slovenia

SLOSIM Officers

President	Rihard Karba, *rihard.karba@fe.uni-lj.si*
Vice president	Leon Žlajpah, *leon.zlajpah@ijs.si*
Secretary	Aleš Belič, *ales.belic@fe.uni-lj.si*
Treasurer	Milan Simčič, *milan.simcic@fe.uni-lj.si*
Repr. EUROSIM	Rihard Karba, *rihard.karba@fe.uni-lj.si*
Deputy	B. Zupančič, *borut.zupancic@fe.uni-lj.si*
Edit. Board SNE	Rihard Karba, *rihard.karba@fe.uni-lj.si*
Web EUROSIM	Milan Simcic, *milan.simcic@fe.uni-lj.si*

Last data update December 2009

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.

→ www.uksim.org.uk

✉ david.al-dabass@ntu.ac.uk

✉ UKSIM / Prof. David Al-Dabass
Computing & Informatics,
Nottingham Trent University
Clifton lane, Nottingham, NG11 8NS
United Kingdom

UKSIM Officers

President	David Al-Dabass, *david.al-dabass@ntu.ac.uk*
Secretary	A. Orsoni, *A.Orsoni@kingston.ac.uk*
Treasurer	B. Thompson, *barry@bjtcon.ndo.co.uk*
Membership chair	K. Al-Begain, *kbegain@glam.ac.uk*
Univ. liaison chair	R. Cheng, *rchc@maths.soton.ac.uk*
Repr. EUROSIM	Richard Zobel, *r.zobel@ntlworld.com*
Edit. Board SNE	Richard Zobel, *r.zobel@ntlworld.com*

Last data update March 2009 (partially)

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 the group.

→ www.cea-ifac.es/wwwgrupos/simulacion

→ simulacion@cea-ifac.es

✉ CEA-SMSG / María Jesús de la Fuente,
System Engineering and AutomaticControl department,
University of Valladolid,
Real de Burgos s/n., 47011 Valladolid, SPAIN

CAE - SMSG Officers

President	María J. la Fuente, *maria@autom.uva.es*
Repr. EUROSIM	Emilio Jiminez, *emilio.jiminez@unirioja.es*
Edit. Board SNE	Emilio Jiminez, *emilio.jiminez@unirioja.es*

Last data update March 2009

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.

→ briedis.itl.rtu.lv/imb/

✉ merkur@itl.rtu.lv

✉ LSS / Yuri Merkuryev, Dept. of Modelling
and Simulation Riga Technical University
Kalku street 1, Riga, LV-1658, LATVIA

LSS Officers

President	Yuri Merkuryev, *merkur@itl.rtu.lv*
Repr. EUROSIM	Yuri Merkuryev, *merkur@itl.rtu.lv*
Edit. Board SNE	Yuri Merkuryev, *merkur@itl.rtu.lv*

Last data update December 2008

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 Moldavia.

→ www.ici.ro/romsim/

✉ sflorin@ici.ro

✉ ROMSIM / Florin Stanciulescu,
National Institute for Research in Informatics, Averescu Av. 8 – 10, 71316 Bucharest, Romania

ROMSIM Officers

President	Florin Stanciulescu, *sflorin@ici.ro*
Vice president	Florin Hartescu, flory@ici.ro Marius Radulescu, *mradulescu@ici.ro*
Secretary	Zoe Radulescu, *radulescu@ici.ro*
Repr. EUROSIM	Florin Stanciulescu, *sflorin@ici.ro*
Deputy	Florin Hartescu, *flory@ici.ro*
Edit. Board SNE	Florin Stanciulescu, *sflorin@ici.ro*

Last data update March 2009

ASIM – German Simulation Society

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 40 institutional or industrial members (plus about about 300 affiliated members). The electronic *ASIM Newsletter* (three times a year) can be downloaded freely from ASIM website www.asim-gi.org. Organisational details / contacts see *Societies Short Info* before.

SNE - Simulation News Europe. ASIM is publishing together with EUROSIM and ARGESIM the journal SNE, which is regularly sent to all ASIM members (as part of their membership; 900 issues) and spread for promotion purposes at conferences (300 issues). The electronic version eSNE is available to all members in high-resolution quality.

ASIM Books / ASIM Notes. ASIM co-operates with Springer Verlag (Berlin), with Shaker Verlag (Aachen), and with ARGESIM Publishers (Vienna University of Technology), in publication of book series (Fortschritte in der Simulationstechnik - Frontiers in Simulation and Fortschrittsberichte Simulation - Advances in Simulation) and in publication of Proceedings. The trademark *ASIM Mitteilungen* (*ASIM Notes*) stands for all publications of ASIM and ASIM Working Groups, in order to mark publications within the 'ASIM environment'. At present (2008-2010) an extended review is undertaken in order to classify all publications since 1982, and to make them electronically available for ASIM members, in full form or abstract form.

ASIM Working Groups. ASIM is part of GI - Gesellschaft für Informatik (Society for Informatics) and is itself structured into working groups (WG), which address various areas of modelling and simulation.

ASIM Conferences. ASIM organises the conference series *Symposium Simulation Technique* (also known as previously annual ASIM Conference), the ASIM working groups organise annual workshops (up to 150 participants) and bi-annual conferences (more than 150 participants. ASIM cooperates in organising the tri-annual EUROSIM Congress and other EUROSIM and SCS conferences.

Furthermore, ASIM co-organises local conferences, e.g. the bi-annual ASIM Wismar Workshop, and the three-annual conference series MATHMOD – Mathematical Modelling in Vienna. A big success is the *ASIM Dedicated Conference on Simulation in Production and Logistics*, organised each second year by the WG Simulation in Production and Logistics.

Upcoming Conferences.

At present, the following conferences and workshops with ASIM as organizer, co-organizer or co-sponsor are scheduled for 2010 and 2011:

- EUROSIM 2010 - 7th EUROSIM Congress, Sept. 5 – 9, 2010, Prague, Czech Republic
- 14th ASIM SPL Conference Simulation in Production and Logistics, October 7 – 8, 2010, Karlsruhe, Germany
- 14th ASIM SUG Workshop Modelling and Simulation of Ecosystems, October 27 -29, 2010, Usedom, Germany; luther@tu-cottbus.de
- ASIM 2011 – 21st Symposium Simulation Technique, Sept . 7 – 9, 2011, Winterthur, Schweiz http://elearning.zhaw.ch/moodle/course/view.php?id=3313; www.asim-gi.org

EUROSIM 2010 CONGRESS

7th EUROSIM Congress on Modeling and Simulation

September 5-10, Prague, Czech Republik

www.eurosim2010.org

In 2010, ASIM is co-sponsoring / co-organising EUROSIM 2010 in Prague. Therefore ASIM does not organise the general ASIM conference Symposium Simulation Technique in 2010. ASIM members are cordially invited to take part at EUROSIM 2010, and they are specially invited to take part at the ASIM Special Sessions, which are scheduled for September 9 - 10, 2010.

ASIM is organising with SLOSIM, the Slovenian Simulation Society, special sessions at EUROSIM 2010. These ASIM/SLOSIM Special Sessions are scheduled for September 9 - 10, 2010, in order to facilitate industry people to attend the congress:

- Physical Modelling, Control and Model Exchange – I. Bausch-Gall, P. Schwarz, F. Breitenecker, T. Pawletta
- Advanced and Comparative Approaches in Modelling and Simulation – T. Pawletta, Ch. Deatcu, F. Breitenecker, N. Popper, P. Schwarz

- Education in Simulation / Simulation in Education - ASIM/SLOSIM Special Session; M. Atanasijevic-Kunc, Univ. Ljubljana; F. Breitenecker, Vienna University of Technolgy

ASIM members organize additionally the following Special Sessions:

- Scenarios and Models for Future Transportation; J. Wittmann, D. P. F. Moller, Univ. Hamburg
- Modelling and Simulation in Environmental Informatics; J. Wittmann and V. Wohlgemuth, Univ. of Applied Sciences, Berlin

Furthermore, it is planned to organise an ASIM Assembly during EUROSIM 2010.

ASIM SPL 2010 CONFERENCE

Simulation in Production and Logistics

October 6-8, Karlsruhe, Germany

www.asim-gi.org; www.asim-fachtagung-spl.de/asim2010/en/fachtagung.htm

The 14th ASIM *Dedicated Conference on Simulation in Production and Logistics* deals with the integration aspects of simulation referring to equipment, organization and personnel in production and logistics systems. Thereby, the various functions of simulation tools should be illustrated which allow modelling, visualization as well as evaluation of versatile organizational problems. Herewith, aspects of data management and modelling techniques have to be taken into account. Furthermore, verification and validation issues shall be considered. This largest European 'discrete' simulation conference is well balanced between research, development and industrial use. Scientific innovation and successful application in every day's business are discussed similarly. Participants without simulation experience gain new insights into the possibilities and constraints of simulation, whereas experienced simulation users can compare notes.

The exhibition which is accompanying the conference gives a survey of the state of the art of simulation tools as well as services in the field of simulation. The vendors display their tools at the exhibition and partly during their own sessions, too.

CONFERENCE REPORTS

ASIM GMMS Workshop:
Computational Science and Engineering
March 3 – 5, 2010, Jülich, Germany

With 85 participants, 30 posters and 14 excellent talks the 7th ASIM-Workshop entitled "Trends in Computational Science and Engineering: Foundations of Modeling and Simulation" was the so far largest of its kind, as shown in the following picture:

The 45 minute plenary talks covered a broad range from quantum mechanics to optimization of technical processes. They were given on a high but generally understandable level and gave the opportunity to establish commonalities between the different applications of modeling and simulation. A large number of young participants had the opportunity to discuss their results, build new bridges to PhD-students working on similar topics, widening their scientific horizon and meet the specialists personally. From the many student contribution of high quality two posters from Jülich were awarded among others with a price (see picture below.

Another highlight of the workshop was guided tour to the Jülich Supercomputers with which the research center once again demonstrated its internationally leading position in simulation science.

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 76 members from 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. Organisational details / contact see short info before.

→ www.slosim.si

slosim@fe.uni-lj.si
rihard.karba@fe.uni-lj.si

SLOSIM / Rihard Karba, Faculty of Electrical Engineering, University of Ljubljana,
Tržaška 25, 1000 Ljubljana, Slovenia
Tel.: +386 1 4768 251, Fax.: +386 1 4264 631

Activities

Student Session at ERK Conference. A student session was organized on the ERK conference for the first time. Prof. Borut Zupancic and prof. Maja Atanasijevic-Kunc as the lecturers of the courses: Simulations and Modeling of Processes on the university study on the Faculty of Electrical Engineering, University of Ljubljana, namely organized student seminar works on the mentioned areas. Students chose the problem by themselves, described the modeling and simulation procedure and commented the obtained results. It came out that among 82 works a lot of them were interesting and well prepared, discussing different areas. So the idea was to present the best seminar works also in the special session on ERK Conference. From 18 invited students 13 decided to prepare the contribution in the Conference form. The corresponding committee chose 8 best papers which were finally incorporated in the mentioned session. The works and presentations, being in final phase prepared with the assistance of mentors, were from very actual and attractive domains.

The students attended the Conference first two days with the organized bus transfer and sleeping in the hotel. They received Conference materials, visited some sessions and presented their works what represent an important experience. Student met the mentors and other educational staff also on the common dinner and on the Conference banquet what improved the mutual communications. The action, being in some sense the testing one, was very successful.

The initiative and motivation of students devoting additional time resulted in excellent papers which were included also in the Conference Proceedings. So the conclusion is that different kinds of students inclusion in the program of an ERK Conference must be proceeded also in the future.

Past Events

As every year SLOSIM organized two modeling and simulation oriented sessions on the ERK Conference which took place from September 21st to 23rd, 2009 in Portoroz, Slovenia. Fourteen papers were presented, while seven of them were contributed by the society members. The latter of course took an active part also in some other sessions.

A SLOSIM executive board meeting took place at the Faculty of Electrical Engineering, Univ.of Ljubljana on November 5th, 2009 where some past activities were analysed and some future ones planned.

Coming Events

In 2010 some important events will occur. The first one will be the EUROSIM congress in September where the SLOSIM members will participate as reviewers, session organizers, authors of the papers etc.

EUROSIM 2010 CONGRESS

7th EUROSIM Congress on Modeling and Simulation

September 5-10, Prague, Czech Republik

www.eurosim2010.org

SLOSIM is organising the following special sessions at EUROSIM 2010 (partly with ASIM):.

- Modelling and Simulation in Medicine and Pharmacy; M. Atanasijevic-Kunc, A. Mrhar, Univ. of Ljubljana; J. Drinovec, Univ. Maribor
- Modelling and Simulation for Control, Coordination and Supervision; R. Karba, Univ. Ljubljana
- Education in Simulation / Simulation in Education - ASIM/SLOSIM Special Session; M. Atanasijevic-Kunc, Univ. Ljubljana; F. Breitenecker, Vienna University of Technolgy

The next event will be general and election assembly of SLOSIM in November. Also an excursion for the members in spring and participation on ERK10 as well as some routine activities are planned.

SIMS − Scandinavian Simulation Society (new)

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).

SIMS Board consists of Esko Juuso (chairman), Erik Dahlquist, Brian Elmegaard, Peter Fritzson, Kaj Juslin, Tiina Komulainen, Bernt Lie, Tommy Mølbak, and Vadim Engelson (SIMS coordinator for practical matters). You can contact the chair of the SIMS board:

✉ Esko Juuso, Control Engineering Laboratory, Department of Process and Environmental Engineering, 90014 University of Oulu, Finland

To become a member of SIMS you should join one of the SIMS member organizations (see SIMS web page).

→ www.scansims.org

info@scansims.org

Past Events

SIMS 50 - The 50th Scandinavian Conference on Simulation and Modelling, was organised by DKSIM at Dong Energy, Fredericia, Denmark, October 7-8, 2009.
The programme was focused on modelling and simulation in energy technology. The programme consisted of two keynote and 42 regular papers. Both days were opened with keynote presentations. The topics were: (1) Modelling of climate and climate change: Principles, Applications and Challenges (Shuting Yang, Danish Meteorological Institute), and (2) Challenges in the future energy system (Charles Nielsen, DONG Energy Director R&D). Future energy systems, especially renewable energy in various forms, were important topics throughout the conference.

Advances on simulation tools were presented as well. There were over 50 participants. The proceedings are available online. More info → dksim.dk/sims50/.

15th Nordic Process Control Workshop **(NPCW09)** was arranged at Telemark University College in Porsgrunn, Norway, 29-30 January 2009.
The programme consisted of two invited and 30 regular papers and an award lecture of Dr. John F. MacGregor, who received The Nordic Process Control Award (given for 'lasting and significant contribution to the field of process control'). The topics of the invited speakers were: (1) Modelica as a modeling tool for online optimizing control and estimation (Lars Imsland), and (2) Efficient Computational Methods for MPC (John Bagterp Jørgensen, DTU). The topics of regular sessions were methods of modeling, monitoring and optimization within control applications. There were 60 participants. info→ www.hit.no/TF/npcw_09

The 1st OpenModelica Annual Workshop was organised by The Open Source Modelica Consortium (OSMC) and Linköping University in Linköping, February 2, 2009. OSMC is a non-profit organization supporting the development of the OpenModelica Open-Source implementation of Modelica for industrial and academic usage. There were eight presentations, a panel discussion and Open Source Modelica Consortium Annual Statutory Meeting.
More information → www.openmodelica.org

The 3rd MODPROD Workshop on Model-Based Product Development was organised by The Center for Model-based Product Development (MODPROD) at Linköping Univ., February 3-4, 2009. MODPROD 2009 focused on model-based tools and methods for electronic systems and software.
The workshop was organised in two days: February 3 contained tutorials and academic sessions, and February 4 was the main event, with keynotes and invited presentations. More info → www.modprod.liu.se

18th Automation Seminar was organised in Helsinki, March 17-18, 2009, consisting of keynote, theme and regular presentations, as well as practical demonstrations and a special session on modelling and simulation. More information → www.automaatioseura.fi

IFAC Symposium on Power Plants and Power Systems Control was held in Tampere, July 5-8, 2009. The programme consisted of five plenary presentations and three parallel tracks with 81 regular papers.
More information → www.automaatioseura.fi

Servomøtet was organised by NFA October 20-21, 2009 in Trondheim. The programme consisted of 29 presentations, partly in two parallel sessions.
More information → www.nfaplassen.no

Seminar on Automation providing performance and reliability in energy production was organised by FINSIM together with FSA Energy and Automation Safety Forum in Tallinn, November 4-5, 2009. The programme included 14 presentations.
More information → www.automaatioseura.fi

Coming Events

SIMS 2010

51st Scandinavian Conference on Simulation and Modelling; October 14 – 15, 2010, Oulu, Finland

www.scansims.org

The 51st *Scandinavian Conference on Simulation and Modelling*, will be organised by FINSIM in Oulu, Finland, October 14 – 15, 2010. The purpose of the SIMS conference is to cover broad aspects of modelling and simulation and scientific computation. It will be of interest for model builders, simulator personnel, scientists, engineers, vendors, etc. The scientific program will consist of technical sessions with submitted and invited papers, and is open for poster sessions and vendor demonstrations. The focus areas are energy and environment.

Proceedings of the accepted papers will be distributed at the conference. Presented papers will be considered for publication in the EUROSIM scientific journal *Simulation and Modelling - Practise and Theory* (SIMPRA) published by Elsevier Science. Especially Ph.D. students are encouraged to contribute with papers according to the conference themes.

CONTROL SYSTEMS 2010

September 15-17, 2010, Stockholm, Sweden

www.controlsystems2010.com

The Control Systems conference is held every two years and is the pre-eminent conference in the area of process measurement and control and systems engineering for the pulp and paper industry. The conference attracts presenters and attendees from all major pulp and paper producing countries, so this will be your best opportunity to network with experts from all around the world and to learn more about important ongoing work in the field. The conference will focus towards the practical use of new and emerging technologies and on research on the brink of implementation. The Control Systems Conference 2010 is hosted by SPCI (The Swedish Association of Pulp and Paper Engineers) and Innventia (one of the world's leading R&D companies in the fields of pulp, paper, the graphics media, packaging and logistics).

PSCS – POLISH SOCIETY FOR COMPUTER SIMULATION

PSCS (The Polish Society for Computer Simulation) 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 257 members. The Board of sixth cadence consisting of the following persons directs the affairs of the PSCS: Leon Bobrowski – President, Andrzej Grzyb - Vice President, Tadeusz Nowicki - Vice President, Zenon A. Sosnowski – Treasurer, Zdzislaw Galkowski – Secretary, Roman Bogacz, Zygmunt Strzyzakowski, Andrzej Tylikowski.

→ www.ptsk.man bialystok.pl

✉ PSCS/ Leon Bobrowski, c/o IBIB PAN,
ul. Trojdena 4 (p. 416), 02-109, Warszawa, Poland

Activities. The main activities of the Polish Society for Computer Simulation are annual conferences known as *PSCS Workshops on Simulation in Research and Development*. The PSCS Workshops were organized in: Mielno (1994), Warszawa(1995), Wigry (1996), Jelenia Gora (1997, 1998), Bialystok & Bialowieza (1999), Zakopane – Koscielisko (2000), Gdansk-Sobieszewo (2001), Osieki k/ Koszalina (2002), Zakopane (2003), Białystok & Augustow (2004), Sarbinowo Morskie k/Koszalina (2005), Krynica Zdroj (2006), Zakopane (2008), and Bialystok-Bobrowa Dolina (2009).

Past Events. The annual PSCS Workshop on "Simulation in Research and Development" took place on September 23-26, 2009 in Bialystok-Bobrowa Dolina, Poland. The papers of the workshop covered the following areas: simulation in mechanical engineering, simulation in mathematical problems, artificial intelligence and simulation, simulation in transportation, neural nets and simulation, simulation in automation and control, and simulation tools.

On February 20, 2009 the general assembly of PSCS members was held in Warsaw. This meeting, besides representing an interesting forum to discuss and pro-

mote the activity of the society, was the occasion to elect the Board for the period 2009-2011.

Publications. Proceedings of the 15th PSCS Workshop on „Simulation in Research and Development", T. Nowicki and J. Koszela (Eds.), Warszawa, 2008, (in Polish). The price is 30,- PLN.

Coming Events. Prof. Kazimierz Furmanik will organize the 17th PSCS Workshop on „Simulation in Research and Development" in September 2010 in Poland. E-mail: fukaz@agh.edu.pl

17TH PSCS WORKSHOP

Simulation in Research and Development

Sept. 2010, Poland; www.ptsk.man.bialystok.pl

EUROPEAN SIMULATION GROUPS

YSS - Yugoslav Simulation Society

YSS was established in 1998, with office in Nis and board with V. Litovski – President, P. Petkovic – Vice Pres., D. Milivanovic – Pres. Assembly . **Activities** are

- Periodically organization of workshops: The Small Systems Simulation Symposium, Application of Parallel Computing in Engineering Design.
- Lectures of prominent scientist in the field of simulation and design (2 – 5 per year).
- Regular collective participation to scientific conferences in Serbia and abroad (ETRAN, INDEL).

→ jds.elfak.ni.ac.rs

JDS@jds.elfak.ni.ac.rs

✉ YSS -Jugosovensko Drustvo za Simulaciju, Elektronski Fakultet, Aleksandra Medvedeva 14, 18000 Nis, Serbia

Report: 3rd Small Systems Simulation Symposium organized by YSS, took place at the Faculty of Electronic Engineering of The University of Nis, Serbia, on February 12 – 14, 2010.

18 papers were presented by 33 authors from seven countries. The main topic of the conference was implementation of parallel hardware in electronic simulation: algorithms, alternatives and prospects. Symposium program at → jds.elfak.ni.ac.rs .

KA-SIM in Kosovo

In 2009, IFAC has welcomed KA-CASE, the *Kosova Association for Control, Automation and Systems Engineering* as NMO of IFAC. KA-CASE is situated at UBT, University for Business and Technology, Pristina, Kosovo. As other IFAC NMOs, KA-CASE is working strongly in the area of modelling and simulation, and founded in autumn 2009 a simulation group – KA-SIM.

→ www.ubt-uni.net/ka-case

ehajrizi@ubt-uni.net

✉ MOD&SIM KA-CASE
Att. Dr. Edmond Hajrizi
Univ. for Business and Technology (UBT)
Lagjja Kalabria p.n., 10000 Prishtina, Kosovo

In 2010, KA-CASE and KA-SIM will organise the IFAC SWIIS Workshop in Pristina, with strong sessions on modelling and simulation.

IFAC SWIIS 2010

Supplemental Ways for Improving International Stability

October 27 – 29, Pristina, Kosovo

www.ubt-uni.net/swiis

Scope of SWISS 2010 covers the following areas:

- Modeling of Social and Economic System
- Social Aspects of Technology
- Managing the Introduction on Technological Changes by M & S
- Case Studies of Technological Transfer and Social Changes
- Technology and Environmental Stability
- International System Complexity

KA-SIM will apply for *Observer Membership* in EUROSIM.

EUROSIM 2010

organised by CSSS

September 2010, Prague, Czech Republic

EUROSIM 2010

7th EUROSIM Congress on Modelling and Simulation

Eurosim Congress the most important modelling and simulation event in Europe

September 5-10, 2010, Prague, Czech Republic

Congress Venue
The Congress will take place in Prague, the capital city of Czech Republic, at the Congress Center of Masaryk College, part of Czech Technical University, in cooperation with the Faculty of Electrical Engineering of CTU.

About Czech Technical University in Prague
Czech Technical University celebrates 300 years of its history in 2007. Under the name Estate Engineering Teaching Institute in Prague was founded by the rescript of the Emperor Josef I of 18 January 1707 on the basis of a petition of Christian Josef Willenberg (1676-1731). This school was reorganized in 1806 as the Prague Polytechnic, and, after the disintegration of the former AustroHungarian Empire in 1918, transformed in to the Czech Technical University in Prague.

About EUROSIM
EUROSIM, the federation of European simulation societies, was set up in 1989. Its purpose is to promote, especially through local simulation societies, the idea of modelling and simulation in different fields, industry, research and development. At present, EUROSIM has 14 full members and 4 observer members.

Congress Scope and Topics
The Congress scope includes all aspects of continuous, discrete (event) and hybrid modelling, simulation, identification and optimisation approaches. Contributions from both technical and non-technical areas are welcome. Two basic tracks will be organized: M&S Methods and Technologies and M&S Applications.

Czech Republic - EUROSIM 2010 Host Country
The Czech Republic is a country in the centre of Europe. It is interesting for its 1,000-year-long history, rich culture and diverse nature. The country is open to new influences and opportunities thanks to a high level of industrial infrastructure, safety measures and plural media. The location of the Czech Republic in the very heart of Europe contributes to the fact that one can get there easily and fast. Usually all it takes to enter the country is a valid passport. The Czech Republic belongs to the Schengen zone. The need for a visas to enter the Czech Republic is very exceptional.

Prague - EUROSIM 2010 Host City
Prague is a magical city of bridges, cathedrals, gold-tipped towers and church spires, whose image has been mirrored in the surface of the Vltava River for more than a millennium.Walking through the city, you will quickly discover that the entire history of European architecture has left splendid representatives of various periods and styles. There are Romanesque, Gothic, Renaissance, Baroque and Classicist buildings, as well as more modern styles, such as Art Nouveau and Cubist. A poet once characterized Prague as a symphony of stones.

About CSSS
CSSS (The Czech and Slovak Simulation Society) has about 150 members in 2 groups connected to the Czech and Slovak national scientific and technical societies (Czech Society for Applied Cybernetics and Informatics, Slovak Society for Applied Cybernetics and Informatics). Since 1992 CSSS is a full member of EUROSIM.

Invitation
Czech and Slovak Simulation Society is greatly honored with the congress organisation and will do the best to organise an event with a high quality scientific programme with some other acompanied actions but also with some unforgettable social events.

Mikuláš Alexík, EUROSIM president,
Miroslav Šnorek, president of CSSS, EUROSIM 2010 Chair