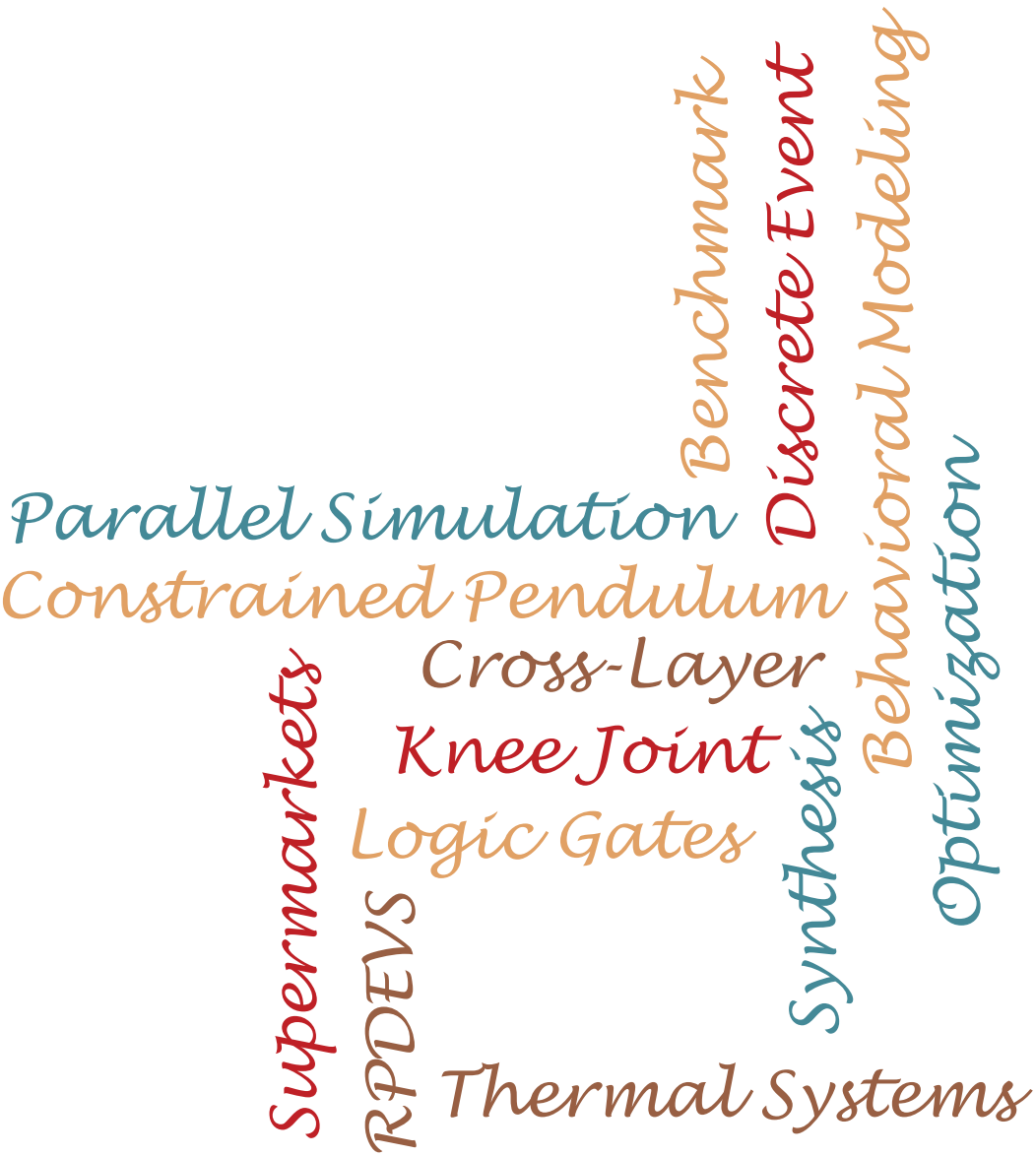
SNE SIMULATION NOTES EUROPE



Journal on Developments and Trends in Modelling and Simulation

EUROSIM Scientific Membership Journal

Vol. 29 No.2, June 2019
ISSN Online 2306-0271
ISSN Print 2305-9974
DOI 10.11128/sne.29.2.1047
ISBN Print 978-3-903024-86-1

TU Verlag an der Technischen Universität Wien

ARGESIM

EUROSIM 2019

10th EUROSIM Congress on Modelling and Simulation

La Rioja, Logroño, Spain, July 1 – 5, 2019

EUROSIM Congresses are the most important modelling and simulation events in Europe. For EUROSIM 2019, we are soliciting original submissions describing novel research and developments in the following (and related) areas of interest: Continuous, discrete (event) and hybrid modelling, simulation, identification and optimization approaches. Two basic contribution motivations are expected: M&S Methods and Technologies and M&S Applications. Contributions from both technical and non-technical areas are welcome.

Congress Topics The EUROSIM 2019 Congress will include invited talks, parallel, special and poster sessions, exhibition and versatile technical and social tours. The Congress topics of interest include, but are not limited to:

- Intelligent Systems and Applications
- Hybrid and Soft Computing
- Data & Semantic Mining
- Neural Networks, Fuzzy Systems & Evolutionary Computation
- Image, Speech & Signal Processing
- Systems Intelligence and Intelligence Systems
- Autonomous Systems
- Energy and Power Systems
- Mining and Metal Industry
- Forest Industry
- Buildings and Construction
- Communication Systems
- Circuits, Sensors and Devices
- Security Modelling and Simulation
- Bioinformatics, Medicine, Pharmacy and Bioengineering
- Water and Wastewater Treatment, Sludge Management and Biogas Production
- Condition monitoring, Mechatronics and maintenance
- Automotive applications
- e-Science and e-Systems
- Industry, Business, Management, Human Factors and Social Issues
- Virtual Reality, Visualization, Computer Art and Games
- Internet Modelling, Semantic Web and Ontologies
- Computational Finance & Economics
- Simulation Methodologies and Tools
- Parallel and Distributed Architectures and Systems
- Operations Research
- Discrete Event Systems
- Manufacturing and Workflows
- Adaptive Dynamic Programming and Reinforcement Learning
- Mobile/Ad hoc wireless networks, mobicast, sensor placement, target tracking
- Control of Intelligent Systems
- Robotics, Cybernetics, Control Engineering, & Manufacturing
- Transport, Logistics, Harbour, Shipping and Marine Simulation

Congress Venue / Social Events The Congress will be held in the City of Logroño, Capital of La Rioja, Northern Spain. The main venue and the exhibition site is the University of La Rioja (UR), located on a modern campus in Logroño, capital of La Rioja, where 7500 students are registered. The UR is the only University in this small, quiet region in Northern Spain. La Rioja is where the Monasteries of San Millán de la Cogolla, cradle of the first words written in the Spanish language, are situated, sites included in UNESCO's World Heritage List in 1996. Of course, social events will reflect this heritage – and the famous wines in la Rioja.

Congress Team: The Congress is organised by CAE CAE-SMSG, the Spanish simulation society, and Universidad de la Rioja.

Info: Emilio Jiménez, EUROSIM President, emilio.jimenez@unirioja.es
Juan Ignacio Latorre, juanignacio.latorre@unavarra.es

www.eurosim2019.com

Editorial

Dear Readers – This issue SNE 29(2) continues the publication strategy of SNE Simulation Notes Europe: quick publication of submitted papers on trends in modelling and simulation as well as overview papers, postconference publications for conferences of EUROSIM societies, and publication of benchmark reports.
The issue starts with an Overview Note on parallel discrete simulation – also suited for use in lectures on this subject, followed by a Technical Note on integrated behaviour modelling and simulation within model-based electric/electronic-architecture (EEA) descriptions. The following two Technical Notes deal with the RPDEVS (Revised Parallel DEVS) modelling formalism which enhances the Parallel Discrete Event System Specification (PDEVS) – underlining the power of DEVS and the progress in DEVS implementations. The next Technical Note presents a simulation-based synthesis and optimization of complex thermal systems for supermarkets – showing benefits of simulation for consumer comfortability and energy saving. The following Short Note checks features of specific physical modelling system for analysis of a knee joint. The issue closes with a Benchmark Note of type Benchmark Report for ARGESIM Benchmark C7 'Constrained Pendulum' – comparing direct implementations in MATLAB and EXCEL. The title page of this issue underlines the presented broad variety of theory, methods and applications on modelling and simulation by a word cloud consisting of terms of the titles of all contributions.

I would like to thank all authors for their contributions to SNE 29(2) showing the broad variety of simulation. And thanks to the editorial board members for review and support, and to the organizers of the EUROSIM conferences for co-operation in post-conference contributions. And last but not least thanks to the SNE Editorial Office for layout, typesetting, preparations for printing, electronic publishing, and much more.

Felix Breitenecker, SNE Editor-in-Chief, eic@sne-journal.org; felix.breitenecker@tuwien.ac.at

Contents SNE 29(2)

Online SNE 29(2), DOI 10.11128/sne.29.2.1047
ARGESIM Publisher, Vienna, www.argesim.org
Print SNE 29(2) ISBN 978-3-903024-86-1,
TU Verlag Vienna, Print-on-Demand, www.tuverlag.at

An Introduction to Parallel Discrete Event Simulation.
O. Ullrich, D. Lückerath 63

Cross-Layer Behavioral Modeling and Simulation of E/E-Architectures using PREEvision and Ptolemy II.
H. Bucher, S. Kamm, J. Becker 73

RPDEVS Abstract Simulator.
F. J. Preyser, B. Heinzl, W. Kastner 79

Simulation of RPDEVS Models of Logic Gates.
C. Fiedler, F. J. Preyser, W. Kastner 85

Approach for Synthesis and Optimization of Complex Thermal Systems for Supermarkets.
J. Kistner, W. Tegethoff, N. Fidorra, J. Köhler 93

Simulation Case Study: Using Simscape for *Human* Knee Joint Models. *R. Leskovar, A. Körner* 101

Direct Implementation of ARGESIM Benchmark C7 'Constrained Pendulum' in MATLAB and EXCEL.
A. E. Stockinger, E. Gütl, S. A. Rath, D. Strasser, M. Bicher, A. Körner, H.Ecker 105

EUROSIM Societies Short Info ..... N1 – N8

SNE Contact & Info

SNE Online ISSN 2306-0271, SNE Print ISSN 2305-9974
→ *www.sne-journal.org*
✉ *office@sne-journal.org, eic@sne-journal.org*

✉ SNE Editorial Office
Johannes Tanzler (Layout, Organisation),
Irmgard Husinsky (Web, Electronic Publishing),
Felix Breitenecker (Organisation, Author Mentoring)
ARGESIM/Math. Modelling & Simulation Group,
Inst. of Analysis and Scientific Computing, TU Wien
Wiedner Hauptstrasse 8-10, 1040 Vienna , Austria

SNE SIMULATION NOTES EUROPE

WEB: → *www.sne-journal.org*, DOI prefix 10.11128/sne
Scope: Developments and trends in modelling and simulation in various areas and in application and theory; comparative studies and benchmarks (documentation of ARGESIM Benchmarks on modelling approaches and simulation implementations); modelling and simulation in and for education, simulation-based e-learning; society information and membership information for EUROSIM members (Federation of European Simulation Societies and Groups).
Editor-in-Chief: Felix Breitenecker, TU Wien, Math. Modelling Group
✉ *Felix.Breitenecker@tuwien.ac.at,* ✉ *eic@sne-journal.org*
Print SNE and Print-on-Demand: Grafisches Zentrum and TU-Verlag, TU Wien, Wiedner Hauptstrasse 8-10, 1040, Vienna, Austria – *www.tuverlag.at*
Publisher: ARGESIM ARBEITSGEMEINSCHAFT SIMULATION NEWS
c/o Math. Modelling and Simulation Group,
TU Wien / 101, Wiedner Hauptstrasse 8-10,
1040 Vienna, Austria; *www.argesim.org,* ✉ *info@argesim.org*
on behalf of ASIM *www.asim-gi.org* and EUROSIM
→ *www.eurosim.info*

SNE - Aims and Scope

Simulation Notes Europe (SNE) provides an international, high-quality forum for presentation of new ideas and approaches in simulation - from modelling to experiment analysis, from implementation to verification, from validation to identification, from numerics to visualisation - in context of the simulation process.

SNE seeks to serve scientists, researchers, developers and users of the simulation process across a variety of theoretical and applied fields in pursuit of novel ideas in simulation and to enable the exchange of experience and knowledge through descriptions of specific applications. **SNE** follows the recent developments and trends of modelling and simulation in new and/or joining application areas, as complex systems and big data. **SNE** puts special emphasis on the overall view in simulation, and on comparative investigations, as benchmarks and comparisons in methodology and application. For this purpose, **SNE** documents the **ARGESIM Benchmarks** on *Modelling Approaches and Simulation Implementations* with publication of definitions, solutions and discussions. SNE welcomes also contributions in education in/for/with simulation.

A News Section in **SNE** provides information for **EUROSIM** Simulation Societies and Simulation Groups.

SNE, primarily an electronic journal, follows an open access strategy, with free download in basic layout. **SNE** is the official membership journal of **EUROSIM**, the *Federation of European Simulation Societies and Simulation Groups* – www.eurosim.info. Members of **EUROSIM** societies are entitled to download **SNE** in an elaborate and extended layout, and to access additional sources of benchmark publications, model sources, etc. **Print SNE** is available for specific groups of **EUROSIM** societies, and starting with Volume 27 (2017) as print-on-demand from TU Verlag, TU Wien. **SNE** is DOI indexed by CrossRef, identified by DOI prefix 10.11128, assigned to the **SNE** publisher **ARGESIM** (*www.argesim.org*).

Author's Info. Individual submissions of scientific papers are welcome, as well as post-conference publications of contributions from conferences of **EUROSIM** societies. **SNE** welcomes special issues, either dedicated to special areas and/or new developments, or on occasion of events as conferences and workshops with special emphasis.

Authors are invited to submit contributions which have not been published and have not being considered for publication elsewhere to the **SNE** Editorial Office.

SNE distinguishes different types of contributions (*Notes*), i.e.

- **TN** Technical Note, 6 – 10 p.
- **PN** Project Note 6 – 8 p.
- **SW** Software Note , 4 – 6 p.
- **ON** Overview Note – only upon invitation, up to 14 p.
- **EN** Education Note –6 – 8 p.
- **SN** Short Note, max. 6 p.
- **BN** Benchmark Note, 2 – 8 p.
- **EBN** Educational Benchmark Note, 2 – 10 p

Further info and templates (doc, tex) at **SNE**'s website.

www.sne-journal.org

SNE Editorial Board

SNE - Simulation Notes Europe is advised and supervised by an international scientific editorial board. This (increasing) board is taking care on peer reviewing of submission to SNE:

An Introduction to Parallel Discrete Event Simulation

Oliver Ullrich*, Daniel Lückerath

Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS, Schloss Birlinghoven, 53757 Sankt Augustin, Germany; *oliver.ullrich@iais.fraunhofer.de

SNE 29(2), 2019, 63-72, DOI: 10.11128/sne.29.on.10471
Received: May 10, 2019
Accepted; May 31, 2019
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna, ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. Some discrete simulation models are too large to be executed on a single processor; in other cases, results might be required faster than a sequential execution can provide them. Such models are candidates for parallelization. Here, models are distributed among several processors, and are then executed with careful synchronization.
This paper provides an introduction to the fundamentals and methods of the parallel execution of simulation models, with a focus on model-based parallelization. The paper describes the two main classes of parallel simulation methods, conservative and optimistic simulation, their respective advantages and shortcomings. A second focus is put on static and dynamic load balancing, with a dynamic load balancing method first developed to accelerate the simulation of transportation systems being introduced in some detail. In addition, the paper describes some typical applications of model-based parallelization.

Introduction

Many discrete simulation models contain a certain degree of inherent concurrency. For example, in the simulation of a light rail network the braking manoeuvres of one vehicle in one region of the network would not directly influence the passenger exchange of a different vehicle in another region. The two vehicles can thus be simulated independently of each other in the majority of cases.

The goal of model-based parallelization is to exploit that existing concurrency through parallel execution of events that take place in different regions of the model on a number of participating processors or processor cores. The basic assumption is that these events can often be executed independently of each other without inducing communication between partial models. During the course of the execution, synchronization issues may arise between these partial models. For example, if a vehicle entity leaves the partial model of one processor it has to be sent to another processor and there has to be integrated with the partial model already being executed.

This paper presents an introduction to background, approaches, and techniques for the parallel execution of simulation models, with a focus on model-based parallelization. It introduces a dynamic load balancing method first developed for the efficient execution of multimodal transportation models. The paper is especially addressed to students of the craft, and to practitioners who might want to look beyond the GUI of their usual modeling tools. While in many cases parallelization methods are hidden in the execution engine of a simulation tool, some applications call for a more hands-on approach. The fundamentals of parallel simulation are easily understood, and its methods are also very powerful. Researchers, students, or practitioners can utilize well-researched parallelization methods to create fast simulation applications executing large models.

The paper continues by sharing some background on the concepts and general approaches to parallel simulation (see Section 1), and then goes on to describe the two main classes of model-based parallelization techniques: conservative and optimistic methods (see Section 2). Subsequently, static and dynamic load balancing approaches are described (see Section 3), followed by an examination of some typical applications (see Section 4). The paper closes with a summary of the lessons learned and recommendations for further reading (see Section 5).

1 Background

Discrete simulation models consist of a set of entities that represent physical or logical components of the examined system, including their behavior and relationships to each other and their state changes over time.

In discrete simulation, a model changes its state at discrete points in simulation time. Here, simulation time – or model time – is the time that elapses from the point of view of the simulated entities (see [1]). Simulation time has to be distinguished from wallclock time, the time elapsing in the real world while the simulation run is executed. In many cases simulation models are executed as fast as possible. In certain applications, however, it is desirable to tie model execution to wallclock time, for example if a human has to react to the changes in the model. This is referred to as real-time execution or scaled real-time execution.

Simulation time can progress in fixed or variable increments. In models with fixed time increments, the model is executed by starting out from simulation time t_{start}, iterating through steps i with a fixed model time increment Δt – the model state can change at any of these points $t_{start} + i * \Delta t$ in simulation time. The entities communicate with each other via messages that might be scheduled with a timestamp in the (model time) future.

With models with a variable time step, simulation time is incremented while processing simulation events. These methods are often called event-based simulation (see [1] or [25]). Each of these events has a timestamp that marks the scheduled time of its occurrence, and often also an attribute that describes the type of the event and various other fields such as a list of the intended receivers and the identity of the sending entity. The events are managed in a Future Event List (FEL), a priority queue that keeps all scheduled events sorted in ascending order of their timestamp. To execute the model, the event with the least timestamp is pulled from the FEL, the simulation time is advanced to its timestamp, and the event is processed – which usually changes the model state. New events can be scheduled during processing; they are then inserted into the FEL.

In order to accelerate model execution, computation can be distributed over parallel processes, for example on several processors or, more and more often, several cores of the same processor. Here, usual goals are to execute the model as fast as possible or in (scaled) real time. An execution that is too fast for a desired real-time binding can be slowed down to the desired speed without any problems.

A central condition for such a parallel execution is that a simulation run in parallel has to deliver identical results as a sequential execution of the same model; the simulation technique must not influence the model behavior (see [6] or [19]).

The central measure for the efficiency of a parallelization method is the speedup. That value determines the ratio of the runtime of the sequential execution of a model to the time needed for parallel execution. The aim of parallel execution is to achieve the highest possible speedup with a given number of processors, or, more precise, given computational resources.

A number of vastly different approaches to parallel simulation exist. For an in-depth discussion of the methods described below – and more – see [6]. *Model-based parallelization* methods, also called space-based parallelization, aim to exploit the parallelism inherent in the model. For this purpose, the model is decomposed into partial models, which are then distributed on the available processors for execution. The different partial models communicate via messages encapsulating simulation events or serialized entities that are sent over the shared cache or the connecting network. The processors p_1 to p_k from the set of processors P each execute a specific partial model – they can be seen as the nodes of a graph, with the messages sent between processors inducing edges.

Any model-based parallelization method has to keep the execution of partial models carefully synchronized. Here, the *local causality constraint* prescribes that each model entity has to process simulation events in a non-descending order regarding their timestamps. If the local causality constraint is not met, the simulation results might be invalidated by causality errors. For example, lets assume that in a light rail simulation a processor p_1 has processed an operational day up until a simulation time of 12:30, while a processor p_2 has only arrived at 12:05. Now a vehicle entity leaves the partial model of p_2. That processor sends a message to p_1 and transfers the vehicle data for further simulation from 12:06. From the point of view of p_1 that message comes from 24 simulation minutes in the past. During these 24 simulated minutes p_1 might have already allocated the resources "rightfully" occupied by the vehicle to other entities. The transferred message can not be processed sensibly; the simulation has to terminate with an error message.

A central concept is the lookahead (see [5] or [13]): If an entity or a partial model is currently processing events at a simulation time t, then a lookahead of L guarantees that no additional simulation events will be generated with a timestamp lesser than $t + L$ (see [6]). For models with a fixed time increment the lookahead corresponds of that increment and is therefore always greater than zero. For event-based models the lookahead usually changes in the course of the simulation; under certain circumstances a lookahead value of zero is possible (see Section 2.1).

2 Model-based Parallelization

Model-based parallelization methods can be categorized based on the way the causality constraint is kept: With conservative methods, causality is guaranteed at all times by only processing simulation events that are explicitly considered safe. With optimistic methods, each entity indiscriminately executes events as quickly as possible. In case a processor receives an event with a timestamp that lies in the past from its local point of view, it rejects corresponding parts of the already executed simulation and restores causality by recalculating them from the timestamp of that event on.

In the following, a selection of important conservative and optimistic parallelization methods are described.

2.1 Conservative Parallelization Methods

Two of the most important conservative parallelization methods are synchronization with null messages and synchronous execution. Both methods – and more – are described in great detail in [6].

Synchronization with null messages. Synchronization with null messages was independently developed as the first conservative parallelization method for event-based simulations by Bryant (see [2]) and Chandy and Misra (see [3]) and explained in detail by Fujimoto (see [6]). Here, the processors p_1 to p_k are regarded as nodes of a graph. In that graph, if a processor p_i sends messages to a processor p_j in the course of the simulation run, a directed edge exists between these nodes.

The method assumes that a processor p_i sends messages to a processor p_j in order of non-decreasing timestamps. A processor stores incoming messages in a series of FIFO queues, each assigned to an incoming edge of the processor graph. It follows that messages are present in each of these queues in non-decreasing order. In a model with variable time progress, messages or events that stay local on one processor are managed in a separate priority queue.

A message with timestamp t is declared secure if there is at least one message with a timestamp not lower than t at the head of each inbound queue. The presence of these messages means that no processor can send messages that lie before t in simulation time. Now the processor selects the message N_{min} with the lowest timestamp from all incoming queues and, if applicable, the local event list. Since no message with a lower timestamp can subsequently occur, the causality condition is maintained when N_{min} is processed.

During event processing, further events with the same or a greater timestamp are sent to neighboring processors if necessary. At that point deadlocks can occur: If for every participating processor not all queues at the incoming edges are filled with at least one message, each processor waits for messages from the other processors to arrive (see Figure 1). Therefore, no events can be declared safe – the simulation is blocked.

To solve this problem, Bryant (see [2]) and Chandy and Misra (see [3]) suggest that each processor, after processing a message, sends so-called null messages to all neighboring processors. These messages are timestamped with the current simulation time plus the lookahead value L of the processor.

The handling of null messages by the receiving processor is the same as that of regular messages. However, when processing a null message, no change is made to the model state apart from advancing the simulation time to its timestamp. Sending null messages during each event processing ensures that messages are always available in the FIFO queues – the development of a deadlock is thus precluded.

The efficiency of the method largely depends on the lookahead value: A small lookahead means that many null messages have to be sent and processed. In addition, the model cannot contain any circles in the graph with a lookahead of $L = 0$, otherwise deadlock situations become possible. Here, the processors involved process only null messages and send (because of $L = 0$) further null messages with the same timestamp to each other. The simulation time never advances, the application is caught in an endless loop.

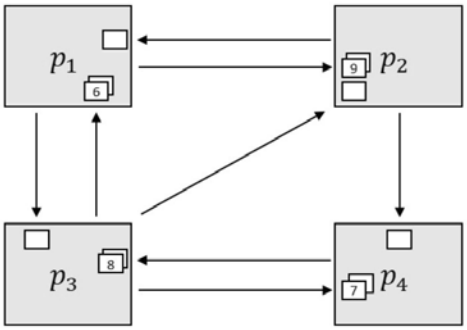


Figure 1: A deadlock occurs when each processor is waiting for its neighbors to send messages; sending null messages can solve this issue.

Synchronous execution. In synchronous execution (see [6]), each processor executes the events or simulation steps of a simulation time interval recognized as safe and then enters a synchronization barrier. Here, each processor waits for all other processors to complete calculation. Then the next interval, now declared safe, is processed. Thus, there is a defined point in wall-clock time when all processors have finished calculating a certain simulation time interval and before any of them starts calculating the next time interval.

To determine what events are safe to be executed, the lookahead $L(i)$ for a step i is used. While $t(i)$ is defined as the simulation time of the next unprocessed message at processor p, $L(i)$ is its local lookahead value. The global lookahead t_L is the minimum value of $t(i)+L(i)$ for all processors. All messages with timestamps of up to t_L are then declared safe.

The synchronization barrier can be implemented in different ways. When synchronizing with tree barriers, the processors are regarded as a balanced span tree, with one processor being designated as a controller. A leaf processor that has completed the calculation step and now wants to enter the barrier sends a barrier message to its parent node in the tree and then waits for a response. An inner node that wants to enter the barrier waits for messages from its daughter nodes. If these are complete, it sends a barrier message to its parent node and then waits for the response. When the controller has finished calculating the interval and has received barrier messages from all daughters, all the processors have reached the barrier phase. To then leave the barrier and initiate the next calculation phase, the controller sends release messages to its daughters, who in turn send them on to their daughters.

A special case of tree barriers is the so-called central barrier. Here all processors are synchronized directly by a controller. The disadvantage of the otherwise very efficient central barrier is the linear growth of the number of messages that have to be processed by the controller, leading to a bottleneck when a large number of processors is involved.

The synchronization messages can be utilized for sending piggybacked data values, such as local lookahead values. The method calls for no other prerequisites than the presence of a positive lookahead for determining the size of the simulation time increments. In particular, there are no requirements for the connections between the individual partial models, since the presence or fill level of FIFO queues need not be taken into account.

2.2 Optimistic Parallelization Methods

The optimistic method Time Warp was first proposed by Jefferson (see [11]) and is described in detail by Fujimoto (see [6]). During the 1990s the method matured with modifications that improve memory consumption (see [22]) as well as reduce costly rollbacks (see [4] and [23]). In Time Warp the parallelization tasks are divided into a local and a global control mechanism. The work carried out by the local mechanism takes place locally on each processor – the processors can work largely independently of each other. The global mechanism performs activities such as input, output, and garbage collection, and requires coordination between the processors.

Local control mechanism. As with other event-based methods, processors execute events from the local Future Event List (FEL), and the state variables of the model are changed if necessary. However, the events are not simply discarded after processing, but stored in another list, the Processed Event List (PEL).

If a message arrives from another processor whose timestamp is greater than or equal to the current simulation time of the local partial model, it is inserted into the FEL and processed normally. If a straggler message N arrives with a timestamp t lesser than the local simulation time, the model has to be rolled back to its state at time t – all state changes from this point on have to be undone. Furthermore, the already processed events with a timestamp greater than t have to be retrieved from the PEL and inserted back into the FEL for reprocessing. Message N is also inserted in the FEL.

There are two general ways to perform this rollback: In copy state saving, the values of all state variables are saved before each event processing. If a straggler arrives, the saved state with the corresponding timestamp is copied back to the state variables – later changes are discarded. With incremental state saving, a log entry notes each change of a state variable. That log entry is then put on a stack keeping a record of all changes.

It may not be enough to reset the local model state: If invalidated messages were sent to other processors, these messages have to be retrieved – or "unsent" – and their effects have to be undone by the receiving processor. Time Warp uses so-called anti-messages for this purpose. Each anti-message N_A corresponds to exactly one regular message N. When an anti-message arrives at a processor, the corresponding regular message is automatically deleted from the corresponding data structure (FEL or PEL). The anti-message is also destroyed.

That mechanism elegantly undoes all invalidated changes and restores causality, but also might lead to a cascade of rollbacks and anti-messages.

Rollbacks do not affect the model state at a time less than or equal to t, i.e. simulation results up to the simulation time of the straggler are retained. It follows that at least the processing of the event with the system-wide least timestamp will not be cancelled. There is therefore a minimum simulation time that might be affected by potential rollbacks – the state of the model before that simulation time will never be invalidated.

Global control mechanism. The Global Virtual Time GVT_t at a time t denotes the minimum timestamp of all unprocessed or partially processed events across all processors involved at a time t. As already described, no rollbacks can take place to times times lesser than GVT_t.

When calculating GVT_t, messages must be taken into account that have already been sent but not yet received by the recipient. Since these transient messages can potentially trigger a rollback and thus reduce the local simulation time, the minimum of local simulation times cannot simply be determined. As a remedy, a simple protocol can be used in which each recipient of a message N confirms the reception to the sender. Until this acknowledgement is received, the sender is responsible for the message N and has to include it in the calculation of the local minimum – afterwards N becomes the responsibility of its receiver. This guarantees that at any point in time the simulation time of N is included in the calculation of GVT_t.

The value of GVT_t is used for a number of administrative tasks, for example the collection of fossil states: backup copies older than it can safely be deleted. As input/output operations generally cannot be undone, simulation events can only order outputs when the current GVT_t has advanced to at least the simulation time of the event. A special case is the processing of program and calculation errors: These can occur due to causality errors, for example a negative number of trains in a depot. The program cannot simply be terminated, as such errors may be reversed by rollbacks.

2.3 Comparison

The best method for the parallel execution of an individual model is largely dependent on its specific properties; no single method is optimal for all applications (as analyzed in detail in [6]).

Generally, conservative methods tend to be less complex in structure (see [9] and [15]). They work with only a single set of state variables, without the need to manage backups. Since conservative methods only execute events or time increments that are explicitly declared safe – they are based on worst-case scenarios –, they do not fully exploit the parallelization potential of a model. Conservative methods can therefore be excessively pessimistic. In general, the greater the lookahead value, the more events can be processed in parallel, so that a higher degree of model-inherent parallelism can be exploited.

An advantage of optimistic methods is that even models with a lookahead of zero can be efficiently executed without further restrictions. Parallel execution is not hindered by all potential dependencies between partial models, as is the case with conservative methods, but only by dependencies that actually occur in the course of a run.

If these dependencies are high, or if the computational loads shift over time, for example resulting from dynamically changing activities in the model, these methods might behave too optimistically, so that a cascade of miscalculations is carried out, that then have to be taken back by complex rollback operations (see [14]). To ensure causality, backups of the model state are necessary for each occuring change. For activities such as input/output, error handling, or memory management, for which the usual library functions can be used in conservative methods, optimistic procedures require specifically implemented rollback-safe functions.

In summary, optimistic methods tend to be more complex than conservative methods. The lower overhead of conservative methods has a positive effect on performance, especially when the lookahead is known – and ideally large in comparison to the event density. However, if a lookahead value is not known or is very low compared to the event density, optimistic methods generally have performance advantages.

3 Load Balancing

Resulting from the typical dependencies in simulation models, the speed of the execution is generally dependent on the partial model that advances most slowly in simulation time. It is therefore beneficial to incorporate a load balancing system into the simulation engine. Such a system does not exclusively aim at high utilization of the processor capacity, but also has to consider a uniform advance in simulation time.

Load balancing schemes employed by parallel simulation methods can be characterized as dynamic, static, adaptive, non-adaptive, local, centralized, or hierarchical (see [16]): A *static* method estimates the load and assigns partial models to processors in a preprocessing step before the start of the simulation run, and thus does not consider dynamic changes in the model activity. In contrast, a *dynamic* load balancing method continuously considers imbalances that develop from shifts in the computational load and re-assigns partial models to appropriate processors while executing the simulation run. *Adaptive* methods consider fluctuations in the available processor power originating from the demand of dynamic processes belonging to third parties. In inhomogeneous computer networks adaptive methods also consider the dissimilar performance power of the respective processors. A *non-adaptive* system ignores those fluctuations and differences. In *local* methods, the processors only exchange data with determined neighborhoods and act on this local information, while *centralized* methods utilize a marked controller process to whom the other processors report. *Hierarchical* methods usually organize communication in a tree topology.

A load balancing method used on a PC or laptop computer should have static and dynamic components, with the latter being also adaptive, and thus consider both the changing computational load of the models, and the changing availability of resources on a non-exclusively used machine. A centralized method is usually simpler to implement and quite adequate for a system with only eight to sixteen processor cores (see [27]); if a method is targeted at a massive parallel system it should avoid a potential bottleneck by utilizing a hierarchical or local scheme.

3.1 Static Load Balancing

At the start of a simulation run, the model entities should be assigned to the participating processors in a way that ensures a balanced load. As a second objective to optimally using the computational potential of the processors, the communication load, resulting from sending and receiving messages from one processor to another, shall be as low as possible. Without further knowledge of model specifics, the static load balancing mechanism uses the number of edges between model partitions as an indicator for communication load. It therefore aims to distribute the model in a way that keeps the number of inter-partition edges at a minumum.

In literature, the decomposition of a graph $G(V,E)$ with $n = |V|$ nodes into k components of simular size is known as the GRAPH PARTITION problem. GRAPH PARTITION is NP complete (see [10]), and can thus – in case $P \neq NP$ holds – not be solved efficiently. For the parallelization of simulation models an exact solution is not necessary, especially since a dynamic change of the load in the course of a simulation run would quickly destroy any optimum static load balance (see [24]).

Kernighan and Lin (see [12]) describe a simple heuristic method suitable for static load balancing. The method starts out from a given partition where all partial models have the same number of nodes (give or take one) – for many models that may be a simple geographical breakdown. The method then iteratively improves the communication load using a hill climbing algorithm (see [18]).

Kernighan and Lin first describe a method to decompose a graph into two partitions K_1 and K_2. Starting out from a given initial partition the method computes for each pair of nodes (v_i, v_j) with $v_i \in K_1$ and $v_j \in K_2$, the impact of a potential movement of v_i to K_2 and v_j to K_1 on the number of inter-partition edges. In case an improvement is possible, the nodes are moved accordingly. The method iterates as long as additional improvements are possible, and thus until a local optimum is reached. It has a computational complexity of $N(n^2)$.

The described method is then extended to disect a graph into $k > 2$ partitions: To that effect each pair K_i

and K_j out of the k partitions are locally optimized using the $k = 2$ method. Usually several iterations are executed, so that the simple method is run $e * (k-1) * (k-2) = O(k^2)$ times. That results in a computational complexity of $O(k^2 * (n/k)^2) = O(n^2)$ for k partitions with n/k nodes – the method is thus independent of the number of participating processors. Kernighan and Lin empirically determine that after two iterations of their method approx. 95% of the potential gain has been reached.

3.2 Dynamic Load Balancing

In many models the majority of inter-entity dependencies are regional in nature: most of the time entities interact with their neighbors, only rarely do they send messages to far away regions of the model. Based on that thought dynamic load balancing generally has two aims:

- ensuring that all participating processors progress uniformly in simulation time by computing loads adequate to their respective performance, and
- keeping the communication load between the processors as low as possible by exploiting regional dependencies in the model.

Generally, the processors perform the load balancing in three steps: *load measuring*: each processor p_i determines its own load and communicates the results to the other processors; *load evaluation*: each processor p_i determines whether any model nodes shall be migrated to adjacent processors and, if so, what nodes shall be migrated to what processor p_j; and *load migration*: the model nodes are encapsulated as messages and sent to adjacent processors.

The dynamic load balancing mechanism described here has been first developed for the parallel simulation of transit systems as part of a conservative, synchronous execution engine (see [24] and [27]).

Measuring loads. To be able to employ effective countermeasures against overload or underload the load of individual processors has to be measured in regular intervals. In conservative parallelization that can be achieved for example as part of the synchronization barrier, in optimistic methods as part of the local control mechanism. The following describes a comparatively simple way to measure load as part of the synchronization barrier that also integrates the available individual performance of a processor – including its change over time, for example through external user or system processes (see [27]).

Each processor $p \in P$ measures its load $l_p(i)$ at time $t(i)$ when all processors have completed their computations regarding simulation step i (see Figure 2). By utilizing the timer functions of the operating system each processor p measures the model-dependent processing time $t_m(p,i)$ it needs to execute the simulation step, and the duration of synchronization time $t_s(p,i)$ that elapses between the completion of the execution and $t(i)$. The load $l_p(i)$ of the processor at step i is now defined as

$$l_p(i) = \frac{t_m(p,i)}{t_m(p,i) + t_s(p,i)} \tag{1}$$

The still available capacity that was wasted as idle time can be determined as

$$f_p(i) = \frac{t_s(p,i)}{t_m(p,i) + t_s(p,i)} \tag{2}$$

The total time $t_g(i)$ used to execute simulation step i is now composed of the processing time, the synchronization time and the communcation time $t_c(p,i)$ used to load balancing and other administrative work (see Equation 3). The values of $t_g(i)$ are equal for all $p \in P$.

$$t_g(i) = t_m(p,i) + t_s(p,i) + t_c(p,i) \tag{3}$$

Based on local load data alone a value of $l_p(i)$ near 1 – and thus a synchronization time $l_s(p,i)$ near 0 – can signal either an optimum load near capacity or a bottleneck caused by overload. To be able to discern, the load data of the other processors in P has to be included. That data basically consists of two numbers that can be exchanged as part of the synchronization process.

The load measurement based on $t_s(p,i)$ takes into account internal and external disturbances, it considers both the progress of simulation time (which in the described, simple case is fixed) and the change of available computing power over time. Based on that load measurement method a dynamic and adaptive load evaluation can be performed.

Evaluating loads. During the load evaluation step, each processor p has to decide whether load balancing has to be performed at all, and if so, how many and which nodes are to be migrated.

Moving model nodes requires computing time and network resources. To avoid over-reaction caused by

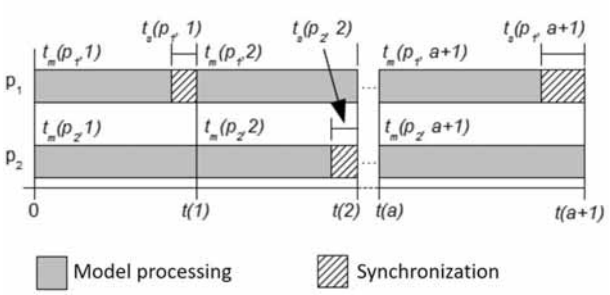


Figure 2: Load measurement on a system with two processors.

only short-term load imbalances, a smoothed value s_i of the synchronization time $t_s(p,i)$ is considered (see Equation 4) when deciding whether load balancing should take place. Nodes are only migrated if s_i is below a threshold value β_i.

$$s_i = \alpha * t_s(p,i) + (1-\alpha) * s_{i-1} \tag{4}$$

An effective method has to prevent overcompensation occuring due to long network runtimes or from attempting to balance even very small imbalances. To avoid thrashing, i.e. nodes being repeatedly sent back and forth between two processors, the value β_i is not constant, but changes over the course of the simulation run between limits β_{min} and β_{max}: If load movements have been performed in step i, the threshold value is decreased: $\beta_{i+1} = max(\beta_i/\gamma; \beta_{min})$, with $\gamma \geq 1$. Further movements are therefore only performed if processor p is heavily overloaded. If no load balancing has been performed for a while, the threshold value is increased: $\beta_{i+1} = min(\beta_i * \gamma; \beta_{max})$.

The number δ of to be migrated nodes from the set of all nodes V_{p_s} managed by the sending processor p_s is determined as $\delta = \max(1, \lfloor |V_{p_s}| * \varphi \rfloor)$, with φ being the ratio of nodes to be moved. Candidates are those nodes that have at least one edge to a node v_j managed by any other processor $p(v_j) \neq p_s$.

The method preferredly (priority 1) selects those nodes v_i for movement to a target processor $p_f(v_i)$ where the number of edges (v_i, v_j) to nodes v_j with $p(v_j) = p_f(v_i)$ is greater than the number of edges to nodes v_k with $p_s = p(v_k)$. That processor $p_f(v_i) = p(v_j)$ then is the target of a potential migration. In addition, any node v_i that has at least one edge to a node v_j managed by a processor $p(v_j) \neq p_s$ not currently running at full capacity can also be moved (priority 2).

Moving a priority 1 node v_i to processor $p_f(v_i)$ reduces the number of edges between model partitions. The load balancing method therefore does not only distribute the computing load evenly, but also reduces the expected communication load.

Moving loads. The load movement itself takes place during a defined time when all processors pause model computation. For optimistic methods that would be during the control mechanism, while conservative methods using barriers typically utilize the synchronization step. At that time changes can be made to the model graph without having to regard ongoing simulation calculations.

Here, each processor $p_s(v)$ encodes each model node v to be relocated as a message N_v, then sends it through the common cache or over the network to the corresponding target processor $p_f(v)$ and, if necessary, informs third processors that contain nodes with edges to v of its relocation. Each received message N_v is decoded and converted to a new node v, which is integrated into the partial model administrated by p_f.

4 Applications

Since their inception, a large number of applications of model-based parallelization and load balancing methods have been developed. A few typical applications reported on during the years are presented below.

Simulation of Electronic Circuits. Schlagenhaft et al. (see [21]) and Schlagenhaft (see [20]) describe a method to parallelize the simulation of the dynamic behavior of logical circuits. Their event-based model is parallelized using the optimistic Time Warp method. The executing processors are not exclusively available to the application, but are also used by third-party processes. The modeled logical circuits consist of switching elements between which dependencies in the form of binary signals exist. In the model, each switching element is mapped as an entity; these are combined into clusters by statically partitioning the model at the start of the simulation run; the clusters are then joined together to form partitions that are then assigned to the individual processors. These clusters are managed individually, with each cluster having its own FEL. Thus, clusters can be moved during load balancing. A further advantage of dividing the partitions into individual clusters comes into play in case of a rollback: Here, the

simulation does not have to be wholly discarded and recalculated for the entire partition, but only for a few clusters – or even only for a single one.

Schlagenhaft, et al. describe a dynamic and adaptive load balancing method to utilize the available resources in the best possible way. The global control mechanism is extended by a load balancing method that can move individual clusters between partitions. Load measurement, load evaluation and load shift are performed as part of the Global Virtual Time GVT_t calculation mechanism. When used on two processors and with a load from external processes (see [21]), the load balancing procedure improves the runtime by approx. 24%. Schlagenhaft (see [20]) reports on improvements of up to 60% when using six processors in networks with external loads.

Simulation of Social Interactions. Permalla (see [17]) presents a parallel discrete event model of the Naming Game, a sociological model of social interactions and consensus building without a central coordinating instance. They utilize the concurrency inherent in the model to implement an efficient application based on a parallel discrete event simulation framework. Analogous to the work of Schlagenhaft (see [20]) the individual entities are bundled into clusters, depending on the indidivual structure of the social network. These clusters are hosted by a processor core each that also administrates one FEL per cluster.

While the parallelization overhead resulting from the step from one to two involved processors significantly increases the runtime, Permalla reports a speedup of 3.43 using 16 processor cores on a single machine.

Simulation of Transit Networks. Ullrich et al. (see [24], [27], and [26]) utilize synchronized execution to parallelize transit simulation models. Their aim is to support the decision-making of operator personnel in the case of major network disturbances. Often the operators only have a short time window at their disposal, as decisions have to be taken in a matter of minutes or even seconds. To be effective, a simulation application enabling the online examination of the impact of potential counter-measures has to run fast, enabling the quick rejection of strategies unsuitable for specific situations. As the traffic operator's desktop computers that also run third-party user processes are the target platform of the resulting simulation tool, the method is specifically aimed at utilizing their small scale parallel processing capacity while being able to quickly shift load to idle processor cores in case external user processes claim resources. To address these issues, the method applies a dynamic and adaptive load balancing scheme analogous to the one described in Section 3.

Ullrich et al. report a speedup of 2.83 for four parallel processor cores with a common cache. Connecting machines over the network with its longer message delays reduces the speedup to 2.25. While the dynamic and adaptive load balancing mechanism only improves run time by a few percent on machines exclusively available to the transit simulation, it has a significantly larger impact when used to compensate for ressources assigned to third-party processes on machines concurrently used by other user processes (see [24]). Experiments with artificial loads demonstrate that effect of load balancing increases with the size of the model.

5 Conclusion

This paper presented an overview of basic concepts and methods of the parallel execution of discrete simulation models, with a focus on model-based parallelization. Following a short description of the background of parallel simulation, the two main classes of methods – conservative and optimistic execution – were presented, complemented by a description of typical static and dynamic load balancing mechanisms. Finally, some typical applications of model-based parallelization we introduced.

Model-based parallelization is a comparatively simple, easy to understand, but also very powerful approach. It is especially useful to accelerate the execution of large models that have to yield results fast. A wide array of applications has been reported on during the last two decades, including communications and electronics, disaster mitigation, health care, logistics, supply management, and transportation.

For further, more detailed study a number of sources authored by Richard Fujimoto, the unrivaled chronicler of the field, can be recommended: His introductionary book "Parallel and Distributed Simulation Systems" (see [6]) covers most concepts described in this article in great detail; it has aged exceedingly well. More recent developments are shared in his tutorial papers for the Winter Simulation Conference (see [7]). For students of the development of parallel (and distributed) simulation since the 1970s the historical overview "Parallel Discrete Event Simulation: the Making of a Field" by Fujimoto et al. (see [8]) is warmly commended.

References

[1] Banks, J., Carson, J. S., Nelson B. L., Nicol D. M.: *Discrete-Event System Simulation*. Pearson, 2010.

[2] Bryant, R. E.: *Simulation of Packet Communication Architecture Computer Systems*. Computer Science Laboratory. Technical Report, Cambridge, Massachusetts, Massachusetts Institute of Technology, 1977.

[3] Chandy, K. M., Misra, J.: Distributed Simulation: A Case Study in Design and Verification of Distributed Programs. In: *IEEE Transactions on Software Engineering*, SE-5(5), 1965, pp. 250-255.

[4] Dickens, P., Reynolds, P.: *SRADS with local rollback*. Institute for Parallel Computation, School of Engineering and Applied Science, University of Virginia, 1990.

[5] Fujimoto, R.: Lookahead in Parallel Discrete Event Simulation. In: Proc. *1988 International Conference on Parallel Processing*, 1988.

[6] Fujimoto, R.: *Parallel and Distributed Simulation Systems*. John Wiley & Son, 2000.

[7] Fujimoto, R.: Parallel and Distributed Simulation. In: Proc. *2015 Winter Simulation Conference*, 2015, pp. 45-59.

[8] Fujimoto, R., Bagrodia, R., Bryant, R. E., Chandy, K. M., Jefferson, D., Misra, J., Nicol, D., Unger, B.: Parallel Discrete Event Simulation: the Making of a Field. In: Proc. *2017 Winter Simulation Conference*, 2017, pp. 262-291.

[9] Fujimoto, R., McLean, T., Perumalla, K., Tacic, I.: Design of high performance rti software. In: Proc. *Fourth IEEE International Workshop on Distributed Simulation and Real-Time Applications*, 2000, pp. 89-96.

[10] Garey, M., Johnson, D., Stockmeyer, L.: Some simplied NP-complete graph problems. In: *Theoretical Computer Science*, 1976, pp. 237-267.

[11] Jefferson, D. R.: Virtual Time. In: *ACM Transactions on Programming Languages and Systems*, Vol. 7, No. 3, 1985, pp. 404-425.

[12] Kernighan, B. W., Lin, S.: An Efficient Heuristic Procedure for Partitioning Graphs.*Bell Syst. Tech Journal*, Vol. 49, No. 2, 1970, pp. 291-307.

[13] Lin, Y. B., Lazowska, E. D.: Exploiting lookahead in parallel simulation. In: *IEEE Transactions on Parallel and Distributed Systems*, Vol. 1, No. 4, 1990, pp. 457-469.

[14] Lubachevsky, B., Schwartz, A., Weiss, A.: Rollback Sometimes Works ... If Filtered. In: Proc. 1989 Winter Simulation Conference, 1989, pp. 630-639.

[15] Mattern, F.: Efficient algorithms for distributed snapshots and global virtual time approximation. In: *Journal of Parallel and Distributed Computing*, Vol. 18, No. 4, 1993.

[16] Meisgen, F.: *Dynamische Lastausgleichsverfahren in heterogenen Netzwerken*. Aachen: Shaker Verlag, 1998.

[17] Permalla, K. S.: Concurrent Conversation Modeling and Parallel Simulation of the Naming Game in Social Networks. In: Proc. *2017 Winter Simulation Conference*, 2017, pp. 1037-1048.

[18] Russell, S. J., Norvig, P.: *Artificial Intelligence: A Modern Approach (2nd ed.)*, Upper Saddle River, New Jersey: Prentice Hall, 2003, pp. 111-114.

[19] Sargent, R. G.: Verification and validation of simulation models. In: Proc. *2010 Winter Simulation Conference*, 2010, pp. 166-183.

[20] Schlagenhaft, R.: Dynamischer Lastausgleich optimistisch synchronisierter, verteilter Simulation. In: Proc. *ASIM-Workshop VSPP*, 1999.

[21] Schlagenhaft, R., Ruhwandel, M., Sporrer, C., Bauer, H.: Dynamic Load Balancing of a Multi-Cluster Simulator on a Network of Workstations. In: Proc. *PADS95*, 1995, pp. 175-180.

[22] Sokol, L. M., Stucky, B. K.: Experimental results for a constrained optimistic scheduling paradigm. In: *Distributed Simulation*, Vol. 22, 1990, pp. 169-173.

[23] Steinman, J. S.: Breathing time warp. In: Proc. *Seventh Workshop on Parallel and Distributed Simulation*, 1993, pp. 109-118.

[24] Ullrich, O.: *Modellbasierte Parallelisierung von Anwendungen zur Verkehrssimulation - Ein dynamischer und adaptiver Ansatz*. Dissertation, Univ. Köln, 2014.

[25] Ullrich, O., Lückerath, D.: An Introduction to Discrete-Event Modeling and Simulation. In: *Simulation Notes Europe (SNE)*, Vol. 27, No. 1, 2017, pp. 9-16.

[26] Ullrich, O., Lückerath, D., Franz, S., Speckenmeyer, E.: Simulation and optimization of Cologne's tram schedule. In: *Simulation Notes Europe (SNE)*, Vol. 22, No. 2, August 2012, pp. 69-76.

[27] Ullrich, O., Lückerath, D., Speckenmeyer, E.: Model-based parallelization of discrete traffic simulation models. In: *Simulation Notes Europe (SNE)*, Vol. 24, No. 3-4, 2014, pp. 115-122.

Cross-Layer Behavioral Modeling and Simulation of E/E-Architectures using PREEvision and Ptolemy II

Harald Bucher[1*], Simon Kamm[2], Jürgen Becker[1]

[1]Karlsruhe Institute of Technology – Institute for Information Processing Technologies, Engesserstr. 5, 76131 Karlsruhe; *bucher@kit.edu

[2]Vector Informatik GmbH, Philipp-Reis-Str. 1, 76137 Karlsruhe

SNE 29(2), 2019, 73 - 78, DOI: 10.11128/sne.29.tn.10472
Received: February 20, 2019 (Selected ASIM GMMS/STS 2019 Postconf. Publ.), Accepted: March 31, 2019
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna, ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. In this paper, an approach for integrated behavior modeling and simulation within model-based electric/electronic-architecture (EEA) descriptions is presented. It leverages actor-oriented and UML state chart behavior modeling to address complex reactive systems. A key contribution is the aggregation of cross-layer behavior specified at the logical function architecture layer and at the hardware layer together with further properties of the EEA like the current consumption of electronic control units (ECUs) and the underlying network topology. The EEA and behavior modeling is done in the common industry tool PREEvision. Using these static descriptions, a unified simulation model is synthesized and executed using Ptolemy II, which extends a previously developed approach. In addition, a concept to feed back the simulation data into PREEvision is briefly described e.g., to further evaluate the gained results. Finally, a proof-of-concept is presented using an Adaptive Cruise Control application.

Introduction

Automotive electric/electronic-architectures (EEAs) are steadily growing in complexity due to the integration of evermore functions [1]. To cope with that complexity at system level, model-based architecture description languages (ADLs) and tools have been established in recent years such as the EAST-ADL [2], EEA-ADL [3] (realized in the tool PREEvision [4,1]) and Vehicle Systems Architect [5], each of which are compliant to the AUTOSAR [6] standard. Each of them offer sophisticated static modeling capabilities from several viewpoints such as requirements, functional network, hardware/software architecture, wiring harness and topology.

A common process is to start with the realization-independent and early stage modeling of the logical function architecture which typically stays stable for years and thus is the basis for further refinements in the development life cycle [1]. Another trend is the architecture-centric modeling of behavior integrated within the model-based EEA descriptions in order to have a common formal format for exchange and subsequent simulation analysis. The trend to amend this is underlined e.g., by the behavioral annexes of the EAST-ADL [2] and the AADL [7] or the integration of UML-compliant state charts into the latest PREEvision release v9.0 [4]. Therefore, recent research is focused on the generation of executable behavior from these static descriptions [8,9,10,11,12,13].

A downside of the behavioral annexes is that they only support the association of architectural components with simple, flat finite state machines (FSMs) which result in state and transition explosion with more complex systems. The mentioned approaches therefore often delegate detailed behavior to external descriptions which results in the loss of the integrated characteristics. In addition, it elicits inconsistencies between the architecture and behavior models and prevents the consideration of lower abstraction layers.

An approach which faces this challenge is presented in [8] by synthesizing and executing a cross-domain simulation from static EEA descriptions designed in PREEvision. In this work we extend that approach to support both actor-oriented and state chart behavior modeling to address complex reactive systems.

Concerning state charts the UML subset provided by PREEvision is leveraged and enhanced to support extended state machines to further handle complexity.

In addition, cross-layer behavior specifications and further EEA properties from lower abstraction layers are synthesized into a unified Ptolemy II (PtII) simulation model. A concept to feed back the simulation data into PREEvision is extended and completes the contributions.

1 Background

The overall baseline approach as proposed in [8] and the extensions addressed in this work are shown in Figure 1. Starting point is a data model e.g., as provided by PREEvision, which captures all relevant abstraction layers of an EEA. For modeling *executable* behavior integrated within the EEA model, a new layer called *Behavioral Logical Architecture (BLA)* is introduced that refines the static logical blocks with detailed behavior by reusing actors [14] from the *PtII Actor Library*. The library contains actors of the heterogeneous modeling and simulation tool Ptolemy II [15] and is imported as a separate library of logical block types into PREEvision. These block types are used to instantiate actors at the BLA layer. In combination with mappings from the LA layer to lower layers they provide the connection of the behavioral blocks of the BLA to domain-specific information at lower layers enabling the cross-domain simulation of the underlying network communication or even electric circuits [16] in an aspect-oriented manner. A variant-sensitive synthesis is also implemented supporting the analysis of architecture variants [17].

2 Concepts

To amend the baseline actor-oriented modeling with state-based behavior we leverage the newly added capability of PREEvision v9.0 to refine architecture artifacts with state charts across several layers including the logical architecture and components of the hardware layer such as ECUs and internal processing units.

The basic principle is to annotate a state chart as child artifact to an architecture artifact. Dependent on the abstraction layer, the interfaces to the state chart comprise different data providers and consumers. At the logical architecture, for instance, communication between functions is done via typed ports, which have attached an interface. The interface specifies the actual data exchanged e.g., in terms of data elements. This follows the AUTOSAR standard.

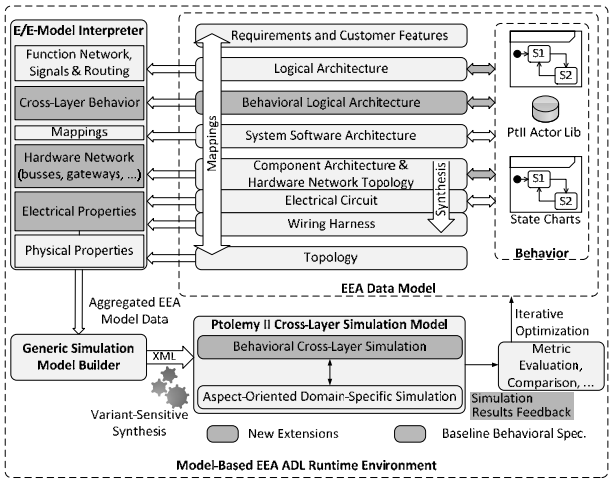


Figure 1: Approach for cross-domain simulation synthesis of model-based EEAs [8] and new extensions to combine cross-layer behavior specifications using UML state charts and actor-oriented library components.

The specified data elements of each port are then available in the state chart of the function to use them in guard and action expressions. The modeling is illustrated in Figure 2. A similar modeling approach applies for hardware components except that state charts are annotated at instance level and data providers differ from data elements.

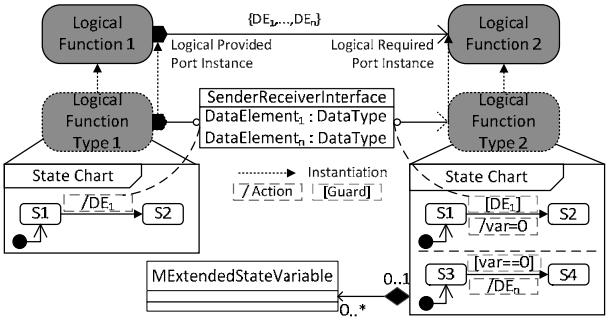


Figure 2: Modeling principle to refine logical function types with state charts. Communication is done via data elements.

2.1 Extended State Machines

A downside of the current state chart modeling capability is the missing support of extended state machines, which can significantly reduce the complexity [15]. Therefore, we propose a meta-model extension by extended state variables. This is indicated in Figure 2 by the composition of the state chart with the proposed meta-class `MExtendedStateVariable`.

2.2 Combining Actor-oriented and State Chart Behavior Simulation

In the baseline approach in Figure 1, behavior is specified by mapping an atomic logical function instance to a composite building block at the BLA layer. Executable actors are instantiated within that building block. Port prototype mappings are generated once to ensure the consistency between the interfaces of the atomic logical function and its refinement building block.

State charts are simulated using modal models [15,18] in PtII. Modal models basically represent a specialized composite actor containing a hierarchical state machine governed by an *FSMDirector*. Each state can contain another state machine refinement or even an actor-oriented sub-model following a distinct execution semantics i.e., a different model of computation (called *Director* in PtII). Modal models are also suitable to deterministically simulate *hybrid systems* [15]. Data exchange between modal models is done via ports. Therefore, a building block of type *ModalModel* is used to identify logical functions which contain a state chart description.

Additional data element sub-mappings are generated once in order to respect the interfaces of the logical functions and to connect the simulation model counterparts during simulation model synthesis. Each port of a building block represents a data element. See Figure 3.

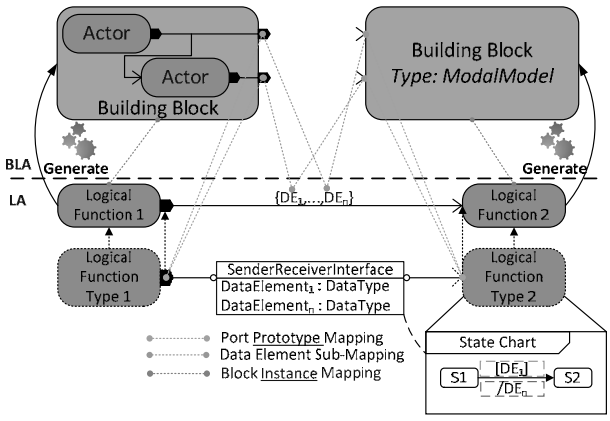


Figure 3: Generation of BLA building block stubs and mappings. State charts are encapsulated in a building block of type *ModalModel*.

2.3 Cross-Layer Behavioral Synthesis

To allow the simulation of cross-layer behavior we leverage the state chart refinements of hardware components. However, the link to higher layers i.e., the logical layer, is missing.

Therefore, we propose to use AUTOSAR-oriented `BasisServiceInterfaces` on logical ports in order to provide additional data elements or operations to communicate with state charts of mapped hardware components. In addition, we propose to reference ECU attributes as state chart variables. For instance, this enables the modeling and simulation of mode-based cross-layer behavior, where functions can request a certain operating mode of the ECU and only perform their functional behavior if the ECU responds it is ready to run. In order to allow spontaneous FSMs [15], which not only react on input events, a `timeout` guard expression (taken from PtII) is introduced e.g., to model the startup time of the ECU. A cross-layer model is exemplarily shown in Figure 4.

Finally, we propose the mapping of current consumption descriptions in terms of PREEvision's meta-class `MCurrentDescriptorType` on state transitions of hardware states. Together with the timed behavior, a mode-based current consumption can be simulated.

In the synthesized PtII model, the function and hardware state charts are encapsulated in distinct modal models communicating via ports which represent the basis service interfaces. An additional output port is generated for the current consumption of the ECU.

Hardware Network. In [8], network communication between functions such as CAN is traced based on their mapping to the hardware and is considered in the resulting simulation in an aspect-oriented way. Together with the state chart refinement of ECUs and processing units, it is possible to automatically include additional behavior along the communication path, such as gateways, by cascaded aspect-oriented simulations. Typically gateways have no logical function counterpart, since they are dependent on the mapping.

2.4 Simulation Data Feedback

To make use of the simulated results in PREEvision in order to perform further analysis and to relate the results with the original EEA model artifacts, a feedback approach is applied. The approach relies on OSGi [19] and is further described in [17]. We reuse and extend the approach by implementing a listener for modal model controllers focusing on the feedback of information about the simulated state machines such as timestamps, current state, previous state, output and variable actions.

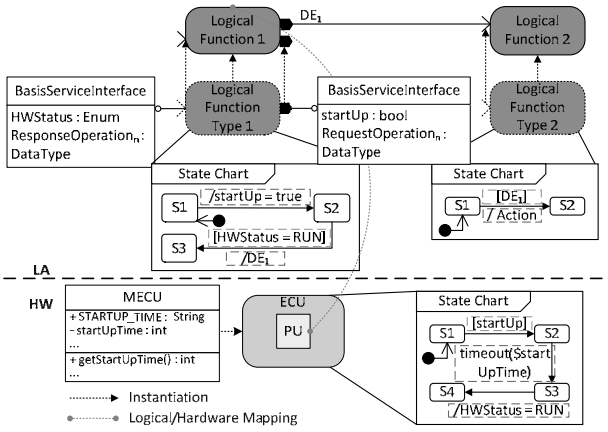


Figure 5: Cross-layer behavior specification using basis service interfaces on logical ports to communicate with the mapped hardware component state chart via additional data elements or operations.

2.5 Transformation Rules

In Table 1 the basic transformation rules between PREEvision's UML state chart subset and modal model artifacts in PtII are summarized.

Note that each generated PtII artifact is suffixed by the UUID of the original EEA model artifact in order to uniquely relate the artifacts and avoid name conflicts on PtII side.

3 Use Case Results

In this chapter, the concepts are demonstrated by means of an *Adaptive Cruise Control (ACC)* application presented in [8] which is enhanced based on Figure 4. The logical function architecture is shown in Figure 5.

UML State Chart Subset	Ptolemy II Modal Models
simple state, choice & junction pseudo-state	state
initial pseudo-state	state with property *isInitialState*
final state	state with property *isFinalState*
composite state	*state machine refinement* state
orthogonal state	*default refinement* state containing a discrete-event director and a modal model composite for each parallel region. Data dependencies between regions are analyzed and communicated via ports between the affected modal models. [18]
deep history state	history transition
state transition	ordinary transition
guard condition	guard expression
IO/variable action	output/set action expression

Table 1: Basic transformation rules between PREEvision's UML state chart subset and PtII modal models.

The *ACC_Testbench* generates the stimuli for the vehicle speed and radar speed sensor functions as well as for the ACC controller in a closed-loop fashion based on the calculated acceleration of the ACC controller. The stimuli values are generated with a sample period of $100ms$. The initial speeds and the distance are set to $15m/s$ and $190m$ respectively. Each of the functions offer `BasisServiceInterfaces` to request or retrieve a certain operating mode of the state chart of their mapped hardware component. The corresponding BLA building block stubs and mappings are generated according to Figure 3.

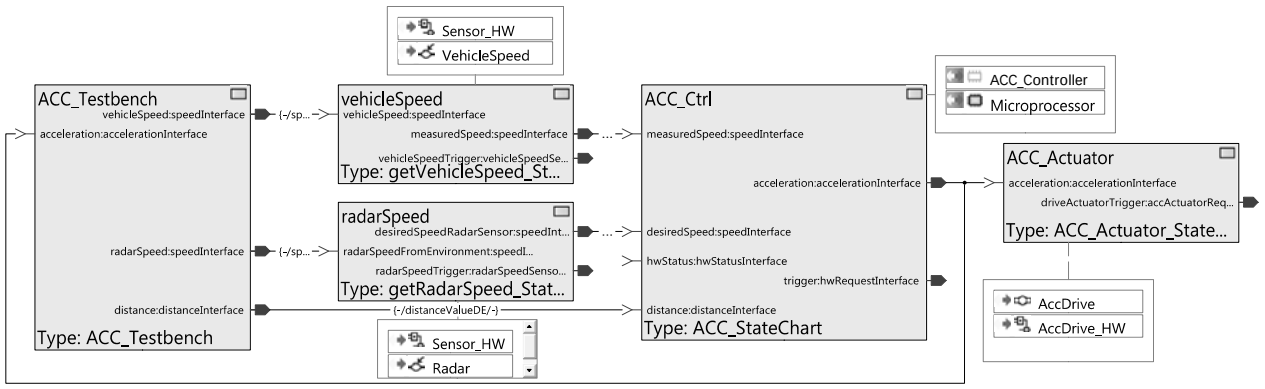


Figure 4: Logical function architecture of the ACC application. Each of the logical functions except for *ACC_Testbench* is refined by a state chart. Their mapping to the hardware layer is illustrated by the annotated text boxes. The behavior of the *ACC_Testbench* is modelled actor-oriented at the BLA layer and is not mapped to the hardware. Thus, a combined actor-oriented and state chart modeling is applied.

3.1 State Charts

The most important state charts are the one of the ACC controller shown in Figure 6 and its corresponding ECU state chart realizing an ECU Manager depicted in Figure 7. The remaining state charts of the sensor and actuator functions and their hardware are modeled simple. They only forward/retrieve the speed/acceleration values and request the sensors/actuator to run as long as they receive values. The ACC is calculating the acceleration only if the ECU is ready to run. The orthogonal state *operate* limits the calculated acceleration by the state variables *aMin* and *aMax*. In the *freeRoad* state the radar detects no vehicle, the own speed reached the desired speed and the sleep mode is requested. A wakeup is triggered when the radar detects a new vehicle. A shutdown is requested when the vehicle stands still.

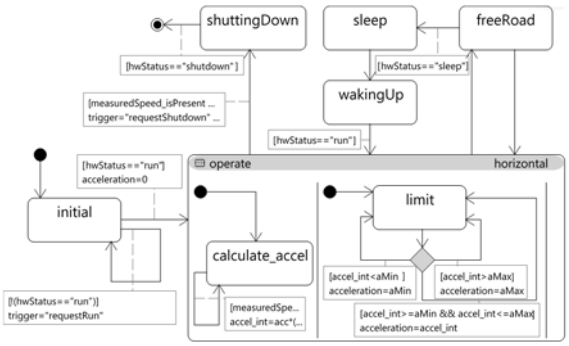


Figure 6: ACC controller state chart. Some transition actions are omitted for space reasons.

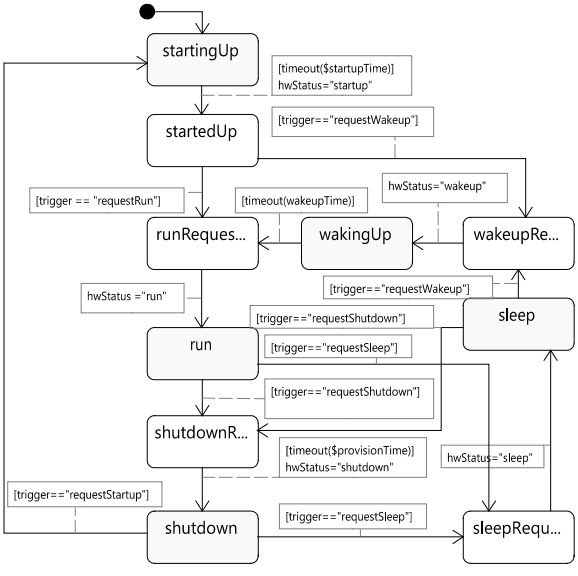


Figure 7: ACC ECU state chart realizing an ECU Manager oriented on the AUTOSAR fixed ECU Manager. Transitions to the yellow states have mapped a current descriptor type in order to simulate a mode-based current consumption.

The ACC ECU state chart represents the different operating modes which can be requested by the ACC controller state chart and sends back the current status via the `BasisServiceInterface`s. In addition, the startup- and provision time attributes of the ECU (50ms and 200s) are referenced as well as a *wakeupTime* state variable (10ms) which are used as timeouts. Provision time is the time a component stays active after its shutdown is requested.

3.2 Simulation Results

Figure 8 shows the PtII plot of the ACC simulation. Until 250s the vehicle is following the leading vehicle. Then the leading vehicle disappears and the ACC accelerates to its desired speed at free road. At 300s a new vehicle is detected at a distance of 200m. At 350s the vehicle is decelerating until it stands still at 380s. At 300s the acceleration is limited to *aMin*.

Figure 9 shows the mode-based current consumption of the hardware components at the key time-points with synthetic values.

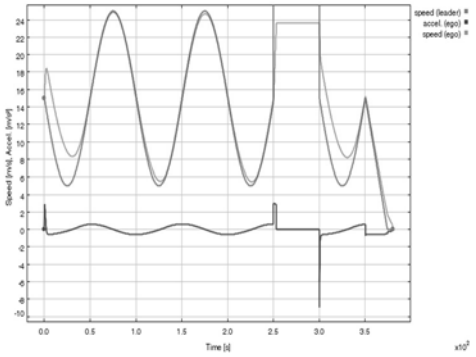


Figure 8: ACC simulation showing the speed of the leading vehicle (red) and the ego vehicle (green) in m/s as well as the acceleration calculated by the ACC (blue) in m/s^2.

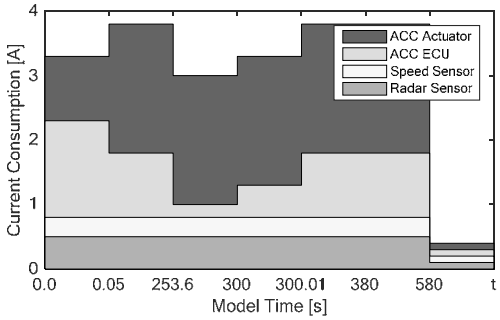


Figure 9: Current consumption of the mapped ACC hardware dependent on the operating mode. Created based on the fed back simulation data which is written to a CSV file in PREEvision.

The sensors are operating until their shutdown. Until 50*ms* the ACC ECU and Actuator are starting up before they are ready to run. At 253.6*s* the vehicle has reached its desired speed at free road and the ACC ECU goes to sleep mode. At 300*s* it wakes up for 10*ms*. At 380*s* the vehicle stands still but all hardware components stay active for the same provision time before they shutdown at 580*s*.

4 Conclusion

In this paper, we presented a set of concepts and their evaluation by an ACC application to model and simulate behavior of model-based EEAs in an integrated manner. The key contribution is the combination of actor-oriented and state chart based behavior across several abstraction layers. This enables new possibilities to analyze model-based EEAs in early development stages dependent on architectural decisions and information.

Future work could include enhanced support of UML state charts, the integration and consideration of behavior at the AUTOSAR-compliant system software architecture layer and envisioning their code generation.

References

[1] Schäuffele J, "E/E Architectural Design and Optimization using PREEvision," in *SAE Technical Paper 2016-01-0016*, 2016. [Online]. https://doi.org/10.4271/2016-01-0016

[2] EAST-ADL Association. (2013) EAST-ADL Domain Model Specification. [Online]. http://www.east-adl.info/Specification

[3] Matheis J, „Abstraktionsebenenübergreifende Darstellung von Elektrik/Elektronik-Architekturen in Kraftfahrzeugen zur Ableitung von Sicherheitszielen nach ISO 26262," Karlsruhe Institute of Technology, Ph.D. thesis ISBN: 978-3-8322-8968-3, 2010.

[4] Vector Informatik GmbH, "PREEvision v9.0 Manual," 2018.

[5] Mentor Graphics. (2018, Oct.) Volcano™ Vehicle Systems Architect. [Online]. https://www.mentor.com/products/vnd/autosar-products/volcano-system-architect

[6] AUTOSAR Consortium. (2018) AUTOSAR 4.4 (Automotive Open System Architecture) Specifications. [Online]. http://www.autosar.org

[7] SAE International, "SAE Architecture Analysis and Design Language (AADL) Annex Volume 2: Annex B: Data Modeling Annex, Annex D: Behavior Model Annex, Annex F: ARINC653 Annex," USA, Standard AS5506/2, Jan. 2011.

[8] Bucher H, Reichmann C, Becker J. "An Integrated Approach Enabling Cross-Domain Simulation of Model-Based E/E-Architectures," in *SAE Technical Paper 2017-01-0006*, Mar. 2017. [Online]. http://papers.sae.org/2017-01-0006/

[9] Weissnegger R et al. "Simulation-based Verification of Automotive Safety-critical Systems Based on EAST-ADL," *Procedia Computer Science*, vol. 83, pp. 245-252, 2016.

[10] Marinescu R et al. "Analyzing Industrial Architectural Models by Simulation and Model-Checking," in *Formal Techniques for Safety-Critical Systems*.: Springer International Publishing, 2015, vol. 476, pp. 189-205.

[11] MAENAD Consortium, "MAENAD Analysis Workbench," Deliverable D5.2.1 V4.0 2014. [Online]. http://www.maenad.eu/public/Deliverables

[12] Lasnier G, Pautet L, Hugues J, Wrage L. "An Implementation of the Behavior Annex in the AADL-Toolset Osate2," in *2011 16th IEEE International Conference on Engineering of Complex Computer Systems*, Apr. 2011, pp. 332-337.

[13] Larsen PG et al. "Integrated Tool Chain for Model-Based Design of Cyber-Physical Systems," in *The 14th Overture Workshop: Towards Analytical Tool Chains*, vol. 4/28, Nov. 2016, pp. 63-79.

[14] Lee EA, Neuendorffer S, Wirthlin MJ. "Actor-Oriented Design Of Embedded Hardware And Software Systems," *Journal of Circuits, Systems, and Computers*, vol. 12, pp. 231-260, 2003.

[15] Claudius Ptolemaeus, *System Design, Modeling, and Simulation using Ptolemy II*.: Ptolemy.org, 2014. [Online]. http://ptolemy.org/books/Systems

[16] Bucher H, Becker J. "Electric Circuit- and Wiring Harness-Aware Behavioral Simulation of Model-Based E/E-Architectures at System Level," in *2018 IEEE International Systems Engineering Symposium (ISSE)*, 2018, pp. 1-8.

[17] Bucher H, Neubauer K, Becker J. "Automated Assessment of E/E-Architecture Variants using an Integrated Model- and Simulation-based Approach," in *SAE Technical Paper 2019-01-0111*, 2019.

[18] Lee EA, Tripakis S. "Modal Models in Ptolemy," in *Proceedings of the 3rd International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, Oct. 2010, pp. 11-21.

[19] Wütherich G, Hartmann N, Kolb BJ, Lübken M. *Die OSGi-Service-Platform: eine Einführung mit Eclipse Equinox*, 1st ed.: dpunkt Verlag, 2008.

RPDEVS Abstract Simulator

Franz J. Preyser*, Bernhard Heinzl, Wolfgang Kastner

Institute of Computer Engineering, Automation Systems Group, TU Wien, Treitlstraße 1-3, 1040 Vienna, Austria; *franz.preyser@tuwien.ac.at

SNE 29(2), 2019, 79-84, DOI: 10.11128/sne.29.tn.10473
Received: March 30, 2019 (Selected ASIM GMMS/STS 2019 Conference Publication); Accepted; May 10, 2019
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna, ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. The Revised Parallel DEVS (RPDEVS) modeling formalism enhances the Parallel Discrete Event System Specification (PDEVS) by the ability to model 'real' Mealy behavior of components. The term 'real' Mealy behavior can be summarized as immediate output response to an input event without a state transition in between. Although this enhancement simplifies model creation, especially of reusable components, it requires a more complex simulation algorithm. In this paper, we present an RPDEVS abstract simulator that describes the simulation execution of RPDEVS models.

Introduction

The Discrete Event System Specification (DEVS) [1] is a modular and hierarchical modeling formalism for systems that process input events, have an internal state, and may produce output events. Basic components can be specified as atomic DEVS which can be coupled with one other in a block diagram manner. The formal definition of an atomic DEVS is similar to a finite automaton (or sequential machine). In [2], the author describes an atomic DEVS as *DEVS Moore Automaton* embedded in additional logic that provides the necessary time events. Automata theory distinguishes between Moore and Mealy automata. The output events of Moore automata solely depend on the system's current state, whereas the output of Mealy automata may also depend on the current input. In theory, these two types of automata are equivalent in the sense that every automaton of the one type can be replaced by a corresponding automaton of the other type. However, in average the Moore model needs about twice the number of states and transitions than the corresponding Mealy model to represent the same system [3].

Both, in classic DEVS and in its most popular revision PDEVS [4], the output function λ solely depends on the internal state of the system. Thus, these two formalisms only allow the modeling of Moore behavior. If Mealy behavior is needed, it has to be modeled with a workaround, using a *transitory state* (a state with zero lifetime). However, as discussed in [5], the use of transitory states leads to a delay of events regarding processing order, which in turn impedes reusability of components. Due to the reasons mentioned above and the experiences we made with applying both, DEVS [6] and PDEVS [7], we decided to revise PDEVS resulting in RPDEVS published in [8]. Basically, the changes include the support of 'true' mealy behavior and the merging of the three state transition functions δ_{int}, δ_{ext}, and δ_{conf} into one generic state transition function δ. As mentioned above, a Mealy automaton needs about half the states compared to the corresponding Moore automaton. Evaluation of RPDEVS shows that formalization of Mealy models simplifies to a similar extent compared to PDEVS. Also merging the state transition functions condenses model definition, since the different transition functions often match at least in parts. However, the price for simplifying modeling is an increase in the complexity of the simulation algorithm.

In this work, we first recap the RPDEVS formalism, before its simulation algorithm is described and presented as *abstract simulator*.

1 RPDEVS Formalism

Equally to classic DEVS and PDEVS, RPDEVS distinguishes between *atomic* and *coupled* components which can be used for modular and hierarchical structuring of complex models (see Figure 1). As shown in [8], RPDEVS also provides *closure under coupling*, which means that for every coupled component an equivalent atomic component can be designed. This assures that couplings can be used within other couplings as if they were atomics.

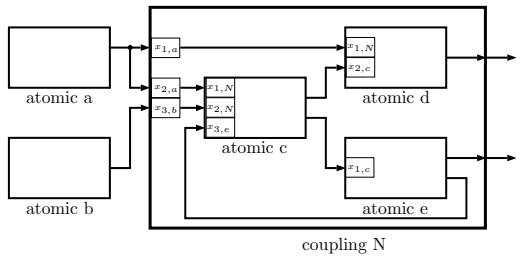


Figure 1: Modular and hierarchical decomposition of a complex model into atomic and coupled RPDEVS components.

1.1 Atomic RPDEVS

Formally, an atomic RPDEVS M is defined as

$$M =< X^b, S, Y^b, \delta, \lambda, ta >,$$

where the single entities have the following meanings:

$X^b \ldots$ set of possible input bags
$S \ldots$ set of possible states (=state space)
$Y^b \ldots$ set of possible output bags
$\delta : Q \times X^b \to S \ldots$ *state transition function*
$\lambda : Q \times X^b \to Y^b \ldots$ *output function*
$ta : S \to [0, \infty] \ldots$ *time advance function*
$Q = \{(s,e) | s \in S, e \in [0, ta(s)]\}$
$e \ldots$ *elapsed time* since last event

Theoretically, X^b is a set of multisets with no particular structure. However, for practical implementation where it is feasible to define input ports which can be connected individually to output ports of other components, the set of possible input bags may be structured into sub-bags, one for each input port. Additionally, the sub-bags can be structured according to the source components the corresponding input messages origin from (see Figure 1). This is especially done in the RPDEVS simulation algorithm presented in Section 2, which has to remember the source component of every input message.

The differences of an atomic RPDEVS compared to PDEVS are the input dependency of the output function λ and the single state transition function δ which replaces the three separated transition functions δ_{int}, δ_{ext}, and δ_{conf} (for details about PDEVS, see [4, 1]). Furthermore, in RPDEVS, λ is called on any kind of event, external, internal, and confluent. The explicit distinction between these three event types is dropped and the behavior of an RPDEVS atomic is the same for each of them:

1. Call the output function λ.
2. Recalculate λ as long as the input bag changes due to (re)calculations of lambda at influencing components (*lambda-iteration*).
3. Conduct state transition δ once (*delta-step*).
4. Call the time advance function ta which returns the time to the next internal event.

If different treatment is necessary depending on whether the event was triggered by the arrival of an input (external event), by the expiration of the current state's lifetime (internal event), or by both happening concurrently (confluent event), this has to be incorporated into the definitions of δ and λ. External events can be recognized by a non-empty input bag ($x^b \neq \emptyset$), whereas internal events imply $e = ta(s)$.

The single transition function δ avoids having to define identical behavior multiple times in cases in which the three transition functions partly match.

According to [9], it frequently happens that calculations necessary for the output event in λ are also necessary for the computation of the next state and thus, have to be repeated in δ_{int}. In the classic DEVS simulator *DesignDEVS* [10], they even merge the two functions λ and δ_{int} to prevent unnecessary recalculations. This is not possible for RPDEVS as λ may have to be called multiple times before the state transition can be conducted. Therefore, for practical implementation, we recommend to split the internal state s of an atomic into two parts $s = (s_\delta, s_\lambda) \in S = S_\delta \times S_\lambda$ which allows to redefine λ and δ as follows:

$$\begin{aligned} \lambda: &\quad S_\delta \times [0,\infty) \times X^b \to Y^b \times S_\lambda,\ (s_\delta, e, x^b) \mapsto (y^b, s_\lambda) \\ \delta: &\quad S_\delta \times S_\lambda \times [0,\infty) \times X^b \to S_\delta,\ (s_\delta, s_\lambda, e, x^b) \mapsto s_\delta \end{aligned}$$

Thus, when λ already needs to calculate a new state value for generating the output y, it can be buffered into s_λ to be reused in δ.

1.2 Coupled RPDEVS

The formal definition of a coupled RPDEVS is identical to that of a coupled PDEVS (see [4]):

$$N =< X^b, Y^b, D, \{M_d\}_{d \in D}, \{I_d\}_{d \in D_N}, \{Z_{i,d}\}_{i,d \in D_N} >$$

with $D_N = D \cup \{N\}$ and

X^b ... set of possible input bags
Y^b ... set of possible output bags
D ... index set
M_d ... child component of N for each $d \in D$
$I_d \subset D \cup \{N\}$ ... influencer set of d
$Z_{i,d}$ ... output translation function

The output translation function $Z_{i,d}$ translates the output events of component i into input events for component d. Theoretically, $Z_{i,d}$ could also alter output events. However, in practice it just forwards events. If the destination component is a coupling, the output translation functions of that coupling further forwards the events to the destinations within the coupling. This is repeated until finally the events reach atomics.

As already mentioned in Section 1.1, the multiset of possible input bags can be structured by input port and source component. In the following, we will not consider ports, but separate the input bags according to the influencers the messages originate from. Such a structuring for a component d has the form

$$X_d^b = \prod_{i \in I_d} X_{i,d}^b,$$

with $X_{i,d}^b$ being the multiset of possible input messages from component i ($Z_{i,d} : Y_i^b \to X_{i,d}^b$). Consequently, every input bag x_d^b of a component d has the form

$$x_d^b = (x_{i_1,d}^b, x_{i_2,d}^b, \ldots, x_{i_l,d}^b), \quad I_d = \{i_1, i_2, \ldots, i_l\}.$$

Thereby, $x_{i_k,d}^b$ is the translated result of the output function of influencer i_k: $x_{i_k,d}^b = Z_{i_k,d}(y_{i_k}^b)$, $\forall k = 1, 2, \ldots, l$.

2 RPDEVS Abstract Simulator

To complete the introduction of RPDEVS started in [8], the definition of an abstract simulator is given. Like in classic DEVS and parallel DEVS, the code consists of a *simulator* part responsible for executing an atomic, a *coordinator* part responsible for executing a coupling, and a *root-coordinator* responsible for the overall model execution. Furthermore, we stick to the format known from [1], using message passing. There are five types of messages used:

`i-message` The initialization message is sent to every component at simulation start. It is used to initialize state variables and gather the times of the first internal events at the single components.

`*-message` In PDEVS, this is the *internal state transition message* because there the output function λ is inseparably connected to the internal and confluent state transitions δ_{int} and δ_{conf}. However, in RPDEVS, λ is calculated in an iterative manner and on every kind of event. Thus, this message is solely used to trigger the λ iteration.

`y-message` The `y-message` is used to transport the output message calculated in λ to the parent coordinator where it is forwarded to the input bag of the receiving component.

`x-message` In RPDEVS, the `x-message` is used to trigger the state transition. Whenever a component receives an `x-message`, it executes δ and then calculates the time of its next internal event t_n.

`done-message` This message is used for synchronization. When the coordinator triggers child components to do their initialization or to conduct their state transition, it has to wait until all of them are done before simulation can proceed.

2.1 Simulator

The simulator of an atomic RPDEVS is nearly identical to the one of an atomic PDEVS (see [1], p. 285):

```
RPDEVS-simulator
  variables:
    parent      // parent coordinator
    tl          // time of last event
    tn          // time of next event
    RPDEVS      // assoc. model with total
                // state (s,e), time advance
                // function, lambda and delta
    (s_i,e_i)   // initial total state
    y           // output message bag

  when receive i-message(i,t)
   (s,e) = (s_i, e_i)
    tl = t - e
    tn = tl + ta(s)
    send done-message(done, tn) to parent

  when receive *-message(*,x,t)
    e = t - tl
    y = lambda(s,e,x)
    send y-message(y,t) to parent

  when receive x-message(x,t)
    s = delta(s,e,x)
```

```
    tl = t
    tn = tl + ta(s)
    send done-message(done, tn) to parent
end RPDEVS-simulator
```

The most important differences compared to the PDEVS simulator are the additional parameter `x` of the `*-message`, and the absence of the case distinction between *internal*, *external*, and *confluent* event when receiving an `x-message`. Like in [11], a `done-message` is used for synchronization during the potentially parallel execution to prevent the problems with Zeigler's PDEVS algorithm described in [12].

2.2 Coordinator

The more interesting part of the abstract simulator is the *coordinator*. We start with the definition of all necessary variables followed by the `i-message` and `done-message` procedures:

```
RPDEVS-coordinator
  variables:
    parent      // parent coordinator
    tl          // time of last event
    tn          // time of next event
    RPDEVS      // associated coupled model
                // including index set D,
                // influencer sets I_d, and
                // output transl. fcts. Z_id
    event-list  // list of elements (d,tn_d),
                // sorted ascending by tn_d
    IMM         // imminent children
    y_coupling  // output message of coupling
    x_dr        // sub input bags:
                // d... sender, r... receiver
    x_r         // input bag of component r
    y_dN        // sub output bag of coupling
                // N, d... sender
    INF         // set of influenced children
                // (with changed input bag)
    INF'        // INF for next lambda-iter.
    DELTA       // set of children who need to
                // conduct a state transition
    CHECK       // components with withdrawn
                // input messages

  when receive i-message(i,t)
    DELTA = D
    for-each d in D do
      send i-message(i,t) to child d
    wait until DELTA = {}
    sort event-list according to tn_d
    tl = max{tl_d : d in D}
    tn = min{tn_d : d in D}
    send done-message(done, tn) to parent

  when receive done-message(done, td) from d
    event-list.(d,tn_d) = (d,td);
    remove d from DELTA
```

At simulation start, the coordinator receives an `i-message` from its parent coordinator. The parent of the uppermost coordinator is the *root-coordinator* (see Section 2.3). The `i-message` is forwarded to all child components `d`$\in$`D` which causes them to calculate their time of next internal event `tn_d`. Then, the coordinator waits until all children have sent their `done-message` (i.e. `DELTA={}`) before the time of the next internal event `tn` of the coupling can be determined.

If a component is imminent (i.e. its time of next event `tn=t`), it receives a `*-message` from its parent coordinator. This message initiates the λ iteration in the coupling. The goal of the λ iteration of a coupling is generating its output message `y_coupling`.

```
when receive *-message(*,x,t)
  y_coupling = {}
  for-each (d,tn_d) in event-list with tn_d=t
    add d to IMM, DELTA and INF
    remove (d,tn_d) from event-list
  for-each r in D with N in I_r
    if x_Nr != Z_Nr(x)
      x_Nr = Z_Nr(x)
      add r to INF and DELTA
      if x_Nr={}
        add r to CHECK
  for-each r in INF
    x_r = {x_dr : d in I_r, x_dr != {}}
  while CHECK != {}
    pick and remove r from CHECK
    if x_r={}
      if r not in IMM
        remove r from INF and DELTA
        for-each d in D with r in I_d
          x_rd = {}
          remove x_rd from x_d
          add d to CHECK
        if r in I_N
          y_rN = {}
  INF'={}
  for-each r in INF
    send *-message(*,x_r,t)
```

In the `*-message` of the coordinator first, the imminent children are determined and collected in `IMM`, `INF`, and `DELTA`. Then, the components' input bag changes caused by the couplings input are calculated. All components whose input bag changed are added to `INF` and scheduled for state transition by adding them to `DELTA`. Finally, `*-messages` are sent to the affected child components triggering their λ execution.

These λ executions result in output messages transported via `y-messages` back to the coordinator. In the coordinator's `y-message` procedure, all output messages of all triggered child components are gathered and converted using the output translation functions `Z_dr`. Depending on the coupling relations, they are converted either into input messages for other children or into coupling output messages. Thereby, all child components whose input bag has changed are collected in `INF'`. After the last element in `INF` has responded to the coordinator with a `y-message`, the components in `INF'` are shifted into `INF`. If `INF` is not empty after that, again a `*-message` is sent to every component in `INF` and their response, in form of `y-messages` is awaited. However, if `INF` is empty at the end of the `y-message` procedure, it means no input bag has changed during the last λ iteration, i.e. they are stable. Thus, the λ iteration of the coupling has terminated and the coordinator can send a `y-messages` to its parent. In [8], it is shown that for models without algebraic loops, the λ iteration always terminates after a maximum of $n = |D|$ iterations. In some cases, algebraic loops can even be solved by the simulation algorithm (see RS flip-flop in [13]).

During the course of λ iterations, it may happen that input messages for child components that were produced in previous iterations may have to be withdrawn from the respective input bag. Thereby, it may occur that the input bag becomes completely empty although it was not in the preceding iteration. These components then need to be checked separately because they may already have produced output in reaction to a non-empty input bag (Mealy behavior) and thereby may have influenced other components. This task is handled via the set `CHECK`.

A coordinator may represent a coupling that is used as component in a parent coupling. In this parent coupling, there is also a λ iteration in progress. Thus, the parent coordinator may send multiple `*-messages` to its child coordinators. This is why the `*-message` procedure of the coordinator also has to check whether formerly received coupling inputs still exist in the new iteration (using `CHECK`).

```
when receiving y-message(y_d,t) from d
  remove d from INF
  if d in I_N
    y_dN = Z_dN(y_d)
  for-each r in D with d in I_r
    if x_dr != Z_dr(y_d)
      x_dr = Z_dr(y_d)
      if x_dr={}
        add r to CHECK
      add r to INF' and DELTA
  if INF = {}
    INF = INF'
    INF' = {}
    for-each r in INF
      x_r = {x_dr : d in I_r, x_dr != {}}
    while CHECK != {}
      pick and remove r from CHECK
      if x_r={}
        if r not in IMM
          remove r from INF and DELTA
          for-each d in D with r in I_d
            if x_rd != {}
               x_rd = {}
               remove x_rd from x_d
               add d to INF, DELTA and CHECK
          if r in I_N
            y_rN = {}
    for-each r in INF
      send *-message(*,x_r,t) to component r
    if INF = {}
      y_coupling={y_dN : d in I_N, y_dN!={}}
      send y-message(y_coupling,t) to parent
```

Receiving an `x-message` means that the λ iteration is finished and the state transitions can be conducted. This is done by sending an `x-message` to every imminent child component and to every child component with non-empty input bag. These components have been gathered in `DELTA` during the λ iteration. After the `x-messages` are sent, the coordinator waits until all of them are done with their state transition. Afterwards, the time of the next internal event can be calculated and the set `IMM` is cleared.

```
when receive x-message(x,t)
  for-each r in DELTA
    send x-message(x_r,t) to r
  wait until DELTA = {}
  sort event-list according to tn_d
  tl = t
  tn = min{tn_d: d in D}
  IMM = {}
  send done-message(done, tn) to parent
end RPDEVS-coordinator
```

2.3 Root Coordinator

Finally, on top of the uppermost coordinator is the root coordinator. It starts the simulation by sending an `i-message` to its child coordinator. Then it advances the simulation time to the time of next event, initiates the λ iteration by sending a `*-message`, waits until the λ iteration is finished and then triggers the state transition

by sending an `x-message`. This is repeated until the simulation time `t` exceeds the final time `tend`.

```
RPDEVS-root-coordinator
  variables:
    tstart // simulation start time
    tend   // simulation end time
    t      // current simulation time
    child  // direct subordinate coordinator

  t = tstart
  send i-message(i,t) to child
  wait for done-message(done,tn) from child
  t=tn
  while t < tend
    send *-message(*,t) to child
    wait for y-message(y,t) from child
    send x-message({},t) to child
    wait for done-message(done,tn) from child
    t=tn
end RPDEVS-root-coordinator
```

3 Conclusion

In this work, we first recapped RPDEVS and pointed out the parallels of PDEVS and RPDEVS to Moore and Mealy automata. Furthermore, we demonstrated how the input bags can be formally split up into sub-bags, one for each influencer. This separation is used by the abstract simulator as it makes it easier to detect input bag changes due to λ recalculations in the influencers. The structure of the abstract simulator is basically similar to the one of Zeigler's PDEVS abstract simulator [1]. For synchronization purposes though, we also added the `done-message` of Chow's algorithm [11].

When implementing the algorithm, especially when facilitating parallelism, aspects like consistent global variable manipulation and execution order have to be taken into account. However, this degree of detail would go beyond the scope of this paper.

Nevertheless, there already exists a proof-of-concept implementation of an RPDEVS simulator. We reprogrammed the simulation engine of the open-source classic DEVS simulator *PowerDEVS* and named it *PowerRPDEVS*. It is available on *SourceForge* [14]. In PowerRPDEVS, a sequential version of the algorithm is implemented. Exploitation of parallelism in the PowerRPDEVS engine is an issue for future work.

Acknowledgement

We thank all partners of the project ASPeCT for their contributions. The presented work is funded by the Austrian Research Promotion Agency within the program *Produktion der Zukunft*, project number 858655.

References

[1] Zeigler BP, Praehofer H, Kim TG. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. Academic Press. 2000.

[2] Joslyn C. The process theoretical approach to qualitative DEVS. *Proc 7th Conf on AI, Simulation, and Planning in High Autonomy Systems*. 1996;.

[3] Klimovich AS, Solov'ev VV. Transformation of a Mealy Finite–State Machine into a Moore Finite–State Machine by Splitting Internal States. *J Comput Syst Sci Int*. 2010;49(6):900–908.

[4] Chow ACH, Zeigler BP. Parallel DEVS: a parallel, hierarchical, modular, modeling formalism. In: *Proc. of WSC'94*. 1994; pp. 716–722.

[5] Preyser FJ, Heinzl B, Raich P, Kastner W. Towards Extending the Parallel-DEVS Formalism to Improve Component Modularity. In: *Beiträge zum WS der ASIM /GI-Fachgruppen STS und GMMS 2016*; pp. 83–89.

[6] Preyser FJ. An Approach to Develop a User Friendly Way of Implementing DEV&DESS Models in PowerDEVS. Masterthesis, TU Wien. 2015.

[7] Raich P, Heinzl B, Preyser F, Kastner W. Modeling Techniques for Integrated Simulation of Industrial Systems Based on Hybrid PDEVS. In: *Proc. of MSCPES'16*, 1. 2016; pp. 1–6.

[8] Preyser F, Heinzl B, Kastner W. RPDEVS: Revising the Parallel Discrete Event System Specification. In: *Proc. of MATHMOD 2018*; pp. 269–274.

[9] Goldstein R, Breslav S, Khan A. Informal DEVS conventions motivated by practical considerations. In: *Proc. of DEVS 2013*; pp. 10:1–10:6.

[10] Goldstein R, Breslav S, Khan A. Practical aspects of the DesignDEVS simulation environment. *Simulation*. 2017;94(4):301–326.

[11] Chow AC, Zeigler BP, Kim DH. Abstract Simulator for the Parallel DEVS Formalism. In: *Proc. of the Fifth Annual Conference on AI, and Planning in High Autonomy Systems*. 1994; pp. 157–163.

[12] Schwatinski T, Pawletta T. An advanced simulation approach for parallel DEVS with ports. In: *Proc. of SpringSim '10*. 2010; pp. 147–154.

[13] Fiedler C, Preyser F, Kastner W. Simulation of RPDEVS Models of Logic Gates. In: *Proc. of ASIM-Workshop Simulation technischer Systeme/Grundlagen und Methoden in Modellbildung und Simulation*. 2019; p. 6.

[14] PowerRPDEVS on sourceforge:. `https://sourceforge.net/projects/powerrpdevs/`. [accessed: 2019-01-04].

Simulation of RPDEVS Models of Logic Gates

Christian Fiedler[1], Franz J. Preyser[1*], Wolfgang Kastner [1]

[1]Institute of Computer Engineering, Automation Systems Group, TU Wien, Treitlstraße 1-3, 1040 Vienna, Austria
[*]*franz.preyser@tuwien.ac.at*

SNE 29(2), 2019, 85-91, DOI: 10.11128/sne.29.tn.10474
Received: April 10, 2019 (Selected ASIM GMMS/STS 2019 Conference Publication); Accepted; May 20, 2019
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna, ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. This paper addresses the simulation of fundamental logic gates (e.g. `AND`, `OR`, `NOT`) using the software *PowerRPDEVS* that is based on the *Revised Parallel Discrete Event System Specification* (RPDEVS) formalism. The formal differences of the models of a `NOR` gate in RPDEVS and PDEVS are analyzed. It is further shown, which possible pitfalls may occur when connecting these logic gates with feedbacks that cause algebraic loops and in which cases these algebraic loops are resolved by the RPDEVS simulation algorithm. For this purpose, a static RS flip-flop, a triggered D flip-flop and a shift register are modeled and simulated in *PowerRPDEVS*. The results are compared to previous research about the simulation of such logic circuits in *Simulink* and *Modelica*.

Introduction

In the theory of computation, the notion of Mealy and Moore automata exists which are forms of finite state automata [1]. The difference between these two types is how the output function is defined. The output function of the Moore type depends only on the internal state of the automaton, whereas for the Mealy type it also depends on the automaton's input. The formal definition of an automaton has a lot in common with the modeling formalism *Discrete Event System Specification* (DEVS) [2] and thus, also with its extension the *Parallel Discrete Event System Specification* (PDEVS), introduced by Chow and Zeigler [3]. However, since the output function in PDEVS depends only on the internal state, in principle, in PDEVS only Moore behavior is supported [4]. For Mealy behavior, a workaround including a *transitory state* (i.e. a state with zero life time) is necessary. This means that PDEVS models which immediately react to an external event with an output first have to enter a transitory state before the output function can be used to set the output.

In *Revised Parallel Discrete Event System Specification* (RPDEVS), introduced by Preyser et al. [5], the formalism was restructured to support Mealy behavior naturally. In RPDEVS immediate reactions to external events can be modeled directly with the output function, which removes the need for transitory states in this context.

As for PDEVS, an abstract simulator for RPDEVS was defined and published in the acompanying work [6]. An implementation is provided with the program *PowerRPDEVS* that also includes a graphical model editor.

With the goal to extend the PowerRPDEVS model library, we created a library with combinational logic and sequential logic elements. Combinational logic gates output the result of a boolean operator applied onto the input values. Thus, when signal delays are not taken into account, their models are all of type Mealy. Sequential logic components, in practice, are usually designed by coupling combinational logic elements with storage elements [7]. As Junglas' findings in [8] show, this can be a cumbersome business in simulation tools.

In this work, first it is analysed how the RPDEVS models of logic gates differ from the corresponding PDEVS models. Afterwards, it is investigated how the RPDEVS simulation algorithm performs when creating sequential logic components and the results are compared to Junglas' work.

1 The PDEVS and RPDEVS Formalisms

PDEVS and RPDEVS are hierarchical modelling formalisms where models are composed of *atomics* and *couplings*. An atomic can receive inputs from other atomics or couplings, it has an internal state and can produce outputs. Couplings can be composed of atomics and other couplings. The formal definition of a coupling is omitted here, it can be found in [3] and [6].

1.1 Atomic PDEVS

In Equation (1) the definition of a PDEVS atomic is given as tuple.

$$A := < X^b, Y^b, S, \delta_{ext}, \delta_{int}, \delta_{conf}, \lambda, ta > \quad (1)$$

X^b … set of possible input bags
Y^b … set of possible output bags
S … set of possible (internal) states of the atomic
$\delta_{ext} : Q \times X^b \to S$ … external state transition function
where $Q = \{(s,e) | s \in S, e \in [0, ta(s)]\}$
$\delta_{int} : S \to S$ … internal state transition function
$\delta_{conf} : S \times X^b \to S$ … confluent state transition function
$\lambda : S \to Y^b$ … output function
$ta : S \to \mathbb{R}_0^+ \cup \{\infty\}$ … time advance function

The time advance function ta determines the time to live $ta(s) \in [0, \infty]$ for every internal state $s \in S$. Whenever this time expires, an internal event is triggered, which first causes the execution of the output function λ and then leads to a state transition conducted by δ_{int}. However, if at the same time an input event occurs, the state transition is performed by δ_{conf}. If the atomic is not *imminent* (i.e. it has no internal event) while an input event x^b is received, δ_{ext} is called and the atomic changes into a new state $s' = \delta_{ext}(s, e, x^b)$ without producing any output. The λ function that sets the output of the atomic is only evaluated right before an internal state transition and relies on the old state of the model. Thus, when an atomic has to respond to an input with a change in output, there has to be a state transition in δ_{ext} (or δ_{conf}) into a transitory state (i.e. a state s' with $ta(s') = 0$). This way, λ can set a new output at the same point in simulation time.

1.2 Atomic RPDEVS

$$A := < X^b, Y^b, S, \delta, \lambda, ta > \quad (2)$$

$\lambda : (Q \times X^b) \to Y^b$ … output function
$\delta : (Q \times X^b) \to S$ … external state transition function
where $Q = \{(s,e) | s \in S, e \in [0, ta(s)]\}$

In RPDEVS (see atomic definition in Equation (2)), the three state transition functions of PDEVS are merged to one single transition function δ which always is preceded by an evaluation of the output function λ. This evalution of λ though, happens iteratively. That is, λ is recalculated every time the input bag has changed due to a λ computation in an influencing component. As shown in [5], this λ *iteration* terminates as long as the model does not contain algebraic loops. Furthermore, it still may terminate if the algebraic loop can be resolved as we will see in Section 1.4. In contrast to PDEVS, λ also depends on the current input bag. Thus, Mealy behavior can be modelled without having to change the internal state. In fact, pure functional blocks can be modeled which do not need an internal state at all, e.g. a logic `NOT` gate just forwards input messages inverted to the output.

As already mentioned in the introduction, a PDEVS model can be compared to a Moore machine ($\lambda(s)$), wheras an RPDEVS can be compared to a Mealy machine because its output is a function of the input and the internal state ($\lambda(s, e, x^b)$).

1.3 `NOR` gate

A `NOR` gate with two inputs is constructed in PDEVS and RPDEVS to demonstrate the differences with an example in the context of logic gates.

$$NOR_{PDEVS} := < X, Y, S, \delta_{ext}, \delta_{int}, \delta_{conf}, \lambda, ta > \quad (3)$$

$$\begin{aligned}
X &= \{1,2\} \times \{0,1\} \\
Y &= \{0,1\} \\
S &= \{0,1\}^2 \times \{0,\infty\}, \quad s = (s_1, s_2, \sigma) \in S \\
\delta_{ext}(s,e,x) &= (a_1(s,x), a_2(s,x), 0) \\
\delta_{int}(s) &= (s_1, s_2, \infty) \\
\delta_{conf}(s,x) &= \delta_{ext}(s, ta(s), x) \\
\lambda(s) &= \neg(s_1 \vee s_2) \\
ta(s) &= \sigma \\
a_i(s, x^b) &= \begin{cases} x_i : & \text{if } (i, x_i) \in x^b \\ s_i : & \text{otherwise} \end{cases}
\end{aligned}$$

Equation (3) shows a PDEVS `NOR` gate. As an external event could update either both or only one input, the model has to keep the last seen values of its inputs in the internal state. A helper function a_i chooses the new input value from the input bag x^b if there is any, or otherwise the saved value from the internal state. The model also keeps the value σ for ta, which is set to 0 in the case of an external event. In this way, every external event is followed by a transitory state used to create an output event.

$$NOR_{RPDEVS} := < X, Y, S, \delta, \lambda, ta > \quad (4)$$

$$X = \{1,2\} \times \{0,1\}$$
$$Y = \{0,1\}$$
$$S = \{0,1\}^2 \ \ldots \ s = (s_1, s_2) \in S$$
$$ta(s) = \infty$$
$$\lambda(s,e,x^b) = \neg(a_1(s,x^b) \vee a_2(s,x^b))$$
$$\delta(s,e,x^b) = (a_1(s,x^b), a_2(s,x^b))$$

When looking at Equation (4) which shows the NOR gate in RPDEVS, one can see that, because λ can access the input bag directly, the transitory state is not needed and therefore, the state space is reduced.

1.4 Static RS Flip-Flop

Flip-flops are sequential logic elements. That means, their outputs not only depend on their current inputs but can also on historic input values [7]. This implies that they have an internal state to store the historic data. A static RS flip-flop is commonly built using two NOR gates connected with their outputs fed back to the input of the other (see Figure 1).

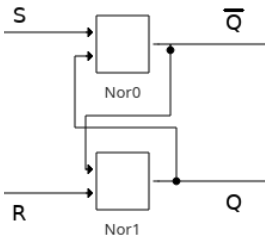


Figure 1: PowerRPDEVS model of the static RS flip-flop composed of two NOR gates.

Notably, the RS flip-flop is constructed from two *combinational* logic elements – the two NOR gates ideally have no state. Thus, deducing the mathematical model from the circuit of Figure 1 results in a system of two implicit equations:

$$Q = \neg(R \vee \overline{Q}) \quad (5)$$
$$\overline{Q} = \neg(S \vee Q) \quad (6)$$

Equations 5 and 6 can be solved as long as the inputs S and R are not both equal to 0 (see Table 1). The defined behavior for a RS flip-flop actually is to keep its previous output values in the case of $R = S = 0$. However, to know the previous value, the system needs to have a memory, i.e. an internal state. A real flip-flop is a continuous system and its signals are exposed to delays. These delays cause the system to still know its previous output, when the input changes.

S	R	Q	$\overline{Q}$
0	0	$\neg\overline{Q}$	$\neg Q$
0	1	0	1
1	0	1	0
1	1	0	0

Table 1: Solutions of Equations (5) and (6).

Due to the discrete event nature, our PDEVS and RPDEVS models have to store the last seen input values and, thus, also possess an internal state. When one of the inputs S or R of the RS flip-flop changes, the affected NOR gate still has stored the previous output of the other NOR gate. Consequently, during the simulation of the PDEVS and RPDEVS models, not Equations (5) and (6) are solved, but a recurrence relation. How this recurrence relation looks like depends on the number of inputs that change concurrently and on whether the simulation algorithm works in parallel or sequentially.

Single input change. We now consider the cases in which only one input changes its value.

If input S changes, the upper NOR gate first calculates its output using the new value of S and the stored value for Q. Then, due to the change in $\overline{Q}$, the lower NOR gate calculates its output, already using the new value for $\overline{Q}$. Thus, the recurrence relation has the form:

$$Q_n = \neg(R \vee \overline{Q}_n) \quad (7)$$
$$\overline{Q}_n = \neg(S \vee Q_{n-1}) \quad (8)$$

In Table 3 the evolution of the recurrence relation in Equations (7) and (8) is depicted for all possible initial states Q_{n-1} and input values S and R. It can be seen, that in all cases a fix point is reached after at least 2 iterations ($Q_{n+1} = Q_n$). Nevertheless, the simulation of this model in PDEVS leads to an infinite loop. When processing the external event due to the change in one of the two inputs, the affected gate schedules an internal event with $ta = 0$. Then λ sets the output and triggers the other gate for an external event. Finally, δ_{int} sets $ta = \infty$. The other gate is activated though, and will do exactly the same afterwards. This again reactivates the first gate and, thus, the simula-

Q_{n-1}	S	R	Q_n	$\overline{Q_n}$	Q_{n+1}	$\overline{Q_{n+1}}$
0	0	0	0	1	0	1
0	0	1	0	1	0	1
0	1	0	1	0	1	0
0	1	1	0	0	0	0
1	0	0	1	0	1	0
1	0	1	0	0	0	1
1	1	0	1	0	1	0
1	1	1	0	0	0	0

Table 2: Solutions of Equations (7) and (8).

tion gets stuck in a loop (the model is *illegitimate*). To mitigate this issue, the PDEVS model must be extended, such that δ_{ext} of the NOR gates only enters a transitory state when the input bit x_i is different from the already stored bit s_i. It should be noted here, that this is an example for how reusability of PDEVS models is impaired due to the need of transitory states for modeling Mealy behavior.

In RPDEVS, the simulation terminates. The change in the input S directly leads to an execution of λ of the upper NOR gate. The produced output respectively input for the lower gate then triggers λ of the lower NOR. The output produced thereby triggers a recalculation of λ at the upper gate. However, if the newly produced output of the upper gate does not differ from its previous one, the lower gate is not triggered again. Consequently, as long as the recurrence relation reaches a fix point in a finite number of steps, the RPDEVS simulation algorithm will find that fix point and will be able to continue simulation.

The case in which the input R changes and S does not change, is completely analog and, thus, is not described.

Concurrent input change. If both inputs S and R change concurrently, it depends on the simulation algorithm, how the recurrence relation that is solved during simulation looks like. In the case of a sequential RPDEVS simulation algorithm, like implemented in PowerRPDEVS [9], first λ of the first block is calculated. Then λ of the second block is calculated, already using the new output of the first block. Thus, the recurrence relation to solve again is the one of first order discussed in the previous paragraph. Consequently, PowerRPDEVS can handle any concurrent change of both inputs S and R without getting stuck in an endless loop.

However, if the RPDEVS simulation engine works in parallel, that is, it calculates λ of both gates concurrently, the recurrence relation to be solved would be of second order (see Equations (9) and (10)).

$$Q_n = \neg(R \vee \overline{Q}_{n-1}) \tag{9}$$

$$\overline{Q}_n = \neg(S \vee Q_{n-1}) \tag{10}$$

This recurrence relation can become unstable though. When both inputs are 1 and then concurrently change to 0, the outputs start to alternate between 0 and 1.

Q_{n-1}	$\overline{Q_{n-1}}$	S	R	Q_n	$\overline{Q_n}$	Q_{n+1}	$\overline{Q_{n+1}}$
X	X	1	1	0	0	0	0
0	0	0	0	1	1	0	0

Table 3: The recurrence relation in Equations (7) and (8) becomes unstable if S and R change concurrently from 1 to 0.

In PDEVS, a concurrent change of both inputs first leads to an execution of δ_{ext} at both gates, storing the new input in the internal state and setting $\sigma = 0$ to enter a transitory state. The transitory state leads immediately to internal events and thus, to a execution of λ in both gates. It does not matter whether λ of the gates is executed in parallel, or consecutively, because both use the old output of the other one that is stored in the component's state. Therefore, for the PDEVS simulation algorithm the concurrent change of both inputs S and R always leads to the solution of the second order recurrence relation in Equations (7) and (8) regardless whether execution is parallel or sequential.

2 Simulation

Simulation was done in *PowerRPDEVS* which is the proof-of-concept implementation of an RPDEVS modelling environment that includes the simulation engine and a graphical model editor. It is open source and available in [9].

2.1 Static RS Flip-Flop

The model of the RS flip-flop in Figure 1 was simulated with the initial values $Q = 0$ and $\overline{Q} = 1$. The input sequence and results are shown in Figure 2. Contrary to

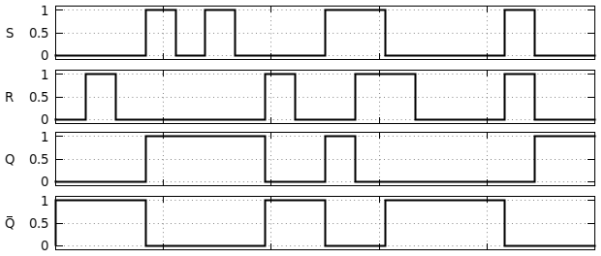


Figure 2: static RS flip-flop – Simulation results

simulation in Simulink [8], no work-arounds are needed and the outputs are not delayed.

A transition to the "forbidden" state $S = R = 1$ and back to $S = R = 0$ is included. As long as $S = R = 1$, the behaviour is actually well-defined as $Q = \overline{Q} = 0$. When S and R change to 0 simultaneously, the behaviour depends on the ordering of the λ function executions. As mentioned in Section 1.4, parallel execution of the `NOR` gates' λ functions would cause an infinite loop (oscillation) in the simulation, but it works in *PowerRPDEVS* because the execution is serialized.

2.2 D Flip-Flop

A (clock triggered) D flip-flop can be constructed from a triggered RS flip-flop and additional wiring at its inputs. The atomic `LogicTriggeredSampling` (`LTS`) was implemented for this example. It can detect edges on its second (lower) input, either triggering for rising edges, falling edges or both, and it either forwards the left ($y(t) = \lim_{\tau \nearrow t} x(\tau)$) or the right limit ($y(t) = \lim_{\tau \searrow t} x(\tau)$) of its first (upper) input when triggered by an edge. This block was placed before the inputs of a static RS flip-flop (see Figure 3). When the `LTS` blocks are set to take the right limit a change in the input that occurs at the same time as the clock edge is accepted by the flip-flop and it is not accepted otherwise.

It was first tried to use a different trigger detection mechanism: a *falling* block as in [8] in the Modelica model of the triggered RS flip-flop. This did not work out well in RPDEVS though, as the block has to send a `1` for an infinitesimal time frame and then switch back to `0` whenever it detects an edge. This means that because of the event-based nature of RPDEVS, the λ output at the time of the edge would be `1` and the block would need to schedule an internal event to set the output `0`. If ta is set to 0 for this purpose a transitory state would be introduced which we are trying to avoid. On the other hand, if it was set to $ta = x \in \mathbb{R}^+$ this would open a time frame during which the flip-flop would accept changes in its inputs, although the clock edge occured in the past. Thus, in the end the triggered RS flip-flop was modeled as depicted in Figure 3.

Figure 4 shows the D flip-flop consisting of the triggered RS flip-flop and a `NOT`. The results of the simulation of the D flip-flop are shown in Figure 5. $\overline{Q}$ is omitted as it always carries exactly the opposite logic level of Q. The triggering clock edge is set to be the falling edge.

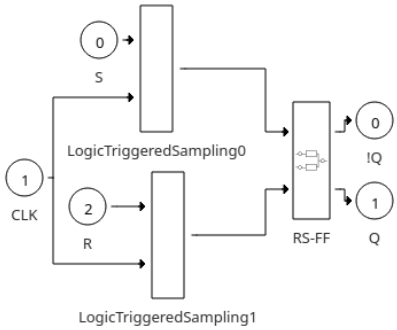


Figure 3: triggered RS flip-flop model

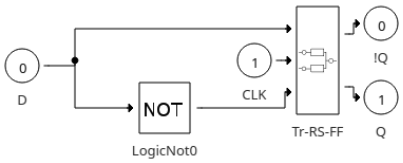


Figure 4: D flip-flop model

During the design of the `LTS` block we recognized that it is actually a D flip-flop in its own right. The block accepts its first input (corresponding to D) as its output only when there is an edge on its second input (corresponding to CLK) which is exactly the behaviour of a D flip-flop.

Replacing the D flip-flop with an `LTS` block yields exactly the same results as in Figure 5.

2.3 Shift register

A shift register is a series of D flip-flops where the input signal is shifted through one flip-flop at a time whenever the clock input triggers.

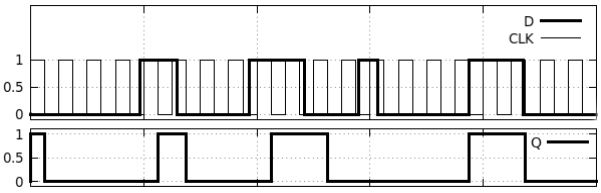


Figure 5: D flip-flop – Simulation results

The shift register in Figure 6 was modelled by using three of the D flip-flops designed above. Note that for the first flip-flop the `LTS` blocks (that are part of the triggered RS flip-flop, see Figure 3) is set to use the right limit and for the other flip-flops it is set to use the left limit of the input.

The reason is that the input signal would otherwise travel through all the flip-flops when the first clock edge arrives, because all D flip-flops are triggered by the same clock and none of them delays the signal.

The results of the simulation are shown in Figure 7. They show the individual D flip-flops' output and how the input signal (first a `1`, then a `0`) propagates through the shift register one stage per clock cycle.

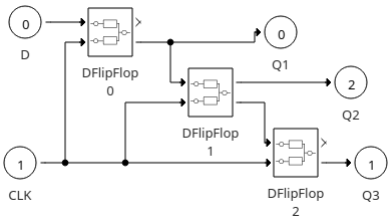


Figure 6: Shift register model

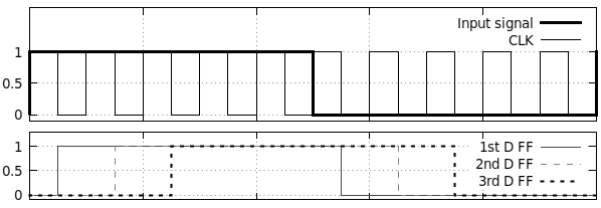


Figure 7: Shift register – Simulation results

In Junglas' tests [8], the Simulink model worked correctly without intervention, but the Modelica model seemed to show a peculiar issue that he mitigated by placing *Pre* blocks between the flip-flops which introduces an infinitesimal delay to break algebraic loops.

3 Conclusion

The *Revised Parallel DEVS* formalism offers new ways to deal with immediate outputs (Mealy behaviour of models) and algebraic loops. Specifically, we discussed a purely functional `NOR` gate in detail, showing that a model of it in RPDEVS can be realized with a smaller state space than in PDEVS, thus, reducing the complexity of the model. The static RS flip-flop was presented to show the behaviour of RPDEVS models with a feedback loop with no delay. The result was that a primitive coupling of `NOR` gates to form an RS latch would almost in any case lead to the expected behaviour of a physical `NOR` gate, but a transition to the "forbidden" state can lead to oscillation if the simulation engine utilizes parallelism.

The simulation of the RS flip-flop with PowerRPDEVS shows the behaviour that is expected from an RS flip-flop, without the need to introduce delay blocks or arrange it in a special way, which is necessary in other simulators.

The triggered D flip-flop model required the creation of the `LTS` block that was capable of forwarding an input event exactly when a clock edge occured which turned out to be a D flip-flop on its own. When they were put together to form a shift register, we needed to take into account the delays that actually make a shift register work and introduce them in our model as infinitesimal delays in the `LTS` block.

Acknowledgement

I want to thank my colleagues Franz Preyser and Wolfgang Kastner for the opportunity to publish and present this paper at the ASIM Workshop in Braunschweig this year. I am also very grateful for its acceptance in this journal propelled by Felix Breitenecker.

References

[1] Sakarovitch J, Thomas R. *Elements of Automata Theory*. 2011.

[2] Zeigler BP, Praehofer H, Kim TG. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. Academic Press. 2000.

[3] Chow ACH, Zeigler BP. Parallel DEVS: a parallel, hierarchical, modular, modeling formalism. In: *Proc. of WSC'94*. 1994; pp. 716–722.

[4] Joslyn C. The process theoretical approach to qualitative DEVS. 1996; .

[5] Preyser FJ, Heinzl B, Kastner W. RPDEVS: Revising the Parallel Discrete Event System Specification. In: *Proc. of MATHMOD 2018*; pp. 269–274.

[6] Preyser FJ, Heinzl B, Kastner W, Breitenecker F. RPDEVS Abstract Simulator. In: *Proc. of ASIM-Workshop Simulation technischer Systeme/Grundlagen und Methoden in Modellbildung und Simulation*. 2019; p. 6.

[7] Cavanagh J. *Sequential Logic*. CRC Press. 2006.

[8] Junglas P. Pitfalls using discrete event blocks in Simulink and Modelica. In: *Tagungsband des Workshops der ASIM/GI-Fachgruppen STS und GMMS 2016*. 2016; pp. 90–97.

[9] PowerRPDEVS on SourceForge. `https://sourceforge.net/projects/powerrpdevs/`. [Online; accessed 27-Dec-2018].

Approach for Synthesis and Optimization of Complex Thermal Systems for Supermarkets

Jonathan Kistner[1*], Wilhelm Tegethoff[1], Nicolas Fidorra[2], Jürgen Köhler[2]

[1]TLK-Thermo GmbH, Hans-Sommer-Straße 5, 38106 Braunschweig, Germany; *j.kistner@tlk-thermo.de*

[2]Institut für Thermodynamik, Technische Universität Braunschweig, Hans-Sommer-Straße 5, 38106 Braunschweig, Germany

SNE 29(2), 2019, 93 - 100, DOI: 10.11128/sne.29.tn.10476
Received: April 15, 2019 (Selected ASIM GMMS/STS 2019 Postconf. Publ.), Accepted: June 10, 2019
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna, ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. The development of thermal systems for supermarkets is a challenging task. Both, heating and cooling demands at different temperature levels have to be satisfied under individual boundary conditions. In combination with a broad range of available technologies and components, a high number of possible system layouts exist. Thus, various types of refrigeration systems can be found in supermarkets: Central refrigeration systems with one or two stages and direct evaporation, central systems with a secondary loop or systems with (semi)-plug-in-cabinets. The system topology and operating strategy depend on climate conditions, building scale, customer's occupancy or evaluation criteria. In practice, established solutions based on experience are used. However, comparing all alternative concepts is difficult. Beside the consideration of investment costs, it is essential to evaluate the energy consumption. For the calculation of energy consumption, considering dynamic interactions between components is crucial.

To compare different system layouts under consideration of dynamic interactions, an optimal operating control has to be applied. Furthermore, the high number of possible topologies makes it necessary to reduce the complexity for the selection of components and their interconnections. Therefore, software based methods are needed to efficiently reduce complexity and evaluate system alternatives in a dynamic environment.

This paper presents a procedure that supports the user to find an optimal system topology under individual conditions. As an example, a secondary-loop refrigeration system with low and medium temperature cabinets is applied.

The user defines ambient conditions and requirements such as cooling load and temperature setpoints. Additionally, a set of transient, non-linear models for available technical equipment is defined. The parametrized, ready-to-use models are managed in a catalogue platform. In the catalogue, additional information is stored, like valid operational ranges, which is used during optimization. On this information basis, an algorithm deduces a reasonable refrigeration system layout. Intermediate result is a ready-to-simulate system. It contains only catalogue models that have physical reasonable interconnections. Subsequently, the system's fluid flow rate of each connection is optimized. The result of the optimization is used for evaluation of the system layout and further reduction of its topology.

The paper shows, that using simple input information, the complexity of the optimization problem can be extremely reduced. The suggested procedure is capable to deploy an optimal system topology under consideration of non-linear dependencies.

Introduction

Many engineering work is spent on developing better thermal energy systems. The effort that is being made touches all sectors of industry and science like cars with electrified drivetrains [14], busses [3][9] and supermarkets [1]. Especially for supermarkets, the energy saving potential is enormous. A broad range of different technologies is available. At the same time, many different thermal demands typically occur. In practice, established solutions based on experience are used. However, comparing all alternative concepts is difficult. The high number of possible topologies makes it necessary to reduce the complexity for the selection of components and their interconnections. Software based methods are needed to efficiently reduce complexity and evaluate system alternatives considering all relevant non-linearities.

This paper presents an approach to support the user finding an optimal system topology under individual conditions. As an example, a secondary-loop refrigeration system with low temperature (LT) and medium temperature (MT) cabinets is applied. The system is synthesized on basis of ready parameterized catalogue models. A steady state parameter optimization of the synthesized system model is performed. Further system reductions are derived from the optimization result. The overall electrical energy consumption of system variants are statistically estimated over one year for different cities in Europe.

1 Supermarket Refrigeration Systems

Supermarket energy systems have big energy saving potentials. Energy savings related to the application of optimal topologies in combination with optimal operating strategies can be tentatively estimated to be in a magnitude of 20 % [2][13].

Furthermore, energy systems of supermarkets typically possess a high complexity caused by many different requirements: The salesroom has to be cooled, heated and dehumidified. Groceries have to be kept at different low temperature levels. The optimal system topology changes with climatic boundary condition, building scale, customers' frequency and evaluation criteria. Improvements of energy efficiency and reduction of emissions are tried to achieve by applying waste heat recovery, regenerative technologies or thermal storages [12][8].

Beside detailed system variants, some general system types for supermarket refrigeration exist: Widely used and under intensive research are central refrigeration systems. Those systems can be huge refrigeration cycles with several evaporators at different temperature levels satisfying both cooling and freezing demands. The cabinets directly contain the evaporators. The main disadvantage is the high complexity of controls and a high charge of refrigerant. In water-loop refrigeration systems every cabinet contains its own little refrigeration cycle. The water loop transfers the waste heat of the condensers to the ambient. For low temperature cabinets, often a secondary brine loop with an additional refrigeration cycle is used to cool the cabinet's condenser. In secondary-loop systems, the cabinets are directly cooled by brine. Different refrigeration cycles provide brine at needed temperature setpoints. Water-loop and secondary-loop systems are more and more focused in current researches. Amongst others, the reasons are small amounts of refrigerant and relatively easy to control and combine in different layouts [1]. A comparison, especially with central refrigeration systems is of big interest in the current scientific and economic discussion [4].

The mentioned aspects encouraged the author to use a secondary-loop system as example for system synthesis and optimization. In fact, this is already a design specification. The example is chosen to be very simple to ease the evaluation of the represented procedure. This paper is a preliminary study for further methodical work to support scientists and planners to develop new system topologies for supermarkets.

2 Procedure for Layout Development

In this chapter, the applied procedure for layout development is represented. It contains three main topics: Firstly, set up of the information basis. Secondly, execution of a system synthesis based on the defined information. Thirdly, optimization of the synthesized system model.

Figure 1 shows the process of the layout development. In the first step, the user starts defining requirements and boundary conditions. At this point, the refrigeration loads and related temperature setpoints are defined. The boundary conditions can be connected to a specific location that is relevant for the system development. Generic boundary conditions are used in this paper. Furthermore, a set of components has to be chosen, that shall be available for the system synthesis. Most information related to the catalogued models can be assumed to be given as meta data. However, the user might have to change some data like temperature setpoints for refrigeration cycles or estimated valid inlet temperatures for cabinets. Once the system is synthesized, the user need to find valid starting conditions for optimization.

In the present study, three main questions were of particular interest:

- How could a synthesis look like on basis of component functionalities without executing any simulations?
- How does a well-suited information basis look like to efficiently support the system synthesis and reduce complexity?
- How can a system layout efficiently be evaluated with less user interactions as possible?

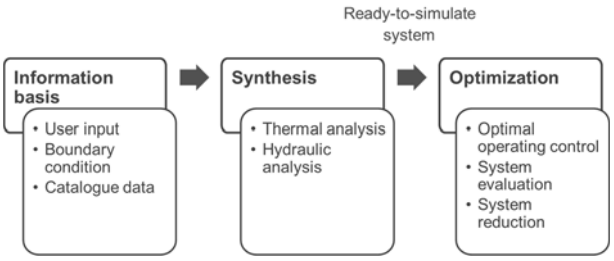


Figure 1: Employed procedure for system synthesis and optimization. The synthesis uses a well-suited information basis to find physical reasonable interconnections. The synthesized ready-to-simulate system model is optimized.

The used component models in this paper can be seen as type representatives. The procedure is not aimed to work for design tasks, which are typically characterized to have a big number of very similar components. Other procedures for system design exist, that could be applied consecutively or be integrated in the represented procedure in future work.

3 Information Basis for System Synthesis

As described in Chapter 2, the information basis contains requirements and specifications of the development task as well as a defined set of component models with related meta data. In this section, all basic information of the applied development task is specified and the available component models are described in detail.

3.1 Catalogue models

All available system components are modeled on basis of TIL [11]. Thus, the modelling language is MODELICA. TILMedia is used for media property calculation.

Figure 2 shows the component models. In the chosen scenario, the models for the cabinets are assumed to be part of the requirement defined by the user. The refrigeration units are inheritors from the same refrigeration cycle with modified parameters. The refrigeration cycle contains a physical based compressor model, two finite volume heat exchangers, an orifice valve and an ideal separator after the condenser. The whole cycle is scalable by one nominal cooling capacity. Additionally, it is capable to automatically switch on and off, depending on the boundary conditions at the condenser and evaporator. The compressor is controlled to maintain a fixed temperature setpoint at the brine outlet of the evaporator. The valve controls the superheat of the refrigerant at the evaporator outlet. The two refrigeration units mainly differ in their type of refrigerant, their scale and their temperature setpoint.

The outdoor units basically are modeled as simple temperature boundaries. The second outdoor unit additionally is capable of regulating the outlet temperature by mixing the ambient temperature with the inlet flow. The cabinets are modeled as heat boundary. An early design decision is made by using cabinet models only for secondary-loop systems.

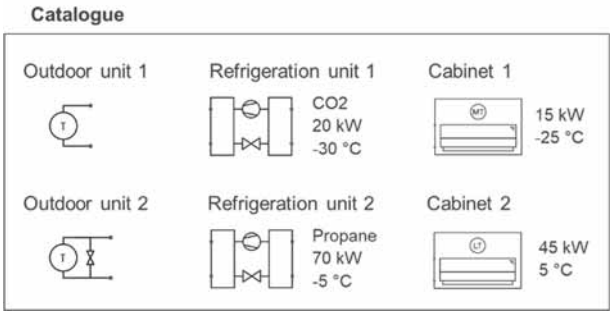


Figure 2: Catalogue models available for the system synthesis. The models for the cabinets are assumed to be part of the requirement. All other components are optional.

The catalogue models contain meta data that describe the functionalities and valid temperature ranges at all in- and outlets. Detailed examples for the meta data used for system synthesis can be seen in Chapter 4.

3.2 Requirements and specifications

The requirements and specifications are typically user inputs that characterize the overall optimization problem. As requirement, a refrigeration load with 45 kW at 5 °C and a load with 15 kW at -25 °C is specified. Furthermore, the user need to adjust the valid temperature ranges at the cabinets' inlet to complete the information basis for system synthesis. The temperature of the ambient air in a range of -10 °C to 40 °C is defined as ambient condition. The supermarket building is not considered in this study.

4 Synthesis of a Secondary-loop Refrigeration System

The system synthesis represented in this paper uses the described information basis in Chapter 3 to deduce a physical reasonable system layout. In the first place, a thermal functional analysis for each inlet and outlet of the available components is executed.

Catalogued temperature information and thermal functionalities of each component are used to estimate reasonable interconnections. Secondly, a hydraulic analysis integrates additional components to complete the hydraulic network and make the model executable.

4.1 Thermal analysis to find reasonable interconnections

Figure 3 shows a simplified view on the meta data that is used for thermal system synthesis. Original catalogue data, defined setpoints and deduced requirements are combined as information basis for the system synthesis.

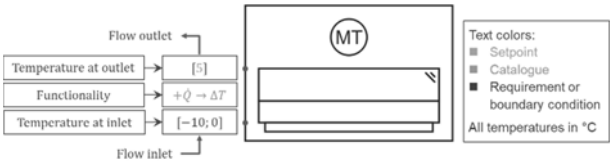


Figure 3: Simplified view of combined meta data as information basis for system synthesis.

The algorithm starts at one port and runs through all other ports of the available components in order to create valid hydraulic loops. When all options are proofed for this starting point, the algorithm restarts from another port until all possible loops are found. Note, that every component of the catalogue can be used only once in the system.

A set of rules reduce complexity and defines valid interconnections. Most important rules are described as follows:

- No interconnections between ports of the same component are allowed
- Only closed loops are allowed: The search of valid ports has to end at the same component (e.g. from evaporator outlet to inlet)
- Every loop must contain complementary functionalities (e.g. at least one positive and one negative heat flow)
- Calculated inlet temperatures have to comply to the valid range of the component port
- A port is only once connected in a loop
- No waste of generated refrigeration (e.g. evaporator to warmer outdoor unit)
- A component is allowed to have interconnections to only one of the outdoor units (best match of temperature range is chosen)

Figure 4 shows the synthesis of one loop (see button 2) with the starting point at the outlet of the MT cabinet.

This loop is valid for ambient temperatures between 0 °C and 20 °C. The direct return flow from the condenser of the LT refrigeration cycle to the MT cabinet is not a valid loop (see button 1 and dashed line). In this case, the temperature range is violated and the functionalities are not complementary. In the end, the whole system layout is built by overlaying all found valid loops. Every interconnection exists only once in the system.

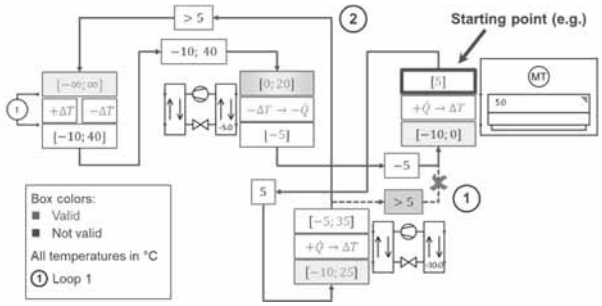


Figure 4: Example for system synthesis algorithm. Starting at the outlet of the MT cabinet, a valid flow connection to the condenser of the LT refrigeration cycle is found. From there other valid connections are found until the loop is closed. Estimated temperatures of the stream are displayed along the blue lines.

4.2 Integration of hydraulic components

The hydraulic concept is to use pumps for every possible interconnection and control their indicated mass flow rate with the optimizer. Thus, no valves, no controllers and no junctions are needed in the hydraulic network. From port to port, pumps are added until every mass flow rate in the hydraulic system is determined. The complete synthesized system layout is illustrated in Appendix A.

5 Optimization of the Synthesized System

During the applied system synthesis, no simulation is used. Therefore, the synthesized system model still has to be evaluated and maybe even further improved by results of simulation. To be able to evaluate the system layout correctly, it must operate in an optimal way while facing the defined boundary conditions. Optimal operating is defined to have minimum power consumption while observing the requirements. For finding valid starting conditions a parameter study for the control inputs of the pumps was applied. Integrations were executed in Dymola with DASSL as solver.

5.1 Optimization problem and algorithm

The applied optimizer is based on the Globalized Bounded Nelder-Mead algorithm (GBNM) [6]. It is capable of constraint handling via penalty function and probabilistic restarts for a globalized optimization. Using a global optimization is crucial for the represented optimization task. Finding good starting conditions can be very time extensive and many local minima are expected for this type of systems.

Another big advantage is, even of the basic Nelder-Mead-Algorithm [7], that it allows a non-gradient evaluation of the model. The used models, especially for the refrigeration cylces, show high non-linear behaviour. Thus, sensible gradient evaluation might restrain the optimization progress or even fail at some operating points. Two other aspects appeared to be important as well: Firstly, with gradient evaluation, a precise normalization of the cost function has a strong impact on the optimization progress. A good normalization equals a good estimation of the cost function value at optimum. This is problematic in the context of topology optimization where a focus on widely automated procedures is crucial and starting conditions can be far away from the optimum. Secondly, unphysical parameter value combinations cannot be avoided. They lead to simulation failures, which impede cost function evaluation. Instead, an artificial penalty cost function value for failed simulations has to be used which can corrupt the gradient sensitivity. Comparison studies with the SLSQP from SciPy [10] based on [5] confirmed that issues.

Since the system model does not contain energy storage components, it is assumed that there is no time constant that is relevant for the power consumption. Thus, a dynamic optimal control problem can be avoided. Instead, a parameter optimization at steady state is applied with a batch of discrete ambient temperatures.

Cost function and constraints. The main object of the optimization is to minimize the sum of all electrical power consumption in the system. Additionally, deviations to setpoint temperatures ($T_{set} - T$) are added to the cost function via penalty function. Furthermore, mass flow rates that are not directly driven by a specific pump shall be only positive for better understanding. Those dependent mass flow rates ($\dot{m}_{dep}$) are considered as constraints, too. Figure 5 shows how the system model evaluation during optimization is done.

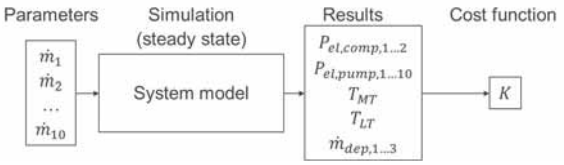


Figure 5: System evaluation during optimization. Optimization parameters are the mass flow rate inputs ($\vec{\dot{m}}$) for all pumps in the hydraulic system. Calculated variables for electrical power consumption (P_{el}), temperatures (T) and dependent mass flow rates ($\dot{m}_{dep}$) are part of the cost function.

Weighting and normalization are part of the penalty function. The optimization problem and the cost function is defined as follows:

Minimize $K(\vec{\dot{m}})$, where

$$K(\vec{\dot{m}}) = \sum_{n=1}^{10} P_{el,pump,n} + \sum_{n=1}^{2} P_{el,comp,n} + f_c\left(T_{MT,set} - T_{MT}\right) + f_c\left(T_{LT,set} - T_{LT}\right) + \sum_{n=1}^{3} f_c(\dot{m}_{dep,n}) \quad (1)$$

With

$\vec{\dot{m}}$ – Vector with $\dot{m}_1, \dots, \dot{m}_{10}$ $\left[{}^{kg}/_{s}\right]$
P_{el} – Electrical Power $[W]$
f_c – Normalized constraint penalty function $[W]$
T – Temperature $[K]$
$\dot{m}_{dep}$ – Dependent mass flow rate $\left[{}^{kg}/_{s}\right]$

5.2 Optimization results

All parallel global optimizations were limited to a maximum duration of about 40 hours, while 2000 iterations for one local search was set as maximum. During one global optimization (at constant ambient temperature) up to 30 local searches took place. In all global runs, several local searches ended up in the same optimum. Furthermore, no better set of parameters could be found by parameter studies. Thus, the optimization results seem to provide valid global minima. Integrations were numerically robust and fast: Failures only occurred due to unphysical parameter values. Speed of integration for one evaluation was about 1400 times real time.

Note that the presented results in this paper are manually simplified for better understanding: Very small mass flow rates are set to zero and some results are adjusted to show the corresponding operating mode clearer.

Figure 6 shows that under the chosen algorithm options, the optimizer is capable to change operating mode even under one local search. Under less precise trunca-

tion criteria, this local search would have stopped at about 800 evaluations and a new local run would have been initiated. With the used truncation criteria, the optimizer was capable to find a new and better minimum within the same local run. Comparing the parameter values at 800 with those at the end of the run, the operating mode changed: In the beginning, the MT cabinet and the condenser of the LT refrigeration cycle is cooled in parallel from the evaporator of the MT refrigeration cycle. At the end of the run, the working fluid that flows to the LT refrigeration cycle is completely passed on to the MT cabinet. Only less than 5 kW of the 45 kW cooling load at the MT cabinet is served directly from the MT refrigeration cycle. In Figure 7, the system simulation with optimized operating control can be seen.

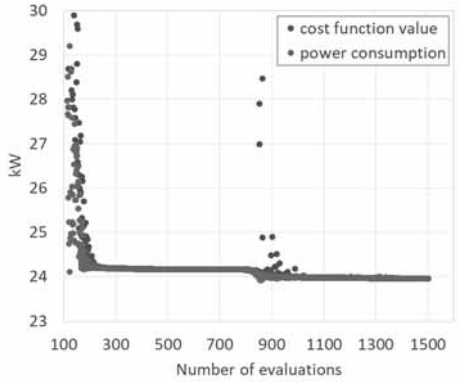


Figure 6: Detail of one local search at 20 °C ambient temperature. After elimination of the constraint penalties, the cost function value equals the power consumption. After 800 evaluations, a set of parameters is found that represents a better operating strategy.

This operating mode is found as global optimum for all ambient temperatures above 5 °C, but of course with slightly different mass flow rates.

Below 0 °C ambient temperature, the optimal operating mode is characterized by cooling the condenser of the LT refrigeration cycle and the MT cabinet by the outdoor unit. About 55 % of the 45 kW cooling load at the MT cabinet is served by mass flow rate passed on from the condenser of the LT refrigeration cycle. Figure 8 shows the optimal mass flow rates at -5 °C ambient temperature. At lower ambient temperatures (about -7 °C), even 100% feedthrough is preferred: All cooling load at the MT cabinet is driven by the mass flow rate through the LT refrigeration cycle's condenser and no direct cooling from the outdoor unit to the MT cabinet is used.

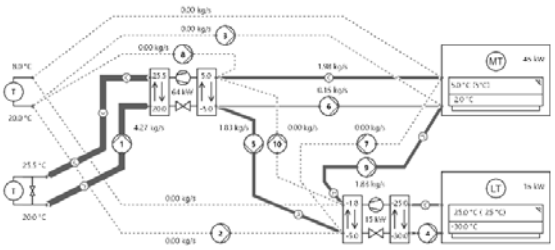


Figure 7: Optimized mass flow rates at 20 °C ambient temperature. The mass flow rate that flows from the evaporator of the MT refrigeration cycle to the condenser of the LT refrigeration cycle is completely passed on to the MT cabinet without direct return.

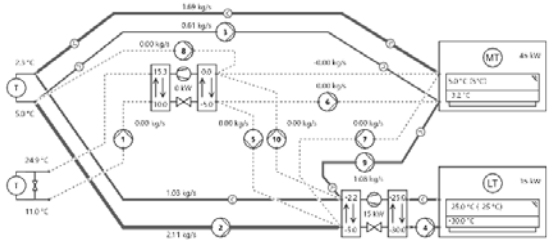


Figure 8: Optimized mass flow rates at -5 °C ambient temperature. MT refrigeration cycle is switched off. The LT refrigeration cycle and the MT cabinet are cooled via outdoor unit. A big amount of working fluid at the condenser outlet is passed on to the MT cabinet.

At ambient temperatures around 5 °C the outdoor unit covers the needed cooling at the condenser of the LT refrigeration cycle with about 60 %. The rest of about 9 kW is served by mass flow rate from the MT cabinet outlet. This additional mass flow rate through the condenser is passed on to the outdoor unit with a few degrees above ambient temperature, cooled down and from there passed on to the inlet of the MT refrigeration cycle. From there it flows again to the MT cabinet. There is a small range of ambient temperature in which this operating mode makes sense. Additionally, the benefit for energy consumption is very small compared to the second best result (see Figure 9 in Chapter 6).

6 Derivation of System Layout Reduction

In this section the capability of the presented procedure to allow topology improvements is proofed. A system topology optimization typically includes a variation of number and type of components as well as their interconnections. The main purpose of this paper is not to optimize the system topology.

However, it is a preliminary study to approach topology optimization in the future. The main idea is to eliminate components and interconnections by analysing the optimized mass flow rates.

Interconnections that show zero mass flow rates at all boundary conditions could be easily eliminated. Since the optimization contains the power consumption of the pumps, in some cases the results of the present study show a splitting of mass flow rate over several interconnections in order to avoid a very high mass flow rate at only one of the interconnections. For instance, providing cooling for the MT cabinet at low ambient temperatures is not only applied by direct cooling from the outdoor unit. Instead, additional mass flows from the outdoor unit to the MT cabinet through the evaporator of the switched off MT refrigeration cycle. In these cases, it is difficult to evaluate the necessity of particular interconnections. Therefore, the known loops from the system synthesis are used to detect corresponding operating modes and hence allow the derivation of second best operating strategies that are referenced to the optimal solution.

Three variations of the synthesized system layout are analysed. Each system layout is quantified by the sum of the power consumptions weighted according to the incidence of relevant ambient temperatures over one year for Berlin, Munich, Barcelona, Madrid and Oslo. Since no component of the simplified secondary-loop refrigeration system could be eliminated, an economical evaluation is not required.

In the first place, the optimization results show never a direct return from LT refrigeration cycle (condenser) to MT refrigeration cycle (evaporator). Instead, all mass is returned via MT cabinet. Thus, this interconnection could be simply eliminated (see Figure 9, pump 10). Secondly, looping mass from the MT cabinet back to MT refrigeration cycle (evaporator) via the condenser of the LT refrigeration cycle and outdoor unit is only used around 5 °C ambient temperature. Second-best option is a higher mass flow rate from the outdoor unit to the condenser of the LT refrigeration cycle. Figure 9 shows the original optimization results at 5 °C. The elimination of the red interconnections only leads to a higher electrical energy consumption over year of about 0.1 % for all considered cities.

The third option that has been analysed is the elimination of all two connections between the LT refrigeration cycle and the MT cabinet. This affects the operating mode for ambient temperatures below 0 °C and above 10 °C. In Appendix B, the corresponding layout reduction can be seen. Compared to the optimization results, the estimated energy consumption over year is slightly higher: Between 1.3 % for Oslo and 1.7 % for Barcelona.

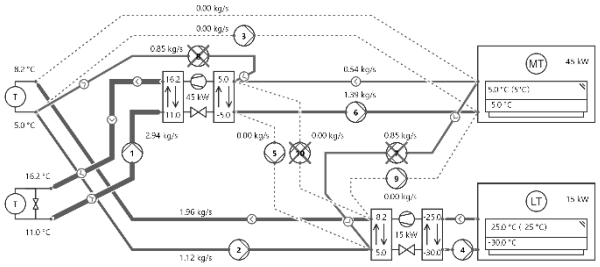


Figure 9: Optimized mass flow rates at 5 °C ambient temperature. The LT refrigeration cycle could be cooled only via outdoor unit. Thus, the red interconnections could be eliminated with almost the same electrical energy consumption. The path with pump 10 could be eliminated anyway. The reduced system layout is valid for all considered ambient temperatures.

7 Conclusion

The represented approach for system synthesis and optimization reduced the number of interconnections from potentially 64 to 10. The synthesized secondary-loop refrigeration system contains all physical reasonable interconnections. The system model covers all relevant non-linearities. At the same time, integration was fast and numerically robust. The applied GBNM algorithm appeared to be robust in finding global minima within an acceptable duration of 1 day.

The optimization results allowed the derivation of further system layout reductions. The catalogued thermal functionalities of the components and the found hydraulic loops during system synthesis were important to interpret optimization results and to detect corresponding operating strategies.

Acknowledgements

This work has been supported by the German Ministry BMBF in the VEOTOP project (FKZ: 01LY1809A). We are much obliged to the BMBF and the project leader DLR.

References

[1] Fidorra N, Kistner J, Tegethoff W, Köhler J. Energetische Bewertung von Wasserkreislauf-Systemen für Supermärkte. In DKV-Tagungsberichte. *Deutsche Kälte- und Klimatagung;* 2017; Bremen.

[2] Fidorra N, Köhler J. Energetische Untersuchung integrierter Supermarktkonzepte. In DKV-Tagungsberichte. *Deutsche Kälte- und Klimatagung;* 2016; Kassel.

[3] Kaiser C, Schröder A, Raabe G. Entwicklung eines CO2-Ejektorkreislaufs für eine umschaltbare Wärmepumpen-Klimaanlage für Omnibusklimaanlagen. Deutsche Bundesstiftung Umwelt - Forschungsbericht (AZ: 30270). 2015.

[4] Karampour M, Sawalha S. Comparison of State-of-the-art CO2 and Alternative Refrigeration Systems for Supermarkets. *Gustav Lorentzen Conference*; 2018; Valencia.

[5] Kraft, D. A software package for sequential quadratic programming. Forschungsbericht DFVLR. 1988; DFVLR-FB 88-28.

[6] Luersen MA, Le Riche R, Guyon F. A constrained, globalized, and bounded Nelder–Mead method for engineering optimization. *Struct Multidisc Optim.* Springer-Verlag; 2003.

[7] Nelder JA, Mead R. A simplex for function minimization; *The Computer Journal*. 1965, Volume 7 (4): 308–313.

[8] Nöding M, Fidorra N, Gräber M, Köhler J. Operation Strategy for Heat Recovery of Transcritical CO2 Refrigeration Systems with Heat Storages. *29th International Conference on ECOS*; 2016; Portoroz.

[9] Peteranderl C, Bernath M, Tegethoff W, Köhler J. City Bus with Modular R744 HVAC System Based on Passenger Car Components, *Thermal Management Systems Symposium*; 2018; San Diego.

[10] SciPy, https://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html (visited 19.12.2018)

[11] TLK Thermo GmbH, TIL-Suite, www.tlk-thermo.de http://www.tlk-thermo.com/index.php/de/softwareprodukte/til-suite (visited 19.12.2018)

[12] Titze M. Dynamische Simulationen zur Wärmerückgewinnung im Supermarkt. *Supermarkt-Symposium ZVKKW;* 2014; Darmstadt.

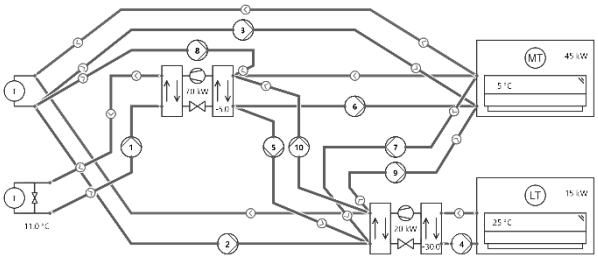
[13] Titze M. Energetic Analysis and Optimisation Strategies of a modern north European Supermarket [dissertation]. TU Braunschweig; 2017.

[14] Weustenfeld T. Heiz- und Kühlkonzept für ein batterieelektrisches Fahrzeug basierend auf Sekundärkeisläufen [dissertation]. Fakultät für Maschinenbau. TU Braunschweig; 2017.

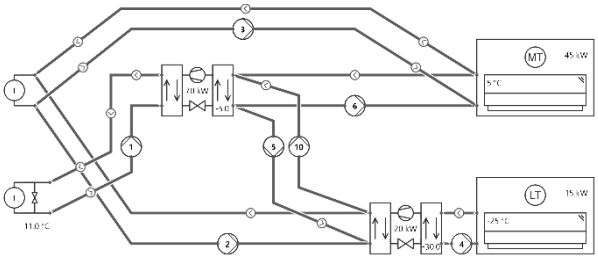
Appendix

Appendix A. The figure below shows the synthesized system model. The red interconnections have no pumps. Their mass flow rate are determined by the other mass flow rates in the hydraulic system. Temperature set-points and refrigeration capacities are displayed as well.

The LT cabinet can only be cooled by the LT refrigeration cycle. The condenser of the MT refrigeration cycle is only be cooled by outdoor unit 2 which can hold a minimum required temperature for condensation. The MT cabinet can be cooled by MT refrigeration cycle and outdoor unit 1. The LT refrigeration cycle (condenser) can be cooled by outdoor unit 1 and MT refrigeration cycle (evaporator). Additionally, several interconnections exist for looping mass between those components at MT temperature level.



Appendix B. The figure below shows the system model of the third identified option for system reduction. Interconnections between the LT refrigeration cycle and the MT cabinet are removed. Additionally, one related interconnection is removed as well. Remaining second-best operating modes are only simple feed and return flows between the components of MT level. Therefore, the path of pump 10 has to remain in the system layout.



Simulation Case Study: Using Simscape for Human Knee Joint Models

Ruth Leskovar*, Andreas Körner, Felix Breitenecker

[1]Institute of Analysis and Scientific Computing, TU Wien, Wiedner Hauptstraße 8-10, 1040 Vienna, Austria; *ruth.leskovar@tuwien.ac.at

SNE 29(2), 2019, 101-104, DOI: 10.11128/sne.29.sn.10477
Received: March 30, 2019 (Selected ASIM GMMS/STS 2019 Conference Publication); Accepted; May 10, 2019
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna, ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. This contribution presents a knee model implemented in Simscape™ and analyses its usability regarding biomechanical aspects. The model simulates the flexion of a human knee. It contains the three bones of the human knee, which are linked together by two revolute joints and one spring damper element representing the patellar tendon. This illustrates a simplification of the human knee joint due to the restriction of degrees of freedom. Finally, this work discusses the advantages and disadvantages of using the multibody library in Simscape for biomechanical models.

Introduction

In the research field of biomechanics, mathematical models for anatomic joints play an important role analysing kinematics and kinetics in the human body. Mainly, two different modelling approaches are used, models based on partial differential equations and multibody systems. They differ in their mathematical description and therefore in application fields as well.

Models described by partial differential equations (PDEs) depend on time and space, which gives the opportunity to analyse even small deformations, which take place in human bone and soft tissue, as ligaments and tendons, under repeated loads.

Multibody models are described by ordinary differential equations (ODEs), which leads to final solutions dependent on time only. These models do not require a precise description as PDE models, because detailed information about the geometries of the bodies does not influence the solution of the model strongly. Multibody models are often used for investigating gross motion and interaction of various connected bodies. Here, a description of movement change over time is needed and no deformations in tissues are investigated.

One possible application is the gait analysis. The study of motion sequences, in case of injuries compared to a normal gait cycle, can give insights to rehabilitation and therapy techniques. Further on, biomechanical models, which simulate the human walk, can be extended to various walking scenarios, as running or climbing stairs.

Results gained from these models can be used in the development of prostheses to improve functions of leg prostheses. For example, multibody models are developed for the investigation of interactions taking place in the human body of amputees wearing prostheses during various falling scenarios as it is done in [1].

1 Multibody Modelling

Physical modelling describes a system based on fundamental physical laws. This technique is often used for systems, where no mathematical equations describing the dynamical behaviour of the system are known. Multibody models are based on the physical modelling approach and describe relative motion between different bodies and the resulting dynamics. Multibody systems consist of bodies and joints, which link them together. Bodies are defined by their physical properties, as mass, centre of mass, density, inertial rotation, etc. There exist various types of joints, which differ in their amount and properties of degrees of freedom, resulting in rotational or translational movement respectively.

Figure 1 shows two rigid bodies, which are connected by a kinematic joint. The positions of the bodies, defined by their local coordinates, can be described in respect to the global coordinate system as it is explained more detailed in [2]. Since the physical model building process does not require mathematical equations, the development of multibody models is often simpler and faster than other methods. The description of a system by PDEs requires more detailed information about

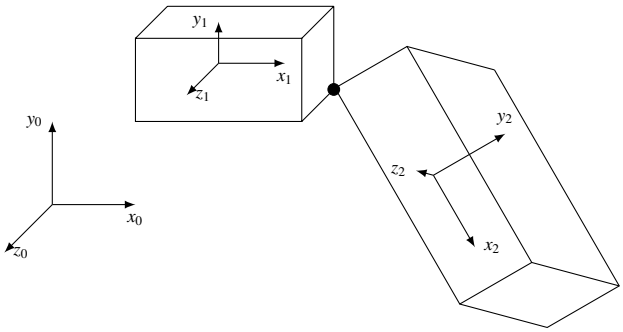


Figure 1: Illustration of a multibody system with local coordinates, indicated by $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, and the global coordinate system, indicated by $\boldsymbol{x}_0$.

geometries defining the included bodies and the numerical solving procedure is more complex than using ODE solver, which are sufficient for multibody models. Furthermore, the physical modelling approach can be used to detect mathematical equations of complex systems to describe their dynamical behaviour. The combination of components, whose dynamics are already known, leads to the desired system description. The analysis and further investigation of simulation results gives the possibility to describe the dynamics by mathematical equations.

In multibody models, the relative motion of a rigid body, which results by applying a force $\boldsymbol{F}$, is given by the set of a second order ordinary differential equations

$$M\ddot{\boldsymbol{x}} + J_{\boldsymbol{x}}^T \lambda = \boldsymbol{F}, \tag{1}$$

with the mass matrix M of the system, the coordinates $\boldsymbol{x}$ of the investigated body, the corresponding Jacobian matrix J and the Lagrange multipliers λ. The derivation can be conducted by using the Lagrange formalism and is explained in [3].

As already mentioned above, multibody models are used to analyse kinematics between bodies in biomechanics. For example, they are used to analyse the interactions in the human body, which take place during motion.

2 Conceptual Model for the Human Knee

The platform `https://simtk.org` offers a repository of biomechanical models. Researchers can share their work, including simulation models and corresponding files as geometries or data. The open access strategy facilitates the development of new models.

The following model simulates the flexion of a human knee and is based on the work of [4], [5] and [6]. The investigated models are mainly implemented in Adams™, a multibody dynamics simulation software. These models describe the flexion of the tibia and the involved movement of the patella in respect to a fixed femur after applying a force at the tibia. Similar to the anatomy of the human knee, ligament forces are included, connecting the bones together and predetermining the direction of motion. Additionally, contact forces influence the movement to prevent interpenetration of the bones.

The usage of the multibody library for Simscape, embedded in the Simulink® environment, requires the introduction of joints, otherwise no movement is possible. A possible simplification of the knee joint reduces the degrees of freedom in the knee to one rotational as it is stated in [7]. For the analysis of load distribution in the knee joint, this simplification is sufficient. This rotational degree of freedom describes the flexion of the tibia and is represented by one revolute joint. This revolute joint j_1, connecting femur and tibia, is placed in the last third of the femoral condyle as it is depicted in Figure 2. A second revolute joint j_2 is introduced, linking the patella to the femur and describing the movement of the patella. This joint is situated at the centre of mass of the femur to ensure that the patella is sliding between the femoral condyles at the front side. In the anatomy of the human knee, the patella is situated in the patellar tendon, which connects the quadriceps to the tibia. In biomechanics, tendons are often implemented using

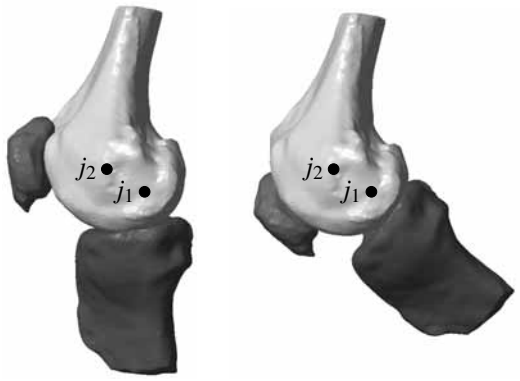


Figure 2: Visualisation of the joint centres in the knee model in the initial position left and in flexion right.

spring damper forces. Therefore, one spring damper element is introduced, which connects tibia and patella. This component applies a linear force acting reciprocally between the connected bodies. This force f is proportional to the distance x between the connected bodies and the resulting velocity $\dot{x}$. It is calculated by

$$f = k \cdot (x - l) + D \cdot \dot{x}, \tag{2}$$

with the spring constant k and damper coefficient D. The natural length of the spring is given by l. This spring damper force does not imply any movement to the bodies, but ensures that the patella is sliding following the rotation of the tibia. The tibia starts to move only after an applied torque at the joint j_1.

3 Simulation Model in Simscape for the Human Knee

Simscape is embedded in the Simulink environment and is intended for the development of physical systems. The library comprises elements for driveline, electrical, fluids and multibody models. Due to the embedment to Simulink, it is possible to use components of Simulink as well. For that purpose, special blocks convert the different signal types from 3D of Simscape to 1D of Simulink and vice versa.

The construction of a simulation model in Simscape is similar to Simulink and realised by drag and drop of components. The final model structure for the flexion of a human knee is shown in Figure 3. Each block represents one component of the model. The global parameters, as gravity and solver settings as well as the global reference frame, are defined in blocks too. Without these blocks, a simulation run of a Simscape model is not possible.

The knee model consists of three bones, femur, tibia and patella, which are implemented as rigid bodies. The corresponding blocks contain `stl`-files defining their geometry and shape as well as physical parameters, namely mass, centre of mass and inertial rotation. These files and parameters are given with the Adams models of [4], [5] and [6] and represent data of the right knee of a 77 year old man.

The stiffness of the spring and damping coefficient for the patellar tendon between tibia and patella are the same values as in the Adams model. The rigid transforms between the components assure the correct attachment points of the tendon to the bones and the right position of joint centres, respectively.

The revolute joints contain parameters for spring stiffness and damping coefficient of the joint and are summarised in Table 1. These parameters are not the same as in the Adams models, because there no rotational joints are considered. These models deal with ligaments only, which contain parameters for translational movements. The conversion of these parameters for rotational movement requires knowledge about rotational stiffness and is not accessible in this case.

	Spring stiffness	Damping coefficient
j_1	553.5 $\frac{\text{N mm}}{\text{deg}}$	1 $\frac{\text{N mm s}}{\text{deg}}$
j_2	33 $\frac{\text{N mm}}{\text{deg}}$	1 $\frac{\text{N mm s}}{\text{deg}}$

Table 1: Parameters for the joints.

Therefore, the parameters for the joints are calculated by calibration and comparing the outputs from the Adams and Simscape model. Both model do not show the same behaviour due to the already discussed restrictions. One more subsequent difference from the model description is the input. In the Adams model, a force is acting at the tibia in posterior direction. The Simscape model requires a torque τ acting on the joint j_1. The torque τ is calculated by using geometrical basics and is finally given by

$$\tau(t) = 28.8 \cdot \sin(2\pi \cdot 0.125(t-1)) \cdot \mathrm{H}(t-1)\ \text{Nm}, \tag{3}$$

with the step function H. The acting torque is implemented in Simscape by using Simulink blocks.

The output of the simulation model is the resulting angle between femur and tibia representing the flexion and extension of the human knee. Figure 4 shows the angle of the Adams model and the Simscape model. The comparison of the output leads to the missing joint parameters. The non-linearity of the Adams model, coming from the ligaments, results in a more steep waveform than the sine wave from the Simscape model.

4 Conclusion and Outlook

The presented knee model is a simplification of the human knee joint and it shows the possibilities and restrictions of using the multibody library in Simscape for biomechanical models. The usage of Simscape for building multibody models is a good option due to the embedment in the Simulink environment. This allows

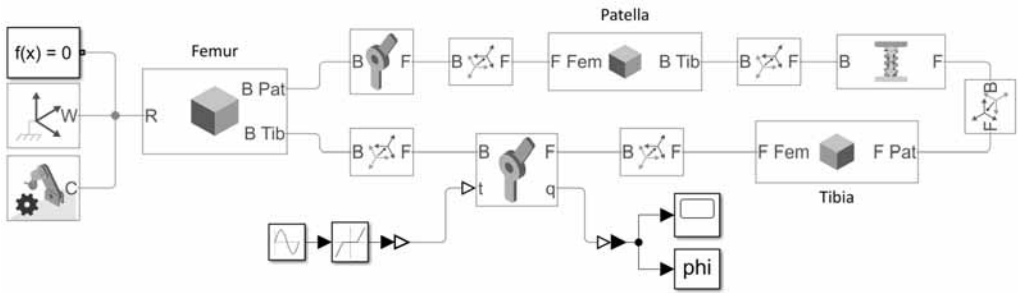


Figure 3: Block structure of the multibody model for the human knee implemented in Simscape.

to use block elements from other libraries and all tools which are available in Simulink. Moreover, the postprocessing in MATLAB® offers many possibilities. This gives flexibility in the development and extension of the model and later in the analysis. For example, the construction of more complex model structures and combination with system based modelling approaches allows to build feedback loops, which extend the areas of application for biomechanical models.

Since MathWorks® does not focus on physical modelling, the library of Simscape is restricted regarding some aspects. Anatomic joints are complex structures due to their composition of various tissues as bones, ligaments and tendons. In order to build a model which fulfils the biomechanical properties of the human knee joint, the consideration of a revolute joint is insufficient. The incorporation of crucial and collateral ligaments to the model increases the degrees of freedom and improves a realistic movement. As it is discussed in [8], ligaments show a non-linear behaviour regarding their stress-strain relationship. Therefore, they can not be modelled with linear spring damper elements but using external functions. Nevertheless, the Simscape multibody library requires the use of joints to create the preconditions for the movements.

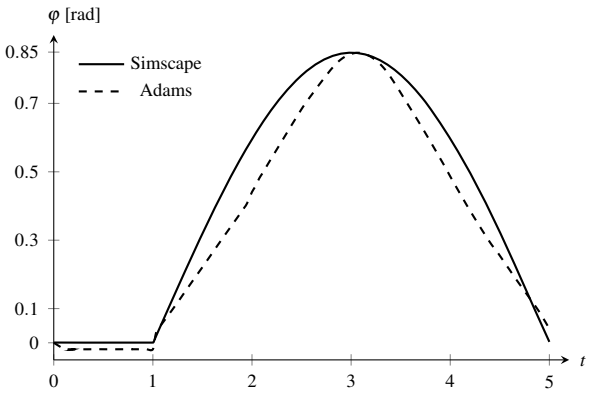


Figure 4: Angle φ between femur and tibia during flexion of the knee for the Adams and Simscape model.

References

[1] Welke B, Schwarze M, Hurschler C, Calliess T, Seehaus F. Multi-body simulation of various falling scenarios for determining resulting loads at the prosthesis interface of transfemoral amputees with osseointegrated fixation. *Journal of Orthopaedic Research*. 2013; 31(7):1123–1129.

[2] Georg Rill TS. *Grundlagen und Methoden der Mehrkörpersimulation*, vol. 3. Springer Vieweg. 2017.

[3] Quental C, Folgado J, Ambrósio J, Monteiro J. A multibody biomechanical model of the upper limb including the shoulder girdle. *Multibody System Dynamics*. 2012;28(1-2):83–108.

[4] Guess TM, Thiagarajan G, Kia M, Mishra M. A subject specific multibody model of the knee with menisci. *Medical Engineering and Physics*. 2010;32(5):505–515.

[5] Guess TM, Liu H, Bhashyam S, Thiagarajan G. A multibody knee model with discrete cartilage prediction of tibio-femoral contact mechanics. *Computer Methods in Biomechanics and Biomedical Engineering*. 2013; 16(3):256–270.

[6] Bloemker KH, Guess TM, Maletsky L, Dodd K. Computational Knee Ligament Modeling Using Experimentally Determined Zero-Load Lengths. *The Open Biomedical Engineering Journal*. 2012;6:33–41.

[7] Brinckmann P, Frobin W, Leivseth G, Drerup B. *Orthopädische Biomechanik*. MV Wissenschaft. 2012.

[8] Blankevoort L, Huiskes R. Ligament-Bone Interaction in a Three-Dimensional Model of the Knee. *Journal of Biomechanical Engineering*. 1991;113(3):263.

Direct Implementation of ARGESIM Benchmark C7 'Constrained Pendulum' in MATLAB and EXCEL

Anna E. Stockinger[1], Elisabeth Gütl[1], Stefan A. Rath[1], Dominik Strasser[1], Martin Bicher[2*], Andreas Körner[1], Horst Ecker[3*]

[1]Mathematical Modelling and Simulation Group, Inst. of Analysis and Scientific Computing
[2]Inst. of Information System Engineering, [3]Inst. of Mechanics and Mechatronics
TU Wien, Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria; *horst.ecker@tuwien.ac.at*

SNE 29(2), 2019, 105 - 110, DOI: 10.11128/sne.29.bne07.10478
Received: October 25, 2018; Revised: January 13, 2019;
Revised: May 10, 2019; Accepted: May 30, 2019
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. This Benchmark Report with educational aspects presents a straightforward and direct implementation of ARGESIM Benchmark C7 'Constrained Pendulum' in MATLAB and in EXCEL: the given models are implemented without any change, and the tasks are directly simulated without any rearrangement. Central issue of this benchmark is the detection and handling of a state event: when the pendulum hits or releases a pin, pendulum length and angular velocity are changing discontinuously.

The MATLAB approach makes use of the event termination feature of the ODE solvers, and a MATLAB script loops between long and short pendulum and handles the event changes. The EXCEL approach solves the overall ODEs by Euler algorithm (a simple EXCEL recursion). The events are synchronized with the chosen time step (detection with delay), and handled by distinction of cases in any state update (no event, hit, release, velocity jump). The MATLAB implementation is straightforward but makes use of different models necessary. The EXCEL implementation shows that a spreadsheet tool – not really designed for simulation - can do simulation by direct implementation of ODE algorithms, but event-handling causes elaborate case-by-case analysis.

Introduction

ARGESIM Benchmark C07 'Constrained Pendulum' requires the simulation of a pendulum that hits and leaves a pin at a certain angular position. Figure 1 shows the schematic structure of the pendulum system. The description of the pendulum is generally given by a nonlinear ODE or a linear (approximating) ODE of second order:

$$m \cdot l \cdot \ddot{\varphi}(t) = -m \cdot g \cdot \sin\varphi(t) - d \cdot l \cdot \dot{\varphi}(t)$$

$$m \cdot l \cdot \ddot{\varphi}(t) = -m \cdot g \cdot \varphi(t) - d \cdot l \cdot \dot{\varphi}(t)$$

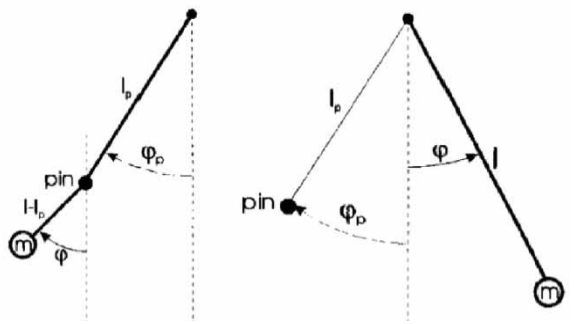


Figure 1: Pendulum hitting a pin.

The angular movement $\varphi(t)$ (measured in radians) is positively counted from the vertical position counterclockwise. Given parameters are pendulum length l, pendulum mass m, damping factor d, the angular position of the pin φ_{pin}, and initial values for angular movement φ_0 and angular velocity $\dot{\varphi}_0$.

If φ_0 or $\dot{\varphi}_0$ is big enough, the pendulum hits the pin if the event function $e(t)$ becomes zero:

$$e(t) = \varphi(t) - \varphi_{pin} = 0.$$

The hit is modelled as state event, changing pendulum length l and angular velocity $\dot{\varphi}$ discontinously.

At hit, the pendulum length shortens to $l_s = l - l_p$, and the angular velocity jumps (increase):

$$\dot{\varphi} \rightarrow \frac{l}{l_s} \cdot \dot{\varphi}$$

After some time the pendulum leaves the pin, causing the 'reverse' event: the pendulum length changes back to l, and the angular velocity jumps at release again (decrease):

$$\dot{\varphi} \rightarrow \frac{l_s}{l} \cdot \dot{\varphi}$$

The tasks of the benchmark are i) time domain simulations with proper detection and handling of the hit and release events, with varying parameters, ii) comparison of nonlinear and linear model description and impact on hit and release events, and iii) a boundary value problem.

These tasks provide different challenges for the MATLAB implementation, and for the EXCEL implementation. Because the solution approach follows a straightforward implementation, the detection and the handling of the hit and release events require special consideration.

1 MATLAB Implementation

The MATLAB ODE solvers require an explicit state space description $\dot{\vec{y}} = \vec{f}(\vec{y}, t)$, which gives for the pendulum description with $y_1(t) = \varphi(t), y_2(t) = \dot{\varphi}(t)$ the following equations:

$$\dot{y}_1 = y_2 = yder_1$$
$$\dot{y}_2 = -\frac{g}{l}\sin y_1 - \frac{d}{m}y_2 = yder_2$$

The MATLAB ODE solvers need these equations as function:

```
function yder = ODE_pendulum(t,y,m,l,d,g)
yder=[y(2);-g*sin(y(1))/l-d/m*y(2)]; end
```

or in linear version

```
function yder = ODE_pendulum_lin(t,y,m,l,d,g)
yder=[y(2);-g*sin(y(1))/l-d/m*y(2)]; end
```

1.1 Task a: Time Domain Analysis – MATLAB Implementation

Since the pendulum swings because of hit and release of the pin in the original length *l* and in the shortened length l_s, a permanent change between two differential equations must take place. This was solved by a *while loop* in combination with an *event function* characterized by the position of the pin, which terminates the ODE solver at hit or at release. With the *event function* it is possible to toggle between the differential equations by interrogating the momentary angle of the pendulum.

In the next listing the *event function* is shown as code: the event occurs when the current angle equals the angle of the pin (**position** equals zero); **isterminal = 1** means that the differential equation is only executed until the first occurrence of the event; **direction = 0** means that it does not matter which side of the pin angle the event originates from:

```
function [position,isterminal,direction] =
event_pendelum(t,y,pinangle)
position=y(1)-pinangle;
isterminal=1;
direction=0; end
```

For solving the differential equations, MATLAB's *ode45* solver (standard) is used. The solver is called in a loop (terminating with given simulation time), which toggles between long and short pendulum. After the *event function* has terminated the ODE solver at the event time, the event is handled: change of length, change of angular velocity; then the solution is concatenated to the previous ones. The MATLAB code is:

```
stopevent = odeset('Events', @(t, y) Event_pendulum
                                    (t, y, pinangle));
if(startingangle>pinangle)
  %Pendulum starts in the long state
  [t,y,te,ye,ie]=ode45(@(t,y) DGL_pendulum(t,y,m,l,d,g),
   [0 10],[startingangle;Initialangularvelocity],stopevent);
else
  %Pendulum starts in the short state
  [t,y,te,ye,ie]=ode45(@(t,y) DGL_pendulum(t,y,m,l2,d,g),
  [0 10],[startingangle;Initialangularvelocity],stopevent);
  i=2;
end
while t(end) < 10;  %as long as the end time not reached
 if(mod(i,2) ~= 0);  %i = odd -> short pendulum
  [t2,y2,te,ye,ie]=ode45(@(t2,y2)  ODE_pendulum (t2,y2,m,l2,
          d,g), [t(end) 10],[y(end,1);y(end,2)*l/l2],stopevent);
 else       %i  = even -> long pendulum
  [t2,y2,te,ye,ie]=ode45(@(t2,y2) DGL_pendulum(t2,y2,m,l
         ,d,g),[t(end) 10],[y(end,1);y(end,2)*l2/l],stopevent);
  end
  y=vertcat(y,y2);    t=vertcat(t,t2);    i=i+1;
end
```

This implementation works independent of position of pin and of initial angle, so that simulations for the two different initial angles can be performed (*Task a1* and *Task a2*). Section 3 shows results of the simulations, in comparison with the results from the EXCEL implementation results.

1.2 Task b: Comparison of Nonlinear and Linear Model – MATLAB Implementation

For *Task b: Comparison of Nonlinear and Linear Model*, simply the simulation is repeated with the linear model, using the same MATLAB script as for *Task a: Time Domain Analysis* but with changed derivative function.

The *ode45* solver performs stepsize control, so that results for nonlinear and linear model are calculated at different grid points, and the number of grid points differs. A comparison can be done simply graphically – for a numerical comparison of results, interpolation of both results must be used (graphical results in Section 3).

1.3 Task c: Boundary Value Problem – MATLAB Implementation

Task c: Boundary Value Problem requires to reach a target angle of the pendulum ($-\pi/2$), by proper choice of the initial angular velocity $\dot{\varphi}_0$. For this task a line search is implemented, which varies the initial angular velocity with variable increments, until the desired angular position is reached.

For controlling the search, a deviation is defined, the difference between the target angle and the maximally reached angle of the shortened pendulum. Depending on the current deviation, the initial angular velocity is increased or decreased with respect to the deviation size, unless the deviation is smaller than allowed. The following code is self-explanatory:

```
while(perdeviation<dev)
 if(startingangle>pinangle); % starts in  long state
  [t,y,te,ye,ie]=ode45(@(t,y) DGL_pendulum(t,y,m,l,d,g),
    [0 10],[startingangle;Initialangularvelocity],stopevent);
    [t2,y2,te,ye,ie]=ode45(@(t,y) DGL_pendulum(t,y,m,l2 ,d,g),
    [t(end) 10],[y(end,1);y(end,2)*l/l2],stopevent);
      y=vertcat(y,y2);  t=vertcat(t,t2);
 else   %Pendulum starts in short state
    [t,y,te,ye,ie]=ode45(@(t,y) DGL_pendulum(t,y,m,l2,d,g),
   [0 10],[startingangle;Initialangularvelocity*l/l2],stopevent);
 end
 maxangle= min(y(:,1));%until pendulum max turns
  dev= abs(targetangle-maxangle);
  if(dev>0.5)   delta=0.1;
  elseif (dev<=0.5 & dev>0.1)  delta=0.01;
  elseif (dev<=0.1 & dev>0.01) delta= 0.001;
  else  delta=0.0001;  end
  if(targetangle<maxangle)
     Initialangularvelocity=Initialangularvelocity-delta;
  else
     Initialangularvelocity=Initialangularvelocity+delta;
  end;    steps=steps+1; end
```

This MATLAB script results in an initial angular velocity of $\dot{\varphi}_0 = -2{,}187$ rad/s in order to reach the angle $-\pi/2$ after one hit. Graphical results are given in Section 3.

2 EXCEL Implementation

EXCEL is not really a simulator, it is mainly used in the area of simulation in economics, also for dynamic processes. 'Basic' EXCEL does not offer ODE solvers - but the spreadsheet structure allows an easy implementation of simpler ODE solvers, as Euler solver, or Heun solver with fixed appropriate small step size: columns calculate time advance and state advance in a recursive manner.

Because of the fixed stepsize, state events cannot be localized – they can only be detected at the next timestep after the event has happened. The handling of the event causes complicated if -then – else constructs, depending on the quality of the event – in this case two actions with the change of the parameter length – easy – and with the jump of the angular velocity – complicated.

2.1 Euler Solver for Pendulum Equations

This contribution makes use of the EULER ODE solver, in state space notation $\dot{\vec{y}} = \vec{f}(\vec{y}, t)$ generally given by

$$\vec{y}_{n+1} = \vec{y}_n + h \cdot \vec{f}(\vec{y}_n, t_n), \;\; t_{n+1} = t_n + h$$

With $y_1(t) = \varphi(t), y_2(t) = \dot{\varphi}(t)$ the state space

$$\dot{y}_1 = f_1(y_1, y_2) = y_2$$
$$\dot{y}_2 = f_2(y_1, y_2) = -\frac{g}{l}\sin y_1 - \frac{d}{m} y_2$$

results in the following Euler algorithm for solving the ODEs:

$$y_{1,n+1} = y_{1,n} + h \cdot f_1(y_{1,n}, y_{2,n}) = y_{1,n} + h \cdot y_{2,n}$$
$$y_{2,n+1} = y_{2,n} + h \cdot f_2\big(y_{1,n}, y_{2,n}\big) =$$
$$= y_{2,n} + h \cdot \left(-\frac{g}{l}\sin y_{1,n} - \frac{d}{m} y_{2,n}\right)$$

2.2 Euler Solver with Event Handling

Together with the state space update due to Euler algorithm, in each time step t_{n+1} also the event function $e(t) = \varphi(t) - \varphi_{pin}$ must be checked and compared with the previous time step:

$$e_n = \varphi_n - \varphi_{pin} = y_{1,n} - \varphi_{pin}$$
$$e_{n+1} = \varphi_{n+1} - \varphi_{pin} = y_{1,n+1} - \varphi_{pin}$$

If the event function has changed sign, then the event has happened between t_n and t_{n+1}, and the quality of the sign change determines the event: hit or release.

The event must be handled now at t_{n+1}. As for t_{n+1} all time and state values are already calculated, the event changes must be considered at the next integration step

$$t_{n+2}, y_{1,n+2}, y_{2,n+2}$$

which now makes uses of a changed length (and also the further steps) and which must make use (only once) of the jump in the angular velocity, i.e. in case of hit

$$\dot{\varphi}_{n+1} = y_{2,n+1} \rightarrow \frac{l}{l_s} \cdot \dot{\varphi}_{n+1} = \frac{\dot{l}}{l_s} \cdot y_{2,n+1}$$

As consequence, the handling of the event requires actions within three timesteps – resulting in complicated case-by-case analysis of the status of the states in each time step. The update equation for the angular velocity

$$y_{2,n+1} = y_{2,n} + h \cdot \left(-\frac{g}{l}\sin y_{1,n} - \frac{d}{m} y_{2,n}\right)$$

is extended by nested *if – then- else* calculations, especially because of the singular jump of the angular velocity.

2.3 Event Handling in EXCEL

There are several ways to implement the complicated case-by-case analysis for the events *hit* and *release*. In any case, the implementation is based on EXCEL's *if-then-else* formula for calculating the value in a cell:

cellvalue **=** **IF(***condition***, THEN-***formula***, ELSE-***formula***);**

For proper implementation of the changes at events *hit* and *release*, these *if-then-else* formulas must be nested, i.e. **THEN-***formula* and **ELSE-***formula* are themselves *if-then-else* formulas. For documentation, the order of the *if*-formulas is identified by a number, i.e. **IF1**, **IF2**, **and IF3**.

The first check **IF1** asks if the angle of the pendulum φ at time t is greater than the pin angle φ_{pin}. If *true* – **THEN-***formula* of **IF1** – another *if*-formula **IF2** is necessary. *if*-formula **IF2** decides, that if the angle of the pendulum φ is less than the pin angle φ_{pin} at the time $t - h$ (checking **IF2**), then the extended update equation for the angular velocity (with change of length to l and with jump of angular velocity with factor l_s/l) is used; if this is not the case (φ greater than φ_{pin} at $t - h$) the standard update equation for the Euler update of the angular velocity with length l is used. The **ELSE-***formula* of **IF1** is another *if*-formula **IF3**. It decides that if the angle of the pendulum φ is greater than the pin angle φ_{pin} at the time $t - h$ (checking **IF3**), then the extended update equation for the angular velocity with shortened length l_s and with jump of angular velocity with factor l/l_s) is used (**THEN-***formula* of **IF3**); if this is not the case, the standard update equation for the Euler update of the angular velocity with length l_s is used **ELSE-***formula* of **IF3**.

The combined *if*-formulas **IF1**, **IF2**, and **IF3** now setup the overall update formula for the angular velocity:

```
= IF ( H16 > phi_p, IF ( H15 < phi_p,
        ls/l * G16 * (1-h*d/m) – h * g/l* SIN(H16),
        G16*(1-h*d/m)-h*g/l*SIN(H16)],
    IF ( H15  >phi_p,
        l/ls * G16  *(1-h*d/m) - h*g/ls * SIN(H16),
       G16 * (1-h*d/m) – h * g/ls * SIN(H16)))
```

Here, the angular velocity at the time step t is calculated in cell **G17**; **H16** is the angle value of the pendulum at the time step $t - h$ and **H15** the angle value at $t - 2h$. **G16** is the value of the angular velocity at the time step before $(t - h)$.

2.4 Task a: Time Domain Analysis – EXCEL Implementation

The implementation follows the classical spreadsheet usage – time update and update of states in columns defined by recursive formulas, and parameters are defined by names, for better understanding of formulas.

Figure 2 shows definitions for the parameters, and Figure 3 sketches the structure of the spreadsheet with columns for time – step size 0.001 –, angle, and angular velocity.

EXCEL allows graphical representations of various kinds; result graphs can be easily produced from the result columns – results see Section 3.

	A	B	C
1	**CONSTRAINED PENDULUM**		
2			
3	DGL:		
4	phi''=	=-g*sin(phi)/l-d/m*phi '	
7	PARAMETER:		
8	g	gravity	9,81
9	m	mass of the pendulum	1,02
10	l	length of the pendulum	1
11	ls	shortend length of the pendulum	0,3
12	d	damping factor	0,2
7	INITIAL CONDITIONS:		
8	phi_0	starting angle	0,5235988
9	phi'_0	starting angular velocity	0
10	phi_P	angle of the pin	-0,262

Figure 2: Definition of parameters – EXCEL implementation.

	D	E	F	G	H
13					
14					Expl. Euler
15		time		phi'	phi
16		0		0	0,5235988
17		0,001		-0,004905	0,5235988
18		0,002		-0,00980904	0,5235939
19		0,003		-0,01471207	0,5235841
20		0,004		-0,01961406	0,5235693
21		0,005		-0,02451497	0,5235497
22		0,006		-0,02941474	0,5235252
23		0,007		-0,03431335	0,5234958
24		0,008		-0,03921075	0,5234615
25		0,009		-0,04410689	0,5234223
26		0,01		-0,04900175	0,5233782

Figure 3: EXCEL spreadsheet structure for state updates.

2.5 Task b: Comparison of Nonlinear and Linear Model – EXCEL Implementation

For *Task b: Comparison of Nonlinear and Linear Model*, simply two further columns are added, calculating the linear equations, using same case-by-case analysis as with the nonlinear equations.

Figure 4 sketches the structure of the spreadsheet with –for time – step size 0.001 –, nonlinear results (angle, angular velocity), and linear results (angle, angular velocity). The fixed step size allows a direct calculation of the deviation between nonlinear and linear results, shown in Figure 4 with columns for deviations of angle and angular velocity.

			Expl. Euler						
time		phi'	phi		phi'_lin	phi_lin		dev phi'	dev phi
0		0	0,26179939		0	0,26179939		0	0
0,001		-0,002539015	0,26179939		-0,00256825	0,26179939		-0,00253901	0
0,002		-0,005077781	0,26179685		-0,00513625	0,26179682		-0,00507778	2,9237E-08
0,003		-0,007616274	0,26179177		-0,00770398	0,26179168		-0,00761627	8,7709E-08
0,004		-0,01015447	0,26178415		-0,0102714	0,26178398		-0,01015447	1,7541E-07
0,005		-0,012692345	0,261774		-0,01283849	0,26177371		-0,01269234	2,9234E-07
0,006		-0,015229875	0,26176131		-0,01540523	0,26176087		-0,01522987	4,3848E-07
0,007		-0,017767035	0,26174608		-0,0179716	0,26174546		-0,01776704	6,1384E-07
0,008		-0,020303803	0,26172831		-0,02053756	0,26172749		-0,0203038	8,184E-07
0,009		-0,022840154	0,26170801		-0,02310309	0,26170696		-0,02284015	1,0522E-06
0,01		-0,025376064	0,26168517		-0,02566817	0,26168385		-0,02537606	1,3151E-06
0,011		-0,027911508	0,26165979		-0,02823277	0,26165818		-0,02791151	1,6072E-06
0,012		-0,030446464	0,26163188		-0,03079687	0,26162995		-0,03044646	1,9285E-06

Figure 4: EXCEL spreadsheet structure for state updates: results nonlinear model, results linear model, and deviation.

2.6 Task c: Boundary Value Problem – MATLAB Implementation

EXCEL provides as standard feature the *Goal Seeking Function* in the *What If Analysis* – suitable for approximating the initial value for angular velocity $\dot{\varphi}_0$, with goal reaching an angle of pendulum ($\pi/2$).

The *Goal Seeking Function* needs as input the cell of the parameter to be iterated – here $\dot{\varphi}_0$ in cell **I9**, the goal function evaluation – here the maximal angle after one hit in cell **I14**, and the goal value – here given with $\pi/2$ in cell **I13**. Additionally an accuracy parameter can be given – here 0.01 in cell **I14** (Figure 5).

	E	F	G	H	I
6					
7	INITIAL CONDITIONS:				
8	phi_0	starting angle			0,5235988
9	phi'_0	starting angular velocity			-2,17
10	phi_P	angle of the pin			-0,262
11					
12	maximum allowable deviation				0,01
13	required maximum angle of the pendulum				**-1,5707963**
14	measured maximum angle of the pendulum				-1,5684524

Figure 5: EXCEL goal seeking function – parameters for boundary value problem for initial angular velocity.

After start of the *What If Analysis,* EXCEL performs an optimizing search for the initial angular velocity, performing several simulation runs with changing values for the initial angular velocity.

Figure 5 shows the input cells for the *What If Analysis* and the results for initial angular velocity $\dot{\varphi}_0 = -2.17$ in cell **I9** and the reached goal angle φ_{end} in cell **I14** ($\varphi_{end} = -1.5684524 \sim -\frac{\pi}{2} = -1.5707963$). Graphical results for the solution are shown in Section 3.

An alternative is use of an EXCEL macro, which changes the initial velocity similar to the controlled line search in the MATLAB implementation.

3 Results – MATLAB and EXCEL

In the following graphical results from the three tasks for the MATLAB implementation and for the EXCEL implementation are shown and commented.

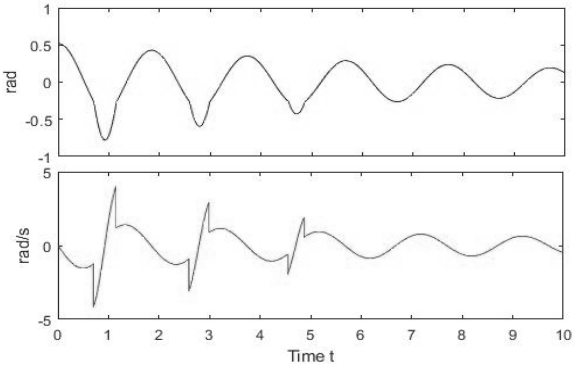


Figure 6: Simulation results for Task a1 – MATLAB.

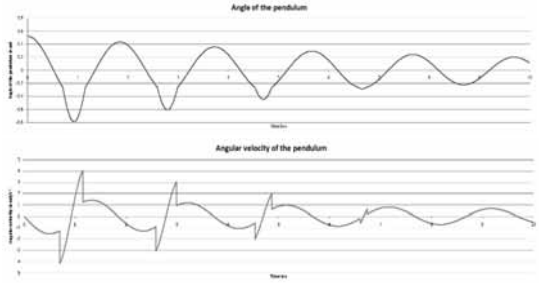


Figure 7: Simulation results for Task a1 – EXCEL.

At first glance, MATLAB results and EXCEL results for *Task a1* look the same. But the EXCEL result comes along with four hits, whereas the MATLAB implementation detects only three hits. It is evident from other solutions, that a fourth event pair *hit – release* exists.

Curiously MATLAB fails, although MATLAB makes use of a much more accurate ODE solver with step size control. Here MATLAB outmanoeuvres itself: the step size control choses because of high order a relatively big step size, so that the very close fourth event pair *hit – release* simply is not recognized (between one step both

events take place). As prevention, the ODE solver parameters must be better tuned.

For *Task b: Comparison of Nonlinear and Linear Model*, the MATLAB implementation repeats the simulation with the linear model. The *ode45* solver performs step size control, so that results for nonlinear and linear model are calculated at different grid points, additionally the total number of grid points differs.

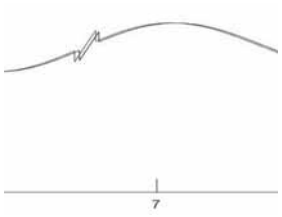


Figure 8: Comparison of linear and nonlinear model with zoom – MATLAB.

But both results can be plotted over the same time scale and look the same; only a zoom-in allows to recognize differences for later time values (Figure 8).

For a numerical comparison of results, interpolation of both results must be used.

The EXCEL implementation calculates nonlinear and linear model in parallel. The fixed step size allows a direct calculation of the deviation between nonlinear and linear results, shown in Figure 4 with columns for deviations of angle and angular velocity.

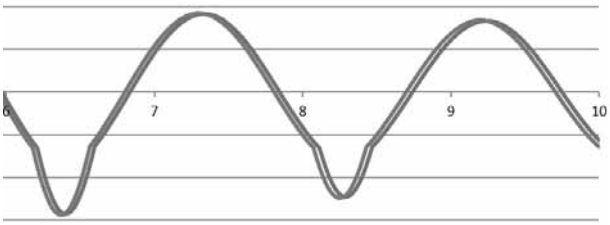


Figure 9: Comparison of linear and nonlinear model with zoom – EXCEL.

The EXCEL implementation shows bigger differences, because of less accuracy – see Figure 9.

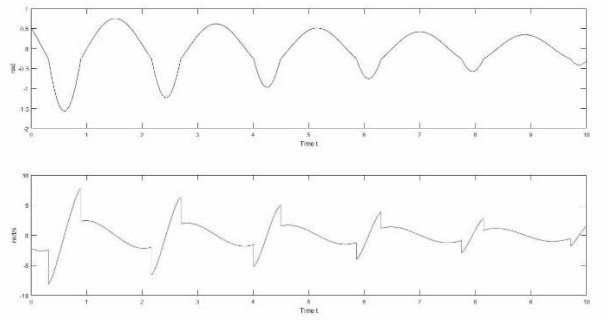


Figure 10: Time course of angle and angular velocity for solution of Task c: Boundary Value Problem - MATLAB.

Task c: Boundary Value Problem requires to reach a target angle of the pendulum $(-\pi/2)$, by proper choice of the initial angular velocity $\dot{\varphi}_0$. Both implementations work with an iterative approach to determine the initial angular velocity, with sufficient similar results: MATLAB gives $\dot{\varphi}_0 = -2.187$, and EXCEL gives $\dot{\varphi}_0 = -2.17$. For completeness, Figure 10 shows the time course of angle and angular velocity for solution of the boundary value problem

4 Conclusion

A spreadsheet tool as EXCEL is definitely not a simulator – modelling features for ODEs, processes, events, etc. are missing. But spreadsheet programs are an excellent experiment environment with statistical analysis, optimisation, what-if analysis, date handling, etc. Of course, macros and external programming could be used, but to some extent the standard features allow to implement this benchmark with sufficient accuracy, using explicit Euler integration.

The crucial task in the EXCEL implementation is the handling of events. Events must be realized by elaborate nesting IF-formulas. As a result, the entire algorithmic model is complicated and lacks clarity.

MATLAB is a classical programming and simulation tool, and allows quick solutions in a standardized and comfortable manner. The MATLAB ODE solvers allow event functions, which terminate the integration, the overall model must be put together in a loop. Nevertheless, parameters for ODE solver and for event functions must be properly tuned.

Both implementations produce quite similar results. MATLAB allows an increase of accuracy (parameters of ODE solvers), EXCEL is limited in choice of step size, because each integration step adds a now row into the spreadsheet (here about 12000 rows).

It was generally the intention to compose a direct implementation, without model reformulation, without toolboxes or macros. On the other side, simple reformulation would allow much easier event handling, especially in EXCEL. If instead of the angular velocity $\dot{\varphi}$ the tangential velocity $v = l \cdot \dot{\varphi}$ is used as state variable, at the events *hit* and *release* the (tangential) velocity remains unchanged; event handling is then simply switching between different values for pendulum length. And also, the boundary value problem can be avoided. Reaching exactly the angle $-\pi/2$ implies, that the pendulum must swing back; this happens only, if the angular velocity is zero for angle $-\pi/2$. As consequence, an initial value problem with reverse time can replace the boundary value problem.

SNE Simulation News

EUROSIM Data and Quick Info

EUROSIM 2019

10th EUROSIM Congress on Modelling and Simulation

La Rioja, Logroño, Spain, July 1 - 5, 2019

Contents

Short Info EUROSIM N2
Short Info ASIM, CEA-SMSG N3
Short Info CSSS, DBSS, LIOPHANT, LSS N4
Short Info KA-SIM, MIMOS, NSSM, PSCS N5
Short Info SIMS, SLOSIM, UKSIM N6
Short Info ROMSIM, Albanian Society N7
Short Info ARGESIM, SNE N8

Simulation Notes Europe SNE is the official membership journal of **EUROSIM** and distributed / available to members of the **EUROSIM Societies** as part of the membership benefits.

If you have any information, announcement, etc. you want to see published, please contact a member of the editorial board in your country or the editorial office. For scientific publications, please contact the EiC.

This *EUROSIM Data & Quick Info* compiles data from **EUROSIM** societies and groups: addresses, weblinks, and officers of societies with function and email, to be published regularly in **SNE** issues. This information is also published at **EUROSIM**'s website *www.eurosim.info*.

SNE Editorial Office /ARGESIM

→ *www.sne-journal.org, www.eurosim.info*
office@sne-journal.org, *eic@sne-journal.org*
SNE Editorial Office
Johannes Tanzler (Layout, Organisation)
Irmgard Husinsky (Web, Electronic Publishing)
Felix Breitenecker EiC (Organisation, Authors)
ARGESIM/Math. Modelling & Simulation Group,
Inst. of Analysis and Scientific Computing, TU Wien
Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria

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 simulation societies and groups to promote 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 *Full Members* and *Observer Member*s, and member candidates.

ASIM	Arbeitsgemeinschaft Simulation *Austria, Germany, Switzerland*
CEA-SMSG	Spanish Modelling and Simulation Group *Spain*
CSSS	Czech and Slovak Simulation Society *Czech Republic, Slovak Republic*
DBSS	Dutch Benelux Simulation Society *Belgium, Netherlands*
KA-SIM	Kosovo Simulation Society, *Kosovo*
LIOPHANT	LIOPHANT Simulation Club *Italy & International*
LSS	Latvian Simulation Society; *Latvia*
PSCS	Polish Society for Computer Simulation *Poland*
MIMOS	Italian Modelling and Simulation Association, *Italy*
NSSM	Russian National Simulation Society *Russian Federation*
ROMSIM	Romanian Society for Modelling and Simulation, *Romania, Observer Member*
SIMS	Simulation Society of Scandinavia *Denmark, Finland, Norway, Sweden*
SLOSIM	Slovenian Simulation Society *Slovenia*
UKSIM	United Kingdom Simulation Society *UK, Ireland*

Societies in Re-Organisation:

CROSSIM	*Croatian Society for Simulation Modeling* Croatia
FRANCOSIM	*Société Francophone de Simulation* *Belgium, France*
HSS	*Hungarian Simulation Society;* Hungary
ISCS	*Italian Society for Computer Simulation* *Italy*

EUROSIM Board / Officers. **EUROSIM** is governed by a board consisting of one representative of each member society, president and past president, and representatives for **SNE** Simulation Notes Europe. The President is nominated by the society organising the next **EUROSIM** Congress. Secretary, Secretary to the Board, and Treasurer are elected out of members of the board.

President	Emilio Jiménez (CAE-SMSG), *emilio.jimenez@unirioja.es*
Past President	Esko Juuso (SIMS) *esko.juuso@oulu.fi*
Secretary	M. Mujica Mota (DBSS), *m.mujica.mota@hva.nl*
Treasurer	Felix Breitenecker (ASIM) *felix.breitenecker@tuwien.ac.at*
Secretary to the Board	Andreas Körner *andreas.koerner@tuwien.ac.at*
Webmaster	I. Husinsky, *irmgard.husinsky*@tuwien.ac.at
SNE Representative	Felix Breitenecker *felix.breitenecker@tuwien.ac.at*

SNE – Simulation Notes Europe. **SNE** is a scientific journal with reviewed contributions 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 **SNE** as official membership journal. **SNE** Publishers are **EUROSIM**, **ARGESIM** and **ASIM**.

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

→ *www.sne-journal.org*,

✉ *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 2019, the **10th EUROSIM Congress**, will be organised by **CEA-SMSG**, the Spanish Simulation Society, in La Rioja, Logroño, Spain, July 1 - 5, 2019.

Chairs / Team EUROSIM 2019

Emilio Jiménez, EUROSIM President,
emilio.jimenez@unirioja.es
Juan Ignacio Latorre, *juanignacio.latorre@unavarra.es*

→ *www.eurosim2019.com*

EUROSIM Member Societies

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 400 individual members (including associated), and 90 institutional or industrial members.

→ *www.asim-gi.org* with members' area

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

✉ ASIM – Inst. of Analysis and Scientific Computing
Vienna University of Technology (TU Wien)
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, *thorsten.pawletta@hs-wismar.de*
	A. Körner, *andreas.koerner@tuwien.ac.at*
Secretary	Ch. Deatcu, *christina.deatcu@hs-wismar.de*
	I. Husinsky, *Irmgard.husinsky@tuwien.ac.at*
Treasurer	Anna Mathe, *anna.mathe@tuwien.ac.at*
Membership Affairs	S. Wenzel, *s.wenzel@uni-kassel.de*
	Ch. Deatcu, *christina.deatcu@hs-wismar.de*
	F. Breitenecker, *felix.breitenecker@tuwien.ac.at*
Repr. EUROSIM	F. Breitenecker, *felix.breitenecker@tuwien.ac.at*
	A. Körner, *andreas.koerner@tuwien.ac.at*
Internat. Affairs – GI Contact	O. Rose, *Oliver.Rose@tu-dresden.de*
	N. Popper, *niki.popper@dwh.at*
Editorial Board SNE	T. Pawletta, *thorsten.pawletta@hs-wismar.de*
	Ch. Deatcu, *christina.deatcu@hs-wismar.de*
Web EUROSIM	I. Husinsky, *Irmgard.husinsky@tuwien.ac.at*

Last data update September 2018

ASIM is organising / co-organising the following international conferences:

- **ASIM** Int. Conference 'Simulation in Production and Logistics' – biannual
- **ASIM** 'Symposium Simulation Technique' – biannual
- **MATHMOD** Int. Vienna Conference on Mathmatical Modelling – triennial

Furthermore, **ASIM** is co-sponsor of *WSC* – Winter Simulation Conference, of SCS conferences *SpringSim* and *SummerSim*, and of *I3M* and *Simutech* conference series.

ASIM Working Committees	
GMMS	Methods in Modelling and Simulation Th. Pawletta, *thorsten.pawletta@hs-wismar.de*
SUG	Simulation in Environmental Systems Jochen Wittmann, *wittmann@informatik.uni-hamburg.de*
STS	Simulation of Technical Systems Walter Commerell, *commerell@hs-ulm.de*
SPL	Simulation in Production and Logistics Sigrid Wenzel, *s.wenzel@uni-kassel.de*
EDU	Simulation in Education/Education in Simulation A. Körner, *andreas.koerner@tuwien.ac.at*
BIG DATA	Working Group Data-driven Simulation in Life Sciences; *niki.popper@dwh.at*
WORKING GROUPS	Simulation in Business Administration, in Traffic Systems, for Standardisation, etc.

CEA-SMSG – Spanish Modelling and Simulation Group

CEA is the Spanish Society on Automation and Control and it is the national member of IFAC (International Federation of Automatic Control) in Spain. Since 1968 **CEA-IFAC** looks after the development of the Automation in Spain, in its different issues: automatic control, robotics, *SIMULATION*, etc. The association is divided into national thematic groups, one of which is centered on Modeling, Simulation and Optimization, constituting the **CEA** Spanish Modeling and Simulation Group (**CEA-SMSG**). It looks after the development of the Modelling and Simulation (M&S) in Spain, working basically on all the issues concerning the use of M&S techniques as essential engineering tools for decision-making and optimization.

→ *http://www.ceautomatica.es/grupos/*

→ *emilio.jimenez@unirioja.es*
simulacion@cea-ifac.es

✉ CEA-SMSG / Emilio Jiménez, Department of Electrical Engineering, University of La Rioja, San José de Calasanz 31, 26004 Logroño (La Rioja), SPAIN

CEA - SMSG Officers	
President	Emilio Jiménez, *emilio.jimenez@unirioja.es*
Vice president	Juan Ignacio Latorre, *juanignacio.latorre@unavarra.es*
Repr. EUROSIM	Emilio Jiménez, *emilio.jimenez@unirioja.es*
Edit. Board SNE	Juan Ignacio Latorre, *juanignacio.latorre@unavarra.es*
Web EUROSIM	Mercedes Perez *mercedes.perez@unirioja.es*

Last data update February 2018

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*). **CSSS** main objectives 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*
Scientific Secr.	A. Kavička, *Antonin.Kavicka@upce.cz*
Repr. EuroSim	Miroslav Šnorek, *snorek@fel.cvut.cz*
Edit. Board SNE	Mikuláš Alexík, *alexik@frtk.fri.utc.sk*
Web EuroSim	Petr Peringer, peringer@fit.vutbr.cz

Last data update December 2012

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.DutchBSS.org*

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	M. Mujica Mota, *m.mujica.mota@hva.nl*
Vice president	A. Heemink, *a.w.heemink@its.tudelft.nl*
Treasurer	A. Heemink, *a.w.heemink@its.tudelft.nl*
Secretary	P. M. Scala, p.m.scala@hva.nl
Repr. EuroSim	M. Mujica Mota, *m.mujica.mota@hva.nl*
Edit. SNE/Web	M. Mujica Mota, *m.mujica.mota@hva.nl*

Last data update June 2016

LIOPHANT Simulation

Liophant Simulation is a non-profit association born in order to be a trait-d'union among simulation developers and users; Liophant is devoted to promote and diffuse the simulation techniques and methodologies; the Association promotes exchange of students, sabbatical years, organization of International Conferences, courses and internships focused on M&S applications.

→ *www.liophant.org*

info@liophant.org

✉ LIOPHANT Simulation, c/o Agostino G. Bruzzone,
DIME, University of Genoa, Savona Campus
via Molinero 1, 17100 Savona (SV), Italy

LIOPHANT Officers	
President	A.G. Bruzzone, *agostino@itim.unige.it*
Director	E. Bocca, *enrico.bocca@liophant.org*
Secretary	A. Devoti, *devoti.a@iveco.com*
Treasurer	Marina Massei, *massei@itim.unige.it*
Repr. EuroSim	A.G. Bruzzone, *agostino@itim.unige.it*
Deputy	F. Longo, *f.longo@unical.it*
Edit. Board SNE	F. Longo, *f.longo@unical.it*
Web EuroSim	F. Longo, *f.longo@unical.it*

Last data update June 2016

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.

→ *www.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*
Vice President	Egils Ginters, *egils.ginters@rtu.lv*
Secretary	Artis Teilans, *artis.teilans@rta.lv*
Repr. EuroSim	Egils Ginters, egils.ginters@rtu.lv
Deputy	Artis Teilans, *artis.teilans@rta.lv*
Edit. Board SNE	Juri Tolujew, *Juri.Tolujew@iff.fraunhofer.de*
Web EuroSim	Vitaly Bolshakov, *vitalijs.bolsakovs@rtu.lv*

Last data update June 2019

KA-SIM Kosovo Simulation Society

Kosova Association for Modeling and Simulation (KA-SIM, founded in 2009), is part of Kosova Association of Control, Automation and Systems Engineering (KA-CASE). KA-CASE was registered in 2006 as non Profit Organization and since 2009 is National Member of IFAC – International Federation of Automatic Control. KA-SIM joined EUROSIM as Observer Member in 2011. In 2016, KA-SIM became full member.

KA-SIM has about 50 members, and is organizing the international conference series International Conference in Business, Technology and Innovation, in November, in Durrhes, Albania, and IFAC Simulation Workshops in Pristina.

→ *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

KA-SIM Officers	
President	Edmond Hajrizi, *ehajrizi@ubt-uni.net*
Vice president	Muzafer Shala, *info@ka-sim.com*
Secretary	Lulzim Beqiri, *info@ka-sim.com*
Treasurer	Selman Berisha, *info@ka-sim.com*
Repr. EUROSIM	Edmond Hajrizi, *ehajrizi@ubt-uni.net*
Deputy	Muzafer Shala, *info@ka-sim.com*
Edit. Board SNE	Edmond Hajrizi, *ehajrizi@ubt-uni.net*
Web EUROSIM	Betim Gashi, *info@ka-sim.com*

Last data update December 2016

MIMOS – Italian Modelling and Simulation Association

MIMOS (Movimento Italiano Modellazione e Simulazione – Italian Modelling and Simulation Association) is the Italian association grouping companies, professionals, universities, and research institutions working in the field of modelling, simulation, virtual reality and 3D, with the aim of enhancing the culture of 'virtuality' in Italy, in every application area.

MIMOS became **EUROSIM** Observer Member in 2016 and **EUROSIM** Full Member in September 2018.

→ *www.mimos.it*

roma@mimos.it – info@mimos.it

✉ MIMOS – Movimento Italiano Modellazione e Simulazione; via Ugo Foscolo 4, 10126 Torino – via Laurentina 760, 00143 Roma

MIMOS Officers	
President	Paolo Proietti, *roma@mimos.it*
Secretary	Davide Borra, *segreteria@mimos.it*
Treasurer	Davide Borra, *segreteria@mimos.it*
Repr. EUROSIM	Paolo Proietti, *roma@mimos.it*
Deputy	Agostino Bruzzone, *agostino@itim.unige.it*
Edit. Board SNE	Paolo Proietti, *roma@mimos.it*

Last data update December 2016

NSSM – National Society for Simulation Modelling (Russia)

NSSM - The Russian National Simulation Society (Национальное Общество Имитационного Моделирования – НОИМ) was officially registered in Russian Federation on February 11, 2011. In February 2012 NSS has been accepted as an observer member of **EUROSIM**, and in 2015 NSSM has become full member.

→ *www.simulation.su*

yusupov@iias.spb.su

✉ NSSM / R. M. Yusupov,
St. Petersburg Institute of Informatics and Automation RAS, 199178, St. Petersburg, 14th lin. V.O, 39

NSSM Officers	
President	R. M. Yusupov, *yusupov@iias.spb.su*
Chair Man. Board	A. Plotnikov, *plotnikov@sstc.spb.ru*
Secretary	M. Dolmatov, *dolmatov@simulation.su*
Repr. EUROSIM	R.M. Yusupov, *yusupov@iias.spb.su* Y. Senichenkov, *senyb@dcn.icc.spbstu.ru*
Deputy	B. Sokolov, *sokol@iias.spb.su*
Edit. Board SNE	Y. Senichenkov, *senyb@mail.ru*, *senyb@dcn.icc.spbstu.ru*,

Last data update February 2018

PSCS – Polish Society for Computer Simulation

PSCS was founded in 1993 in Warsaw. PSCS is a scientific, non-profit association of members from universities, research institutes and industry in Poland with common interests in variety of methods of computer simulations and its applications. At present PSCS counts 257 members.

→ *www.eurosim.info*, www.ptsk.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	Tadeusz Nowicki, *Tadeusz.Nowicki@wat.edu.pl*
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	Tadeusz Nowicki, *tadeusz.nowicki@wat.edu.pl*
Edit. Board SNE	Zenon Sosnowski, z.sosnowski@pb.ed.pl
Web EUROSIM	Magdalena Topczewska *m.topczewska@pb.edu.pl*

Last data update December2013

SIMS – Scandinavian Simulation Society

SIMS is the *Scandinavian Simulation Society* with members from the five Nordic countries Denmark, Finland, Iceland, 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 one board member).

SIMS Structure. SIMS is organised as federation of regional societies. There are **FinSim** (Finnish Simulation Forum), **MoSis** (Society for Modelling and Simulation in Sweden), **DKSIM** (Dansk Simuleringsforening) and **NFA** (Norsk Forening for Automatisering).

→ *www.scansims.org*

erik.dahlquist@mdh.se

SIMS / Erik Dahlquist, School of Business, Society and Engineering, Department of Energy, Building and Environment, Mälardalen University, P.O.Box 883, 72123 Västerås, Sweden

SIMS Officers	
President	Erik Dahlquist, *erik.dahlquist@mdh.se*
Vice president	Bernt Lie, *Bernt.Lie@usn.no*
Treasurer	Vadim Engelson, *vadim.engelson@mathcore.com*
Repr. EUROSIM	Erik Dahlquist, *erik.dahlquist@mdh.se*
Edit. Board SNE	Esko Juuso, *esko.juuso@oulu.fi*
Web EUROSIM	Vadim Engelson, *vadim.engelson@mathcore.com*

Last data update February 2018

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 90 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 / Vito Logar, Faculty of Electrical Engineering, University of Ljubljana,
Tržaška 25, 1000 Ljubljana, Slovenia

SLOSIM Officers	
President	Vito Logar, *vito.logar@fe.uni-lj.si*
Vice president	Božidar Šarler, *bozidar.sarler@ung.si*
Secretary	Simon Tomažič, *simon.tomazic@fe.uni-lj.si*
Treasurer	Milan Simčič, *milan.simcic@fe.uni-lj.si*
Repr. EUROSIM	B. Zupančič, *borut.zupancic@fe.uni-lj.si*
Deputy	Vito Logar, *vito.logar@fe.uni-lj.si*
Edit. Board SNE	R. Karba, *rihard.karba@fe.uni-lj.si*
Web EUROSIM	Vito Logar, *vito.logar@fe.uni-lj.si*

Last data update December 2018

UKSIM - United Kingdom Simulation Society

The UK Simulation Society is very active in organizing conferences, meetings and workshops. UKSim holds its annual conference in the March-April period. In recent years the conference has always been held at Emmanuel College, Cambridge. The Asia Modelling and Simulation Section (AMSS) of UKSim holds 4-5 conferences per year including the EMS (European Modelling Symposium), an event mainly aimed at young researchers, organized each year by UKSim in different European cities. Membership of the UK Simulation Society is free to participants of any of our conferences and their co-authors.

→*uksim.info*

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	A. Orsoni, *A.Orsoni@kingston.ac.uk*
Membership chair	G. Jenkins, *glenn.l.jenkins@smu.ac.uk*
Local/Venue chair	Richard Cant, *richard.cant@ntu.ac.uk*
Repr. EUROSIM	A. Orsoni, *A.Orsoni@kingston.ac.uk*
Deputy	G. Jenkins, *glenn.l.jenkins@smu.ac.uk*
Edit. Board SNE	A. Orsoni, *A.Orsoni@kingston.ac.uk*

Last data update March 2016

EUROSIM Observer Members

ROMSIM – Romanian Modelling and Simulation Society

ROMSIM has been founded in 1990 as a non-profit society, devoted to theoretical and applied aspects of modelling and simulation of systems. ROMSIM currently has about 100 members from Romania and Moldavia.

→ *www.eurosim.info/societies/romsim/*

florin_h2004@yahoo.com

✉ ROMSIM / Florin Hartescu,
National Institute for Research in Informatics, Averescu Av. 8 – 10, 011455 Bucharest, Romania

ROMSIM Officers	
President	N. N.
Vice president	Florin Hartescu, *florin_h2004@yahoo.com* Marius Radulescu, *mradulescu.csmro@yahoo.com*
Repr. EUROSIM	Marius Radulescu, *mradulescu.csmro@yahoo.com*
Deputy	Florin Hartescu, *florin_h2004@yahoo.com*
Edit. Board SNE	Constanta Zoe Radulescu, *zoe@ici.ro*
Web EUROSIM	Florin Hartescu, *florin_h2004@yahoo.com*

Last data update June 2019

MEMBER CANDIDATES

Albanian SIMULATION Society

At the Department of Statistics and Applied Informatics, Faculty of Economy, University of Tirana, Prof. Dr. Kozeta Sevrani at present is setting up an Albanian Simulation Society

The society – constitution and bylaws are being worked out – will be involved in different international and local simulation projects, and is engaged in the organisation of the conference series ISTI – Information Systems and Technology. The society intends to become a EUROSIM Observer Member.

→ *www.eurosim.info/societies/albsim/*

kozeta.sevrani@unitir.edu.al

✉ Albanian Simulation Goup, attn. Kozeta Sevrani
University of Tirana, Faculty of Economy
rr. Elbasanit, Tirana 355 Albania

Albanian Simulation Society- Officers (Planned)	
President	Kozeta Sevrani, *kozeta.sevrani@unitir.edu.al*
Repr. EUROSIM	Kozeta Sevrani, *kozeta.sevrani@unitir.edu.al*
Edit. Board SNE	Albana Gorishti, *albana.gorishti@unitir.edu.al* Majlinda Godolja, *majlinda.godolja@feut.edu.al*

Last data update June 2019

Societies in Re-Organisation

The following societies are at present inactive or under re-organisation:

- CROSSIM - *Croatian Society for Simulation Modelling*
- FRANCOSIM – Société Francophone de Simulation
- HSS – Hungarian Simulation Society
- ISCS – Italian Society for Computer Simulation

Association Simulation News

ARGESIM is a non-profit association generally aiming for dissemination of information on system simulation – from research via development to applications of system simulation. **ARGESIM** is closely co-operating with **EUROSIM**, the Federation of European Simulation Societies, and with **ASIM**, the German Simulation Society. **ARGESIM** is an 'outsourced' activity from the *Mathematical Modelling and Simulation Group* of TU Wien, there is also close co-operation with TU Wien (organisationally and personally).

→ *www.argesim.org*

→ *office@argesim.org*

→ ARGESIM/Math. Modelling & Simulation Group,
Inst. of Analysis and Scientific Computing, TU Wien
Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria
Attn. Prof. Dr. Felix Breitenecker

ARGESIM is following its aims and scope by the following activities and projects:

- Publication of the scientific journal **SNE** – *Simulation Notes Europe* (membership journal of **EUROSIM**, the *Federation of European Simulation Societies*) – *www.sne-journal.org*
- Organisation and Publication of the **ARGESIM Benchmarks** for *Modelling Approaches and Simulation Implementations*
- Publication of the series **ARGESIM Reports** for monographs in system simulation, and proceedings of simulation conferences and workshops
- Publication of the special series **FBS Simulation** – Advances in Simulation / Fortschrittsberichte Simulation - monographs in co-operation with ASIM, the German Simulation Society
- Organisation of the Conference Series **MATHMOD Vienna** (triennial, in co-operation with **EUROSIM**, **ASIM**, and **TU Wien**) – www.mathmod.at
- Organisation of Seminars and Summerschools on Simulation
- Administration of **ASIM** (German Simulation Society) and administrative support for **EUROSIM** *www.eurosim.info*
- Support of ERASMUS and CEEPUS activities in system simulation for TU Wien

ARGESIM is a registered non-profit association and a registered publisher: **ARGESIM Publisher Vienna**, root ISBN 978-3-901608-xx-y, root DOI 10.11128/z…zz.zz. Publication is open for **ASIM** and for **EUROSIM Member Societies**.

SNE – Simulation Notes Europe

The scientific journal **SNE** – *Simulation Notes Europe* provides an international, high-quality forum for presentation of new ideas and approaches in simulation – from modelling to experiment analysis, from implementation to verification, from validation to identification, from numerics to visualisation – in context of the simulation process. **SNE** puts special emphasis on the overall view in simulation, and on comparative investigations.
Furthermore, **SNE** welcomes contributions on education in/for/with simulation.

SNE is also the forum for the **ARGESIM Benchmarks** on *Modelling Approaches and Simulation Implementations* publishing benchmarks definitions, solutions, reports and studies – including model sources via web.

→ *www.sne-journal.org,*

→ *office@sne-journal.org*, *eic@sne-journal.org*

→ SNE Editorial Office
ARGESIM/Math. Modelling & Simulation Group,
Inst. of Analysis and Scientific Computing, TU Wien
Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria
EiC Prof. Dr. Felix Breitenecker

SNE, primarily an electronic journal, follows an open access strategy, with free download in basic layout. **SNE** is the official membership journal of **EUROSIM**, the *Federation of European Simulation Societies*. Members of **EUROSIM Societies** are entitled to download **SNE** in high-quality, and to access additional sources of benchmark publications, model sources, etc. On the other hand, **SNE** offers **EUROSIM Societies** a publication forum for post-conference publication of the society's international conferences, and the possibility to compile thematic or event-based **SNE Special Issues**.

Simulationists are invited to submit contributions of any type – *Technical Note*, *Short Note*, *Project Note*, *Educational Note*, *Benchmark Note*, etc. via **SNE**'s website:

EUROSIM 2019

10th EUROSIM Congress on Modelling and Simulation

La Rioja, Logroño, Spain, July 1 – 5, 2019

Programme & Scheduling

The Congress includes three days of work (Tuesday to Thursday) combined with cultural and social activities, a previous day of simulation courses and pre-reception (Monday), a day devoted to cultural and technical activities (Friday), and finally, as an extension, the possibility of a trip (Saturday) to the festivities of San Fermines in Pamplona. All this is detailed in the Programme:

Monday 1st July

Morning: Free courses on continuous and discrete events simulation.
Lunch and coffee breaks
Evening: Dinner and Rioja Wine Tasting Course

Tuesday 2nd July

Morning: Plenary sessions. Reception at the city hall.
Lunch and coffee breaks
Afternoon: Parallel sessions (optional visit to the Wine Museum for those who cannot visit it on Saturday)
Evening: Dinner in the old town

Wednesday 3rd July

Morning: Parallel sessions
Lunch and coffee breaks
Afternoon: Parallel sessions
Evening: Walk for the Santiago's road in the city to the Gala Dinner in a Winery (and visit to the winery)

Thursday 4th July

Morning: Parallel sessions
Lunch and coffee breaks
Afternoon: Parallel sessions
Evening: Guided visit to the old town and Dinner of typical tapas at bars of Laurel Street

Friday 5th July

Technical and Cultural visits, including the Wine Museum, Wineries (such as the famous Marques de Riscal, of architect Frank Gehry), and the well-known monasteries (Suso, Yuso, Cañas, Santa María La Real, San Millán, and Valvanera)

Saturday 6th July

As an extension of the activities, and taking advantage from the vicinity of Pamplona, the well-known city for the festivities of *San Fermines* and the bullfights for its streets during the festivities, immortalized by Hemingway, there will be an excursion to the event of the official beginning of the festivity, with "*el chupinazo*" (rocket announcer)

Congress Team: The Congress is organised by CAE CAE-SMSG, the Spanish simulation society, and Universidad de la Rioja.

Info: Emilio Jiménez, EUROSIM President, emilio.jimenez@unirioja.es
Juan Ignacio Latorre, juanignacio.latorre@unavarra.es

www.eurosim2019.com

www.tuverlag.at
ISBN 978-3-903024-86-1