Education in Modelling and Simulation using ARGESIM Comparisons/Benchmarks with Physical Modelling

Felix Breitenecker[1], Florian Judex[1], Eduard Nigsch[1], Nikolas Popper[2]
[1]Vienna Univ. of Technology, [2]Drahtwarenhandlung – Simulation Services, Vienna; Austria

Corresponding Author:
Florian Judex, Inst. f. Analysis and Scientific Computing
Vienna University of Technology, Austria
Wiedner Hauptstrasse 8-10, 1040 Wien, Austria
Phone: +43 1 58801-11455, email: efelo@fsmat.at

Abstract. ARGESIM started in 1990 the series *Comparison of Simulation Software* in the journal Simulation News Europe (SNE). These software comparisons developed towards benchmarks not only for simulation tools but also for modelling tools and for modelling techniques and modelling approaches.
The solutions allow comparisons of different modelling approaches, of features of simulators, of development of simulators, etc. Furthermore, the solutions – many of them with source codes – may be used as examples in simulation courses, etc. The ARGESIM Comparisons have proven a big success: up to now 291 solutions have been published in SNE, and the comparison models are used worldwide as examples and benchmarks in teaching.
It has turned out, that the ARGESIM Comparisons are a valuable source for demos, exercises, or benchmark studies in education on modelling and simulation. As the comparisons tend towards modelling approaches, they can be used not only in simulation software classes, but also in more or less general classes on modelling in natural sciences, in computer science and computer engineering, etc.
This contribution first gives an overview on comparisons and benchmarks for modelling and simulation, following by a presentation of the ARGESIM Comparisons. Then the extension of the comparisons necessary for education in modelling is discussed: information on modelling procedure and information on background of the application area. This education project is documented by two physical comparisons, C1 'Lithium Cluster Dynamics', dealing with formation and decay of molecule clusters during electron bombardment, and C12 'Collision of Spheres' representing the dynamics and impact of spheres in a row.

1. Evaluation, Benchmarks and Comparisons for Modelling and Simulation

Modelling and simulation has become the third pillar of gaining knowledge, replacing or supporting essentially the classic pillars theory and experiment. Modelling and simulation of a process makes use of a simulation software, and quality of results and investigation time depend dramatically on the choice of the most appropriate simulator. Consequently, bases for this choice or at least serious hints are necessary.

1.1 General Evaluation

Special hints for the choice of a simulator depend substantially on the present time. If a publication with such hints is published, it is already obsolete. Consequently, only general hints can be given, and methods can be shown, how to compare features of simulators and approaches to modelling and simulation.

The following categories for evaluating software are important and give general hints:

- Flexibility to model a variety of systems
- Hierarchical modelling structures, modelling libraries
- Different modelling approaches
- Debugging aids, execution speed, links
- Support for animation and run time graphics
- Appropriate statistical capabilities random number generation, input probability distributions, output analysis, experimental design
- Numerical capabilities: ODE and DAE solvers, Jacobian calculation, eigenvalue analysis, optimisation, etc.
- Symbolical capabilities: non-causal modelling, index reduction, sensivity analysis

Banks ([1]) gives also some warnings, like 'Execution speed is really important, because also development time counts', or 'Beware of advertising claims and demonstrations', or 'Beware of checklists with yes and no as the entries for features'.

1.2 Feature Comparisons and Benchmarking

It is evident, that the choice of a simulator must be based on the knowledge of the features of a simulator for the planned simulation project. It is necessary to get known, how the simulator works with the planned project or with a similar and analogue process. This leads to the necessity to compare simulation software on a more or less standardised basis. In the 1980s, in general two methods for comparing simulation software were developed.

The *Checklist Method* listed desired features, and for each simulator 'yes' or 'no' entries were put in the lists (feature comparisons). There are two major disadvantages: first, very often it is important, how a particular feature is implemented, and not the existence itself; second, such checklists are becoming obsolete very quickly. Nowadays the development cycle for a new release of simulation software is less than two years, so such a checklist must become obsolete in very short time. The checklist method may be seen as formalisation of a list of special hints for choosing a simulator, and, as given before, it is not the best method.

The *Benchmark Method* makes use of so-called benchmarks. The term benchmark may have different meanings; in general a benchmark is a more or less normalised test of the quality of product, of a process, etc. Benchmarking means, measuring and comparing products like bicycles or software, in order to choose the most appropriate product for a certain purpose. In modelling and simulation, benchmarks like PHYSBE, Pilot Ejection, etc. – are well known. They check certain features of simulators, based on fixed models with defined experiments. These benchmarks are complex, so that it takes time, if more simulators have to be benchmarked for a certain purpose.

Until the midst of the nineties, benchmarks for general purpose simulation software were up-to-date. In the last ten years the emphasis of benchmarking has shifted towards special purpose simulators, like wastewater treatment processes, or performance modelling of networks ([2], [3]). Especially in process simulation and in power plant simulation there was a claim for standardised benchmarking. Up to now, benchmarks are not officially standardised (ISO), industrial standards are given e.g. by SPEC benchmarks. Benchmarks may be restricted also only to a certain class of problems or applications, e.g. on hybrid systems. At the web here a lot of sources can be found, which interestingly link to the *ARGESIM Comparisons*.

1.3 Choice by Decision-Support

Another possibility is to standardise and to automatise the choice based on decision support software. From 1998 on, at Brunel University a decision support tool called SimSelect was developed ([4]), that provides a support to the users when selecting simulation software. Following a specification of user's requirements, the system queries a database and finds a simulation package as well as alternatives suitable to the user. The problem with this system is maintenance, and the quality of data provided by distributor of simulation software.

1.4 ARGESIM Comparisons.

At Vienna University of Technology, in 1990 comparisons for simulation software were set up, which try to overcome the disadvantages of checklists on the one side, and the complexity of benchmarks on the other side.

These comparisons, the *ARGESIM Comparisons of Modelling and Simulation Techniques and Tools* are published in the journal *SNE - Simulation News Europe*. These software comparisons developed towards small benchmarks not only for simulation tools but also for modelling tools and for modelling techniques. Furthermore, the solutions – many of them with source codes – may be used as examples in simulation courses, etc. Details are given in the next section.

1.5 Special Benchmarks

Frome time to time special benchmarks appear. They are related to certain events or certain occasions. Elder simulationists may remember the Coffeepot Benchmark, set up by R. Huntsinger on occasion of bad coffee at a simulation conference. Unfortunately this benchmark was not documented officially.

Another special benchmark, the MATHMOD Yo-Yo Benchmark was initiated at the 4th MATHMOD Conference in Vienna. Participants had opportunity to train their skilfulness with a real yo-yo (a promotion gadget). On occasion of 6th MATHMOD Conference Vienna (February 2006) the Yo-Yo Benchmark is published officially. This benchmarks checks hybrid features, state events and mechanical modelling. A definition for the *MATHMOD Yo-Yo Simulation Challenge* is published in SNE 44/45 [7].

2. The ARGESIM Comparisons / Benchmarks on Simulation Tools and Modelling Approaches

ARGESIM started in 1990 the series Comparison of Simulation Software in the journal Simulation News Europe (SNE). These software comparisons developed towards benchmarks not only for simulation tools but also for modelling tools and for modelling techniques and modelling approaches.

The new comparisons *C16 Restaurant Business Dynamics*, *C17 Spatial Dynamics of Epidemic*, *C18 Classical vs. Neural Net Models*, and *C19 Ground Water Flow* address also non-classical modelling techniques, like agent-based simulation, neural nets and cellular automata. They can be analysed by various software systems, not only by simulation systems. Furthermore, they underline the importance of spatial dynamics, coupled with temporal dynamics.

The solutions allow comparisons of different modelling approaches, of features of simulators, of development of simulators, etc. Furthermore, the solutions – many of them with source codes – may be used as examples in simulation courses, etc.

The ARGESIM Comparisons have proven a big success: up to now 291 solutions have been published in SNE, and the comparison models are used worldwide as examples and benchmarks in teaching.

2.1 Development of the Comparisons

ARGESIM, the Working group Simulation at Vienna University of Technology takes care on definition of these comparisons, on publication of the solutions and of evaluation of the solutions. Since 2005, work on a data-based driven evaluation and classification is going on, to be presented in a new ARGESIM web server (end 2006).

The principle idea of the ARGESIM Comparisons is a mixture of a general simple comparison of features within 'yes / no' – tables and the well-known benchmark problems, which are relatively big (like PHYSBE). The ARGESIM Comparisons are based on relatively simple, easily comprehensible processes. Different modelling techniques and their implementation as well as features of modelling and experimentation within simulators, also with respect to application area, are compared.

The comparisons solutions have to consist of two parts. The first part is the description of the model and of the modelling procedure with the simulator used, and the second parts has to present the procedure and the results of three so-called tasks, which are experiments with the model – from simple to complex.

The comparisons started in 1990, and since that time there have taken place new developments in software and algorithms. Consequently also the comparisons developed further on, from comparisons of simulation software towards comparisons of modelling and simulation techniques and tools. This development is based on following facts:

- Nowadays different modelling approaches are offered by simulators – it makes sense to work on different solutions with the same simulator.
- The paradigm of Classes and Objects has changed software engineering dramatically. Also in modelling and simulation OO approaches give better insight into structures. Consequently, OO approaches may give better insight into the modelling procedure – it makes sense to compare classical and OO approaches.
- Hybrid approaches become more and more important; and as simulators offer environments with complex features, hybrid approaches can now be set up easily. Hybrid processes may be tackled by different hybrid modelling structures – it makes sense to compare these approaches, from total hybrid decoupling of models until complete overall models.
- Symbolic computation is an alternative to analysis in the time domain. Nowadays Computer Algebra Systems can analyse also nonlinear systems and can handle complex semi-numerical tasks and pure numerical tasks. Furthermore, they usually offer a very good environment for experimenting with models. It makes sense, to include Computer Algebra Systems, and to specify also tasks with analytical background.
- For modelling and simulation of discrete processes, not only classical discrete simulation systems, based on DEVS, can be used. It makes sense to look also for different or alternative approaches, like Petri nets and Markov chains, and to make use also general statistical tools and environments. Furthermore, in the OO world of Java a lot of libraries for discrete process modelling are available, which start to compete with classical simulators.
- The classic basis of continuous modelling and simulation was analysis and simulation in the time domain, and spatial dynamics was shifted to the world of finite differences, finite volume, and finite elements. Nowadays it is necessary to combine these "different" worlds by "re-considering" the couples temporal and spatial dynamics.
  Consequently, it makes sense to study the different approaches for incorporating also spatial behaviour in the ARGESIM Comparisons.
- Although non-classical modelling techniques did not replace the classical ODEs, DAEs, and PDEs, alternative methods like cellular automata, agent-based approaches, and fuzzy and neural models have become very important for certain classes of problems.
  Consequently it makes sense, to extend the range of the ARGESIM Comparisons also towards these alternative approaches and to compare them with classical ones.
- And last but not least, many simulators have been developed continuously. So it makes sense to solve a comparison from time to time with the new version of a specific simulator, to show the advances and new features of the system.

2.2 Comparison Definitions, Simulation Tasks, and Problems to be Investigated

Up to now 20 comparisons were defined. The following list shortly introduces the comparisons and sketches special problems (SP), which could be observed for a special comparison. Clearly, some defined tasks seem to be simple, but they prove tricky, so that they must cause problems in implementation, etc.

C1 Lithium-Cluster Dynamics, SNE 0 (11/1990),
checks integration of stiff systems, parameter variation, and steady state calculation.
SP: loops with logarithmic increments, correct double – logarithmic plots, steady state calculation.

C2 Flexible Assembly System, SNE 2 (3/1991),
discrete system, compares features for submodel structures, control strategies, and optimisation.
SP: complex control strategies, analytical considerations before modelling very helpful, optimisation avoidable.

C3 Generalised Class-E Amplifier, SNE 2 (7/1991),
simulation of electronic circuits, table functions, eigenvalue analysis, and complex experiments.
SP: use of same model for analytical and numerical analysis, up to now accuracy, table function evaluation vs. piecewise functions.

C4 Dining Philosophers I, SNE 3 (11/1991),
general comparison, involving not only simulation but also analysis e.g. by Petri nets and, etc.
SP: network analysis for deadlocks, simultaneous events, results difficult to compare.

C5 Two State Model, SNE 4 (3/1992),
checks high- accuracy features and state event handling.
SP: analytical approach possible, but ill-conditioned; fully discrete approach possible, accuracy of state event handling.

C6 Emergency Department - Follow-up Treatment, SNE 6 (11/1992),
discrete system, tests features for modelling, concepts of availability, and complex control strategies.
SP: no strict separation of entities and resources, complex routing and priority problems.

C7 Constrained Pendulum, SNE 7 (3/1993),
checks features for hybrid modelling, comparison of models, state events, and boundary value problems.
SP: choice of states, different levels of hybrid approaches.

CP1 Parallel Simulation Techniques, SNE 10, (3/1994),
deals with the benefits of distributed and parallel computation for simulation tasks; three test examples test parallelisation techniques.
SP: results not encouraging wrt parallelisation, very often direct programming necessary.

C8 Canal-and-Lock System, SNE 16 (3/1996),
discrete system, checks features for complex logic control, validation and variance reduction.
SP: complex logic control, analytical considerations necessary; support for advanced statistical analysis (variance reduction methods) often missing.

C9 Fuzzy Control of a Two Tank System, SNE 17, (7/1996), asks for approaches and for implementations of modules for fuzzy control.
SP: support for fuzzy control, two-dimensional calculations for control surface, pure discrete approach possible.

C10 Dining Philosophers II, SNE 18 (11/1996),
reviews discrete simulators with respect to concurrent access to resources and with deadlocks.
SP: discrete random variables, simultaneous events, deadlock recognition.

C11 SCARA Robot, SNE 22 (3/1998),
deals with implicit and hybrid systems with state events.
SP: implicit model, different approaches for collision event and action.

C12 Collision of Spheres, SNE 27 (11/ 1999),
allows numerical or analytical analysis as well as continuous or discrete approaches
SP: broad variety of approaches (numerical - continuous, numerical – discrete, numerical – analytical, analytical – symbolic), collision limit.

C13 Crane Crab with Embedded Control, SNE 31 (3/2001), revised SNE 35/36 (11/2002),
checks techniques and features for embedded digital control with sensors and with DAE-systems.
SP: implicit model, discrete control coupled with sensor diagnosis, complex experiments.

C14 Supply Chain, SNE 32/33 (11/2001), SNE 34 (7/2002),
addresses discrete simulators - features for supply chain systems (messages, strategies).
SP: distinction between material flow and order flow, distance-dependent control strategies.

C15 Clearance Identification, SNE 35/36 (11/2002),
checks identification features (based on measured data) and influences of noise.
SP: identification algorithms, short-term input functions (Dirac-like), support of statistics.

C16 Restaurant Business Dynamics, SNE 40 (5/2004),
addresses agent-based simulation as well DEVS approach and classical programming
SP: renaissance of activity scanning, coordination of run samples, optimisation

SNE	COMPARISON																				
	Sum	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**CP**	**C8**	**C9**	**C 10**	**C 11**	**C 12**	**C 13**	**C 14**	**C 15**	**C 16**	**C 17**	**C 18**	**C 19**
0		D																			
1	5	5	D																		
2	8	4	4	D																	
3	10	4	3	3	D																
4	13	1	5	5	2	D															
5	8	4	-	1	1	2															
6	5	-	2	-	2	1	D														
7	7	1	2	1	2	-	1	D													
8	5	-	1	-	-	-	1	3													
9	5	-	-	-	-	-	2	3													
10	7	1	2	-	-	-	1	2	D/1												
11	8	2	2	1	-	1	-	-	2												
12	7	1	-	1	-	-	-	2	3												
13	4	-	-	-	-	-	-	3	1												
14	6	3	-	1	-	-	-	2	-												
15	2	-	-	1	-	1	-	-	-												
16	3	1	-	-	-	-	-	1	-	D/1											
17	6	-	-	1	-	1	-	1	1	1	D/1										
18	5	-	-	-	-	-	-	2	2	-	-	D/1									
19	6	-	-	-	-	-	-	-	1	1	1	3									
20	5	-	-	-	-	-	-	1	-	1	1	2									
21	10	-	1	-	1	-	1	5	-	-	-	2									
22	10	1	-	1	-	1	-	5	-	1	-	-	D/1								
23	5	-	2	-	-	-	-	-	-	-	2	-	1								
24	7	1	-	-	1	-	-	2	-	-	-	1	2								
25	7	-	1	-	-	1	-	-	-	-	3	1	1								
26	11	2	1	2	1	1	-	1	-	-	1	1	1								
27	5	-	-	1	-	-	-	-	-	-	2	1	-	D/1							
28	7	-	-	2	-	1	-	-	-	-	2	-	-	2							
29/30	11	1	1	1	1	-	3	-	-	-	-	1	-	3							
31	6	-	-	-	-	-	-	1	-	-	-	-	1	3	D/1						
32/33	10	-	1	-	-	-	-	1	-	-	2	1	-	4	-	D/1					
34	5	1	1	-	-	-	1	-	-	-	-	1	-	1	-	-					
35/36	13	2	-	-	1	2	-	3	-	-	-	1	-	-	2	1	D/1				
37	11	-	1	-	-	1	-	2	-	-		2	-	-	2	2	1				
38/39	14	1	2	1	1	1	3	-	1	-	-	1	1	-	1	-	1				
40	9	-	1	2	-	1	1	-	-	-	-	-	1	-	-	1	1	D/1			
41/42	9	1	-	-	-	-	-	-	-	1	-	1	-	1	-	1	1	2	D/1		
43	8	-	-	-	1	-	-	1	-	1		1	-	-	1	-	1	1	-	D/1	
44/45	8	1	-	-	-	1	1	-	-	-	-	-	-	1	-	-	-	2	-	1	D/1
Total	**291**	**38**	**33**	**25**	**14**	**16**	**15**	**41**	**12**	**7**	**15**	**21**	**9**	**16**	**7**	**6**	**6**	**6**	**1**	**2**	**1**
	Sum	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**CP**	**C8**	**C9**	**C 10**	**C 11**	**C 12**	**C 13**	**C 14**	**C 15**	**C 16**	**C 17**	**C 18**	**C 19**

Table 1: Definitions and solutions of ARGESIM Comparions in SNE - per issue, per comparison, summed up for comparisons and issue, and summed up in total

C17 Spatial Dynamics of Epidemic, SNE 41/42 (12/204),
analyses temporal and spatial behaviour of the process by cellular automata models.
SP: proper features for cellular automata in simulation systems, comparison of spatial/temporal results with pure temporal results.

C18 Neural Networks vs. Transfer Functions, SNE 43, (7/2005),
compares transfer function modelling and neural net modelling for given data of a nonlinear process.
SP: proper features for neural net modelling in the simulation system, combination of transfer functions with neural nets for parameter tuning.

C19 Ground Water Flow, SNE 44/45, (12/2005),
studies the flow of contamination in the ground water in 2D-space and time, allowing different modelling approaches for the spatial behaviour (numerical PDE solution, discretisation to ODEs, cellular automata, etc.
SP: features for description of spatial dynamics, combination of spatial/temporal behaviour with temporal behaviour of control inputs.

Table 1 shows definitions and solutions of *ARGESIM Comparisons* in *SNE* - per issue, per comparison, summed up for comparisons and issue, and summed up in total

2.3 Solutions of the Comparisons

Not only readers of *SNE*, but also each simulationist is invited to participate in these comparisons by providing a "solution" with the simulator under investigation.

A solution should consist of:

i. a short description of the simulator,
ii. description of modelling technique,
iii. model description,
iv. results of the three tasks,
v. and additionally we ask for model sources

The printed solution should fit into one page of *SNE* – templates are found at our web page. Solutions sent in are reviewed. Table 1 shows a summary of comparison solutions.

A little bit of statistics:

- 20 comparisons
- 40 SNE issues
- 291 comparison solutions
- 7.3 solutions / SNE issue
- most popular comparisons:
  C7 – 14.1%, C1 – 13.1%, C2 – 11.3%

3. ARGESIM Comparisons as Education Tool

It has turned out, that the *ARGESIM Comparisons* are a valuable source for demos, exercises, or benchmark studies in education on modelling and simulation [5], [6]. As the comparisons tend towards modelling approaches, they can be used not only in simulation software classes, but also in more or less general classes on modelling in natural sciences, in computer science and computer engineering, etc.

Up to now, the model descriptions for all comparisons were given, as ODE, DAE, DEVS, or in another form. For education it is necessary, to study also the analytical modelling procedure, the derivation of the model, and the background of the laws which govern the model. Within a project for master theses and PhD theses, ARGESIM will extend the comparison definitions by information on modelling procedure, on the physical background, etc.

In order to make an *ARGESIM Comparison* a self-contained part of a lecture, each comparison should consist of

i. Model description and derivation
ii. Application area and background
iii. Comparison definition
iv. Various solutions
v. Various implemented models (sources)

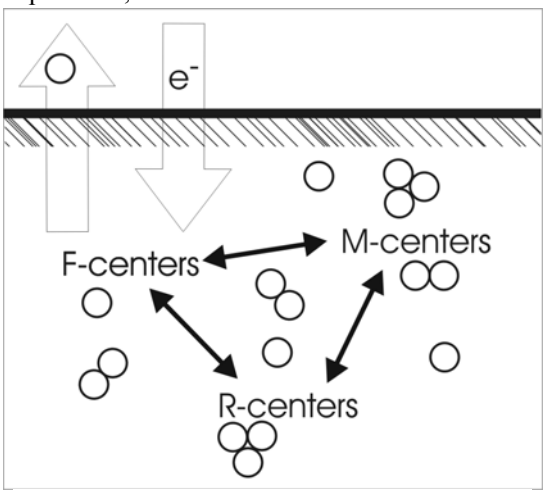
SNE starts with this completion of the comparisons in *SNE* Issue 46 [8]: a contribution on physical background and laws for modelling for Comparison 1 Lithium Cluster Dynamics", is in preparation. The following sections present modelling and background of two physical comparisons.

4. Comparison C1 – Lithium Cluster Dynamics - Modelling and Simulations of Desorption Kinetics

This comparison deals with the desorption kinetics of Li atoms from LiF under electron bombardment. The comparison itself requires the time domain solution of the process using a model with three 'centers', performing three tasks: i) efficient solution of the stiff ODEs (comparisons of algorithms), ii) logarithmic parameter variation and double-logarithmic plot, and iii) calculation of steady states.

4.1 Physical Modelling

When a LiF[1] surface with a temperature of approx. 700 K is being irradiated by an electron beam emission of lithium atoms can be measured. This typical behaviour of alkali halides is well known and due to their simple crystal structure this process is extensively studied. The specific dynamics depend on the temperature of the material, but at temperatures above 500 K the rate of evaporation from the surface is not the limiting factor in the experiment, so the effect of an electron beam can be studied in more detail.



When electrons hit the surface with sufficient energy, so-called *F*-centers (a term from solid state physics) are formed in the crystal. Those can either immediately cause desorption of a lithium atom, or they agglomerate near the surface of the material and form aggregates of various sizes, called $F^{(i)}$-centers where $I = 1, \ldots, n$ denotes the size. These break down after a characteristic time into $F^{(1)}$ -centers (short: *F*-centers) again, which reach the surface and lead to emission of lithium atoms even after the electron beam has been turned off. This so-called *delayed emission* is depicted in figure 1.

Experiments described in [9] show that during electron bombardment the Li emission rate is constant but drops significantly in a very short time (< 20 ms) when bombardment stops. It follows that during this short time interval one can still observe a delayed emission of Li atoms.

In order to build a model of the desorption dynamics some simplifications were made in [9]. These are justified by the small penetration depth of the electrons and the high temperature:

- The diffusion time of *F* -centers to the surface is very small and can be neglected.
- Every *F* -center reaching the surface causes desorption.
- There is no spatial dependence of the F^i -center concentration, i.e. centers are uniformly distributed.
- Growth and decay of a center is only due to absorption or evaporation of an $F^{(1)}$ -center, i.e. $F^{(i)} + F^{(i)} \leftrightarrow F^{(i+1)}$

The following system of differential equations is found to describe the dynamics of the $F^{(i)}$ -centers accordingly to the assumptions made:

$$\frac{dF^{(1)}}{dt} = P - aF^{(1)} + \sum_{i=3}^{n} l_i F^{(i)} - \sum_{h=2}^{n-1} k_h F^{(1)} F^{(h)} + 2l_2 F^{(2)} - 2k_1 (F^{(1)})^2 \tag{1}$$

$$\frac{dF^{(m)}}{dt} = l_{m+1} F^{(m+1)} - k_m F^{(1)} F^{(m)} - l_m F^{(m)} + k_{m-1} F^{(1)} F^{(m-1)}, \; m = 2 \ldots n-1 \tag{2}$$

$$\frac{dF^{(n)}}{dt} = -l_n F^{(n)} + k_{n-1} F^{(1)} F^{(n-1)} \tag{3}$$

In (2), *m* ranges from 2 to *n*-1; *P* denotes the rate by which $F^{(1)}$ -centers are formed, *a* the rate by which they get emitted from the surface. The l_i and k_h are constants describing the rate at which $F^{(i)}$ -centers decay into $F^{(1)}$-centers and $F^{(h)}$-centers combine with $F^{(1)}$-centers to form $F^{(h+1)}$-centers, respectively. Their value (taken from [8]) results from fitting the model to physical measurements the details of which are not of importance here; the l_i and k_h for centers of size up to $n = 6$ are:

$$a = 700,\ l_2 = 17,\ l_3 = 8,\ l_4 = 7,\ l_5 = 5,\ l_6 = 2.5,\ k_1 = 2,\ k_2 = 2.5,\ k_3 = 2.5,\ k_4 = 2.7,\ k_5 = 5.5$$

The equations (1) - (3) are made up of terms describing how each kind of center is formed by other centers and the electron beam and breaks down into smaller centers or gets desorbed. They represent a balance, with no centers entering or leaving the system except by electron bombardment (*P*) and atom absorption ($-aF^{(1)}$).

[1] Lithum (Li) and fluoride (F) form a crystalline salt called an *alkali halide.*

A detailed description of all appearing terms shows how the centers interact:

- P … the rate at which $F^{(1)}$-centers are formed; the electron beam doesn't cause formation of higher-order centers.
- $-aF^{(1)}$) … only $F^{(1)}$-centers get emitted from the surface with a rate proportional to their number; a is the according rate constant.
- $\sum_{i=2}^{n} l_i F^{(i)}$ in (1), $l_{i+1}F^{(i+1)}$ in (1), (2) for i =1, ..., n-1 $\leftrightarrow -l_i F^{(i)}$ for i =2, ..., n in (2), (3) …
  a $F^{(i)}$-center ($i \in \{2,\ldots,n\}$) breaks down into an $F^{(i-1)}$-center and a $F^{(1)}$-center with rate l_i.
- $-\sum_{h=1}^{n-1} k_h F^{(1)} F^{(h)}$ in (1), $k_i F^{(1)} F^{(i)}$ in (1), (2) for $i \in \{1,\ldots,n-1\}$
  $\leftrightarrow$ $k_{i-1} F^{(1)} F^{(i-1)}$ in (2), (3) for I = 2, …, 5
  … $F^{(i)}$-center combines with a $F^{(1)}$-center to form a $F^{(i+1)}$-center with rate k_i.

This model describing the dynamics of the $F^{(i)}$-centers is discussed in the following sections. At this point it should be noted that the above system is *stiff*: as will be examined later, stiff solvers perform better than non-stiff ones, and the stiffness index (ratio of largest to smallest eigenvalue) is high. Most of the results were obtained by the use of Mathematica.

4.2 Comparing Different Numbers of Centers

A major question is how many centers should be used. Using more centers of higher order results also in more parameters to be identified. A comparison of $F^{(1)}$-centers depending on the number of higher order centers used may be of help.
This behaviour for different values of n is depicted in Figure 2. We examined the model (1)-(3) with the given rate constants for n = 3, 4, 5, and 6. It can be seen that the higher n is, the more the differential equations describe the delayed emission behaviour which was observed in experiments ([9]).
In constructing this graph, we started with $F^{(i)}(0) = 0$, for all i, simulated bombardment with P = 10000 for 10 seconds and turned off the electron beam at t = 10.

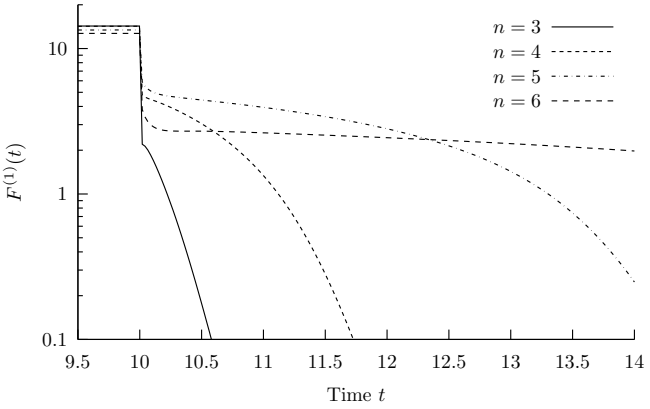


Figure 2: $F^{(1)}$(t) for different number n of centers higher order.

4.3 Model with 3 Centers – Comparison C1

Three centers (n = 3) are sufficient to get interesting results that can be validated by experimental data (see [9]). The variables and constants are renamed in order to be consistent with the definition of the *Comparison C1*: $F^{(1)} \rightarrow f$, $F^{(2)} \rightarrow m$, $F^{(3)} \rightarrow r$; $a \rightarrow l_f$, $d_3 \rightarrow d_r$, $d_2 \rightarrow d_m$, $k_1 \rightarrow k_f$, $k_2 \rightarrow k_r$. The system can then be rewritten as follows:

$$\begin{aligned} \frac{df}{dt} &= d_r r + 2d_m m - k_r mf - 2k_f f^2 - l_f f + p \\ \frac{dm}{dt} &= d_r r - d_m m + k_f f^2 - k_r mf \\ \frac{dr}{dt} &= -d_r r + k_r mf \end{aligned} \tag{4}$$

The parameters used are:

- Constants: k_r =1, k_f =0.1, l_f = 1000, d_r = 0.1, d_m = 1
- The initial values are those taken after constant electron bombardment with P = 10000 for approximately 10 seconds: $f(0)$ = 9.975, $m(0)$ = 1.674, $r(0)$ = 84.99
- Time Interval: $t \in [0, 10]$

A plot showing a typical curve describing the concentration of F -centers in time can be seen in Figure 3, with constant bombardment (P = 10000) during $t \in$ [-5, 0] and no bombardment (P = 0) afterwards, during $t \in$ [0, 10]. When the electron beam is turned off (t = 0) an immediate decay is observed.
Logarithmic plots (Figure 4 and 5) reveal more of the dynamics: there is a local minimum shortly after the beam is turned off (t = 0) followed by a local maximum shortly afterwards, when the decaying higher-order centers cause desorption of further lithium atoms. The position of the extrema depends on the desorption rate a and the maximum size n of aggregates considered.

In order to compare solver performance and to calculate steady states the model has been implemented in MATLAB using the SIMULINK package. The model diagram can be seen in Figure 6.

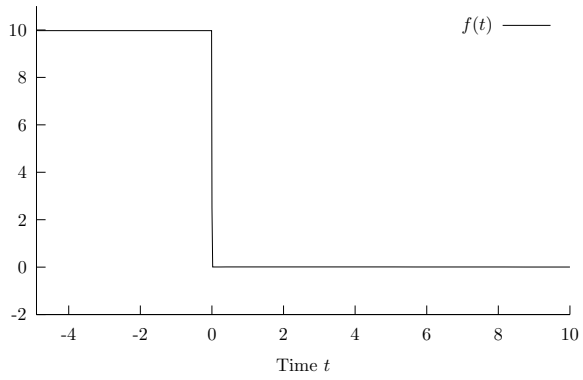


Figure 3: A typical plot of $f(t)$.

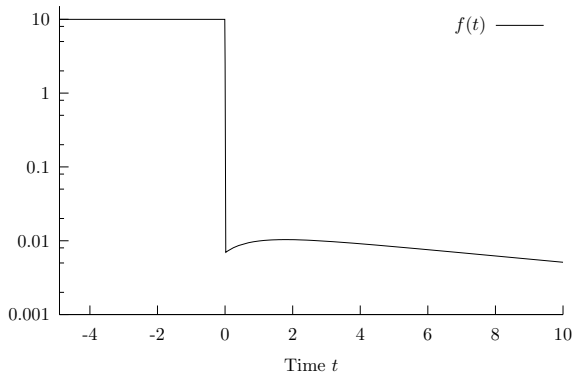


Figure 4: A typical plot of $f(t)$, half logarithmic.

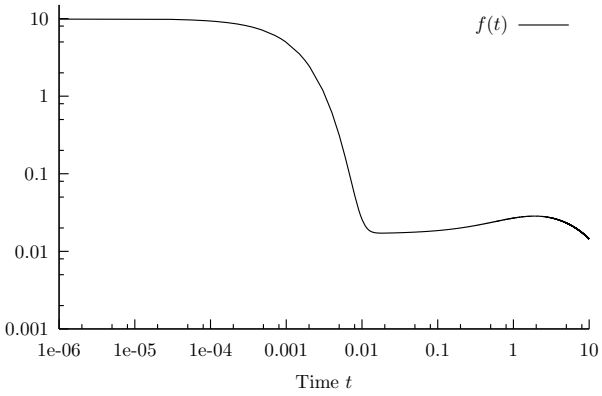


Figure 5: A typical plot of $f(t)$, logarithmic.

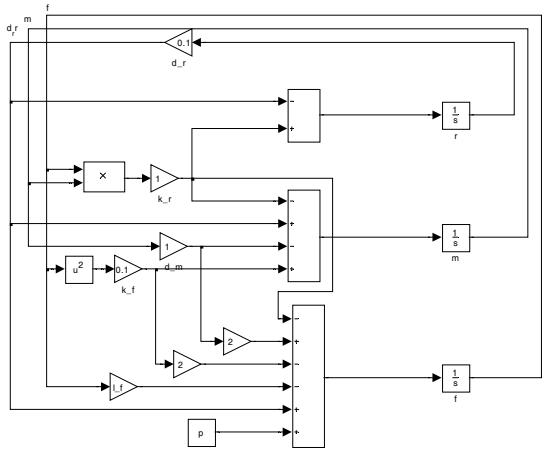


Figure 6: The SIMULINK model for n=3.

Solver performance – Comparison C1 Task a.

The computing time of various algorithms for solving the differential equation system for solving for $t \in [0, 10]$ in Matlab are compared in Table 2. The results show that stiff solvers perform better than non-stiff ones.

Algorithm	Time [sec.]
ode45 (Dormand-Prince)	0.0688
ode23 (Bogacki-Shampine)	0.0627
ode113 (Adams)	0.1250
ode15s (stiff/NDF)	0.0200
ode23s (stiff/Mod. Rosenbrock)	0.0203
ode23t (mod. Stiff/Trapezoidal)	0.0200
ode23tb (stiff/TR-BDF2)	0.0200

Table 2: Performance of MATLAB ODE solvers.

Variation of l_f – Comparison C1 Task b.

The desorption parameter l_f is varied through the interval 10^2 to 10^4 to using logarithmic steps while the value $f(t)$, i.e. the number of centers of size 1, is displayed in a plot which is logarithmic on both scales (Figure 7). It can be seen that after an initial desorption the speed of which depends on lf, the decay of centers of greater size entails delayed emission.

The MATLAB commands used to produce this plot are:

```
for i=1:n
    a=10^(2+1*(i-1)/(n-1));
    sim('c1c', [0, 10]);
    fc=xout.signals(3).values;
    ti=xout.time;
    loglog(ti, fc, '-');
    if(i==1) hold on;end
end
```

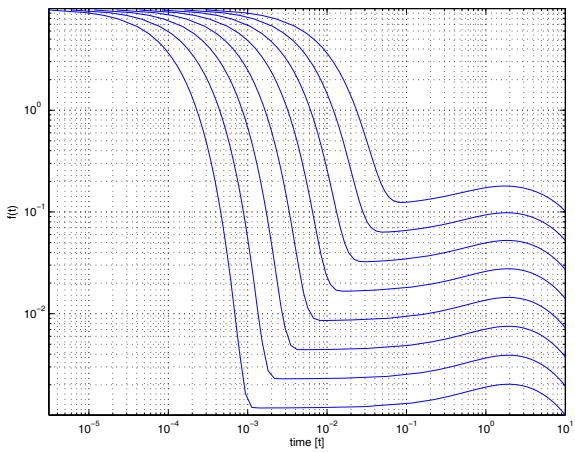


Figure 7: $f(t)$, for varying l_f, using ode23t.

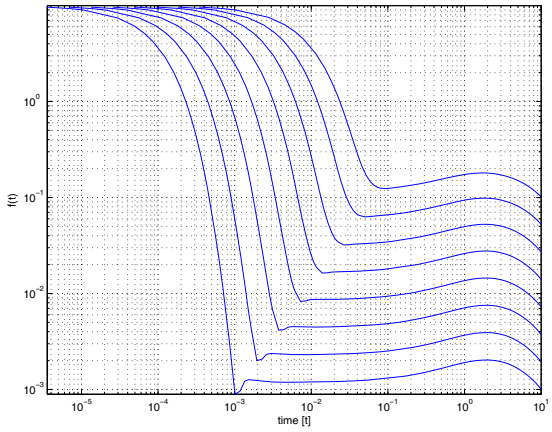


Figure 8: $f(t)$ for varying l_f, using ode15s.

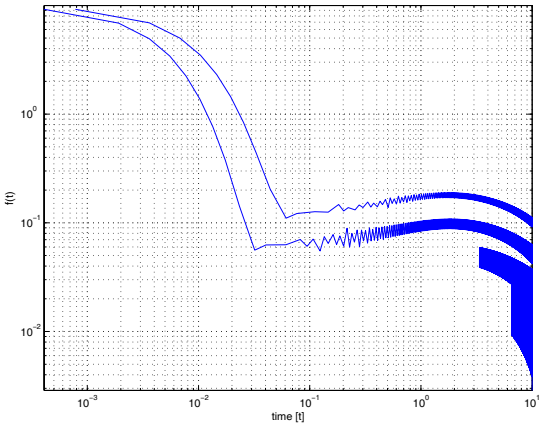


Figure 9: $f(t)$ for varying l_f, using ode23.

In Figure 7 the solver ode23t was used. Although both ode23t and ode15s are stiff solvers, they differ in that ode15s produces an artefact in the plot at the local minimum which can be seen in Figure 8. Non-stiff solvers fail badly; see Figure 9 for an example using ode23.
Examining the artefact of ode15s in more detail for the case $l_f = 1000$ (the corresponding plot of $f(t)$ can be seen in Figures 3-5) we have to look at the Jacobian matrix $J(t)$ and their eigenvalues λ_i, i=1, 2, 3, in order to see if the system is stiff:

$$J(t) = \begin{pmatrix} -1000 - 0.4f(t) - m(t) & 2 - f(t) & 0.1 \\ 0.2f(t) - m(t) & -1 - f(t) & 0.1 \\ m(t) & f(t) & -0.1 \end{pmatrix}, \quad S_I(t) = \frac{\max_i |\lambda_i|}{\min_i |\lambda_i|}$$

The eigenvalues λ_i of $J(t)$, evaluated using the numerical solution obtained previously, are all negative, which means that the system is stable (Figure 10). The ratio $S_I(t)$ between the largest and smallest eigenvalues, called the stiffness index, is plotted in Figure 11 and found to be sufficiently high to call the system stiff.
The inverted spikes in Figure 8 can be explained by the fact that the solver uses a stepsize that is too small to follow the dynamics of the system - the width of the spikes is about $5 \cdot 10^{-4}$. They disappear, however, if the default relative error tolerance of 10^{-3} is set to a lower value of the order 10^{-6}. Therefore, it is suggested that the default error tolerance in Matlab/SIMULINK is not sufficient to follow the quick dynamics of the system around $t = 0.01$.
It is noteworthy that the numerical difficulties in this model do not arise - as could be expected - in the beginning where the stiffness index is largest but just before $t = 0.01$ where the stiffness index smaller. The explanation for this is that the values of all the $F^{(i)}$ get very small (about 10^{-2}) in the latter region and, as the system is ill-conditioned, numerical difficulties arise.

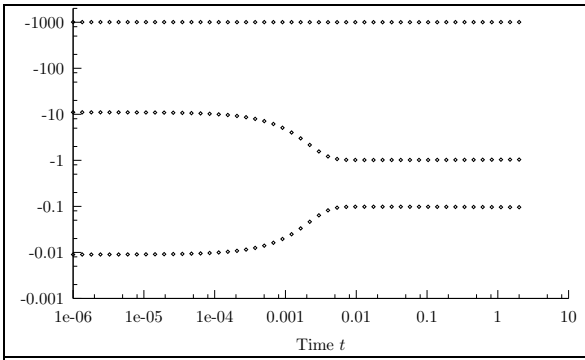


Figure 10: Eigenvalues of $J(t)$.

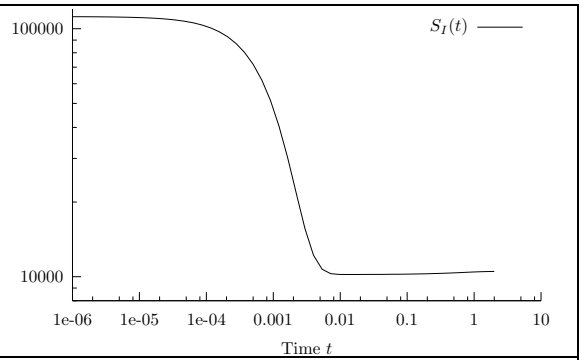


Figure 11: Stiffness index $S_I(t)$.

Calculation of Steady States – Comparison C1 Task c.

The steady states of the system of differential equations (4) during constant bombardment ($P = 10^4$) and without bombardment ($P = 0$), respectively, can be computed both analytically and numerically.
For an analytical solution, one has to solve for

$$\frac{df}{dt} = 0, \quad \frac{dm}{dt} = 0, \quad \frac{dr}{dt} = 0 \qquad \rightarrow \qquad f(t) = \frac{p}{l_f}, \quad m(t) = \frac{k_f p^2}{d_m l_f^2}, \quad r(t) = \frac{k_r k_f p^3}{d_r d_m l_f^3}$$

The analytic calculations give results shown in Table 3. Using any algorithm for finding roots of (3), for example Newton's algorithm, a numerical solution can be obtained; in MATLAB there is the command `trim`, which uses a more efficient so-called *sequential quadratic programming algorithm*[2] to find steady states. We see that the obtained solution is identical except for $F^{(3)}$ at $P = 0$, but the difference is very small.

P	$F^{(1)}$	$F^{(2)}$	$F^{(3)}$
0	0	0	0
10000	10	10	1000

Table 3: Steady states, analytical solution.

P	$F^{(1)}$	$F^{(2)}$	$F^{(3)}$
0	0	0	$-2\cdot10^{-12}$
10000	10	10	1000

Table 4: Steady states, numerical solution.

4.4 Analytical methods

The structure of the model equation allow to use analytical methods without big difficulties. The methods applied are Taylor series expansion for $f(t)$ at appropriate time instants and linearisation of the system (4) around certain time instants.

Taylor series expansion.

The Taylor series of degree k for $f(t)$ at t_0 is given by $T_k(f,t_0)(t) = \sum_{l=0}^{k} \frac{f^{(l)}(t_0)}{l!}(t-t_0)^l$.

The differential equation system (4) can be used to recursively calculate the derivatives of f at $t = 0$ or any other point t where the values of $f(t_0)$, $m(t_0)$, $r(t_0)$ are given numerically.
For example, $T_{10}(f, 0)(t)$ is given by

$$\begin{aligned} T_{10}(f,0)(t) = {} & 9.975 - 9999.75\cdot t + 5.0282\cdot10^{6}\cdot t^{2} - 1.69222\cdot10^{9}\cdot t^{3} \\ & + 4.30468\cdot10^{11}\cdot t^{4} - 8.89326\cdot10^{13}\cdot t^{5} + 1.57505\cdot10^{16}\cdot t^{6} \\ & - 2.51393\cdot10^{18}\cdot t^{7} + 3.8043\cdot10^{20}\cdot t^{8} - 5.7126\cdot10^{22}\cdot t^{9} \\ & + 8.7323\cdot10^{24}\cdot t^{10} \end{aligned}$$

As can be seen in Figure 12, the Taylor approximation around $t_0 = 0$ of a given degree is only valid until a certain time (about $t_0 = 0.01$), after which it rapidly diverges to $\pm\infty$, even at higher degrees.

Expanding about $t_0 = 0.01$ (Figure 13) gives nearly the same figure. Using $t_0 = 1$ as point of expansion (Figure 14) gives bad approximations that diverge to $\pm\infty$ very quickly.

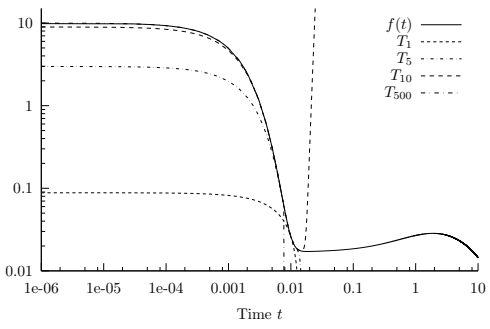


Figure 12: Taylor approximations, $t_0 = 0$.

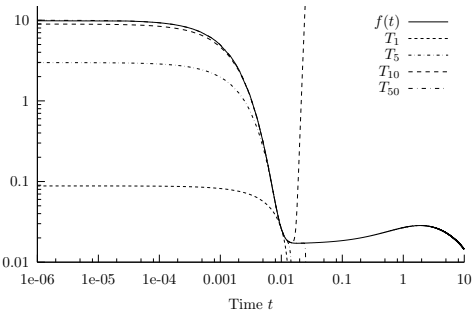


Figure 12: Taylor approximations, $t_0 = 0.01$.

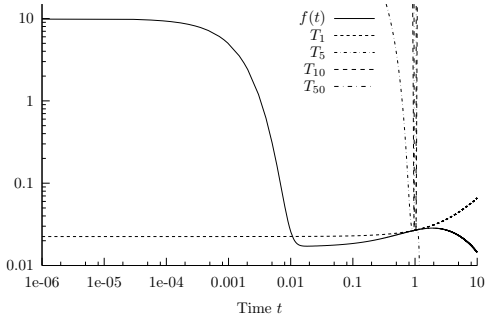


Figure 12: Taylor approximations, $t_0 = 1$.

[2] see http://www.mathworks.com/access/helpdesk/help/toolbox/optim/ug/f26622.html

Linearisation

The model can be linearised and then analytically solved using linear approximations of the right sides of the system (4):

$$\begin{pmatrix} \frac{df}{dt} \\ \frac{dm}{dt} \\ \frac{dr}{dt} \end{pmatrix} = \begin{pmatrix} f_1(f(t),m(t),r(t)) \\ f_2(f(t),m(t),r(t)) \\ f_3(f(t),m(t),r(t)) \end{pmatrix} \approx \begin{pmatrix} f_1(f(t_0),m(t_0),r(t_0)) \\ f_2(f(t_0),m(t_0),r(t_0)) \\ f_3(f(t_0),m(t_0),r(t_0)) \end{pmatrix} + J(t_0) \begin{pmatrix} f(t) \\ m(t) \\ r(t) \end{pmatrix}$$

The Jacobian $J(t)$ is given by

$$J(t) = \begin{pmatrix} -k_r m(t) - 4k_f f(t) - l_f & 2d_m - k_r f(t) & d_r \\ 2k_f f(t) - k_r m(t) & -d_m - k_r f(t) & d_r \\ k_r m(t) & k_r f(t) & -d_r \end{pmatrix}$$

and, evaluated at $t = t_0$, gives us the linear system

$$\begin{pmatrix} \frac{df}{dt} \\ \frac{dm}{dt} \\ \frac{dr}{dt} \end{pmatrix} = \begin{pmatrix} f_1(f(t_0),m(t_0),r(t_0)) \\ f_2(f(t_0),m(t_0),r(t_0)) \\ f_3(f(t_0),m(t_0),r(t_0)) \end{pmatrix} + \begin{pmatrix} -1000 - 0.4f(t_0) - m(t_0) & 2 - f(t_0) & 0.1 \\ 0.2f(t_0) - m(t_0) & -1 - f(t_0) & 0.1 \\ m(t_0) & f(t_0) & -0.1 \end{pmatrix} \begin{pmatrix} f(t) \\ m(t) \\ r(t) \end{pmatrix}$$

This system is a linearisation of the original system and therefore resembles its behaviour in the neighbourhood of $t = t_0$.

The exactness of the resulting approximation was examined at various points of the scale, $t_0 = 0$, $t_0 = 0.01$, $t_0 = 0.04$, $t_0 = 0.05$, and $t_0 = 1$.
Apparently, it is only good for the first part of the numerical solution. Similarly to Taylor expansion, linearising after $t_0 = 0.05$ yields solutions that diverge very quickly.

Surprisingly, linearising at $t_0 = 1$ results in a solution that is valid also only around $t_0 = 0.05$ (Figure 15).

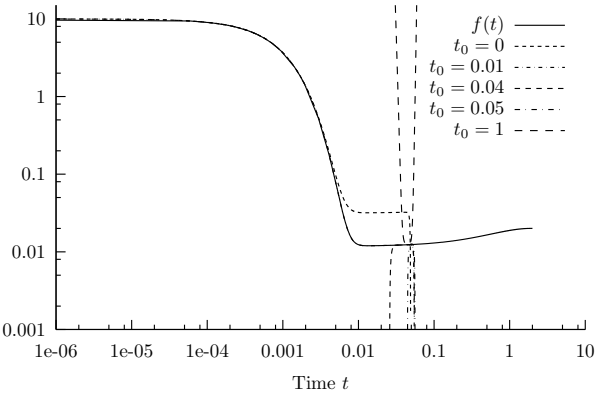


Figure 15: Linearisation at different time instants.

4.5 Model with Five Centers

As rate equations can be given for arbitrary n (the size of the largest centers considered), we will now look at the results for $n = 5$ and compare them to those for $n = 3$ discussed earlier. The full differential equation system and rate constants for $n = 5$ are given below. The initial values $F^{(i)}(0)$ are taken from a simulation that starts at $t = -10$ with $F^{(i)}(-10) = 0$ for all i and runs 10 seconds with $P = 10000$:

$$\begin{aligned}
\frac{dF^{(1)}}{dt} &= P - aF^{(1)} + \sum_{i=3}^{n} l_i F^{(i)} - \sum_{h=2}^{n-1} k_h F^{(1)} F^{(h)} + 2l_2 F^{(2)} - 2k_1 (F^{(1)})^2 \\
\frac{dF^{(2)}}{dt} &= l_3 F^{(3)} - k_2 F^{(1)} F^{(2)} - l_2 F^{(2)} + k_1 (F^{(1)})^2 \\
\frac{dF^{(3)}}{dt} &= l_4 F^{(4)} - k_3 F^{(1)} F^{(3)} - l_3 F^{(3)} + k_2 F^{(1)} F^{(2)} \\
\frac{dF^{(4)}}{dt} &= l_5 F^{(5)} - k_4 F^{(1)} F^{(4)} - l_4 F^{(4)} + k_3 F^{(1)} F^{(3)} \\
\frac{dF^{(5)}}{dt} &= -l_5 F^{(5)} + k_4 F^{(1)} F^{(4)}
\end{aligned} \tag{5}$$

$a = 700,\ l_2 = 17,\ l_3 = 8,\ l_4 = 7,\ l_5 = 5,$
$k_1 = 2,\ k_2 = 2.5, k_3 = 2.5,\ k_4 = 2.7$

$F^{(1)}(0) = 13.4267,\ F^{(2)}(0) = 13.8909,\ F^{(3)}(0) = 42.7877,$
$F^{(4)}(0) = 187.849,\ F^{(5)}(0) = 1340.56$

The SIMULINK model for the case of 5 centers is very similar to the case of 3 centers. The computing time of the different MATLAB solvers for solving for $t \in [0, 10]$ are compared in Table 5, where stiff solvers perform slightly better, again.
A plot showing what happens when a varies from 100 to 1000 in logarithmic steps can be seen in Figure 16 and Figure 17 for the ode23t (ode15s looks the same) and ode23 solvers, respectively. Again, the non-stiff solver ode23 fails blatantly when the $F^{(i)}(t)$ get very small.

Algorithm	Time [sec.]
ode45 (Dormand-Prince)	0.2874
ode23 (Bogacki-Shampine)	0.2453
ode113 (Adams)	0.3015
ode15s (stiff/NDF)	0.0601
ode23s (stiff/Mod. Rosenbrock)	0.2784
ode23t (mod. Stiff/Trapezoidal)	0.1201
ode23tb (stiff/TR-BDF2)	0.1573

Table 5. Computing time for different ODE solvers

Using the Jacobian matrix one can again calculate the eigenvalues and the stiffness index (Figures 18 and 19). There are 5 eigenvalues that give a ratio similar to the case of $n = 3$ up to $t = 1$, but then it drops below 100.

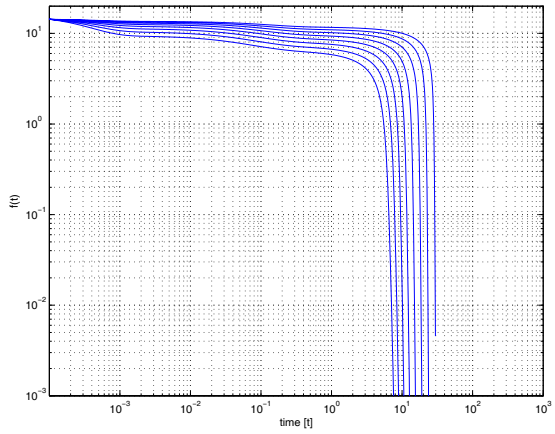


Figure 16: $f(t)$ for varying l_f, using ode23t.

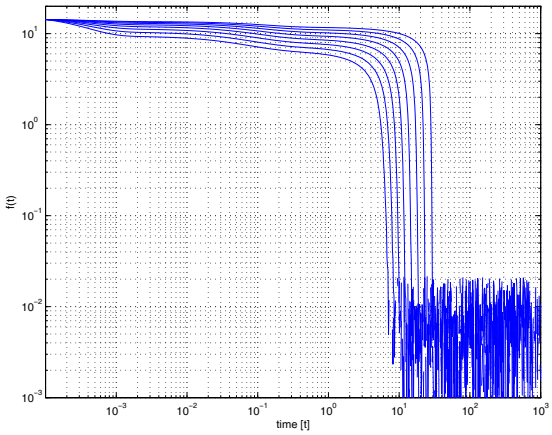


Figure 17: $f(t)$ for varying l_f, using ode23.

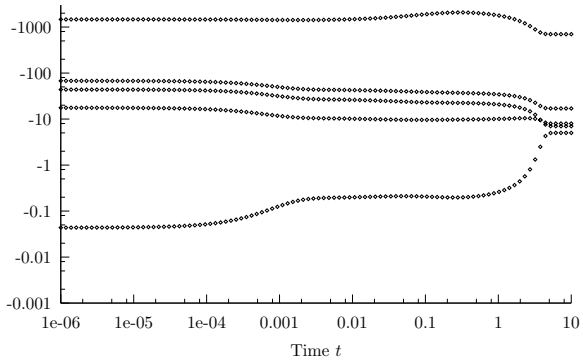


Figure 18: Eigenvalues of $J(t)$.

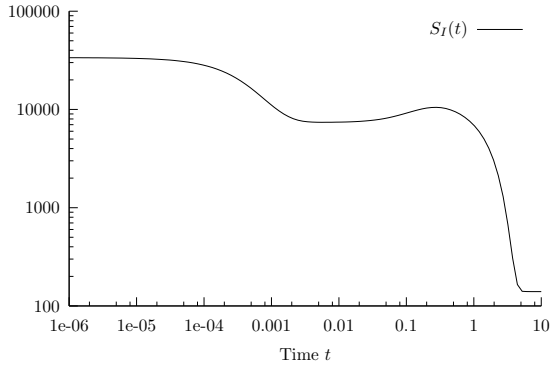


Figure 19: Stiffness index $S_I(t)$

Analytically and numerically determined steady states are given in Table 6 and Table 7, respectively. For an analytical solution, one has to solve for derivatives getting zero. Again, there are slight errors in the numerical solution.

$$\frac{dF^{(i)}}{dt} = 0, \quad i = 1,\ldots,5 \qquad \longrightarrow$$

$$F^{(1)}(t) = \frac{p}{a}, \quad F^{(2)}(t) = \frac{k_1 p^2}{a^2 l_2}, \quad F^{(3)}(t) = \frac{k_1 k_2 p^3}{l_2 l_3 a^3},$$
$$F^{(4)}(t) = \frac{k_1 k_2 k_3 p^4}{l_2 l_3 l_4 a^4}, \quad F^{(5)}(t) = \frac{k_1 k_2 k_3 k_4 P^5}{l_2 l_3 l_4 l_5 a^5}$$

P	$F^{(1)}$	$F^{(2)}$	$F^{(3)}$	$F^{(4)}$	$F^{(5)}$
0	0	0	0	0	0
10000	14.2857	24.0096	107.186	546.866	4218.68

Table 6: 5 Centers, steady states, analytical solution.

P	$F^{(1)}$	$F^{(2)}$	$F^{(3)}$	$F^{(4)}$	$F^{(5)}$
	$0.0851 \cdot 10^{-21}$	$-0.0817 \cdot 10^{-21}$	$0.0132 \cdot 10^{-21}$	$-0.3706 \cdot 10^{-21}$	$0.2118 \cdot 10^{-21}$
10000	14.3	24.0	107.2	546.9	4218.7

Table 7: 5 Centers, steady states, numerical solution.

The Taylor expansion behaves the same way here as for $n = 3$. There is a characteristic point up to which it is usable, and after that it diverges very quickly around the point of expansion (Figures 20, 21 and 22). The linearised version of the model gives analogous results: it is a good approximation up to a certain point beyond which, however, the linear model is insufficient to describe the dynamics (Figure 23).

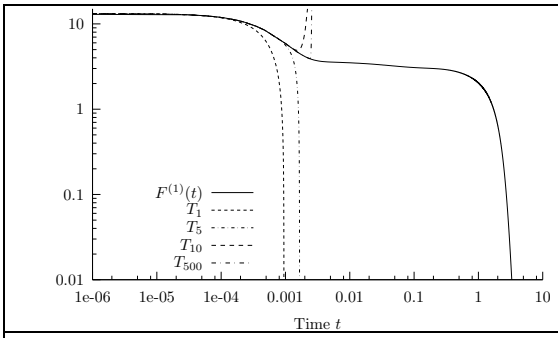


Figure 20: Taylor approximations, $t_0 = 0$.

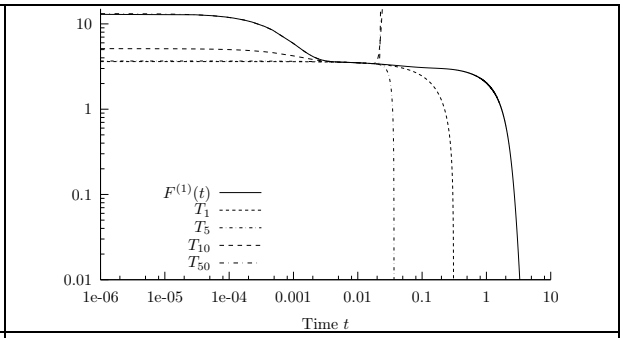


Figure 21: Taylor approximations, $t_0 = 0.01$.

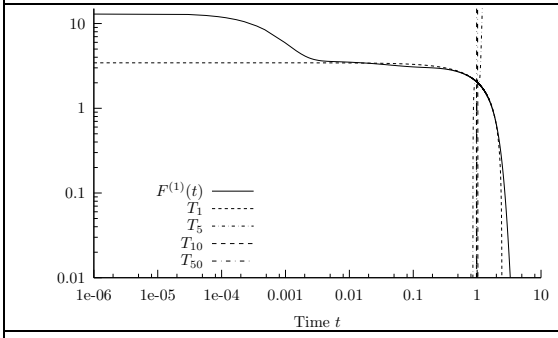


Figure 22: Taylor approximations, $t_0 = 1$.

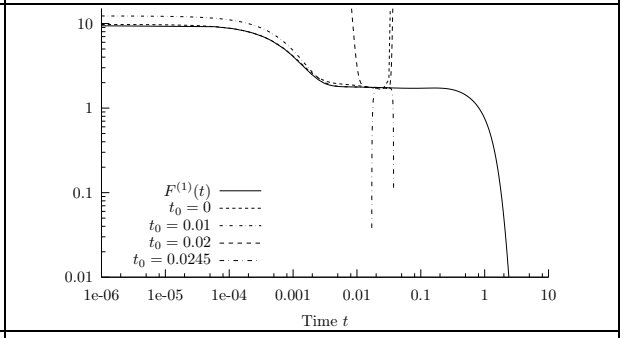


Figure 23: Linearisation at different time instants.

5. Comparison 12: Collision Processes in Rows of Spheres

The *ARGESIM Comparisons C12 'Collision of Spheres'*, defined by R. Hohmann [10], University Magdeburg, deals with a model of mechanics: four spheres in a row are colliding. The movement of the spheres is given by linear movement, so that for calculation of dynamics also analytical methods can be used. The features to be compared represent a large number of collision events, the numerical accuracy, the iteration of a boundary value, and stochastic parameter variations. Piecewise, constant velocities permit both a continuous and a discrete treatment.

Subject of the investigation are sequences of collisions, caused by the impact of a sphere on a resting row of spheres. In the elastic case only one impact occurs between neighbouring spheres, whereas one can observe many interactions if elasticity decreases. Numerical problems result from the peculiarity, that the relative distances and velocities at a low elasticity can be smaller by orders of magnitude than the absolute variables. In order to avoid small faulty differences of great values, the relative quantities are used as variables, and absolute quantities are obtained by summation.

5.1 Physical Modelling: Partially Elastic Collision of two Masses

The collision shall take place at $t = 0$ with the velocities v_1, v_2 (Figure 24a). The force $F(t)$ being exerted from both masses on each other, rises first with t and reaches its maximum at $t = t^*$ (Figure 24b). In this compression phase, the bodies are increasingly deformed in the immediate vicinity of the contact place. At the end (maximum deformation) both bodies have the same velocity v^*. In the following restitution period the deformations disappear partially or completely, concurring with a reduction of the contact force $F(t)$. After the time interval t_S the collision process is finished and both masses move with velocities $\bar{v}_1$ and $\bar{v}_2$, respectively.

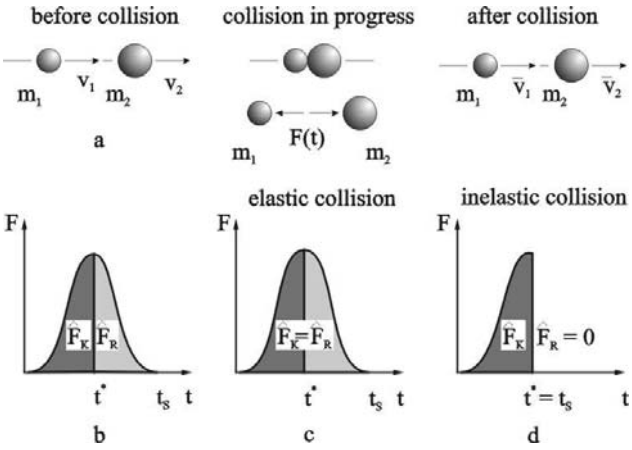


Figure 24: Central impact of two masses

The force impulses $\hat{F}_K$ and $\hat{F}_R$, exerted during both periods, determine the momentum change. They are represented by the areas below the force curve $F(t)$. The force impulse in the restitution phase reaches at most the value of the compression phase with e restitution coefficient (collision coefficient):

$$\hat{F}_R = e \cdot \hat{F}_K,\ 0 \le e \le 1 \tag{6}$$

An elastic impact has the collision coefficient $e = 1$, whereas an inelastic collision is known to have no restitution phase ($e = 0$). In general, partially elastic case the collision coefficient takes on values of $0 < e < 1$.

Using the momentum conservation law, the new velocities in the next period of time follow this piecewise description:

$$\bar{v}_1 = v_1 - (1+e)\frac{m_2}{m_1+m_2}(v_1 - v_2), \quad \bar{v}_2 = v_2 + (1+e)\frac{m_2}{m_1+m_2}(v_1 - v_2) \tag{7}$$

After the limiting process of the collision time $t_S \to 0$, the impact shall be modelled in the following as a state event that takes place immediately.

5.2 Physical Modelling: Mathematical Model of a Spheres' Row

In order to obtain an ideal translation, the p spheres arranged in a row are tied up with infinite long threads without any friction (Figure 25). The model consists of $p = 4$ spheres; for all collisions e takes on a constant value that does not depend on the velocities. A further precondition is the equality of the diameters d for all spheres, their masses m_i and distances a from each other.

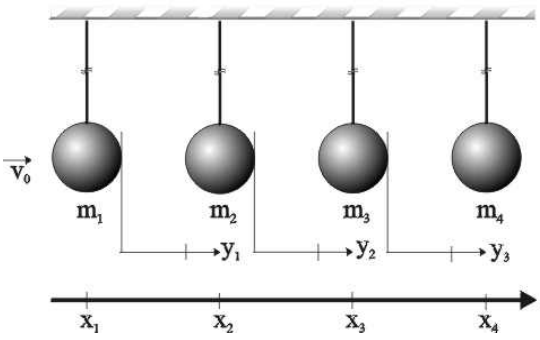


Figure 25: Collision pendulum of four spheres

In the model description the relative quantities are variables:

$$\begin{aligned} y_1 = x_2 - x_1 - d,\ y_2 = x_3 - x_2 - d,\ y_3 = x_4 - x_3 - d \\ \dot{y}_1 = \dot{x}_2 - \dot{x}_1,\ \dot{y}_2 = \dot{x}_3 - \dot{x}_2,\ \dot{y}_3 = \dot{x}_4 - \dot{x}_3 \end{aligned} \tag{8}$$

For determination of the remaining absolute quantities by summation equations of motion for the inner distances y_i (i=1,2,3) and the absolute variable x_1 are needed. The initial conditions are chosen so that sphere 1 strikes the motionless other three spheres with velocity v_0. An influence of external forces is not considered.

Equations of motion and absolute quantities are:

$$\begin{aligned} &\ddot{x}_1 = 0, \quad \dot{x}_1(0) = v_0, \quad x_1(0) = 0; \qquad \ddot{y}_1 = 0, \quad \dot{y}_1(0) = -v_0, \quad y_1(0) = a \\ &\ddot{y}_2 = 0, \quad \dot{y}_2(0) = 0, \quad y_2(0) = a; \qquad \ddot{y}_3 = 0, \quad \dot{y}_3(0) = 0, \quad y_3(0) = a \end{aligned} \tag{9}$$

$$\begin{aligned} x_2 = x_1 + y_1 + d, \quad x_3 = x_2 + y_2 + d, \quad x_4 = x_3 + y_3 + d \\ \dot{x}_2 = \dot{x}_1 + \dot{y}_1, \quad \dot{x}_3 = \dot{x}_2 + \dot{y}_2, \quad \dot{x}_4 = \dot{x}_3 + \dot{y}_3 \end{aligned} \tag{10}$$

The expressions on the right side of equations (10) describing the velocities after a collision contain the relative velocities at the moment of impact as derivatives of the distance variables y_i, that determine the time of collision.

Collision 1-2:

$$\dot{x}_1 = \dot{x}_1 + (1+e)\cdot m_2/(m_1+m_2)\cdot \dot{y}_1, \quad \dot{y}_2 = \dot{y}_2 + (1+e)\cdot m_1/(m_1+m_2)\cdot \dot{y}_1, \quad \dot{y}_1 = -e\cdot \dot{y}_1 \tag{11a}$$

Collision 2-3:

$$\dot{y}_1 = \dot{y}_1 + (1+e)\cdot m_3/(m_2+m_3)\cdot \dot{y}_2, \quad \dot{y}_3 = \dot{y}_3 + (1+e)\cdot m_2/(m_2+m_3)\cdot \dot{y}_2, \quad \dot{y}_2 = -e\cdot \dot{y}_2 \tag{11b}$$

Collision 3-4

$$\dot{y}_2 = \dot{y}_2 + (1+e)\cdot m_4/(m_3+m_4)\cdot \dot{y}_3, \quad \dot{y}_3 = -e\cdot \dot{y}_3 \tag{11c}$$

Insignificant or positive relative velocities $(\dot{y}_1 \geq 0) \wedge (\dot{y}_2 \geq 0) \wedge (\dot{y}_3 \geq 0)$, i.e., monotonously increasing absolute velocities, establish the termination criterion for a simulation run, that is, no further collisions will occur and the velocities will not change.

5.3 Tasks for Comparison C12

As in all comparisons, three tasks have to be performed.

Task a. (a1) Graphical representation of the distance-time functions $y_1(t)$, $y_2(t)$ and $y_3(t)$ for parameter values $e = 0.2$, $d = 1$ and initial values $a = 1$, $v_0 = 1$ in time interval $0 \leq t \leq 15$ (termination criterion met). Initial values and sphere diameter d remain valid in the following.
(a2) Final values of the velocities for $e = 1$ (elastic case) and for the quasi-plastic case in which velocities are sufficiently equal.

Task b. (b1) Number of collisions as a function of the restitution coefficient $n(e)$ which should be varied from $e = 1$ to a value for which the quasi-plastic case is reached.
(b2) Graphical representation of the final velocities $\dot{x}_1, \dot{x}_2, \dot{x}_3, \dot{x}_4$ as a function of values of e for $e \leq 1$ up to the quasi-plastic case.

Task c. (c1) As a boundary value problem the restitution coefficient e is to be determined such that the final velocity be $v_4 = v_0/2$.
(c2) The restitution coefficient e, which is equal for all spheres, is now a normally distributed stochastic variate with mean value $m = 0{,}5$ and standard deviation $s = 0{,}05$. The distribution function of v_4, mean value, standard deviation and confidence interval with confidence probability of 95% for a sufficiently large sample size are to be determined.

5.4 Graphical Results – Comparison C12 'Collision of Spheres'and Solution approaches

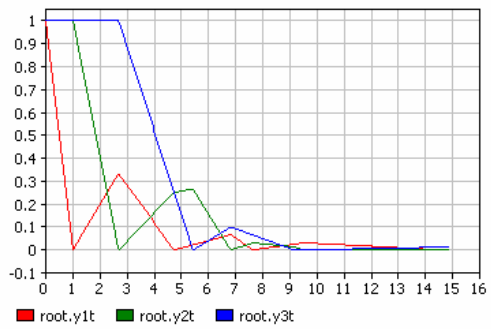


Figure 26: Distances of spheres over time

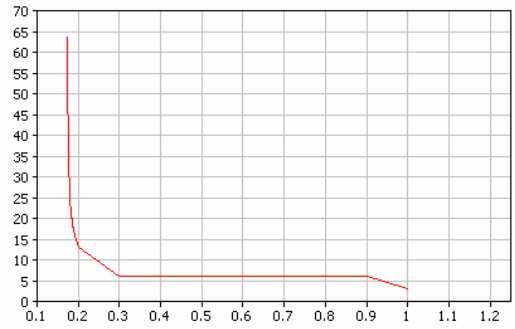


Figure 27: Number of collisions vs. restitution coefficient

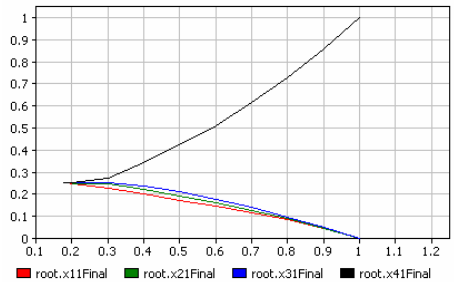


Figure 28: Final velocities vs. restitution coefficient

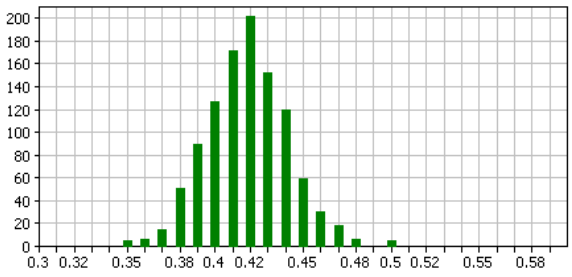


Figure 29: Distribution of final velocity of 4th sphere for stochastic restitution coefficient

Before sketching the various approaches to this comparison, the graphical results are presented. Figure 26 shows the results for task a, for distance-time functions $y_1(t)$, $y_2(t)$ and $y_3(t)$ $e = 0.2$, $d = 1$ and initial values $a = 1$, $v_0 = 1$ in time interval $0 \le t \le 15$.
Figure 27 shows the number of collisions versus the restitution coefficient: starting with $n(e) = 3$ collisions for $a = 1$, increasing to $n(e) = 6$ at $e = 0.9$, staying constant until $a = 0{,}3$, increasing to $n(e) = 13$ at $e = 0.2$, and $n(e)$ going to infinity for $a \to 0$.

Figure 28 displays the final velocities of the spheres, depending from the restitution coefficient: clearly, for $a = 1$ the first three spheres do not move, and the fourth sphere has got all the energy for moving further on with velocity 1; decreasing a lets the first three spheres move with increasing velocity, while the fourth sphere gets slower; for $a \to 0$ velocities of all spheres tend 0.25.
Finally, Figure 29 shows the distribution of the final velocity of 4th sphere for a stochastic restitution coefficient.

5.5 Solution Approaches – Comparison C12 'Collision of Spheres'

This general comparison allows a broad variety of approaches. In principle, the system is governed by four ODEs for the position of the spheres. Numerical investigations show, that its is highly recommended to use instead of the four ODEs for position three ODEs for the differences of the positions. Do not taking into account the simplicity of the ODEs, the collisions are state events, which have to be recognised, localised with arbitrary accuracy, and handled (impact). On the other hand, the ODEs can be solved analytically, and the collision times can be formulated as analytical formula. In both cases, especially in case of small values for the restitution factor a, high accuracy plays a dominant role for detecting or determining the collisions. As solutions are calculated numerically with minimal stepsizes (as well in case of ODE solvers or solving the analytical collisions equations), one also may use a discretised time base with minimal stepsize, so that also event driven approaches can be used.

The principle approaches are :

- Solution of the ODEs, with collisions as state events
- Analytical formula for collision events
- Event- based determination of collisions
- Mixed approaches
- High accuracy approaches

In the following, some of the approaches of the comparison solutions sent in are sketched.

Analytical Formula for Collision Events - FORTRAN [11].
A FORTRAN 90 program was used to (1) determine the next two spheres which will hit each other and (2) to update the positions and the velocities of the spheres after the collision according to the rules for partially elastic collisions. All calculations were carried out in double precision.
The program terminates if all relative velocities are not negative (no more collisions) or if the next two colliding spheres cannot be determined uniquely (two pairs of spheres are numerically equally likely to collide next). Assuming that the last pair of spheres to collide will not also be the next to collide, simplifies the algorithm. Additionally, only spheres with a negative relative velocity are considered.

```
!  previous hit of spheres 1 and 2
IF (.NOT. t12_log) THEN
  IF ( (v23 .LT. 0) .AND. (v34 .LT. 0) ) THEN
    IF (d23/v23 .EQ. d34/v34) THEN
      EXIT
    END IF
    t_neg_max = max( d23/v23 , d34/v34 )
    IF (t_neg_max .EQ. d23/v23) THEN
      nexthit = 23
    ELSE IF (t_neg_max .EQ. d34/v34) THEN
      nexthit = 34
    END IF
  ELSE IF ( v23 .LT. 0 ) THEN
    nexthit = 23
  ELSE IF ( v34 .LT. 0 ) THEN
    nexthit = 34  …….
```

Analytical Formula for Collision Events / SLX/LEDA – High Accuracy [12].

In order to deal with this problem by a discrete event simulator we determine both the time to the next interaction and the two spheres involved in this collision. Our algorithm uses the relative velocities of neighbouring spheres to find the minimum among all possible future interaction times. Based on that we identify the spheres to execute the next collision and determine their resulting velocities as initial conditions for the next iteration step.

```
for (j = 1; j <= N-1; j++)
if (dv[j] > 0.0) dt[j] = dx[j] / dv[j];
else dt[j] = 1e+10;
k = 1; dtmin = dt[k];
for (j = 2; j <= N-1; j++)
        if (dt[j] < dtmin)
        { dtmin = dt[j]; k = j; }
```

LEDA (Library of Efficient Data types and Algorithms) is a software library which introduces the `algebraic real` data type to C++. The `reals` are the best approximation of the mathematical real numbers IR. They offer exact results for the operators +, -, *, /, the *k*-th root for any natural number *k* and for comparative operators.

The `reals` 'memorize' the calculations executed in an `expression dag`. Only if two `reals` are compared, or a number of the type `real` is to be output, the result is computed 'as precise as necessary' in order to execute an exact comparison or in order to output the result with a given number of digits. This is however connected to a dramatically increasing runtime.
LEDA is coupled with SLX in order to increase the accuracy of the calculations essentially.

Analytical Formula for Collision Events / ODEs for Movement - MATLAB [13].

In this solution, first the time of the next impact is analytically determined repeatedly.

Second, the resulting time intervals between the collisions, together with the initial values, the positions and velocities of the spheres after the last collisions control a numerical integration (of the equations of motion). The algorithm calculates the positions of the spheres by solving numerically these differential equations (using MATLAB's algorithm `ODE23`):

```
if ydot(j)<0
time=abs(y(j)/ydot(j));
     if time<dt(i,1)
       dt(i,1)=time; dt(i,2)=j;
     end
  if dt(i,1)==0
     dt(i,1)=time; dt(i,2)=j;
  end
```

```
ODE23(@odefun_y,[0,dt(i,1)],y',[],ydot');
y=y_ode(size(y_ode,1),:);
function [ydot]=odefun_y(T,Y,ydot)
function [ydot]=hit23(ydot,e,m2,m3)
ydot(1)=ydot(1)+(1+e)*m3/(m2+m3)*ydot(2);
ydot(3)=ydot(3)+(1+e)*m2/(m2+m3)*ydot(2);
ydot(2)=-e*ydot(2);
```

ODE Solution with State Event Handling for Collision Events / DYMOLA [14].

Model: With the following discussed hybrid model approach, the capability of Dymola to handle hybrid systems is tested. In this approach the implementation of the model is made in Dymola because of the simplicity of the Model, no need for a graphical based modelling is given. A hybrid model in Dymola consists of differential, algebraic and discrete equations. Furthermore it offers three variable step size algorithms for the numerical treatment of such Differential-Algebraic-Equations (DAEs).
The Dymola user manual suggests using the statement `when (cond)` then to handle events in continuous systems. To reinitialise a variable in case of an event Dymola provides the `reinit` statement. The implementation is then straightforward:
In Dymola, all differential, algebraic and discrete equations are treated as synchronous in time.

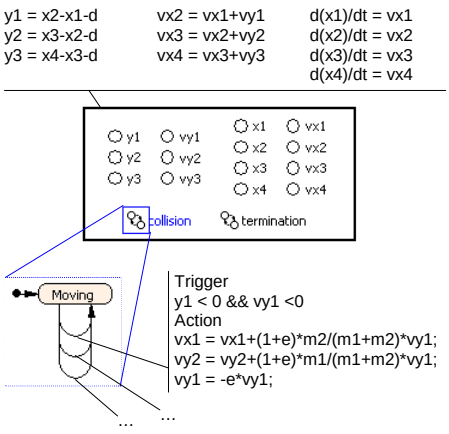
```
class Comparison12
 .. declaration of vari-
ables…

algorithm
when y1<=0
  reinit(x1p, x1p +
(1+e)*m2/(m1+m2)*y1p);
  reinit(y2p, y2p +
(1+e)*m1/(m1+m2)*y1p);
  …..
end
….other events …
equation
der(x1)  = x1p;
der(x1p) = 0;
… other equations…
end Comparison12
```

Therefore in Dymola / Modelica the assumption is used that all equations in a model may potentially be active at the same time instant.
It is easy then to construct conflicting equations. To handle this problem multiply events have to be implemented in the so-called `algorithm` section.

5 ODE Solution with State Automata- Controlled Event Handling for Collision Events / AnyLogic [15].

The system of colliding spheres is modelled as one AnyLogic object with several variables representing the positions of spheres (x1, x2, x3, x4), their relative distances (y1, y2, y3), and their absolute (vx1, vx2, vx3, vx4) and relative (vy1, vy2, vy3) velocities. The equations for motion of the spheres are associated with the object.

The state chart has a single state and three transitions representing collisions of spheres. These transitions are triggered by change events – Boolean expressions over variables.

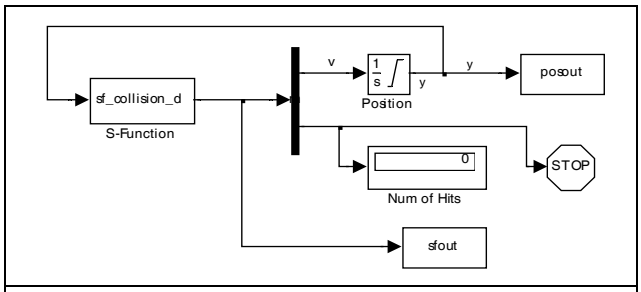


The hybrid state chart can be entered in AnyLogic model editor, and the simulation engine automatically detects change events, changes the working set of equations and readjusts the numerical methods.

ODE Solution with State Event Handling for Collision Events / SIMULINK [16].

The main components of the SIMULINK model are the s-function that provides the velocities and times of collisions and an integrator that calculates the positions of the spheres.

Since the zero-detection-block is not available in the presence of an s-function, integration was limited to $[0, \infty)$. For the s-function an event-driven approach was chosen: using state- (current velocities) and input- (current distances) information the s-function calculates state update and the time when it should be called again (time of next collision).




```
function sys=mdlGetTimeOfNextVarHit(t,v,u,e)
global thit stop hittype
% get time of next collision
v_0=-eps;y_0=realmin;
thit = 0;
hittype = 0;
if (v(2) < v_0) & (u(2) > y_0)
    thit = - u(2) / v(2); hittype = 1;
end
if (v(3) < v_0) & (u(3) > y_0)
```

6. References

1. Banks J., (Ed). (1998): Handbook of Simulation. New York: John Wiley and Sons.
2. Jeppsson, Ulf: A Comparison of Simulation Software for Wastewater Treatment Processes - A Cost 682 Program Perspective. Report TEIE-7078, 1994.
3. Ludwin W., Chlebus E.: Performance comparison of simulators for cellular mobile networks, Applied Mathematical Modelling 20 (8) (1996) pp. 585-587.
4. Hlupic V., Mann A. S.: SimSelect: a system for simulation software selection**.** Proc. 27th Wintersim, Arlington, Virginia, USA; ISBN:0-7803-3018-8, 1995, p 720 – 727.
5. Breitenecker F., ARGESIM Comparisons on Simulation Technique: Classification, Evaluation as Source for WWW-supported Simulation Courses. Proc. Conf. IEEE - ERK'2001 Portoroz, IEEE Slovenia, ISBN 961-6062-21-2; p. 279-282, p.
6. Breitenecker F.; ARGESIM Comparisons of Modelling and Simulation Techniques and Tools – Simulation Benchmarks. SNE Simulation News Europe 44/45, 2005, 4 p.
7. Ecker H., Breitenecker F., Troch I.: The MATHMOD Yo-Yo Benchmark – Comparison of Simulator Features. SNE Simulation News Europe 44/45, December 2005, pp 60.
8. F. Breitenecker, F. Judex, E. Nigsch, G. Betz: Phsyical Modelling for the Benchmark 'Collisions of Spheres'. Simulation News Europe SNE 46 (Juli 2006), 8 pages, to appear.
9. O. Kreitschitz, C. Polster, W. Husinsky, and G. Betz, Desorption kinetics of Li atoms from LiF under electron bombardment, Nuclear Instruments and Methods in Physics Research B 58 (1991), 490-495
10. Rüdiger Hohmann, Christian Gotzel, Carsten Pöge: Definition of ARGESIM Comparison C12 'Spheres Collision', SNE Simulation News Europe 27, pp 36 - 37.
11. M. Willensdorfer, F. Breitenecker: C12 'Spheres Collision' - Solution with FORTRAN; SNE 31, pp36.
12. Christian Gotzel, Rüdiger Hohmann, Carsten Pöge,Jörg Schwerdt: C12 'Spheres Collision' - Solution with SLX LEDA SNE 31, pp35.
13. W. Weidinger, D. Schachinger, G.Langs: C12 'Spheres Collision' - Solution with MATLAB; SNE32/33, pp52
14. Michael Wibmer:C12 'Spheres Collision- Solution with Dymola; SNE 32/33, pp53.
15. Alexei Filippov, Alexei Kornev: C12 'Spheres Collision! Solution with AnyLogic; SNE 32/33, pp50
16. J. Scheikl: C12 'Spheres Collision' - Solution with SIMULINK; SNE 31, pp34.