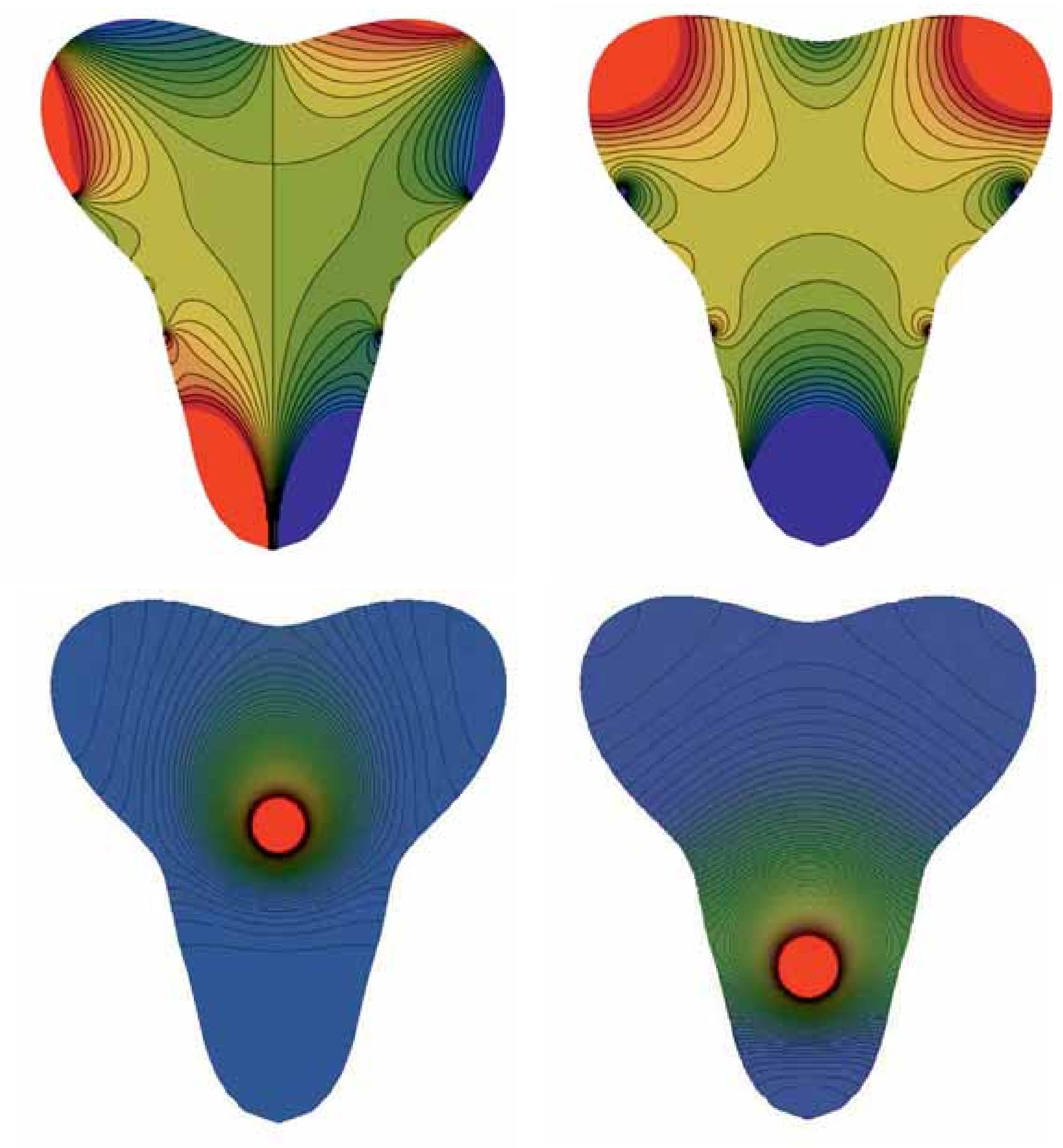
SNE SIMULATION NOTES EUROPE



Journal on Developments and Trends in Modelling and Simulation

EUROSIM Scientific Membership Journal

Vol. 31 No.1, March 2021
ISSN Online 2306-0271
ISSN Print 2305-9974
DOI 10.11128/sne.31.1.1055
ISBN Print 978-3-903311-15-2

TU Verlag
an der Technischen Universität Wien

ARGESIM

New Schedule for EUROSIM Conferences and Congress

Due to Corona Virus, also **EUROSIM** changed the schedule of **EUROSIM Conferences** and the **EUROSIM Congress** – all events will take place one year later. To bridge the 2020 conference gap the **EUROSIM** societies organise virtual conferences, and the **EUROSIM Board** started in June 2020 **VESS** – the **Virtual EUROSIM Seminar**, a series of online presentations discussing trends in modelling and simulation and preparing emphasis of future **EUROSIM** events.

DBSS
Dutch Benelux Simulation Society
www.eurosim2023.eu
VESS - VIRTUAL EUROSIM SIMULATION SEMINAR
free online seminar series

The **EUROSIM Board** and **DBSS** started in June 2020 **VESS** – the **Virtual EUROSIM Seminar**, a series of online presentations discussing trends in modelling and simulation. These international online simulation seminars – monthly or bimonthly – are open to everybody, via Zoom, lasting 60 minutes (45 minutes presentations, 15 minutes Q & A).
Information and informal registration via website **www.eurosim2023.eu**

SIMS EUROSIM 2021 Conference on Modelling and Simulation
21-23 September 2021 in Oulu, Finland

The First **SIMS EUROSIM Conference** on Modelling and Simulation, **SIMS EUROSIM 2021** takes place in Oulu, Finland, September 21-23, 2021. The 62nd International Conference of Scandinavian Simulation Society, SIMS 2021, is embedded with **SIMS EUROSIM 2021**. The **SIMS EUROSIM** conference will be organized every third year by **SIMS** and **EUROSIM**. The background of this conference series is in the 60-years history of Scandinavian Simulation Society, **SIMS**.
The program of the **SIMS EUROSIM 2021 Conference** will have a multi-conference structure with several special topics related to methodologies and application areas. The program includes invited talks, parallel, special and poster sessions, exhibition and versatile technical and social tours – info **www.scansims.org**

MATHMOD 2022
Vienna
10th Vienna International Conference
on Mathematical Modelling
February 16 - 18, 2022, Vienna, Austria

MATHMOD organizers continue the conference series one year later, with **10th MATHMOD 2022**, February 16-18, 2022. **MATHMOD 2022**, one of **EUROSIM**'s main events, provides a forum for professionals, researchers, and experts in the field of theoretic and applied aspects of mathematical modelling for systems of dynamic nature.
The scope of the **MATHMOD 2022** conference covers theoretic and applied aspects of various types of mathematical modelling (equations of various types, automata, Petri nets, bond graphs, qualitative and fuzzy models) for systems of dynamic nature (deterministic, stochastic, continuous, discrete or hybrid) – info and details **www.mathmod.at**

EUROSIM 2023, the **11th EUROSIM Congress**, will take place in Amsterdam, The Netherlands, Spring/Autumn 2023. It will be organized by the Dutch Benelux Simulation Society (www.dutchbss.org) supported mainly by their corporate members like TU Delft, Amsterdam University of Applied Sciences, EUROCONTROL and IGAMT (www.igamt.eu).
Due to the growth of Simulation and its relationship with other analytical techniques like Big Data, AI, Machine Learning, Large Scale Simulation and others, the event will be structured, for the first time, in dedicated tracks focused on different areas and applications of Simulation ranging from aviation to health care and humanitarian activities. We have the ambition to attract at the congress participants from Academia, industry and governmental representatives to share the latest developments in Simulation and related activities and applications.
Please follow the news and activities towards the **EUROSIM 2023** at **www.eurosim2023.eu**

Editorial

Dear Readers, SNE Volume 31 starts in the March Issue SNE 31(1) with three topics. First, this issue finishes the subject 'high math meets simulation' – Martin Holzinger presents his third contribution on PDE solution by conformal mappings presenting a surprising application, the modelling and simulation of thermal distributions in teeth.

Second, SNE 31(1) starts with post-conference publication of contributions from ASIM 25. Symposium Simulation Technique. Two contributions deal with co-simulation: T. Moshagen presents a stable but explicit co-simulation coupling method, and I. Hafner and N. Popper investigate stability properties of hierarchical co-simulation. And H. Folkerts, T. Pawletta, and C. Deatcu introduce model generation for multiple simulators using SES/MB and FMI.

And third, two contributions – Benchmark Notes 'BN' – continue the series on ARGESIM Benchmarks. In 2019, SNE has introduced extended possibilities for submitting a benchmark contribution: i) a Benchmark Solution with concise description of model implementation and experiment tasks (as before, two pages SNE), ii) or a Benchmark Report with sufficient detailed description of model implementation with variants and adequate experiment formulations (four to six pages SNE), iii) or a Benchmark Study presenting e.g. different/alternative/comparative modelling approaches and sketching analysis variants or supplemental model experiments (six to ten pages SNE). Additionally, content may concentrate on educational aspects (Educational Benchmark Note 'BNE'), and student authors may submit a benchmark contribution from their views (Student Benchmark Note 'BNS').

So, Peter Junglas et al. have realised a very comprehensible benchmark report to ARGESIM Benchmark C21 'State Events and Structural-dynamic Systems' with Simulink, and M. Batliner et al. have developed a benchmark study for ARGESIM Benchmark C11 'SCARA Robot' with extended trajectory tracking control comparing model approaches and simulation results in MATLAB, Simulink and SimMechanics The next issue will continue with Student Benchmark Notes.

I would like to thank all authors for their contribution, 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 31(1)

Online SNE 31(1), DOI 10.11128/sne.31.1.1055
ARGESIM Publisher, Vienna, www.argesim.org

Print SNE 31(1) ISBN 978-3-903311-15-2
TU Verlag Vienna, Print-on-Demand, www.tuverlag.at

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: TU-Verlag, 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.
- **BNE** Edu B.Note, 2 10 p
- **STN** Student Note, 4-8 p, inv.

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:

PDE-based Simulation of Thermal Distributions in Teeth by Conformal Parametrization

Martin Holzinger[1]*, Pauline Steinkogler[2]

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

[2]BG XIII/1c, Fichtnergasse 15, 1130 Vienna, Austria; paulinsky.st@icloud.com

SNE 31(1), 2021, 1-8, DOI: 10.11128/sne.31.tn.10551
Received: October 20, 2020; Revised: February 10, 2021;
Accepted: February 20, 2021
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna,
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. With the conformal map from unit square to unit disk analytically known, in our last contribution we investigated ways to numerically map the disk to more general (but star-shaped) domains. Such point-mappings of the complex planes are now to be interpreted as transformations of co-ordinates, hence the domains are parametrized by the square. Using general, curvilinear co-ordinates one has to take the shape of the fundamental tensor and other related quantities into account. The also numerically known derivatives of the map act as metric quantities flowing in and correcting a pre-given PDE in Cartesian co-ordinates on such a domain. On the other hand, a formulation of physical laws in co-ordinate free manner gives an even smarter access to implement a simulation code of a given problem in *Mathematica*. In this article, we focus on a practical problem: let the domain be the cross-section of a tooth and the task be to find the temperature distribution on its boundary with respect to a heat source moving in the interior of the domain. This model can then be interpreted as a decision-finding issue to parameter identification when treating a tooth with a laser pulse. Considering the problem in three dimensions by using rotational symmetry will turn out to be essential with respect to the obtained results.

Introduction

The treatment of tooth root inflammations by root resection is (in the author's own experience) one of the less desirable aspects in the whole digestion process. Moreover, success of this dolorous treatment can not be ensured and more commonly leads to extraction of the affected tooth after repeatedly abortive attempts.

To try to rescue a tooth from extraction anyhow, laser treatment can be applied as an alternative method.

The purpose of this alternative and widely painless endodontic laser treatment approach [1] is to annihilate malicious bacteria by means of thermal demolition with the obvious side condition that the *Parodontium* anchoring the tooth must not be damaged by temperatures too high. The area in question is sketched by the red line in Figure 1, labeled with *Cementum* there. For this undertaking, a laser source is moved along the root canal.

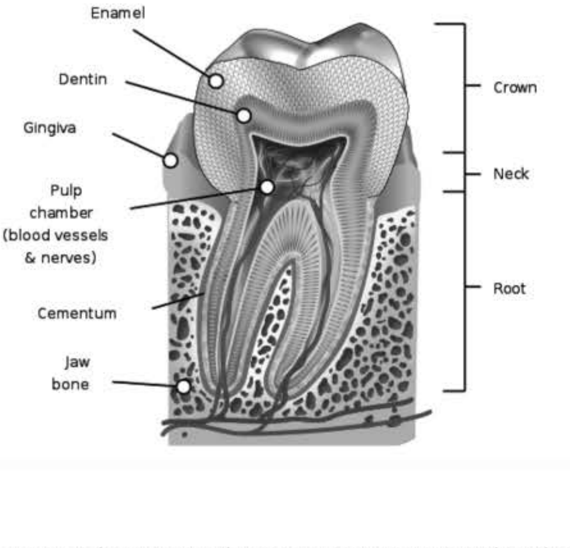


Figure 1: Schematic tooth cross-section (Source: Wikipedia).

With the gained knowledge of our previous articles (cf. [3] & [4]) we can set out to establish a simulation model on this issue. Hereto we adapt the developed heat equation simulation code on unit square (and unit disk) with the insights provided by the conformal map. While basically we are already ready-to-run, deducing a model in three spacial dimensions is most desirable. This extension has yet to be discussed as well as an apt implementation of the laser heat source. Moreover, let us focus on the temperature distribution on the boundary.

1 Calculating the Tooth Grid

In our previous contribution [4, p.90f], for three different pre-given domains we investigated how to numerically establish the function of boundary correspondence, $\varphi \mapsto \theta$, relating the angle φ circling around unit disk $\mathfrak{E}$ to angle θ on domain $\mathfrak{G}$ with given boundary $\mathfrak{C}$ as a smooth, closed Jordan arc. Due to worst error behaviour we will focus on geometry $GEO3$, described by means of Hermite interpolation. This domain may be used to derive a tooth cross-section simulation grid.

1.1 Notation

Using Cartesian co-ordinates (ξ, η) on $\mathfrak{G}$, we assumed rotational symmetry with respect to η-axis. Introducing the complex ζ-plane, a point of $\mathfrak{G}$ can be addressed via $\zeta = \xi + i\eta$ by means of its complex representation. Let then w denote the complex w-plane where $\mathfrak{E}$ resides so its points are reached by the complex representation $w = u + iv$. Furthermore, in a third complex plane we represent the unit square $\mathfrak{Q}$ in the z-plane, $z = x + iy$, where the computational grid is located (cf. [3, p.43]).

For the subsequent task to numerically construct the map $\overline{\mathfrak{Q}} \to \overline{\mathfrak{G}}$, the unit disk $\mathfrak{E}$ will be interposed. Hence, let $w = h(z)$ be the elliptic function evolved in [3, p.47] acting as a closed-form conformal map $\mathfrak{Q} \to \mathfrak{E}$. We note that implementing and calculating h and its derivatives numerically with *Mathematica* is straight-forward.

Finally, with $\zeta = g(w)$ presented in [4, p.87] mapping the closures $\overline{\mathfrak{E}} \to \overline{\mathfrak{G}}$ we find the conformal map from square onto domain by composition of g and h, thus $f = g \circ h : \overline{\mathfrak{Q}} \to \overline{\mathfrak{G}}$ where $\zeta = f(z) = g(h(z))$.

It is well worth explicitly pointing out the chain rule for the case that $g(w) = w \cdot e^{P(w)}$ is established by means of a trigonometric polynomial (see again [4, p.87]),

$$P(w) = \frac{a_0}{2} + \sum (a_k - ib_k) w^k,$$

applying the first derivative then results in

$$f'(z) = \xi_x + i\eta_x = g'(w)h'(z) = h'(z)e^{P(w)}(wP'(w)+1),$$

whereas for the second derivative we get

$$\begin{aligned} f''(z) &= \xi_{xx} + i\eta_{xx} = g''h'^2 + h''g' \\ &= e^P \left[h'' + P'(2h'^2 + wh'' + wh'^2P') + wh'^2P''\right]. \end{aligned}$$

Term by term differentiation of P is trivial for *Mathematica*, so is the numeric evaluation of f, f' and f''.

1.2 Remarks on g and g′

Concerning trigonometric interpolation of $\ln\rho(\theta(\varphi))$ to get hold of the Fourier coefficients establishing the map g we noticed that series convergence degrades depending on the ε-condition as well as in case that slope discontinuities arise in the parametrization of $\mathfrak{C}$.

Of course, with $GEO3$ we will not face any problems: smoothness of boundary is guaranteed by means of Hermite interpolation and the ε-condition has been investigated as well as convergence and error behaviour on $[0, 2\pi]$. But let us have a look at the non-trigonometric case in this section just for completeness.

For example, by considering the unit square itself as image domain $\mathfrak{G}$, in [4, p.89] we showed that high-precision $\theta(\varphi)$ can be achieved nonetheless by switching to more apt base functions than trigonometric ones.

Suppose therefore that $\theta(\varphi)$ is known to and implemented in *Mathematica* as well as $\rho(\theta)$ describing $\mathfrak{C}$ is at hand. Then we distinguish four cases [5, p.56f] to snatch numeric values for g and g':

Unit disk boundary point images:

$$g(e^{i\varphi}) = \rho(\theta(\varphi)) \cdot e^{i\theta(\varphi)}$$

Unit disk boundary point derivatives:

$$g'(e^{i\varphi}) = \theta'(\varphi) \cdot e^{i(\theta(\varphi)-\varphi)} \cdot \left[\rho(\theta(\varphi)) - i\rho'(\theta(\varphi))\right]$$

Unit disk interior point images:

$$g(w) = w \cdot \exp\left[\frac{1}{2\pi}\int_0^{2\pi} \ln\rho(\theta(\vartheta)) \frac{e^{i\vartheta} + w}{e^{i\vartheta} - w} \,\mathrm{d}\vartheta\right]$$

Unit disk interior point derivatives:

$$g'(w) = \frac{g(w)}{w} + \frac{g(w)}{2\pi}\int_0^{2\pi} \frac{\ln\rho(\theta(\vartheta)) + i(\theta(\vartheta) - \vartheta)}{(e^{i\vartheta} - w)^2} e^{i\vartheta} \,\mathrm{d}\vartheta.$$

$$g'(0) = \frac{1}{2i\pi}\oint_\gamma \frac{g(\zeta)}{\zeta(\zeta - 0)} \,\mathrm{d}\zeta = \frac{1}{2\pi}\int_0^{2\pi} \frac{\rho(\theta(\vartheta))e^{i\theta(\vartheta)}}{e^{i\vartheta}} \,\mathrm{d}\vartheta.$$

If necessary, formulae for higher-order derivatives can be developed likewise by regarding the generalized CAUCHY integral formula for derivatives [2, S.246, Theorem 4.7d] like we did for points in the interior of the unit disk to evaluate $g'(w)$. Finally note that the singularity arising in the origin, $g'(0)$, was removed by subsequent application of CAUCHYs integral formula.

1.3 Grid Visualization

Setting up and saving the grids on unit square, disk and $GEO3$ is done in complex arrays Z, W and ZETA:

```
For[i=1,i<=n,i++,For[j=1,j<=n,j++,
Z[[i,j]]=-1+(i-1)dx+I(-1+(j-1)dy);];]
X = Re[Z]; Y = Im[Z];
```

Real and imaginary parts of these arrays are also defined for the ease of data access. Whereas these objects are used in the simulation process (for example, Z can be fed directly in functions h and f and other arrays are defined to save the derivative values on the grid), the grids themselves are stored in separate structures:

```
Zgrid=Flatten[Table[
X[[i,j]],Y[[i,j]],i,1,n,j,1,n],1];
```

With these definitions, ListPlot outputs can be generated in a convenient way. Figure 2 shows the result of such an output ($n = 101$). While Figure 3 has already been presented in [3, p.47], Figure 4 shows the image grid following from the conformal mapping f from square to $GEO3$ with the unit disk interposed. Note in particular how the square corners are being mapped. Despite these present singularities no further impact on the simulation process can be expected.

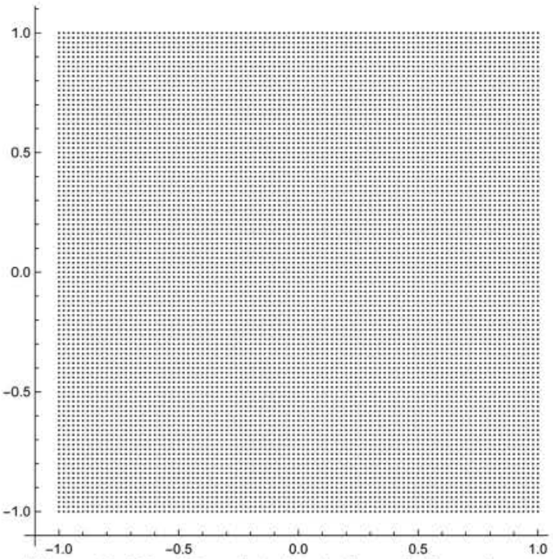


Figure 2: 101×101-point simulation grid in z-plane.

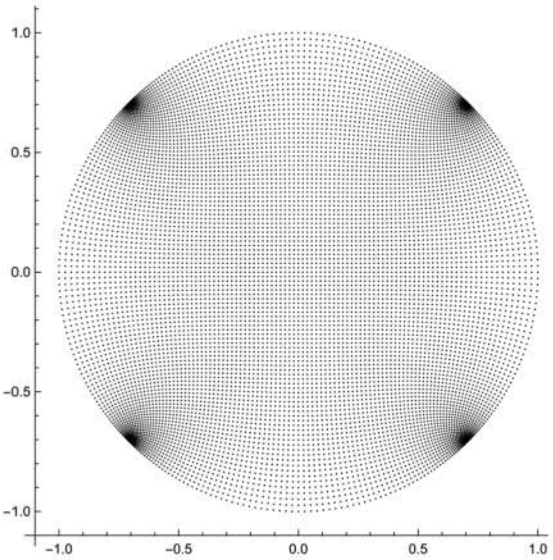


Figure 3: Result of the mapping $h : \overline{\mathfrak{Q}} \to \overline{\mathfrak{C}}$ in w-plane.

In the visualization context, all of these mappings act as point transformations. In our subsequent examinations, f will alternatively be interpreted as a transformation of co-ordinates. With the metrics calculated in addition, we are ready to run a simulation model on the tooth cross-section parameterized by the unit square.

2 Setting up the Model

It is one of the major benefits of our approach in using conformal parametrization of a domain by means of the unit square that - as soon as the conformal map and its derivatives are numerically determined and externally saved - an existing PDE-model can easily be ported to an other domain of interest.

Basically, one starts with an investigation of the underlying PDE itself in co-ordinate free formulation. Regarding the fundamental tensor involved and knowing $g_{ij} = \delta_{ij}$ in Cartesian co-ordinates, the proper transformation laws have to be taken into account to determine its new shape as well as vector or tensor valued state variables must be transformed. Right here the conformal transformation f and its derivatives come into play.

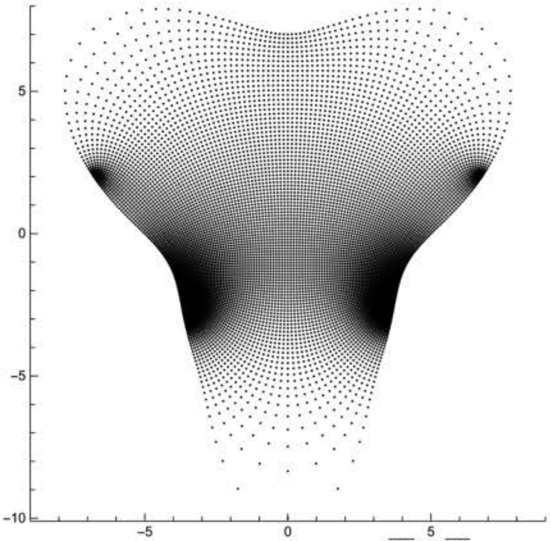


Figure 4: Result of the mapping $f = g \circ h : \overline{\mathfrak{Q}} \to \overline{\mathfrak{G}}$ in ζ-plane.

2.1 3D considerations

Starting from what we have implemented so far considering the simulation models on square and disk, switching to our new f' paying tribute to domain $GEO3$, the heat equation on the tooth cross-section can be simulated and one might think that we are already done.

However, such an approach in two dimensions is worth to be critically impugned. In fact, dropping the third Cartesian co-ordinate, $\mathfrak{z}$, to work with a 2D model results in the assumption of a constant temperature profile in this direction. Thus, this proceeding will lead us to a 3D geometry similar to a rod with infinite length, having a cross-section $\mathfrak{G}$ in each fixed point $\mathfrak{z}$.

This is definitely not in our intention and a comparison of mean temperature distributions will in fact reveal a 44% deviation with regard to the 3D model to be developed below. Let us therefore pack rotational symmetry by introducing cylindrical co-ordinates in our model to prevent dentists from having a formidable tooth braai.

Let thus be $\mathfrak{G}^* \subset \mathbb{R}^3$ a three-dimensional domain and $(\xi, \eta, \mathfrak{z})$ Cartesian co-ordinates with η denoting the axes of rotation. Let further be

$$f(x+iy) = \xi(x,y) + i\eta(x,y)$$

a conformal map such that for $\mathfrak{z} = 0$ the unit square in the (x,y)-plane is mapped to the cross-section $\mathfrak{G} \subset \mathfrak{G}^*$ located in the ζ-plane. Then the desired symmetry is incorporated by the co-ordinate transformation

$$\xi = \xi(x,y)\sin\varphi, \quad \eta = \eta(x,y), \quad \mathfrak{z} = \xi(x,y)\cos\varphi.$$

The new shape of the covariant metric tensor when switching from Cartesian to curvilinear co-ordinates, $(\xi, \eta, \mathfrak{z}) \to (x, y, \varphi)$, is determined by (cf. Figure 5)

$$\bar{g}_{ij} = \begin{pmatrix} \xi_x^2 + \eta_x^2 & 0 & 0 \\ 0 & \xi_x^2 + \eta_x^2 & 0 \\ 0 & 0 & \xi^2 \end{pmatrix}.$$

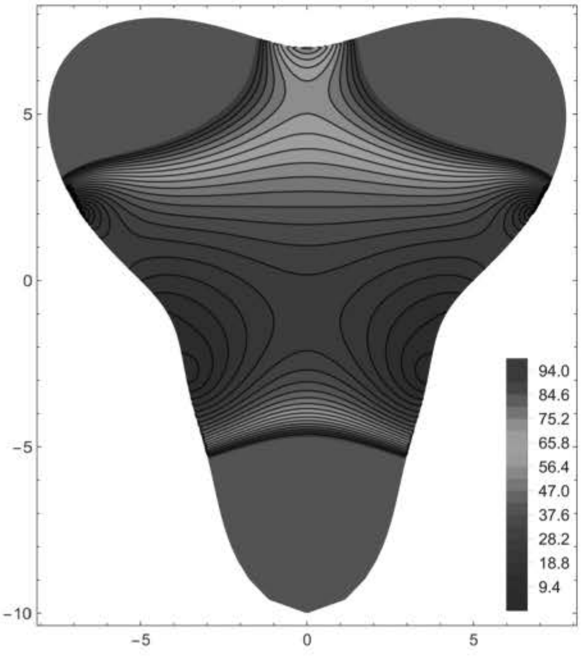


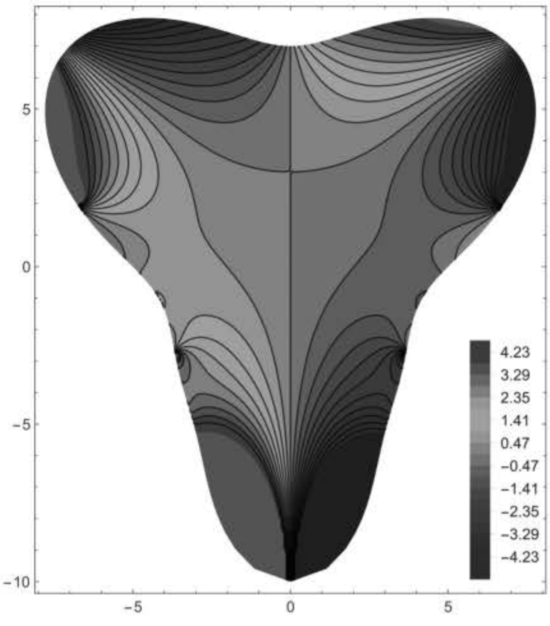
Figure 5: Fundamental tensor on $\mathfrak{G}$ with $g_{11} = g_{22}$.

The transformation itself obviously has impacts on the expressions involved in differential operators, too. For example, examination of the LAPLACE operator given in [4, p.92] yields (let be $\chi = \xi_x^2 + \eta_x^2$)

$$\triangle u = \frac{1}{\chi}\left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{1}{\xi}\left(\xi_x \frac{\partial u}{\partial x} - \eta_x \frac{\partial u}{\partial y}\right)\right] + \frac{1}{\xi^2}\frac{\partial^2 u}{\partial \varphi^2}.$$

The assumption of a rotational symmetric, angular constant temperature field now entitles us to drop the last summand in the expression above.

Comparing the remaining expression with the 2D simulation rod approach - the Laplacian is corrected by a factor $1/\chi$ in this case - we note that in addition, first derivatives of the temperature field $u(x,y,t)$ appear. Also, the sense of this expression on the axes of symmetry itself (singularity for $\xi = 0$) has to be questioned.



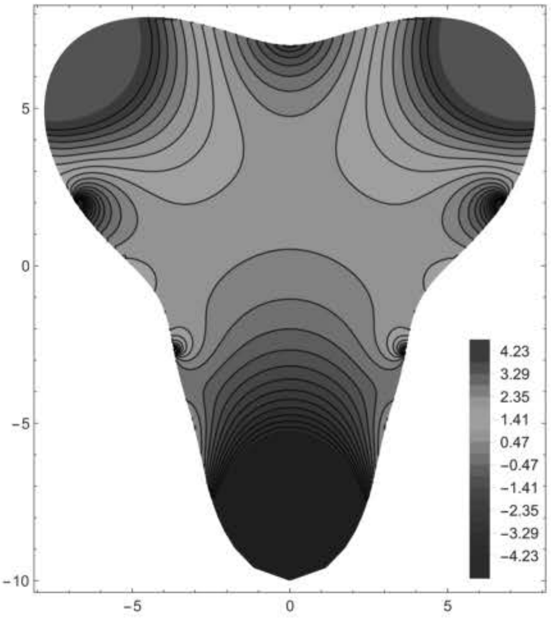


Figure 6: The two independent CHRISTOFFEL symbols Γ^1_{11} (top) and Γ^1_{12} (bottom) on tooth cross-section $\mathfrak{G}$ can be derived by evaluating f' and f'' in each grid point.

In the course of setting up $f(z)$ by means of trigonometric interpolation, we also showed how to derive expressions for f' and f''. With these functions implemented, we are able to visualize the CHRISTOFFEL symbols shown in Figure 6. Although not needed for simulating the heat equation it is a good idea to calculate and externally save these numeric values for each grid point for the case that the base PDE has to be adapted.

2.2 Impacts on grid and model

As it can be expected that switching over to a 3D rotational geometry yields a far more realistic modeling approach, there are other good news concerning the grid size: the temperature field has to be η-symmetric per definition, meaning restricting the computational area to $[0,1]\times[1,1]$ is sufficient. For an odd number n^2 of grid points on the unit square this results in a reduction to an effective number of $(n^2+n)/2$ calculation points.

As an example, for the 101×101-grid shown in Figure 2 instead of solving a system of 9.801 coupled ODEs we get along with 4.851 equations. This gain in simulation resources (primarily memory and CPU time) can be used for additional grid refinement if necessary.

These thoughts reflect the fact that the PDE itself only holds in the *interior* of the domain and one has to treat the boundary gridpoints in a different way. In this sense, the nature of reasonable boundary conditions in grid points $\zeta = i\eta, \eta\in[-1,1]$ has yet to be clarified. Aside the axes of rotation we do not face any problems, DIRICHLET type conditions can be implemented as well as NEWTON/NEUMANN ones just like we previously did in our heat simulations on square and disk.

An heuristic approach to this boundary axes problem could claim smoothness of the temperature field when traversing the axes. This physically motivated demand would result in $\nabla_{\mathbf{n}} u = 0\ \forall\eta\in[-1,1]$.

Finally, a mathematical approach can also legitimate prescription of the homogeneous NEUMANN boundary condition on the η-axes: with knowledge of the modified Laplacian holding in the interior, examination of

$$\lim_{\xi\to 0+}(\xi_x u_x - \eta_x u_y)/\xi$$

reveals that zeroing the numerator is at least a necessary condition for the expression to make sense. Taking now into account that $\zeta = f(z)$ is η-symmetric, one first shows that the y-axes is mapped onto the η-axes. With $\xi\,(0,y)=0$, $\xi_x(0,y)\neq 0$ and $\eta_x(0,y)=0\ \forall y$ comes $u_x = 0$ and we are done (details in [5, p.63, fn.6]).

2.3 Dental laser source

Introducing a heat source into our model to simulate the thermal treatment with a laser equipment inserted into the dental root canal with successive pulsing or movement can be done in different ways. With respect to rotational symmetry of our 3D model developed so far focusing on the η-axes (representing the canal) for the sake of not breaking up the symmetry seems reasonable.

One practicable approach could be altering the homogeneous NEUMANN boundary condition on the η-axes to $\nabla_{\mathbf{n}} u = \alpha(0,\eta,t)$. But aside the necessity of a correct parameter identification it is also to be expected that we are running into numerical problems with this modeling approach. Moreover, adjusting the depth of heat penetration would also be a desirable aspect but cannot be incorporated by this modeling procedure.

Seeming more flexible is the introduction of an additional source term into the heat equation itself which holds in the *interior* of the region. For this purpose, a Gaussian curve in two dimensions would be appropriate, that is we make use of

$$\sigma(\xi,\eta,t) = u_{amp} \cdot \exp\left[\frac{-\xi^2 - (\eta - \eta_0 - vt)^2}{s}\right]$$

as our heating source base signal function. Being dependent on both space and time variables this signal can then be further adjusted to the desired needs within a simulation run. For $t = 0$, the signal starts moving at point $(0,\eta_0)$ up- or downwards the axes of symmetry with velocity v (positive or negative).

Penetration depth is controlled by constant s. As for the amplitude u_{amp}, we leave the correct unit determination as well as considerations concerning plausible numeric values to the physicist. Having a look at the two-dimensional normal distribution can extend the signal to pay tribute to a certain direction (using correlation).

Furthermore we see that grid points on the η-axes will not be affected by this modeling approach, indeed the homogeneous NEUMANN boundary condition, $\nabla_{\mathbf{n}} u = 0$, has to hold during the whole simulation run. The peak u_{amp} of the Gaussian is calculated on the boundary by solving this condition *within simulation*.

Finally, caused by the fact that we are dealing with an invariant, adding the source term $\sigma(\xi,\eta,t)$ to the heat equation has no further impact concerning conformal transformation laws. Yet another advantage compared to heating via boundary condition where transformation of the normal vector has to be considered.

3 Parameters and Results

In [3, p.43] we assumed for κ, the coefficient of thermal diffusion being responsible for heat propagation, for sake of convenience $\kappa = 1$. This is by chance not too far from reality, in literature [7, Fig.5] one finds for dentine and enamel $\kappa \approx 2 - 5 \times 10^{-7}[m^2 s^{-1}]$. Supposing $[mm]$ units like pointed out in the cross section figures thus entitles to define $\kappa = 0,3[mm^2 s^{-1}]$.

With $u_0 = 36\,^{\circ}C$ as prescribed initial temperature distribution the set of boundary conditions also has to be adapted to the level of this reference temperature u_0,

$$\nabla_{\mathbf{n}} u = \begin{cases} 0, & x = 0,\ y \in (-1,1) \\ \alpha_r(u_0 - u), & x \in (0,1),\ y = -1 \\ \alpha_r(u_0 - u), & x = 1,\ y \in (-1,1) \\ \alpha_c(u_0 - u), & x \in (0,1),\ y = 1. \end{cases}$$

Herein, for convenience the junction point from root to crown was determined to be – in computational coordinates (x,y) – at point $(1,1)$. With α_c and α_r being positive heat is being dissipated to the exterior. For the heat transition coefficients (α_c: enamel→air and root α_r: cementum→bone) one finds in literature [6, p.1640] $\alpha_r = 10^{-4}[W mm^{-2} K^{-1}]$ and $\alpha_c = 10^{-5}[W mm^{-2} K^{-1}]$.

Using a canonical model first, impact of the different geometries on the results and rate of convergence can be studied. Mean temperatures and temperatures at $\zeta = 0$ for different grid spacing are presented in Table 1.

Finally, for the results shown in Figure 7, a constantly emitting heating source with $u_{amp} = 400$ and $s = 2/10$ was placed at $t = 0$ in point $\zeta = 5i$. Moving down the axes of symmetry with velocity $v = -1/5$, the point $\zeta = -7i$ is reached within $t = 60s$.

Geo	Grid	$u(0)$	$\Delta\%$	$u\varnothing$	$\Delta\%$
	26×51	0.5076	—	0.4637	—
1	51×101	0.5079	0.05	0.4649	0.25
	101×201	0.5079	0.005	0.4654	0.1
	1.326	0.298	—	0.2745	—
2	5.151	0.2988	0.285	0.2761	0.59
	20.301	0.299	0.047	0.2765	0.16
	1.326	0.1975	—	0.1875	—
3	5.151	0.1998	1.17	0.1905	1.6
	20.301	0.2004	0.283	0.1913	0.42

Table 1: Investigating convergence ($t = 50$) on the three different cross-sections presented in [4, p.91].

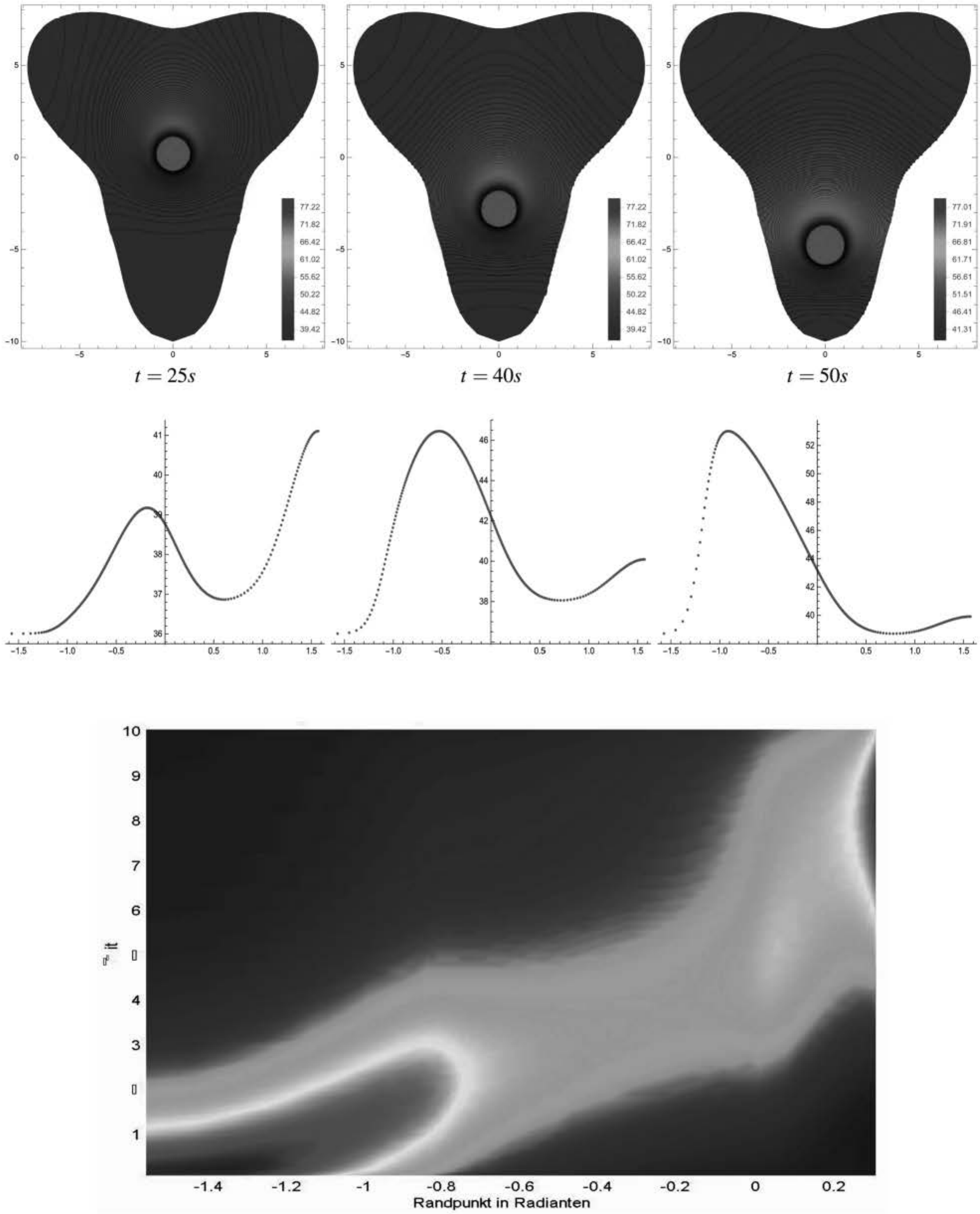


Figure 7: Simulation results: Isothermal lines for a heat source moving down from $\eta = 5$ to $\eta = -7$ (t) and associated temperature distributions on the boundary (c). Visualization on the boundary with time proceeding (b).

Summary

The present *SNE* article completes our conformal transformation simulation trilogy where it was our intention to point out an alternative approach to the classical method of Finite Elements by using Finite Differences on particular curvilinear grids to produce a system of coupled ordinary differential equations.

Hence, we demonstrated how conformal mappings are utilized to simulate two-dimensional or symmetric three-dimensional Initial Boundary Value Problems. Parametrization of a 2D-region by means of the unit square is achieved when combining the analytically given map from square to unit disk with numerically constructed conformal transformations. The introduction of such special co-ordinates thus makes the usage of rectangular structured computational grids possible.

The method of lines (we also focused on deriving generalized expressions for the sake of extensible approximation quality) can then be used to transform a given system of PDEs to a system of ODEs. The latter can be treated by standard methods available for systems with lumped parameters. All computations were carried out by using the program *Mathematica 11.3*.

Starting with Cartesian co-ordinates, linking up the derivatives of the map with the fundamental tensor is all to be done to get access to the metrics of the transformation. This approach proves to be highly flexible concerning an application to a wide range of PDEs.

In contrary, as theory of conformal mapping is bound to complex analysis, the field of application *a priori* is limited to two-dimensional domains which can be considered as a serious restriction. Knowing the special shape of the metric tensor in the conformal case, an attempt to construct an analogon in three dimensions would be an interesting topic for future research indeed.

Focusing on a thermal heat conduction problem for tooth laser treatment at last, the calculated temperature distribution on the boundary can be used to adjust source intensity for the sake of avoiding injuries. Possible further model extensions are discussed in [5, p.67ff].

References

[1] Douschan, P. *Benefit der Anwendung des Diodenlasers in der Endodontie.* Med. Univ. Graz, diploma thesis (in German), 2014.

[2] Henrici, P. *Applied and Computational Complex Analysis, Volume I.* John Wiley & Sons, New York-London-Sydney-Toronto 1974, ISBN 0-471-37244-7.

[3] Holzinger, M. *CTDS – Mauling Heat Equation on Unit Disk by Conformal Parametrization.* Simulation Notes Europe 2020; 30(2): 43-50. DOI: 10.11128/sne.30.tn.10511.

[4] Holzinger, M. *CTDS – High Precision Conformal Map of the Unit Disk to Star-Shaped Domains.* Simulation Notes Europe 2020; SNE 30(3): 85-94. DOI: 10.11128/sne.30.tn.10521.

[5] Holzinger, M. *Konforme Abbildungen zur Simulation von Modellen mit verteilten Parametern.* Dissertation/PhD (in German), Wien, 2020. http://katalog.ub.tuwien.ac.at/AC15652638

[6] Martins, M.A., Oliveira, L.S., Valle R.M. *Adaptivity for finite volume on unstructured triangular meshes: a study on thermal injury in teeth.* International Journal for Numerical Methods in Engineering, 2004; 61(10): 1625-1643.

[7] Panas, A.J., Żmuda, S., Terpiłowski, J., Preiskorn, M. *Investigation of the Thermal Diffusivity of Human Tooth Hard Tissue.* Intl. Journal of Thermophysics, Vol. 24, Number 3, pp. 837-848, 2003. Online ISSN 1572-9567.

A Stable but Explicit Cosimulation Coupling Method

Thilo Moshagen

Wismar Univ. of Applied Sciences, Mechanical Engineering/Process and Environmental Engineering,
Philipp-Müller-Straße 14, 23966 Wismar, Germany; *thilo.moshagen@hs-wismar.de*

SNE 31(1), 2021, 9-15, DOI: 10.11128/sne.31.tn.10552
Received: November 10, 2020 (Selected ASIM vSST 2020 Postconf. Publ.); Revised February 10, 2021;
Accepted: February 28, 2021
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna,
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. The term *co-simulation* denotes the coupling of some simulation tools for dynamical systems into one big system by having them exchange data at points of a fixed time grid and extrapolating the received data into the interval, while none of the steps is repeated for iteration. From the global perspective, the simulation thus has a strong explicit component. Frequently, among the data passed across subsystem boundaries there are flows of conserved quantities, and as there is no iteration of steps, system-wide balances may not be fulfilled: the system is not solved as one monolithic equation system. If these *balance errors* accumulate, simulation results become inaccurate. Balance correction methods which compensate these errors by adding corrections for the balances to the signal in the next coupling time step have been considered in past research. But establishing the balance of one quantity *a posteriori* due to the time delay in general cannot establish the balances of quantities that depend on the exchanged quantities, usually energy. In most applications from physics, the balance of energy is equivalent to stability. In this paper, a method is presented which allows users to choose the quantity that should be balanced to be that energy, and to accurately balance it. This establishes also numerical stability for many classes of stable problems.

Introduction

With the rise of simulation software for technical systems emerged the desire to couple those simulations in order to take into account the influence the systems exercise onto each other. In other words, these systems are now viewed as subsystems which form one big system.

One now wants to simulate this large system, using the subsystems' simulator software and coupling it by sharing data. What used to be a parameter when the systems were calculated separately is given now by a state variable of the other subsystem, reading:

$$\begin{aligned} S_1 : \dot{x}_1 &= f_1(x_1, x_2, z_1, z_2) && (1)\\ 0 &= g_1(x_1, x_2, z_1, z_2) && (2)\\ S_2 : \dot{x}_2 &= f_2(x_1, x_2, z_1, z_2) && (3)\\ 0 &= g_2(x_1, x_2, z_1, z_2). && (4) \end{aligned}$$

Here, the (x_1, x_2) are the differential, the (z_1, z_2) are the algebraic states. The setting generalizes to n subsystems in a straightforward way, and it includes parabolic partial differential equations. We require that the derivatives $d_{z_i} g_i$ have full rank. Such each of the S_i is an index-1 differential-algebraic system if the $(x_{k \neq i}, z_{k \neq i})$ are seen as parameters of it. The influence of x_2, z_2 in a split setting is therefore modeled by parameters u_{12} in S_1 and x_1, z_1 as parameters u_{21}:

$$\begin{aligned} S_1 : \dot{x}_1 &= f_1(x_1, z_1, u_{12}) && (5)\\ 0 &= g_1(x_1, z_1, u_{12}) && (6)\\ S_2 : \dot{x}_2 &= f_2(x_2, z_2, u_{21}) && (7)\\ 0 &= g_2(x_2, z_2, u_{21}). && (8) \end{aligned}$$

When coupled, the u_{ij} are determined by the coupling conditions

$$\begin{aligned} 0 &= h_{21}(x_1, z_1, u_{21}) && (9)\\ 0 &= h_{12}(x_2, z_2, u_{12}) && (10) \end{aligned}$$

that have to be fulfilled, and exchanged at fixed time nodes T_k. Between them, the u_{ij} are extrapolated. To establish coupling, the h_{ik} must be solvable with respect to the u_{ik}. The $d_{z_i} g_i$ have full row rank, too. Such, the differential-algebraic system given by Equations (5) - (10) is again of index 1.

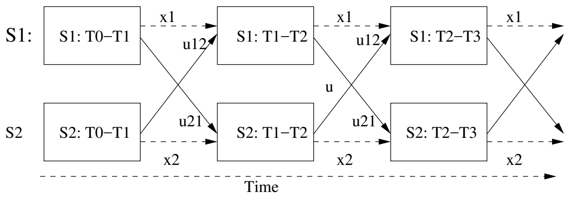


Figure 1: Explicit co-simulation scheme.

This description of the setting is widespread [2].

It is commonly said that the coupling is done by a *co-simulation scheme* if the u_{ij} are calculated from Equations (9) and (10) at exchange time nodes T_k and then passed on to S_2 and S_1, respectively. Of course, some extrapolation of u_{ij} into $[T_k, T_{k+1})$ is required. Considerable research has been done on coupling [6, 7, 5]. A lot of methods repeat the timestep after the calculation with an extrapolation that has been improved with respect to some objective. Thus, they are implicit, see e.g. [5]. For a convergence proof, see [2]. The convergence of explicit co-simulation methods for ODE and index one DAE not surprisingly as well improves with the extrapolation order of subsystems input [8, 2]. The situation here in simulator coupling mirrors the one in ODE solvers: The explicit solvers are quick in each step but not stable [12, 8], while the implicit ones require iterations within each step but usually ensure some stability. When used for stiff problems, the explicit schemes require such small stepwidths that the implicit schemes are finally cheaper. Also, implicit algorithms for coupled solvers require additional programming and storage. Therefore, the co-simulation scheme, where one just proceeds to the next timestep (Figure 1), is still popular.

So far, it has been common sense that the usual stability classifications like A- and B-Stability cannot be achieved with explicit algorithms [14]. A solution for these stability issues would be helpful in many applications and is the subject of this contribution.

It is important to note that all results and figures herein have been published before in [1]. This contribution is a highly condensed presentation of that content for the purpose of reaching the engineering community rather than novelty.

1 The Lack of Stability

1.1 Stability classifications

For readability, we present the concepts of stability classifications of methods.

Definition 1.1 (Stable points of ODE) *Let x^* be an equilibrium point of the ODE $\dot{x} = f(x)$ and $\phi^t x$ the solution for the initial value $x(t_0) = x$. Then x^* is*

- stable *if* $\forall \varepsilon > 0 \ \exists \delta > 0 : \|x - x^*\| < \delta \Rightarrow \|\phi^t x - x^*\| < \varepsilon \ \forall t \in [t_0, T]$
- asymptotically stable *if* $\exists r > 0 : \|x - x^*\| < r \Rightarrow \lim_{t \longrightarrow \infty} \phi^t x = x^*$.

Definition 1.2 (Stable Point of Difference equation) *Let x^* be an equilibrium point of the $k-th$ order difference equation $x_{n+1} = f(x_n, ..., x_{n-k})$. Then x^* is classified as in Definition* (1.1) *where x is replaced by x_n and $\forall t \in [t_0, T]$ by $\forall n \in \{1, ..., N\}$ and furthermore $t \longrightarrow \infty$ by $n \longrightarrow \infty$.*

Using these two definitions, stability classifications like zero-, A- or B-stability are defined: The respective stability of a method is the inheritance of the stability of an equilibrium point of a certain ODE class to the equilibrium point of the difference equation yielding from the application of the numerical scheme.

Stability, consistency and convergence. In this framework, zero stability of a numerical method means that the difference equation that one gets by applying the method to $\dot{x} = 0$ is stable. It is well-known that this is a necessary condition for convergence [13, 14]. But this condition is fulfilled by all one-step methods as $x_{n+1} = x_n + 0$ is a stable equation. (One-step methods can be written as $x_{n+1} = x_n + h\psi(x_n, t_n, h_n)$, and $\psi(x_n, t_n, 0) = f$, where f is the ODE's right hand side.) So unlike for multistep methods, there is no need here to examine zero-stability when one examines convergence of one-step methods. It frequently causes confusion that zero stability in the original paper [13] was labeled *stability* only, and with this nomenclature Lax's and Richtmyers' theorem is given in an equation-like form *stability* + *consistency* = *convergence*.

1.2 Stability

These results were confirmed numerically in [8, Sec.3.2] using the two-dimensional linear problem

$$\dot{x} = Ax, \tag{11}$$

which with

$$A = \begin{pmatrix} 0 & 1 \\ -\frac{c}{m} & 0 \end{pmatrix}, \qquad x = \begin{pmatrix} x \\ \dot{x} \end{pmatrix} \tag{12}$$

can be interpreted as linear spring-mass oscillator with mass m and spring constant c. This problem is the most simple problem possible that is linear and can be splitted. The original problem is marginally stable, so stable, as its spectrum is purely imaginary.

Spring		Mass
	System States	
$x_1 := s = x$		$x_2 := v = \dot{x}$
	Outputs	
$u_{21} := F = -cx$		$u_{12} := v = \dot{x}$
	Inputs	
u_{12}		u_{21}
	Equations	
$\dot{x_1} = \mathrm{Ext}(u_{12}) = v$		$\dot{x_2} = -\frac{1}{m}\mathrm{Ext}(u_{21}) = -\frac{F}{m}$

Spring		Mass
	System States	
…		…
	Outputs	
$u_{21} := (f, \dot{f}) = (-cx, -cv)$	(13)	$u_{12} := (v, a) = (\dot{x}, \dot{f}/m)$
	Inputs	
u_{12}		u_{21}
	Equations	
⋮		⋮

(14)

Table 1: Standard Co-simulation schemes for the spring-mass system, top constant, down linear extrapolation. When there is no difference, dots have been used.

Written as a co-simulation problem, Problem (11) with (12) yields Table 1. In [8] and [12] it is shown that co-simulation schemes are not stable for linear problems, even not for stable subsystem solvers. The stability for linear problems replaces the notion of A-stability, as the one-component equation used there cannot be split. When treated with a co-simulation scheme (output of the spring is the force $f = -cx$, that of the mass is the velocity $v = \dot{x}$) the emerging (method-induced) ODE

$$\begin{cases} \dot{x}_1 = a_{1,1}x_1 + a_{1,2}\,\mathrm{Ext}(x_2) \\ \dot{x}_2 = a_{2,2}x_2 + a_{2,1}\,\mathrm{Ext}(x_1) \end{cases} . \tag{15}$$

is obviously unstable [8, Section 2.5].

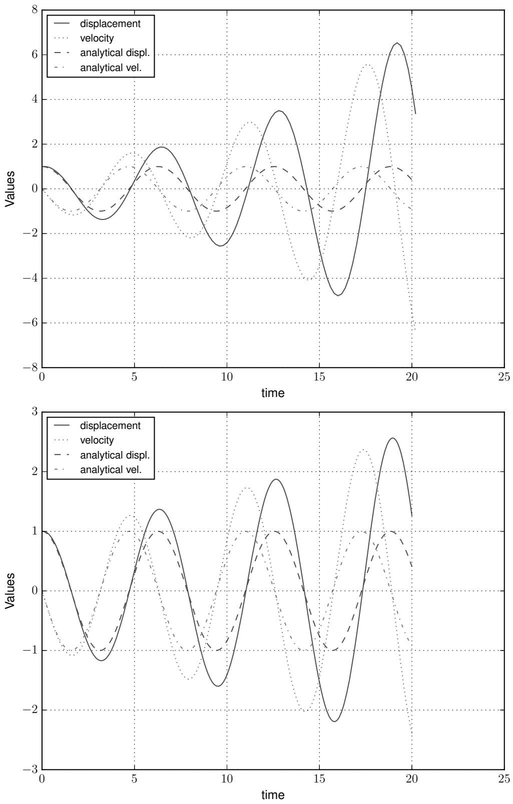


Figure 2: Simulation of the system (11)-(12) in the co-simulation scheme with constant extrapolation, varying the exchange step size H. Upper row, left: $H = 0.2$, right: $H = 0.1$. Previously published in [12].

Its numerical solution is shown in Figure 2, – there is no linear stability for general step sizes. This means $\|x\| \longrightarrow \infty$ for $t \longrightarrow \infty$. The energy of our system is $E = \frac{1}{2}mv^2 + \frac{1}{2}cs^2 = \langle x, \frac{1}{2}\,\mathrm{diag}(m,c)x \rangle = \langle x, x \rangle_{\frac{1}{2}\,\mathrm{diag}(m,c)} = \|x\|_{\frac{1}{2}\,\mathrm{diag}(m,c)}$, which is an equivalent norm, so lack of stability is equivalent to energy augmentation.

Using piecewise constant extrapolation of inputs, the force, as it is seen by the mass, is effectively shifted to later times: a value from time T_i is used for all future times $t \in (T_i, T_{i+1})$. The analogy with the reactive power and the real power of an electrical network is apparent. Work from oscillating systems with phase shift contains an integral over a constant and thus grows unbounded (Figure 2). Similar arguments hold for linear extrapolation co-simulation.

2 Enforcing Balance by Sharing the View on Potential Flow

2.1 The proposed method

The key feature to establish energy balance is exchanging the value of power and calculating the variable of interest from that power. Consider a co-simulation problem with subsystems S_1 and S_2 as given by Equations (5) - (10) with states x_1 and x_2 respectively and inputs u_{21} and u_{12}. We suggest the following procedure to enforce energy balance between subsystems S_1 and S_2:

1. At data exchange timepoint T_n the powers P_{ij} as the flux of energy are calculated in both subsystems, using up-to-date input u^n_{ji}. In general $P_{ij} \neq P_{ji}$.
   Applied to the $S_1 - S_2$ setting, P_{21} (the power calculated in S_1 for passing to S_2), is calculated using fresh u_{12}, the input into S_1. Now in the input vectors u_{ij} one component is replaced by P_{ij} and that new vector $u_{21,\text{bal}}$ is exchanged between the subsystems. Applied to $S_1 - S_2$ setting, the value P_{21} replaces one component $(u_{21})_m$ of u_{21}, and respectively, P_{12} replaces $(u_{12})_n$ of u_{12}.
   This means S_1's point of view about the power has been passed on to S_2 and vice versa.

2. Now both subsystems have the same information and thus the opportunity to draw the same conclusion on what energy exchange should be assumed. We denote this assumed energy exchange as

   $$\hat{P}_{12}(P_{21}, P_{12}) = -\hat{P}_{21}, \qquad (16)$$

   a straightforward choice is $\hat{P}_{21} = (P_{12} - P_{21})/2 = -\hat{P}_{12}$, where now it is necessary to define flow directions: P_{ij} shall be negative if it leaves S_j, so it is counted with opposite sign in S_i.
   Again remember that P_{21} is the power calculated in S_1 for passing to S_2, calculated using u_{12}, the input into S_1. The former input $(u_{12})_n(t)$ now is calculated subject to

   $$P_{21}(x_1(t), u_{12\backslash n}, (u_{12})_n(t)) = \text{Ext}(\hat{P}_{12}). \qquad (17)$$

   Analogously $(u_{21})_m(t)$ s.t. $P_{12}(x_2, u_{21\backslash m}, (u_{21})_m) = \text{Ext}(\hat{P}_{21})$ is calculated. The expression $12 \setminus k$ in subscript is to say that the k-th component of the vector is left out. For the unique inversion of P_{ij} it is required that the maps $(u_{ji})_k \longrightarrow P_{ij}(.,.,(u_{ji})_k)$ are strictly monotone.

   As $\text{Ext}(\hat{P}_{12}) = -\text{Ext}(\hat{P}_{21})$, now it is established that the inputs of S_1 and S_2 are consistent in terms of energy conservation for all t.

2.2 Example

To apply the scheme given in Section 2.1 above to a spring-mass system (12), replacing the standard co-simulation scheme from Table 1, one first calculates the energies of the systems parts, powers acting on subsystems boundaries, and their derivatives. As $P_i = \dot{W}_i$, $P_i < 0$ indicates that energy leaves S_i.

Spring	Mass
Energy	
$W = \int -f\,ds$ $= \int -fv\,dt$	$W = \int f\,ds$ $= \int mav\,dt$
Power	
$P = \dot{W}$ $= -fv = cxv$	$P = \dot{W}$ $= mav = fv$
Derivative of Power	
$\dot{P} = c(v^2 + sa)$	$\dot{P} = m(a^2 + v\dot{a})$ $= m(a^2 + v\frac{\dot{f}}{m})$

The derivative of force $\dot{f}$ is available as output of spring, as it is usually needed for linearly extrapolating the input. With this, the scheme yields Table 2.

Spring	Mass
System States	
$x_1 := s = x$	$x_2 := v = \dot{x}$
Outputs	
$(u_{21,\text{Std}})_1 := f = -cx$	$(u_{12,\text{Std}})_1 := v = \dot{x}$
$(u_{21,\text{Std}})_2 := \dot{f} = -cv$	$(u_{12,\text{Std}})_2 := \dot{v} = f/m$
(intermediately exchanging $u_{ij,\text{Std}}$)	
$(u_{21})_1 = P(x_1, u_{12})$ $= cxv = cx_1(u_{12})_1$	$(u_{12})_1 = P(x_2, u_{21})$ $= fv = (u_{21})_1 x_2$
$(u_{21})_2 = \dot{P}(x_1, u_{12})$ $= c(v^2 + xa)$ $= c((u_{12})_1^2 + x_1(u_{12})_2)$	$(u_{12})_2 = \dot{P}(x_2, u_{21})$ $= m(a^2 + v\frac{\dot{f}}{m})$ $= m\left(\frac{(u_{21})_1}{m}^2 + x_2\frac{(u_{21})_2}{m}\right)$
Inputs	
$(u_{12})_1 := \hat{P}$	$(u_{21})_1 := -\hat{P}$
$(u_{12})_2 := \hat{\dot{P}}$	$(u_{21})_2 := -\hat{\dot{P}}$
Input variables of standard method u_{std} depending on Power	
$v = \frac{\text{Ext}(\hat{P})}{cs} = \frac{\text{Ext}(u_{12})_1}{cx_1}$	$f = -\frac{\text{Ext}(\hat{P})}{v} = \frac{\text{Ext}(u_{21})_1}{x_2}$
Equations	
$\dot{x}_1 = v$	$\dot{x}_2 = \frac{f}{m}$

Table 2: Method form Section 2.1 applied to the spring-mass system.

3 Stability of power balanced schemes

As discussed in Section 1.2 and shown in [8], stability for linear systems of a partly explicite scheme is not given. This section shall relate energy conservation of our method to stability. The class of problems under consideration are all stable *gradient flow problems*

$$\dot{x} = -M\nabla_x \mathscr{P}^T, \tag{18}$$

which is a huge class, containing entropy driven and energy conserving problems. The *mobility Matrix M* determines the systems stability - it is positive definite if the system is dissipative and skew if energy conserving. This behavior must be inherited to the ODE that is induced by our splitting method. We give an outline of the arguments:

1. Switch to gradient flow view. In this, inserting (18) into the time derivative of the respective potential $\dot{\mathscr{P}}(x)$ yields

   $$\dot{\mathscr{P}}(x) = \langle \nabla_x \mathscr{P}(x), \dot{x} \rangle = \left\langle \nabla_x \mathscr{P}(x), -M\nabla_x \mathscr{P}(x)^T \right\rangle \tag{19}$$

   with the scalar product $\langle .,. \rangle$.

2. Introduce split system
   - Identify coupling contributions
   - Characterize potential conservation/dissipation properties (see below)

3. See method as decoupling ODE – Insert calculation of inputs from power into original equations

4. Relate decoupled ODEs stability properties to stability of original systems
   - Show that negotiated exchange conserves $\dot{\mathscr{P}} \leq 0$. It can be shown and there are straightforward arguments that there is no unphysical power production when sharing subsystems agree on the exchanged energy
   - Use Lyapunov's direct method on the decoupled system.
   - Additionally, one can argue that maximum stable stepwidth for dissipative systems is augmented (method is closer to B-stability than extrapolation of inputs method).

5. If such stable subsystems ODEs are solved with methods preserving that stability, overall solution will be stable.

Items (2) and also (4) need closer consideration. The split systems potential production $\dot{\mathscr{P}}(x)$ according to

Eq. (19) in subsystem-wise block matrix form reads

$$\dot{\mathscr{P}}(x) = P_k + P_l + \dots \tag{20}$$

$$= \begin{pmatrix} \cdot \\ (\nabla_x \mathscr{P}(x))_{I_k} \\ \cdot \\ (\nabla_x \mathscr{P}(x))_{I_l} \\ \cdot \end{pmatrix} \cdot \dots \tag{21}$$

$$\begin{pmatrix} & & * & & \\ \dots & -(M)_{I_k,I_k} & \dots & -(M)_{I_k,I_l} & \dots \\ & & * & & \\ \dots & -(M)_{I_l,I_k} & \dots & -(M)_{I_l,I_l} & \dots \\ & & * & & \end{pmatrix} \begin{pmatrix} \cdot \\ (\nabla_x \mathscr{P}(x))_{I_k} \\ \cdot \\ (\nabla_x \mathscr{P}(x))_{I_l} \\ \cdot \end{pmatrix} \tag{22}$$

$$= \underbrace{\left\langle \nabla_{x_{I_k}} \mathscr{P}(x), -M_{I_k,I_k} \nabla_{x_{I_k}} \mathscr{P}(x)^T \right\rangle}_{P_{kk}} \tag{23}$$

$$+ \underbrace{\left\langle \nabla_{x_{I_k}} \mathscr{P}(x), -M_{I_k,I_l} \nabla_{x_{I_l}} \mathscr{P}(x)^T \right\rangle}_{P_{kl}}$$

$$+ \underbrace{\left\langle \nabla_{x_{I_l}} \mathscr{P}(x), -M_{I_l,I_l} \nabla_{x_{I_l}} \mathscr{P}(x)^T \right\rangle}_{P_{ll}} \tag{24}$$

$$+ \underbrace{\left\langle \nabla_{x_{I_l}} \mathscr{P}(x), -M_{I_l,I_k} \nabla_{x_{I_k}} \mathscr{P}(x)^T \right\rangle}_{P_{lk}} + \dots,$$

we identify

$$P_{kl} := \left\langle (\nabla_x \mathscr{P}(x))_{I_k}, -(M)_{I_k,I_l} (\nabla_x \mathscr{P}(x)^T)_{I_l} \right\rangle \tag{25}$$

as the *potential production* in S_k by S_l's variables, or power acting from subsystem l onto subsystem k. Item (4) now means that those eliminate in the suggested scheme, as the exchanging subsystems agree on their value. So, there is no contribution to $\dot{\mathscr{P}}$ by the extrapolation during coupling.

Theorem 3.1 *For a Lyapunov stable (asymptotically stable) gradient flow initial value problem (IVP), the IVP resulting from the energy balancing method as described in Section 2.1 is also stable (asymptotically stable).*

4 Discussion, Conclusion and Future Work

The suggested method overcomes the decade-old issue of stability in coupled simulation for a huge class of problems.

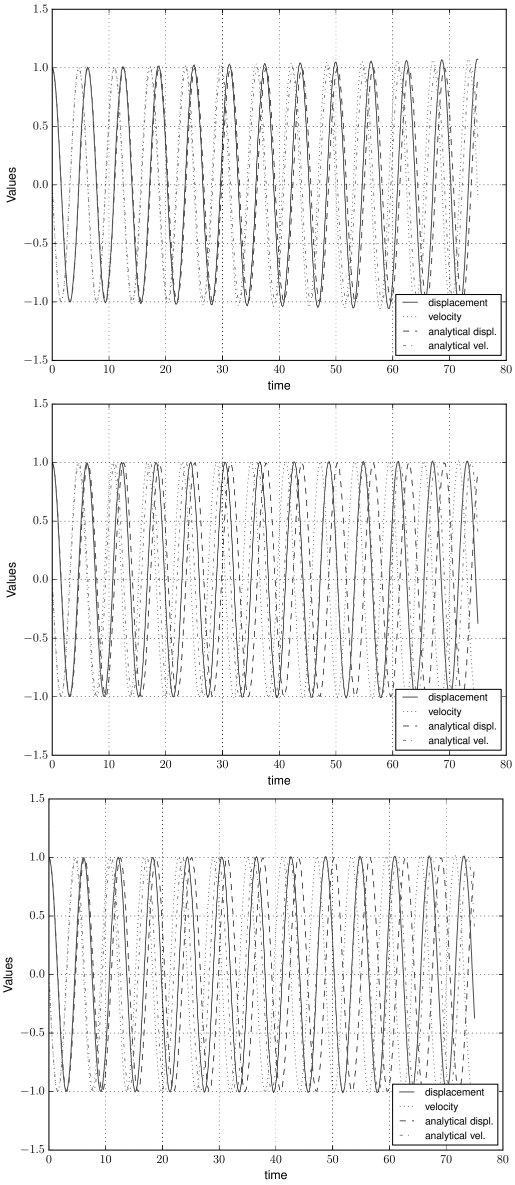


Figure 3: Stability of cosimulation schemes applied to spring-mass system: Top: Linear extrapolation, middle and bottom: Power balanced scheme. The image on bottem is an amplification of the lower right corner of the image above to confirm that there is no energy gain. $T_{\text{end}} = 75$, exchange stepwidth $H = 0.2$, subsystems refinement decisions left to subsystems solvers, stable *vode* used on subsystems.

Moreover, the method has a clear interpretation in physics: the enforcement of the power balance in systems interactions. It can therefore be implemented by anyone with understanding of the systems they want to couple, without deep knowledge of numerical analysis.

For simulations in industrial research and development, the new method enables stable calculations with big timesteps and few programming effort and is thus a big step forward.

References

[1] Moshagen, Thilo, *On Meeting Energy Balance Errors in Co-Simulations*, Mathematical and Computer Modelling of Dynamical Systems, 25, pp 139-166, Publisher: Taylor & Francis, 2019, Available at `doi:10.1080/13873954.2019.1595667`.

[2] M. Arnold and M. Günther, *Preconditioned Dynamic Iteration for Coupled Differential-Algebraic Systems*, BIT Numerical Mathematics 41 (2001), pp. 1–25, Available at `http://dx.doi.org/10.1023/A3A1021909032551`.

[3] M. Arnold, C. Clauss, and T. Schierz, *Error Analysis and Error Estimates for Co-Simulation in FMI for Model Exchange and Co-Simulation v2.0*, Progress in Differential-Algebraic Equations 60.1 (2013), pp. 75–94, Available at `doi:10.2478/meceng-2013-0005`.

[4] M. Arnold, C. Bausch, T. Blochwitz, C. Clauss, M. Monteiro, T. Neidhold, J.V. Peetz, and S. Wolf, *Functional Mock-up Interface for Co-Simulation* (2010), Available at `https://svn.modelica.org/fmi/branches/public/specifications/v1.0/FMI_for_CoSimulation_v1.0.pdf`.

[5] M. Busch, *Zur effizienten Kopplung von Simulationsprogrammen*, Ph.D. diss., Universität Kassel, 2012, Available at `http://books.google.de/books?id=0qBpXp-f2gQC`.

[6] R. Kübler and W. Schiehlen, *Modular Simulation in Multibody System Dynamics*, Multibody System Dynamics 4 (2000), pp. 107–127, Available at `http://dx.doi.org/10.1023/A:1009810318420`.

[7] S.B.E. Elmqvist, *Interface Jacobian-based Co-Simulation*, International Journal for Numerical Methods in Engineering 98 (2014), pp. 418–444, Available at `http://dx.doi.org/10.1002/nme.4637`.

[8] T. Moshagen, *Convergence of explicitly coupled Simulation Tools (Co-Simulations)*, Journal of Numerical Mathematics (2018), p. 27.

[9] R. Kossel, *Hybride Simulation thermischer Systeme am Beispiel eines Reisebusses*, Ph.D. diss., Techn. Univ. Braunschweig, 2012.

[10] D. Scharff, C. Kaiser, W. Tegethoff, and M. Huhn, *Ein einfaches Verfahren zur Bilanzkor-rektur in Kosimulationsumgebungen*, in *SIMVEC - Berechnung, Simulation und Erprobung im Fahrzeugbau*. 2012.

[11] M. Wells J.; Hasan and C. Lucas, *Predictive Hold with Error Correction Techniques that Maintain Signal Continuity in Co-Simulation Environments*, SAE Int. J. Aerosp (2012), pp. 481–493.

[12] D. Scharff, T. Moshagen, and J. Vondřejc, *Treating Smoothness and Balance during Data Exchange in Explicit Simulator Coupling or Cosimulation* (2017), p. 30, Available at `https://arxiv.org/abs/1703.05522`.

[13] P.D. Lax and R.D. Richtmyer, *Survey of the stability of linear finite difference equations*, Communications on Pure and Applied Mathematics 9 (56), pp. 267–293, Available at `https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.3160090206`.

[14] P. Deuflhard and F.A. Bornemann, *Numerische Mathematik II*, de Gruyter, 1994.

[15] *Scipy Integration and ODEs Web Documentation*. Available at `https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.ode.html`.

[16] K. Brenan, S. Campbell, and L. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, Society for Industrial and Applied Mathematics, 1995, Available at `http://epubs.siam.org/doi/abs/10.1137/1.9781611971224`.

[17] T. Moshagen, *Diffuse Grenzflächen thermodynamisch scharf - ein voll physikalisch eingebettetes Multiphasenfeldmodell*, Ph.D. diss., Universität Bremen, 2011, Available at `http://nbn-resolving.de/urn:nbn:de:gbv:46-00101865-17`, Online-Ressource (PDF: 208 S., 22,5 MB).

[18] A.V.D. Schaft, *Port-Hamiltonian Systems: an introductory Survey* (2006). Available at `http://www.icm2006.org/proceedings/Vol_III/contents/ICM_Vol_3_65.pdf`.

[19] A. van der Schaft and B.M. Maschke, *Port-hamiltonian Systems: a Theory for Modeling, Simulation and Control of Complex Physical Systems* (2003). Available at `http://www-lar.deis.unibo.it/euron-geoplex-sumsch/files/lectures_1/Van%20Der%20Schaft/VDSchaft_01_PCHS.pdf`.

[20] J. Dormand and P. Prince, *A Family of embedded Runge-Kutta Formulae*, Journal of Computational and Applied Mathematics 6 (1980), pp. 19 – 26, Available at `http://www.sciencedirect.com/science/article/pii/0771050X80900133`.

Investigation on Stability Properties of Hierarchical Co-Simulation

Irene Hafner[1]*, Niki Popper[2]

[1]dwh GmbH, Neustiftgasse 57–59, 1070 Vienna, Austria; *irene.hafner@dwh.at

[2]Institute of Information Systems Engineering, Research Unit of Information and Software Engineering, TU Wien, Favoritenstraße 9–11, 1040 Vienna, Austria

SNE 31(1), 2021, 17-24, DOI: 10.11128/sne.31.tn.10553
Received: November 10, 2020 (Selected ASIM vSST 2020 Post-conf. Publ.); Revised January 10, 2021;
Accepted: February 2, 2021
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna,
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. This paper introduces the concept of hierarchical co-simulation and presents an investigation on stability properties of this method. In conventional co-simulation methods, all participating simulations are executed on the same level via one co-simulation. Hierarchical co-simulation, on the other hand, enables the introduction of several levels of co-simulation by allowing participating subsystems to consist of co-simulated systems themselves, thus nesting co-simulations within co-simulations. While on the one hand, certain stability issues can arise by the introduction of more co-simulation layers, this method enables the usage of different synchronization references for parts of the overall system according to varying dependencies between the subsystems, which can increase accuracy and numerical stability.

Introduction

Co-simulation has become an important instrument to approach the simulation of large-scale heterogeneous systems in recent years. While definitions for the term co-simulation vary depending on the field of origin, in this paper we refer (in accordance with the terminology found in [1]) to co-simulation as the coupling of two or more simulations which differ in at least one aspect out of simulation tool, solver algorithm or step size.

Hierarchical approaches or multi-level descriptions have already been introduced in other fields within modeling and simulation (f.i. DEVS [2], multi-level agent-based modeling [3], MPC [4] or partitioned multi-rate approaches [5]). However, hierarchical co-simulation as explained in the following has to the authors' knowledge not been investigated up to now, although several frameworks and standards do not prohibit the realization of further co-simulations within a co-simulation. The idea is illustrated in Figure 1.

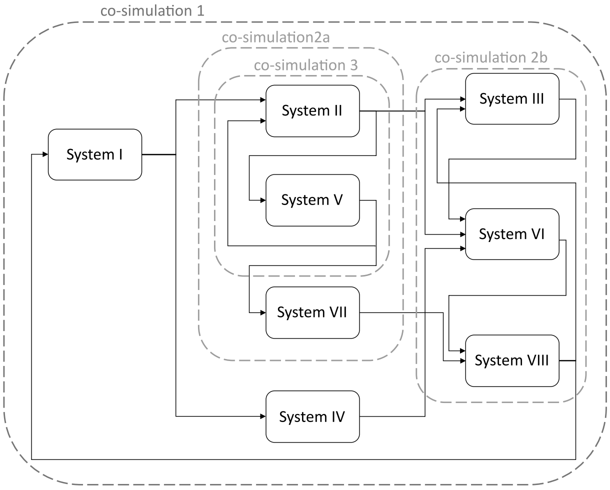


Figure 1: Schematic depiction of a hierarchical co-simulation approach.

In a traditional co-simulation approach all eight participating subsystems would have been co-simulated in one overall co-simulation, probably requiring all systems to synchronize at the same points in time. In the hierarchical approach, systems III, VI and VIII are coupled in another co-simulation (2b) as well as systems II and V, the coupled system of which (co-simulation 3) is again co-simulated with system VII, before the resulting co-simulation (2a) represents a system coupled in the top-level co-simulation 1.

A coupling structure like this could be motivated by the usage of different synchronization intervals on every co-simulation level, thus enabling frequent exchanges between subsystems which are sensitive to changes in their respective exchanged values while allowing larger communication intervals with other, slower reacting system parts, which can speed up the overall execution. In the course of this paper, we will show that these ideas are valid and that a hierarchically structured co-simulation approach indeed allows to enhance stability at a low computational cost.

1 Consistency

A valid method to bound the global co-simulation error is local error control, which justifies to investigate the consistency error, i.e. the error of the method in one step. For traditional co-simulation, it has been shown that consistency can be maintained, but possibly reduced to the extrapolation order of values from other systems, see for example [6, 17]. Since consistency is defined locally (i.e. per step), and it is a property regarded for the limit of step size $h \to 0$, the value present at the most recent point in time where the method sets a step is considered to be the exact solution - a property that is not affected by the method used in the respective other subsystems or the time steps and further synchronizations happening there in-between. This means that consistency in hierarchical co-simulation is also maintained with its order depending on the applied extrapolation method.

2 Zero-stability

Zero-stability, i.e. convergence of a method for infinitesimal step sizes, has been analyzed for certain co-simulation approaches in [7], on which we base our investigation. The mathematical description of coupled DAEs is given in [7] as follows:

$$\dot{\boldsymbol{x}}^i(t) = \boldsymbol{f}^i(\boldsymbol{x}^i, \boldsymbol{u}^i, t), \quad \boldsymbol{x}^i(t_0) = \boldsymbol{x}^i_0 \tag{1a}$$

$$\boldsymbol{y}^i(t) = \boldsymbol{g}^i(\boldsymbol{x}^i, \boldsymbol{u}^i, t) \tag{1b}$$

with $i = I, \ldots, N$, $\boldsymbol{x}^i \in \mathbb{R}^{n^i_x}$, $\boldsymbol{u}^i \in \mathbb{R}^{n^i_u}$, $\boldsymbol{y}^i \in \mathbb{R}^{n^i_y}$ and

$$\boldsymbol{u}^i = \boldsymbol{L}^i \boldsymbol{y} \tag{1c}$$

where

$$\boldsymbol{L}^i = \begin{bmatrix} \boldsymbol{L}^{i,I} & \ldots & \boldsymbol{L}^{i,i-1} & 0 & \boldsymbol{L}^{i,i+1} & \ldots & \boldsymbol{L}^{i,N} \end{bmatrix},$$

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}^I & \ldots & \boldsymbol{y}^{i-1} & \boldsymbol{y}^i & \boldsymbol{y}^{i+1} & \ldots & \boldsymbol{y}^N \end{bmatrix}^T$$

with $\boldsymbol{L}^{i,j} \in \mathbb{R}^{n^i_u \times n^j_y} \quad \forall i, j \in \{I, \ldots, N\}$ and the elements of $\boldsymbol{L}^{i,j}$ being equal to zero or one.

Under certain assumptions (given in [7], p. 100), the outputs can be written as $\boldsymbol{y}^i = \overline{\boldsymbol{g}}^i(\boldsymbol{x}^i) + \boldsymbol{D}^i(\boldsymbol{x}^i)\boldsymbol{u}^i$, yielding the discretized output equations

$$\boldsymbol{y}^i_{k+1} = \overline{\boldsymbol{g}}^i + \boldsymbol{D}^i \boldsymbol{u}^i_k \tag{2}$$

with constant $\overline{\boldsymbol{g}}^i$, $\boldsymbol{D}^i$. Using this, it holds for the outputs of global system

$$\boldsymbol{y}_{k+1} = \overline{\boldsymbol{g}} + \underbrace{\begin{bmatrix} \boldsymbol{0} & \boldsymbol{D}^I\boldsymbol{L}^{I,II} & \ldots & \boldsymbol{D}^I\boldsymbol{L}^{I,N} \\ \boldsymbol{D}^{II}\boldsymbol{L}^{II,I} & \boldsymbol{0} & \ldots & \boldsymbol{D}^{II}\boldsymbol{L}^{II,N} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{D}^N\boldsymbol{L}^{N,I} & \boldsymbol{D}^N\boldsymbol{L}^{N,II} & \ldots & \boldsymbol{0} \end{bmatrix}}_{=:\boldsymbol{D}} \boldsymbol{y}_k \tag{3}$$

that stability is guaranteed if the spectral radius ρ of $\boldsymbol{D}$ is less than or equal to 1. This is fulfilled in particular if $\rho(\boldsymbol{D}) = 0$ which for the case of two participating subsystems means that there is no algebraic loop.

To determine zero-stability properties of hierarchical co-simulation, a co-simulation of N systems is considered, where w.l.o.g. systems M to N are combined in a second-level co-simulation as depicted in Figure 2.

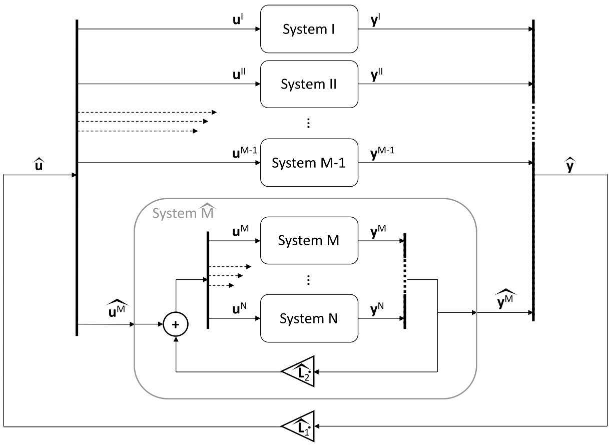


Figure 2: Hierarchical co-simulation of N systems on two levels.

System $\widehat{M}$ replaces systems M to N of the original co-simulation on one level (coupled via 1c and called CS_0 henceforth). We obtain coupling equations (4) for the upper co-simulation level CS_1:

$$\begin{bmatrix} \boldsymbol{u}^I \\ \boldsymbol{u}^{II} \\ \vdots \\ \boldsymbol{u}^{M-1} \\ \widehat{\boldsymbol{u}^M} \end{bmatrix} = \widehat{\boldsymbol{L}_1} \begin{bmatrix} \boldsymbol{y}^I \\ \boldsymbol{y}^{II} \\ \vdots \\ \boldsymbol{y}^{M-1} \\ \widehat{\boldsymbol{y}^M} \end{bmatrix} \tag{4}$$

with

$$\widehat{\boldsymbol{L}_1} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{L}^{I,II} & \dots & \boldsymbol{L}^{I,M-1} & \widehat{\boldsymbol{L}^{I,M}} \\ \boldsymbol{L}^{II,I} & \boldsymbol{0} & \dots & \boldsymbol{L}^{II,M-1} & \widehat{\boldsymbol{L}^{II,M}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{L}^{M-1,I} & \boldsymbol{L}^{M-1,II} & \dots & \boldsymbol{0} & \widehat{\boldsymbol{L}^{M-1,M}} \\ \widehat{\boldsymbol{L}^{M,I}} & \widehat{\boldsymbol{L}^{M,II}} & \dots & \widehat{\boldsymbol{L}^{M,M-1}} & \boldsymbol{0} \end{bmatrix}$$

and $\widehat{\boldsymbol{u}^M}$ as input to the new subsystem $\widehat{M}$, $\widehat{\boldsymbol{y}^M}$ as its output and

$$\begin{aligned} \widehat{\boldsymbol{L}^{i,M}} &= \begin{bmatrix} \boldsymbol{L}^{i,M} & \boldsymbol{L}^{i,M+1} & \dots & \boldsymbol{L}^{i,N} \end{bmatrix}, i = I \dots M-1 \\ \widehat{\boldsymbol{L}^{M,i}} &= \begin{bmatrix} \boldsymbol{L}^{M,i} \\ \boldsymbol{L}^{M+1,i} \\ \vdots \\ \boldsymbol{L}^{N,i} \end{bmatrix}, i = I \dots M-1. \end{aligned}$$

Thus, the only difference between $\boldsymbol{L}$ and $\widehat{\boldsymbol{L}_1}$ is the increased number of zeroes in the lower right corner.

The discretized output equations of CS_1 are (5):

$$\begin{aligned} \boldsymbol{y}^I_{k+1} &= \bar{\boldsymbol{g}}^I + \boldsymbol{D}^I \boldsymbol{u}^I_k \\ \boldsymbol{y}^{II}_{k+1} &= \bar{\boldsymbol{g}}^{II} + \boldsymbol{D}^{II} \boldsymbol{u}^{II}_k \\ &\vdots \\ \widehat{\boldsymbol{y}^M_{k+1}} &= \widehat{\bar{\boldsymbol{g}}^M} + \widehat{\boldsymbol{D}^M} \widehat{\boldsymbol{u}^M_k} \end{aligned} \tag{5}$$

While $\widehat{\boldsymbol{y}^M}$ in general corresponds to the stacked output vectors $\boldsymbol{y}^M \dots \boldsymbol{y}^N$ of CS_0, the input vectors don't as the coupling with the outputs of systems M to N is considered within the new system $\widehat{M}$, cf. Figure 2 and (6).

The outputs of the global system can with (5) be written as

$$\begin{bmatrix} \boldsymbol{y}^I_{k+1} \\ \vdots \\ \boldsymbol{y}^{M-1}_{k+1} \\ \widehat{\boldsymbol{y}^M_{k+1}} \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{g}}^I \\ \vdots \\ \bar{\boldsymbol{g}}^{M-1} \\ \widehat{\bar{\boldsymbol{g}}^M} \end{bmatrix} + \underbrace{\begin{bmatrix} \boldsymbol{D}^I & & & \boldsymbol{0} \\ & \ddots & & \\ & & \boldsymbol{D}^{M-1} & \\ \boldsymbol{0} & & & \widehat{\boldsymbol{D}^M} \end{bmatrix} \widehat{\boldsymbol{L}_1}}_{=:\boldsymbol{D}_{CS_1}} \begin{bmatrix} \boldsymbol{y}^I_k \\ \vdots \\ \boldsymbol{y}^{M-1}_k \\ \widehat{\boldsymbol{y}^M_k} \end{bmatrix}$$

In analogy to the case of one co-simulation level, the co-simulation of the upper level is stable if $\rho(D_{CS_1}) \leq 1$. The only unknown in comparison to $\boldsymbol{D}$ of CS_0 is $\widehat{\boldsymbol{D}^M}$, for which we have to take a look at the second-level co-simulation CS_2. The coupling equations within this system can be written (cf. Figure 2) as follows:

$$\begin{bmatrix} \boldsymbol{u}^M_k \\ \boldsymbol{u}^{M+1}_k \\ \vdots \\ \boldsymbol{u}^{N-1}_k \\ \boldsymbol{u}^N_k \end{bmatrix} = \widehat{\boldsymbol{L}_2} \cdot \begin{bmatrix} \boldsymbol{y}^M_k \\ \boldsymbol{y}^{M+1}_k \\ \vdots \\ \boldsymbol{y}^{N-1}_k \\ \boldsymbol{y}^N_k \end{bmatrix} + \widehat{\boldsymbol{u}^M_k} \tag{6}$$

where

$$\widehat{\boldsymbol{L}_2} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{L}^{M,M+1} & \dots & \boldsymbol{L}^{M,N-1} & \boldsymbol{L}^{M,N} \\ \boldsymbol{L}^{M+1,M} & \boldsymbol{0} & \dots & \boldsymbol{L}^{M+1,N-1} & \boldsymbol{L}^{M+1,N} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{L}^{N-1,M} & \boldsymbol{L}^{N-1,M+1} & \dots & \boldsymbol{0} & \boldsymbol{L}^{N-1,N} \\ \boldsymbol{L}^{N,M} & \boldsymbol{L}^{N,M+1} & \dots & \boldsymbol{L}^{N,N-1} & \boldsymbol{0} \end{bmatrix}.$$

The discretized output equations are

$$\boldsymbol{y}^i_{k+1} = \bar{\boldsymbol{g}}^i + \boldsymbol{D}^i \boldsymbol{u}^i_k,\ i = M \dots N. \tag{7}$$

Thus follows for the global output of CS_2

$$\widehat{\boldsymbol{y}^M_{k+1}} = \begin{bmatrix} \boldsymbol{y}^M_{k+1} \\ \vdots \\ \boldsymbol{y}^N_{k+1} \end{bmatrix} = \widehat{\bar{\boldsymbol{g}}^M} + \begin{bmatrix} \boldsymbol{D}^M & \dots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \dots & \boldsymbol{D}^N \end{bmatrix} \widehat{\boldsymbol{u}^M_k} \tag{8}$$

with

$$\widehat{\bar{\boldsymbol{g}}^M} = \begin{bmatrix} \bar{\boldsymbol{g}}^M \\ \vdots \\ \bar{\boldsymbol{g}}^N \end{bmatrix} + \begin{bmatrix} \boldsymbol{D}^M & \dots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \dots & \boldsymbol{D}^N \end{bmatrix} \widehat{\boldsymbol{L}_2} \begin{bmatrix} \boldsymbol{y}^M_k \\ \vdots \\ \boldsymbol{y}^N_k \end{bmatrix}.$$

The part containing $\boldsymbol{y}^i_k$, $i = M \dots N$ can be included in $\widehat{\bar{\boldsymbol{g}}^M}$ as these are only internal states of CS_2 which are unknown in CS_1. Hence we obtain

$$\widehat{\boldsymbol{D}^M} = \begin{bmatrix} \boldsymbol{D}^M & \dots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \dots & \boldsymbol{D}^N \end{bmatrix}, \tag{9}$$

which yields

$$\boldsymbol{D}_{CS_1} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{D}^I\boldsymbol{L}^{I,II} & \dots & \boldsymbol{D}^I\widehat{\boldsymbol{L}^{I,M}} \\ \boldsymbol{D}^{II}\boldsymbol{L}^{II,I} & \boldsymbol{0} & \dots & \boldsymbol{D}^{II}\widehat{\boldsymbol{L}^{II,M}} \\ \vdots & \ddots & \dots & \vdots \\ \boldsymbol{D}^{M-1}\boldsymbol{L}^{M-1,I} & \dots & \boldsymbol{0} & \boldsymbol{D}^{M-1}\widehat{\boldsymbol{L}^{M-1,M}} \\ \widehat{\boldsymbol{D}^M}\widehat{\boldsymbol{L}^{M,I}} & \dots & \widehat{\boldsymbol{D}^M}\widehat{\boldsymbol{L}^{M,M-1}} & \boldsymbol{0} \end{bmatrix}.$$

Due to

$$\widehat{\boldsymbol{D}^M}\widehat{\boldsymbol{L}^{M,i}} = \begin{bmatrix} \boldsymbol{D}^M & \dots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \dots & \boldsymbol{D}^N \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{L}^{M,i} \\ \boldsymbol{L}^{M+1,i} \\ \vdots \\ \boldsymbol{L}^{N,i} \end{bmatrix} = \begin{bmatrix} \boldsymbol{D}^M\boldsymbol{L}^{M,i} \\ \boldsymbol{D}^M\boldsymbol{L}^{M+1,i} \\ \vdots \\ \boldsymbol{D}^N\boldsymbol{L}^{N,i} \end{bmatrix}$$

and

$$\begin{aligned} \boldsymbol{D}^i\widehat{\boldsymbol{L}^{i,M}} &= \boldsymbol{D}^i \cdot \begin{bmatrix} \boldsymbol{L}^{i,M} & \boldsymbol{L}^{i,M+1} & \dots & \boldsymbol{L}^{i,N} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{D}^i\boldsymbol{L}^{i,M} & \boldsymbol{D}^i\boldsymbol{L}^{i,M+1} & \dots & \boldsymbol{D}^i\boldsymbol{L}^{i,N} \end{bmatrix}, \end{aligned}$$

the only difference compared to matrix $\boldsymbol{D}$ of system CS_0 is the increased number of zeroes in the lower right corner. In the following, we try to use this to gain information on the properties of the spectral radius of $\boldsymbol{D}_{CS_1}$ using knowledge on $\rho(\boldsymbol{D})$.

We know that for every matrix norm $\|.\|$ and arbitrary matrix $\boldsymbol{A} = (a_{ij})$; $i = 1,\dots,m$; $j = 1,\dots,n$; $m,n \in \mathbb{N}$

$$\rho(A) \leq \|\boldsymbol{A}\| \tag{10}$$

holds ([8], Thm. 5.6.9).

If we consider $\|.\|_\infty$ given as

$$\|\boldsymbol{A}\|_\infty = \max_{i=1,\dots,m} \sum_{j=1}^{n} |a_{ij}|$$

we see at once that $\|\boldsymbol{D}_{CS_1}\|_\infty \leq \|\boldsymbol{D}\|_\infty$. Unfortunately, this does *not* imply $\rho(\boldsymbol{D}_{CS_1}) \leq \rho(\boldsymbol{D})$, see f.i. the following example: Let matrices $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ given as

$$\boldsymbol{A}_1 = \begin{bmatrix} 0 & 0.1 & 0.5 & 0 \\ 0.1 & 0 & 0 & 0.5 \\ 0.2 & 0 & 0 & 0 \\ 0 & 0.2 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{A}_2 = \begin{bmatrix} 0 & 0.1 & 0.5 & 0 \\ 0.1 & 0 & 0 & 0.5 \\ 0.2 & 0 & 0 & -0.1 \\ 0 & 0.2 & -0.1 & 0 \end{bmatrix}.$$

Here $\|\boldsymbol{A}_1\|_\infty = \|\boldsymbol{A}_2\|_\infty = 0.6$ but $\rho(\boldsymbol{A}_1) \approx 0.3702 > \rho(\boldsymbol{A}_2) \approx 0.3317$. This means that in general, stability for hierarchical co-simulation has to be determined anew, even if the starting point is a zero-stable co-simulation on one level. An exception is the case where not only $\rho(\boldsymbol{D}) \leq 1$ but also $\|\boldsymbol{D}\|_\infty \leq 1$, as from this follows further

$$\rho(\boldsymbol{D}_{CS_1}) \leq \|\boldsymbol{D}_{CS_1}\|_\infty \leq \|\boldsymbol{D}\|_\infty \leq 1 \tag{11}$$

which ensures zero-stability of the co-simulation on the upper level CS_1.

For the stability properties of the coupling in CS_2, we are interested in the input-output dependencies within the system only, thus we need to look at the spectral radius of $\boldsymbol{D}_{CS_2}$, which results from (8):

$$\boldsymbol{D}_{CS_2} = \begin{bmatrix} \boldsymbol{D}^M & \dots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \dots & \boldsymbol{D}^N \end{bmatrix} \widehat{\boldsymbol{L}_2} \tag{12}$$

Since we see that $\boldsymbol{D}_{CS_2}$ is composed of a submatrix of $\boldsymbol{D}$, here again $\|\boldsymbol{D}_{CS_2}\|_\infty \leq \|\boldsymbol{D}\|_\infty$ holds, and thus $\rho(\boldsymbol{D}_{CS_2})$ has to be determined separately only if $\|\boldsymbol{D}\|_\infty > 1$.

To sum up, we can conclude that zero-stability of hierarchical co-simulation can be determined analogously to customary co-simulation on one level. To this end, the matrices referring to the global system outputs on every co-simulation level have to be examined - except for the cases where the origin is a stable co-simulation with matrix $\boldsymbol{D}$ fulfilling $\|\boldsymbol{D}\|_\infty \leq 1$, which is satisfied in particular for couplings where no feed-through occurs in at least one system, so $\|\boldsymbol{D}\|_\infty = \rho(\boldsymbol{D}) = 0$. These considerations, of course, can directly be taken further and applied to more than two levels of co-simulation, as well.

3 Numerical Stability

Depending on the coupling method, instabilities can still occur for zero-stable coupling methods due to the errors introduced by extra- or interpolation. A weak coupling approach is called *numerically stable* if it yields a stable solution for a finite macro-step size $H > 0$ [9].

To investigate stability properties for finite communication step sizes, we consider a three-mass oscillator as benchmark example, which is illustrated in Figure 3.

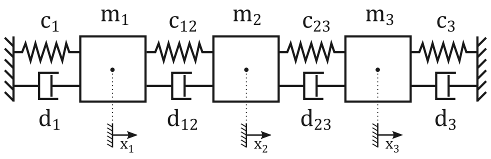


Figure 3: Illustration of a three-mass oscillator.

The underlying equation system can be interpreted as coupled Dahlquist equations, which can be solved analytically and thus provide an eminently suitable test case. The oscillator with two masses has been taken into consideration in numerous investigations on stability of conventional, single-level co-simulation approaches, where it proves highly sensitive to the choice of parameters and macro step size [9].

For the intended co-simulation, the system is split along the individual masses and coupled via force-displacement-coupling (cf. f.i. [10] for further information on the coupling approach), as illustrated in Figure 4.

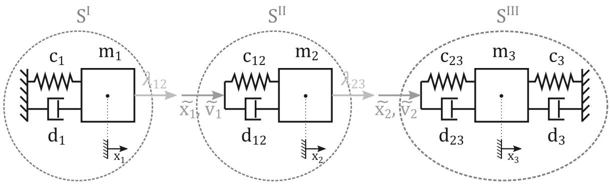


Figure 4: Force-displacement coupling of the three-mass oscillator.

By this coupling approach we obtain the subsystem equations for systems S^I, S^{II} and S^{III}:

$$S^I: \quad \begin{aligned} \dot{x}_1 &= v_1 \\ m_1\dot{v}_1 &= -c_1x_1 - d_1v_1 + \lambda_{12} \end{aligned} \tag{13a}$$

$$S^{II}: \quad \begin{aligned} \dot{x}_2 &= v_2 \\ m_2\dot{v}_2 &= -c_{12}(x_2 - \widetilde{x_1}) - d_{12}(v_2 - \widetilde{v_1}) + \lambda_{23} \end{aligned} \tag{13b}$$

$$S^{III}: \quad \begin{aligned} \dot{x}_3 &= v_3 \\ m_3\dot{v}_3 &= -c_{23}(x_3 - \widetilde{x_2}) - d_{23}(v_3 - \widetilde{v_2}) \\ &\quad + c_3(-x_3) + d_3(-v_3) \end{aligned} \tag{13c}$$

With the coupling conditions

$$\lambda_{12} - c_{12}(x_2 - x_1) - d_{12}(v_2 - v_1) = 0 \tag{14a}$$

$$\widetilde{x_1} - x_1 = 0 \tag{14b}$$

$$\widetilde{v_1} - v_1 = 0 \tag{14c}$$

$$\lambda_{23} - c_{23}(x_3 - x_2) - d_{23}(v_3 - v_2) = 0 \tag{14d}$$

$$\widetilde{x_2} - x_2 = 0 \tag{14e}$$

$$\widetilde{v_2} - v_2 = 0 \tag{14f}$$

Following the considerations from section 2, we obtain for matrix $\boldsymbol{D}$ in (3):

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{D}^I\boldsymbol{L}^{I,II} & \boldsymbol{D}^I\boldsymbol{L}^{I,III} \\ \boldsymbol{D}^{II}\boldsymbol{L}^{II,I} & \boldsymbol{0} & \boldsymbol{D}^{II}\boldsymbol{L}^{II,III} \\ \boldsymbol{D}^{III}\boldsymbol{L}^{III,I} & \boldsymbol{D}^{III}\boldsymbol{L}^{III,II} & \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -c_{12} & -d_{12} & 0 & 0 & 0 & 0 \\ 0 & 0 & -c_{23} & -d_{23} & 0 & 0 \end{bmatrix} \tag{15}$$

whence follows $\rho(\boldsymbol{D}) = 0$, thus guaranteeing zero-stability.

For the hierarchical co-simulation approach, systems S^{II} and S^{III} are combined in a second-level co-simulation.

As expected (cf. section 2), we obtain $\rho(\boldsymbol{D}_{CS_1}) = \rho(\boldsymbol{D}_{CS_2}) = 0$ for

$$\boldsymbol{D}_{CS_1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -c_{12} & -d_{12} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{16}$$

and

$$\boldsymbol{D}_{CS_2} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -c_{23} & -d_{23} & 0 & 0 \end{bmatrix}, \tag{17}$$

thus the conditions for zero-stability are satisfied for both levels of co-simulation.

In the following, several scenarios are performed for both co-simulation approaches to compare numerical stability properties. For all settings, explicit Euler methods are used to solve the individual subsystems. These simple methods have been chosen to enable the focus on the different methods of co-simulation without additional corrections (f.i. by step size control). As synchronization method, Jacobi-type coupling without iteration using zero-order extrapolation for external variables has been used. The initial conditions for all scenarios have been chosen as $x_1 = 1$, $x_2 = 2$, $x_3 = 3$ and $v_1 = v_2 = v_3 = 0$.

Scenario 1. The parameters for the first scenario to be considered are given in Table 1.

c_1	c_{12}	c_{23}	c_3	d_1	d_{12}	d_{23}	d_3	m_1	m_2	m_3
1E-02	1E-01	1	10	0.1	0.4	1	2	10	10	10

Table 1: Parameter settings for Scenario 1.

As can be seen, the spring stiffnesses are chosen to increase from left to right (cf. Figure 3) to result in slower and faster varying subsystems. The step sizes for the individual subsystem solvers are chosen accordingly with $h_1 = 0.005$, $h_2 = 0.0025$, $h_3 = 0.00125$. The monolithic reference system is of the form $\dot{\boldsymbol{y}} = \boldsymbol{A} \cdot \boldsymbol{y}$ and can thus be solved analytically. In addition to the analytical solution, the results of the hierarchical co-simulation are compared to a conventional single-level co-simulation. For the latter, a macro step size H of 0.1 seconds is chosen. The results in Figure 5 show that even if the overall communication step size H_1 is doubled in comparison to the traditional co-simulation, the hierarchical approach yields significantly more accurate results for systems S^{II} and S^{III} if the step size for the second-level co-simulation is chosen adequately ($H_1 = 0.2s$, $H_2 = 0.05s$).

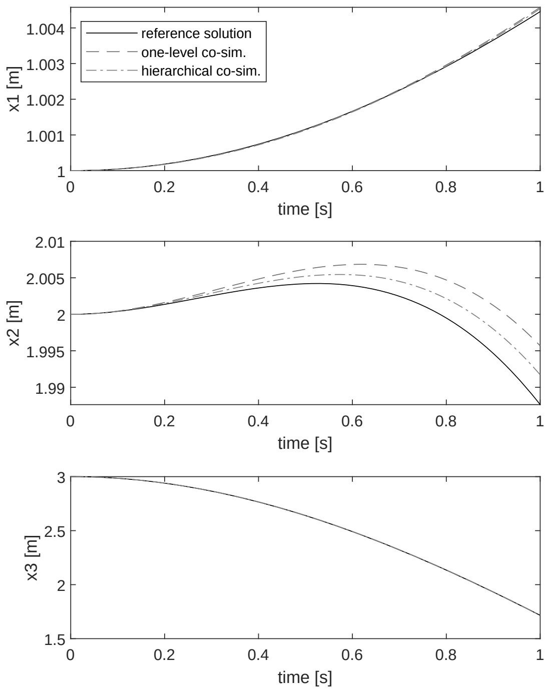


Figure 5: Trajectories of x_1, x_2 and x_3 for Scenario 1 with $H = 0.1s$, $H_1 = 0.2s$ and $H_2 = 0.05s$.

In spite of plainly distinct errors in specific phases, both approximations remain stable, as can be seen in simulations over a longer period of time.

The maximum absolute errors and elapsed time for several different settings are given in Table 2.

appr.	t_{end}	$H\|H_1$	H_2	err_{x_1}	err_{v_1}	err_{x_2}	err_{v_2}	err_{x_3}	err_{v_3}	el. time
trad.	1	0.1		1.21E-04	5.30E-04	8.05E-03	1.53E-02	9.51E-04	6.32E-04	0.0066
hier.	1	0.1	0.025	4.52E-05	2.89E-04	2.10E-03	3.67E-03	8.22E-04	1.38E-03	0.0161
hier.	1	0.2	0.05	8.26E-05	5.34E-04	4.08E-03	7.44E-03	8.51E-04	1.06E-03	0.0099
trad.	25	0.1		3.96E-02	7.13E-03	9.35E-02	3.68E-02	1.78E-02	1.24E-02	0.1845
hier.	25	0.1	0.025	1.76E-02	3.29E-03	2.14E-02	8.75E-03	6.63E-03	5.48E-03	0.4018
hier.	25	0.2	0.05	3.53E-02	6.64E-03	4.28E-02	1.75E-02	9.67E-03	7.20E-03	0.2514

Table 2: Maximum error and elapsed time for the traditional and hierarchical co-simulation approach in Scenario 1.

We see that while the execution time is significantly higher in case of the same step size on the upper level and the traditional co-simulation - which has to be expected due to the additional synchronization on the lower level - the high difference can be overcome while still maintaining better accuracy by increasing both macro step sizes in the hierarchical approach.

Scenario 2. In Scenario 2, the stiffnesses differ to a greater extent (see parameters in Table 3), which can lead to stability issues if communication step sizes are chosen too large. The conventional co-simulation already yields unstable results for the same step size as in Scenario 1 ($H = 0.1$). The solution obtained by the hierarchical approach with the same upper-level communication step size but additional synchronization between subsystems S^{II} and S^{II} remains stable.

c_1	c_{12}	c_{23}	c_3	d_1	d_{12}	d_{23}	d_3	m_1	m_2	m_3
1E-03	1E-01	10	100	0.1	0.4	1	2	10	10	10

Table 3: Parameter settings for Scenario 2.

Even for a larger communication step size on the upper level ($H_1 = 0.2$), stability is maintained with the hierarchical approach, as the coupling between systems S^{II} and S^{II} is the crucial one (cf. Figure 6).

If the synchronization time on the second level is also increased (to $H_2 = 0.05$), qualitative behavior is still maintained but errors are too high to consider the solution still acceptable (cf. Table 4).

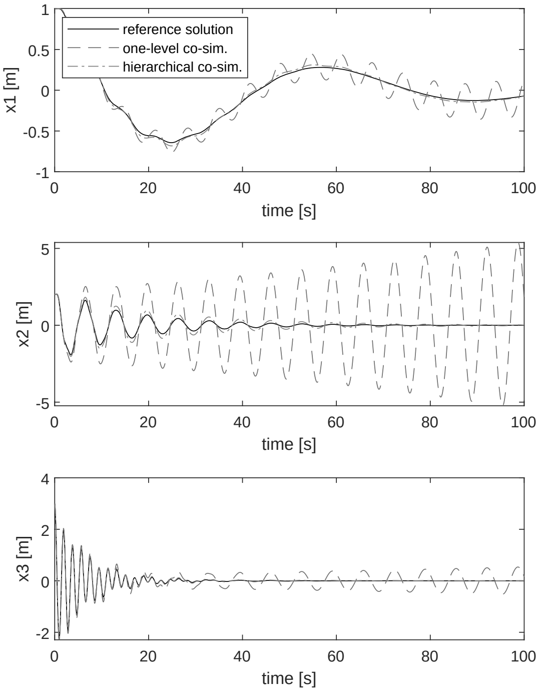


Figure 6: Trajectories of x_1, x_2 and x_3 for Scenario 2 with $H = 0.1s$, $H_1 = 0.2s$ and $H_2 = 0.025s$ from $t_{start} = 0s$ to $t_{end} = 100s$.

Scenario 3. In Scenario 3, the stiffnesses for the springs attached to mass m_1 are increased, too (see Table 5), which leads to unstable results for the traditional as well as hierarchical approach with step sizes $H = 0.1s$, $H_1 = 0.1s$ and $H_2 = 0.025s$, see Figure 7. This makes sense as the increased stiffness for System I can not be compensated by closer communication of Systems II and III.

The macro step sizes H and H_1 would have to be chosen as low as 0.03 to keep the error in bounds at all, even though results are still too far from the reference solution to be of use.

appr.	t_{end}	$H \vert H_1$	H_2	err_{x_1}	err_{v_1}	err_{x_2}	err_{v_2}	err_{x_3}	err_{v_3}	el. time
trad.	3	0.1		1.19E-02	1.56E-02	4.03E-01	4.61E-01	9.78E-02	2.37E-01	0.0202
hier.	3	0.1	0.025	7.59E-03	7.01E-03	9.54E-02	1.10E-01	5.01E-02	1.33E-01	0.0376
hier.	3	0.2	0.025	1.54E-02	1.44E-02	1.94E-01	2.24E-01	6.56E-02	1.66E-01	0.0372
hier.	3	0.2	0.05	1.46E-02	1.29E-02	9.48E-02	1.10E-01	5.00E-02	1.33E-01	0.0258
trad.	100	0.1		2.37E-01	2.28E-01	5.38E+00	5.06E+00	5.35E-01	5.06E-01	2.2885
hier.	100	0.1	0.025	2.57E-02	1.30E-02	2.90E-01	2.79E-01	6.86E-02	1.98E-01	6.4143
hier.	100	0.2	0.025	5.81E-02	3.08E-02	6.96E-01	6.64E-01	1.06E-01	2.53E-01	5.3847
hier.	100	0.2	0.05	3.90E-02	1.47E-02	2.88E-01	2.77E-01	6.85E-02	1.98E-01	2.5811

Table 4: Maximum error and elapsed time for the traditional and hierarchical co-simulation approach in Scenario 2.

c_1	c_{12}	c_{23}	c_3	d_1	d_{12}	d_{23}	d_3	m_1	m_2	m_3
1	10	10	100	0.1	0.4	1	2	10	10	10

Table 5: Parameter settings for Scenario 3.

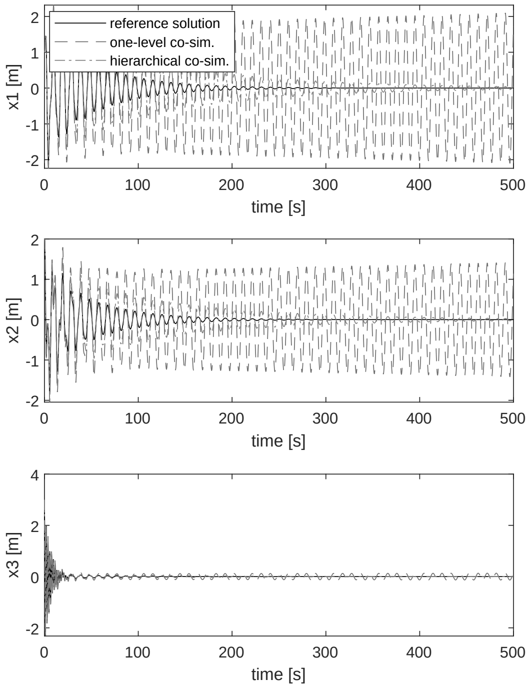


Figure 7: Trajectories of x_1, x_2 and x_3 for Scenario 3 with $H = 0.1s$, $H_1 = 0.1s$ and $H_2 = 0.025s$ from $t_{start} = 0s$ to $t_{end} = 100s$.

4 Conclusion

In this paper, the method of hierarchical co-simulation has been presented and investigated with respect to stability properties. In comparison to hierarchical multirate approaches as presented in [5, 12], the application of the hierarchical co-simulation method presented in this paper does not require any knowledge on the underlying system per se. The subsystems can, as in common co-simulation methods, be treated as black boxes with information on the input and output dependencies without interfering with the subsystem solvers. This can be beneficial when using co-simulation platforms like the BCVTB [11] or standards like the FMI[1], and in particular for interdisciplinary collaborative projects where partial systems are developed independently and possibly protected by company-specific privacy agreements.

While in the experiments above, quite simple subsystem solvers and coupling methods are chosen, improvement methods commonly used in single-level co-simulation approaches like variations of extrapolation order, coupling methods (sequential or mixed algorithms and waveform iteration) and stabilization techniques can of course be utilized in hierarchical co-simulation as well. Detailed studies on the advantages of said techniques for traditional co-simulation are ample in the literature (see for example [13, 14, 15, 16]). In addition, the results from section 3 show that stability issues can be tackled by introducing another layer of communication instead of having to decrease the overall communication step size, thus providing an innovative method for stabilization.

References

[1] Hafner I, Niki P. On the Terminology and Structuring of Co-simulation Methods. In Proceedings of the 8th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools. *EOOLT '17*; 2017; Weßling, Germany. New York, USA: ACM. 67-76. doi: 10.1145/3158191.3158203.

[2] Zeigler BP. *Object-Oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic Systems.*. Saint Louis: Elsevier Science; 2014.

[3] Morvan G. Multi-level agent-based modeling - A literature survey. *arXiv:1205.0561 [cs]*. 2013.

[4] Scattolini R. Architectures for distributed and hierarchical Model Predictive Control - A review. *Journal of Process Control*. 2009; 19(5): 723–731. doi: 10.1016/j.jprocont.2009.02.003.

[5] Günther M, Rentrop P. Partitioning and Multirate Strategies in Latent Electric Circuits. In: Bank RE, Gajewski H, Bulirsch R, Merten K. *Mathematical Modelling and Simulation of Electrical Circuits and Semiconductor Devices*. Basel: Birkhäuser; 1994. p 33–60. doi: 10.1007/978-3-0348-8528-7_3

[6] Trcka, M. *Cosimulation for Performance Prediction of Innovative Integrated Mechanical Energy Systems in Buildings* [dissertation]. Technische Universiteit Eindhoven; 2008.

[7] Kübler R, Schiehlen W. Two Methods of Simulator Coupling. *Mathematical and Computer Modelling of Dynamical Systems*. 2000; 6(2): 93–113. doi: 10.1076/1387-3954(200006)6:2;1-M;FT093.

[8] Horn RA, Johnson CR. *Matrix analysis*. 23. print. Cambridge: Cambridge Univ. Press; 2010.

[9] Busch, M. *Zur effizienten Kopplung von Simulationsprogrammen*. Kassel: Kassel University Press; 2012.

[10] Schweizer B, Lu D. Semi-implicit co-simulation approach for solver coupling. *Archive of Applied Mechanics*. 2014; 84(12): 1739–1769. doi: 10.1007/s00419-014-0883-5.

[11] Wetter M. Co-simulation of building energy and control systems with the Building Controls Virtual Test Bed. *Journal of Building Performance Simulation*. 2011; 4(3): 185–203. doi: 10.1080/19401493.2010.518631.

[12] Striebel, M. *Hierarchical Mixed Multirating for Distributed Integration of DAE Network Equations in Chip Design* [dissertation]. Bergische Universität Wuppertal; 2006.

[13] White J, Odeh F, Sangiovanni-Vincentelli AL, Ruehli AE. *Waveform Relaxation: Theory and Practice* [Technical Report]. [EECS Department]. University of California; 1985.

[14] Larsson J, Krus P. Stability Analysis of Coupled Simulation. In Dynamic Systems and Control, Volumes 1 and 2. *ASME International Mechanical Engineering Congress and Exposition*. 2003 Nov; Washington DC, USA. 861–868. doi: 10.1115/IMECE2003-41192.

[15] Arnold M. Stability of Sequential Modular Time Integration Methods for Coupled Multibody System Models. *Journal of Computational and Nonlinear Dynamics*. 2010; 5(3): 031003. doi: 10.1115/1.4001389.

[16] Schweizer B, Li P, Lu D, Meyer T. Stabilized implicit co-simulation methods: solver coupling based on constitutive laws. *Archive of Applied Mechanics*. 2015; 85(11): 1559–1594. doi: 10.1007/s00419-015-0999-2.

[17] Arnold M, Clauss C, Schierz T. Error Analysis and Error Estimates for Co-Simulation in FMI for Model Exchange and Co-Simulation V2.0. *Archive of Mechanical Engineering*. 2013; LX(1). doi: 10.2478/meceng-2013-0005.

[1] https://fmi-standard.org/

Model Generation for Multiple Simulators Using SES/MB and FMI

Hendrik Folkerts*, Thorsten Pawletta, Christina Deatcu

Research Group Computational Engineering and Automation, University of Applied Sciences Wismar,
Philipp-Müller-Straße 14, 23966 Wismar, Germany; **hendrik.folkerts@cea-wismar.de*

SNE 31(1), 2021, 25-32, DOI: 10.11128/sne.31.tn.10554
Received: November 10, 2020 (Selected ASIM vSST 2020 Post-conf. Publ.); Revised February 20, 2021;
Accepted: February 25,2021
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna, ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. This paper deals with the extension of a Python-based infrastructure for studying the characteristics and behavior of families of systems. The infrastructure allows automatic execution of simulation experiments with varying system structures as well as with varying parameter sets in different simulators. Special focus is put on the support of different simulation environments by creating models implementing the Functional Mockup Interface (FMI). Possible system structures and parameterizations are defined using a System Entity Structure (SES). The SES as a high level approach for variability modeling, particularly in simulation engineering, describes a set of system configurations, i.e. different system structures and parameter settings of system components. In combination with a Model Base (MB), executable models can be generated from an SES. Based on the extended SES/MB approach, tool-supported variability modeling and automatic model generation and execution in different simulation environments using FMI is described. This is done by means of an engineering application.

Introduction

This paper is based on [1] and [2]. It focuses on the more general approach using the tools for variability modeling introduced there by integrating with the Functional Mock-up Interface (FMI) instead of just offering simulator specific solutions.

The high variant diversity with components of different application fields in today's technical systems leads to the need for variability modeling and integration of varying simulation platforms. One application area for variability modeling is e.g. the generation of software for electronic control units, which is often generated by underlying models. Those models are usually of similar type, but still differ in structure and parameterization. To handle modeling and simulation of these so called families of systems, several approaches for variabilty modeling exist. Most approaches make use of 150% models, which means that all possible behavior is put into just one large and complex model and functionality is then adjusted by switching off unneeded model parts. In contrast to 150% modeling, in this paper we describe a method to define, generate and simulate well-tailored and therefore lean models by making use of the System Entity Structure / Model Base (SES/MB) approach. The SES/MB approach [3] originates in the systems theory community and has undergone many extensions over the years [4, 5]. It allows platform-independent variability modeling with subsequent platform-dependent model generation of specific variants. The structures of systems are coded in an SES, while the dynamic models are organized in an MB. The SES links to these dynamic models.

For this approach, a proposal for using one MB in several simulators is detailed. This is achieved by creating an MB of models which implement the FMI. The general tool independent standard for *model exchange* and *co-simulation* FMI [6, 7] enables the exchange of models between different simulators. This makes it possible to combine models from different domains and execute them in several simulation environments.

After the extended SES/MB approach is briefly introduced, the paper presents some software tools implementing the theory. An engineering application example is then discussed in detail to clarify the process of model definition and model generation using FMI.

1 SES/MB Theory and Implementation

This section briefly discusses the general SES/MB theory and the derived extended SES/MB (eSES/MB) infrastructure. Subsequently, an implementation of the infrastructure is presented.

1.1 SES/MB Basics and the eSES/MB Infrastructure

An SES is represented by a tree structure comprising entity nodes, descriptive nodes and attributes. A number of different system structures can be coded in one SES tree. In the context of modeling and simulation *entity nodes* are linked to *basic models* organized in an MB. Attributes of an entity node correspond to the parameters of the associated basic model. *Descriptive nodes* describe the relations among at least two entities and are divided into *aspect*, *multi-aspect* and *specialization* nodes.

In order to derive a specific system configuration all variation points are resolved by evaluating the rules at the descriptive nodes of the SES. This procedure is called *pruning*.

The resulting *Pruned Entity Structure (PES)* represents exactly one system configuration. In conjunction with an MB, a fully configured and executable model can be generated from the PES.

The basic SES/MB framework introduced in [3] was extended by new modeling features, methods and components [4, 5], such as an *Experiment Control (EC)* and an *Execution Unit (EU)* as shown in Figure 1. In this eSES/MB infrastructure, the EC uses an interface to the SES and its methods to derive goal-driven system configurations and to generate models, which are executed by the EU. The results returned by the EU are collected and analyzed by the EC. Thus, the derivation and generation of subsequent system configurations can be controlled reactively based on experiments already carried out.

A set of variables with global scope establish the interface to the SES. They are called *SES variables* (SESvar). *Semantic conditions* can be used to specify permitted value ranges and dependencies between SESvars. *SES functions* (SESfcn) are introduced for the specification of procedural knowledge. Complex variability can often be described more easily with SESfcns. Typical examples include the definition of varying coupling relations or the definition of variable parameter configurations in attributes. For automatic pruning, *selection rules* at descriptive nodes need to be defined, such as *aspectrules* for aspect and multi-aspect siblings or *specrules* at specialization nodes. A special mandatory attribute of multi-aspects is the attribute *number of replications* (numRep). The numRep attribute specifies the number of entities to create at a multi-aspect node during pruning. The *mb-attribute* of leaf entity nodes connects the entity node to a basic model in the MB. Attribute values and selection rules can be specified using SESvars or SESfcns.

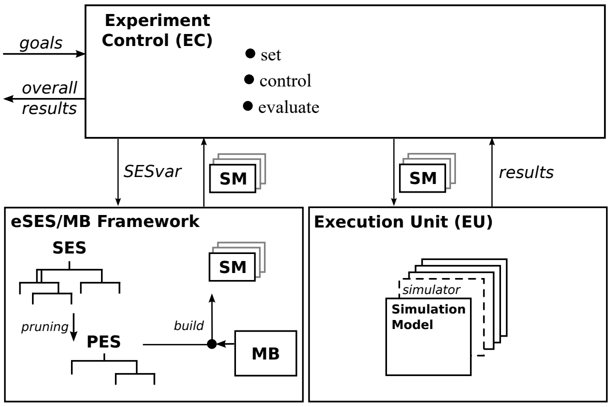


Figure 1: The eSES/MB infrastructure.

1.2 Software Tools

The eSES/MB framework as presented in the lower left part of Figure 1 was implemented in a prototype software tool in MATLAB [8]. The focus of this tool is the modeling and generation of MATLAB/Simulink models. In contrast to the MATLAB prototype, the objective of the software used in this paper is to support the generation and execution of models for different simulation environments. The infrastructure in Figure 1 is implemented as a Python framework as presented in Figure 2. The tools are called *SESToPy*, *SESMoPy*, and *SESEuPy* [9].

SESToPy (**S**ystem **E**ntity **S**tructure **To**ols **Py**thon) implements a graphical editor and all SES related methods. In the editor an SES tree can be specified interactively in a file browser view and attributes and rules can be defined for every node. In addition to the pruning method already mentioned, SESToPy supports some more methods such as *merging* different SES and *flattening* for removing the hierarchy information. Applying the flattening method, a *Flattened Pruned Entity*

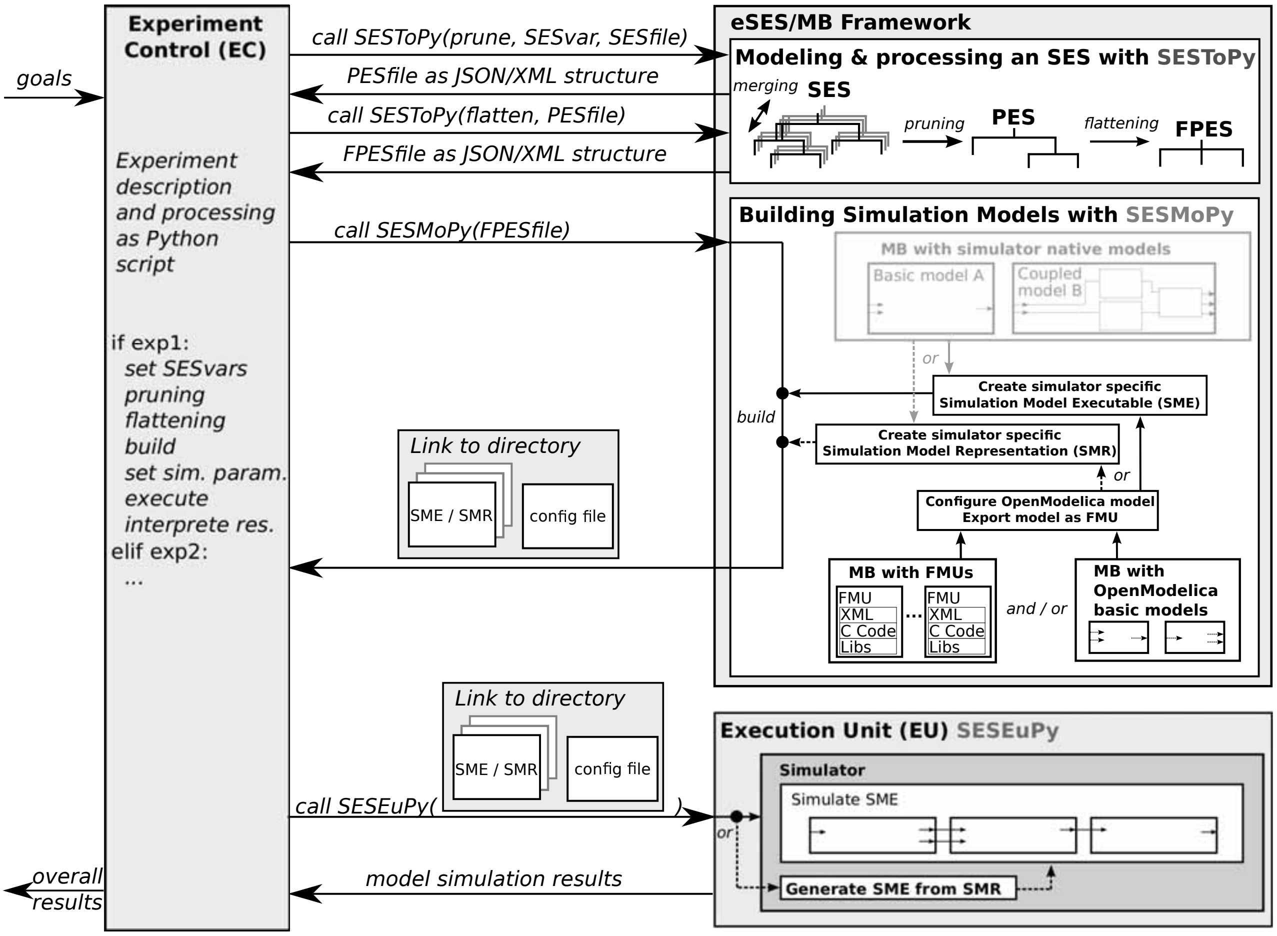

Experiment Control (EC)
goals
Experiment description and processing as Python script
if exp1:
set SESvars
pruning
flattening
build
set sim. param.
execute
interprete res.
elif exp2:
...
overall results
call SESToPy(prune, SESvar, SESfile)
PESfile as JSON/XML structure
call SESToPy(flatten, PESfile)
FPESfile as JSON/XML structure
call SESMoPy(FPESfile)
Link to directory
SME / SMR
config file
call SESEuPy( )
model simulation results
eSES/MB Framework
Modeling & processing an SES with SESToPy
merging
SES
pruning
PES
flattening
FPES
Building Simulation Models with SESMoPy
MB with simulator native models
Basic model A
Coupled model B
or
build
Create simulator specific Simulation Model Executable (SME)
Create simulator specific Simulation Model Representation (SMR)
or
Configure OpenModelica model Export model as FMU
MB with FMUs
FMU
XML
C Code
Libs
and / or
MB with OpenModelica basic models
Execution Unit (EU) SESEuPy
Simulator
Simulate SME
or
Generate SME from SMR

Figure 2: Python-based eSES/MB infrastructure for multiple EUs.

Structure (FPES) is derived.

For generating executable models, SESMoPy (**S**ystem **E**ntity **S**tructure **Mo**del builder **Py**thon) was developed. SESMoPy is a model builder, which implements the *build* method in two different ways and supports several simulation environments. For both approaches, all corresponding basic models must be organized in MBs, as shown in Figure 2. The first approach, called *native model generation*, is the generation of executables for a specific EU using a simulator specific MB. The second approach, this paper focuses at, is the *model generation based on FMI*. A Functional Mock-up Unit (FMU) is a model that implements an FMI [10]. In an FMU models are described by differential, algebraic, and discrete equations with time, state, and step events. In the scope of SESMoPy *FMI for Model Exchange* is used, which enables the simulation environment to generate C code of the FMU. *FMI for Co-simulation* is not discussed in this paper. The generalized interface FMI is supported by a number of established simulators [11], such as Simulink, OpenModelica or Dymola discussed for the use with SESMoPy.

Using the FMI-based approach, an MB with basic models from the simulator OpenModelica and/or an MB with FMUs are defined. SESMoPy creates an OpenModelica model and configures it according to the information passed in the FPES. Thus FMUs in an MB need to be imported into OpenModelica. The configured OpenModelica model is exported as FMU. Depending on the target simulator a specific Simulation Model Executable (SME) or Simulation Model Representation (SMR) is created. Finally SESMoPy returns a link to a directory, where the SME or SMR is placed together with a configuration file with information on the SME or SMR.

Information about the way the model is created can be provided in the EC calling SESMoPy or at the SES level according to the SES enhancements in [4].

The Python software tool SESEuPy (**S**ystem **E**ntity **S**tructure **E**xecution **u**nit **Py**thon) acts as a general EU. It implements a kind of wrapper for the integration of different simulation environments into the framework. SESEuPy takes the link to the directory with the SME or SMR and reads the configuration file. If the model is given as SMR, an SME needs to be built. The SME then can be simulated in the target simulator and simulation results are returned.

In the next section, the components and functionality of the Python framework are explained using the example of an engineering application.

2 Engineering Application

A feedback control system can be modeled using transfer functions describing the behavior of the components in frequency domain. Controlled variables in a feedback control system are usually influenced by disturbances. A common approach for minimizing the influence of predictable disturbances is adding a feedforward control. The system can be mapped to a signal-flow oriented model. In the following paragraphs it is described how the eSES/MB infrastructure can be used to design and test such a system using the introduced tools and FMI-based model generation in combination with the simulation programs Matlab/Simulink, OpenModelica, and Dymola.

2.1 Problem Description

A process unit with a *PT1* behavior shall be controlled using a PID controller. A disturbance with a *PT1* behavior affects the output of the process unit. Different configurations of the PID controller shall be tested. If a defined regulatory goal is met, the current configuration of the PID controller is taken. Otherwise the structure is varied by adding a feedforward control to the system and different configurations of the PID controller are analyzed again. Figure 3 depicts a schematic representation of the application.

The system's behavior follows the *PT1* transfer function in Equation 1 and the step-shaped disturbance affects the output of the process unit with a *PT1* behavior according to Equation 2. The optional feedforward control is realized by subtracting the disturbing signal calculated by Equation 3 from the manipulated variable.

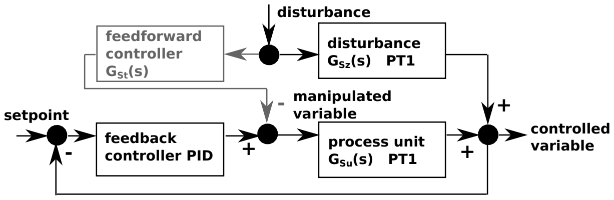

disturbance
feedforward controller $G_{St}(s)$
disturbance $G_{Sz}(s)$ PT1
-
manipulated variable
+
setpoint
feedback controller PID
+
process unit $G_{Su}(s)$ PT1
+
controlled variable
-

Figure 3: Structure of the feedback control system with optional feedforward control.

The control goals are a settling time of less than 15 seconds and a maximum overshoot of less than 5% after a disturbance.

The system has two structure variants, either without or with the feedforward control part, and a range of different configurations for the PID controller can be applied for each structure variant. In the next section, the two structure variants and their possible configurations are specified as an SES.

$$G_{Su}(s) = \frac{1}{20 \cdot s + 1} \tag{1}$$

$$G_{Sz}(s) = \frac{1}{10 \cdot s + 1} \tag{2}$$

$$G_{St}(s) = \frac{G_{Sz}(s)}{G_{Su}(s)} = \frac{20 \cdot s + 1}{10 \cdot s + 1} \tag{3}$$

2.2 Variant Modeling with SESToPy

The specification of the SES describing the feedback control system is done with the tool SESToPy. The tree and all attributes are defined via a graphical user interface. During modeling the SES with SESToPy, checks on the SES and plausibility tests are executed indicating model errors. The SES is saved as a JSON structure.

Figure 4 depicts the SES and its representation in SESToPy. The SES uses some extensions introduced in [5]. In addition to the different system configurations, essential parts for the configuration of simulation experiments are defined.

The root node *exp* of the SES and its subsequent aspect node *expDEC* describe a set of simulation based parameter studies for different system structures. The subtree of the entity node *simModel-ctrlSys* specifies the two system structures, i.e. a variant with and a variant without feedforward controller. The other two entity nodes specify experiment related information: The entity node *simMethod* specifies a target simulation environment for performing simulation runs using the SES-

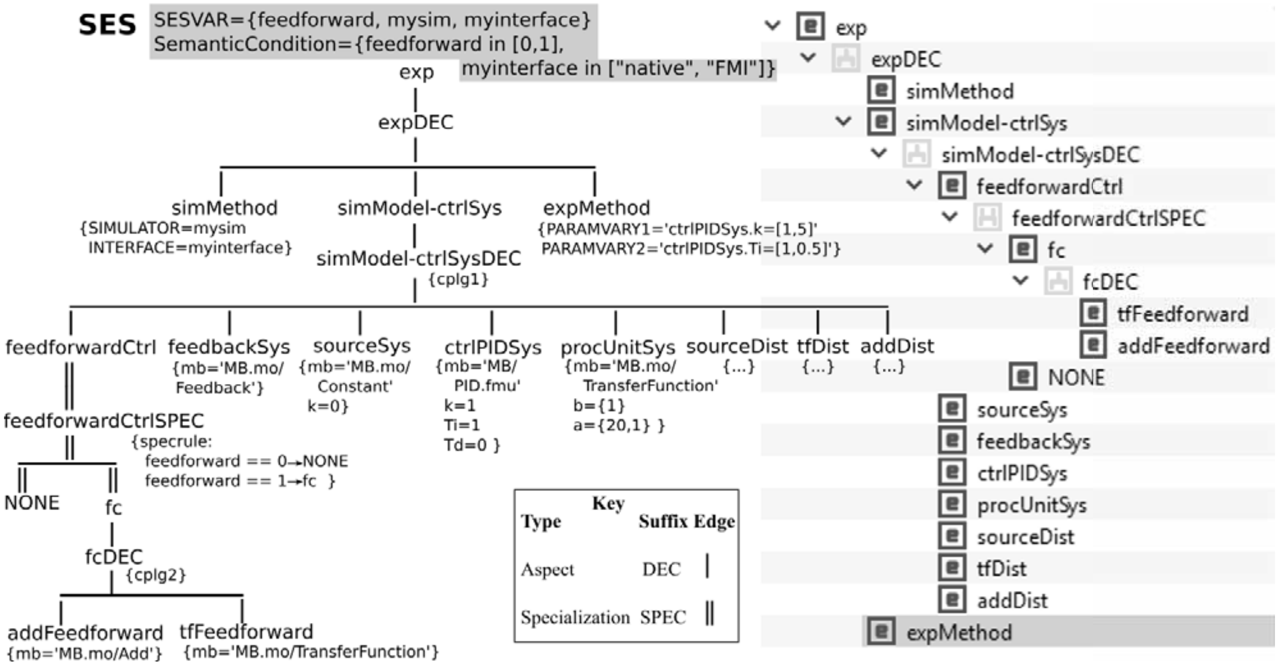


Figure 4: Left: SES specifying the feedback control system study; Right: Part of the SES representation in SESToPy.

var *mysim*. The SESvar *myinterface* specifies whether to use the native or the FMI model generation. Other simulation execution parameters, such as the simulation period, are not specified and are set by the EC later. The entity node *expMethod* specifies the permitted value ranges of two parameters for the PID controller. Besides the different system structures, they are the subject under study. The aspect *simModel-ctrlSysDEC* describes that each system variant consists of the following entities: *feedbackSys*, *sourceSys*, *ctrlPIDSys*, *procUnitSys*, *sourceDist*, *tfDist* and *addDist*. They are mandatory system elements. The optional feedforward control is specified by the subtree of entity *feedforwardCtrl*. The coupling relations of both structure variants are defined in the attribute *cplg1* of aspect *simModel-ctrlSysDEC*.

According to [12], optional parts in an SES are expressed by a specialization node where one of its children is a NONE element. A NONE element means that the entity is not included at all. The selection at a specialization is defined by an attribute called *specrule*. The specrule of the specialization *feedforwardCtrlSpec* defines that either the entity *fc* or *NONE* is selected during pruning. The result of evaluating the specrule at node *feedforwardCtrlSPEC* depends on the value of the SESvar *feedforward*. The SESvar codes the two possible structure variants as values 1 or 0. Therefore, the semantic condition $feedforward \in [0,1]$ applies to the SESvar. The entity *fc* and its subsequent aspect *fcDEC* specifies the feedforward control structure as a composition of the two entities *tfFeedforward* and *addFeedforward*.

Aspects and multi-aspects can define coupling relations as attribute. Couplings specify a composition of entities, which can be linked to basic models. Coupling attributes are abbreviated with *cplg* in Figure 4. Due to the varying system structures specified in the SES, the couplings in attribute *cplg1* of aspect node *simModel-ctrlSysDEC* are defined using an SESfcn. The coupling definitions in *cplg2* at node *fcDEC* are invariable and can therefore be defined without using an SESfcn.

According to Section 1, each leaf node defines an mb-attribute referring to a basic model in the MB. The basic model can be an OpenModelica component or an FMU. The other attributes of the leaf nodes define properties to configure the linked basic models. The values for *k* and *Ti* specified at node *ctrlPIDSys* are only default values, which will be overwritten because they are parameters under study.

2.3 Creating an MB

OpenModelica is an open source simulation platform and defines a set of basic models. It is widely used in different fields of engineering. In this case study OpenModelica basic models as well as FMU basic models are used. The FMUs define the FMI and can thus be exported from any simulation environment.

For the OpenModelica MB a *package* is created that contains basic models. This package is stored as the *file MB.mo* and is referred to as *local OpenModelica library* in this paper. Furthermore the FMUs with the *fileending *.fmu* are stored in a folder on the local filesystem, which is referred to as *local FMU library* in this paper.

The local OpenModelica library is filled with the following basic models whose names correspond to the names in the mb-attributes of the leaf nodes in the SES:

- *Constant* as the setpoint for the controlled variable
- *Feedback* for closing the feedback control loop
- *TransferFunction* for representing the process, the disturbance's behavior, and the feedforward
- *Add* for adding signals

In the local FMU library FMUs are placed like listed. The names correspond to the names in the mb-attributes of the leaf nodes in the SES.

- *Step.fmu* for stimulating the disturbance
- *PID.fmu* is the controller of the feedback control system

Each basic model can be configured according to the attributes of the leaf node which they are linked to in the SES. The local OpenModelica library as well as the local FMU library act as MB for the basic models.

2.4 Experiment Execution

For executing simulation based experiments the experiment process and its goals need to be defined in a Python script. This script implements the EC according to Figure 2. The Python framework provides some EC related template scripts. The goals of the experiment were discussed in Section 2.1. The experiment should start with the study of different PID controller configurations using the control system structure without feedforward controller. The simulation is executed with the simulators OpenModelica, Dymola, and Simulink. In case that the objectives are not achieved by just varying the parameters *k* and *Ti* of the PID controller, the study shall be carried out with the additional feedforward control structure and the simulation programs OpenModelica, Dymola, and Simulink. A snippet of the EC script with essential steps of the experiment process is given next.

```
...
SESfile = ...
if conditions_for_experiment:
  #prune, flatten, build, and execute
  SESvar = [mysim = <simulator>,
            myinterface = "FMI",
            feedforward = 0]
  PESfile = SESToPy("prune",SESvar,
                                SESfile)
  FPESfile = SESToPy("flatten",PESfile)
  smHandle = SESMoPy("build",FPESfile)
  sim_param = [solver=<solver>, ...]
  results = SESEuPy("simulate",smHandle)
...
elif conditions_for_experiment:
  #prune, flatten, build, and execute
  SESvar = [mysim = <simulator>,
            myinterface = "FMI",
            feedforward = 1]
  PESfile = ...
  ...
...
```

The EC starts the experiment by setting the SESvars *mysim*, *myinterface*, and $feedforward$. A target simulator is set for *mysim*. Next, the EC calls SESToPy's API method for pruning with the current SESvar values and a reference to the file defining the SES as JSON structure. The pruning process results in a PES coded as JSON structure. Afterwards, the EC calls SESToPy's API method for flattening the PES. The created FPES is similar to the FPES shown in Figure 5, which represents the more complex FPES for the later SESvar assignment $feedforward = 1$. A reference to the file containing the FPES as a JSON structure is returned to the EC. The EC then calls SESMoPy's API method for the build method and passes the FPES file handle. SESMoPy determines the target simulator from the attribute at the node *simMethod* and the value ranges of the PID controller parameters under study from the attribute at node *expMethod* in the FPES.

Based on the information in the FPES and the basic models from the MB, SESMoPy creates an Open-

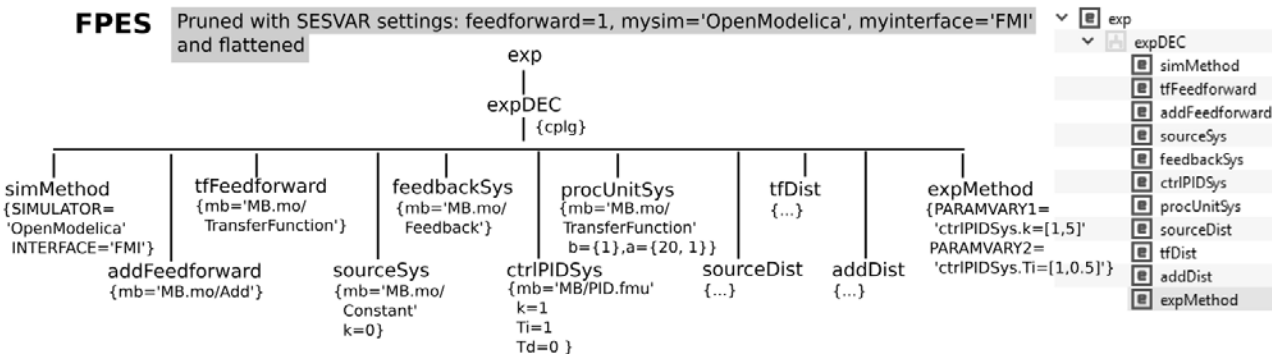


Figure 5: Left: FPES to study the feedback control system structure with feedforward; Right: FPES representation in SESToPy.

Modelica model for each configuration of the simulation model of the control system. FMU basic models need to be imported into OpenModelica. Figure 6 shows the structure of a fully configured OpenModelica model, but with feedforward controller, i.e. for the SESvar assignment $feedforward = 1$. The configured OpenModelica model is exported as FMU, which is called *model FMU* in this context. This model FMU is simulator independent, since it implements the FMI. It represents an SM. Thus only one MB needs to be defined for use with multiple target simulators.

Depending on the target simulator different steps are necessary as discussed before. (i) A Simulation Model Executable (SME) for the target simulators OpenModelica and Dymola or (ii) a Simulation Model Representation (SMR) for the target simulator Simulink is created.
(i) The SME is built by importing the model FMU into the target simulator. Using the interface of the FMU simulator specific code is generated of the model. For execution a file with simulator specific instructions on the execution is generated. Furthermore a configuration file with information about the SME and its target simulator is created.
(ii) The SMR is a file with simulator specific instructions for the import of the FMU in the target simulator. The file is not executed yet. Furthermore a configuration file with information about the SMR and its target simulator is created.

SMs of one structure variant have different configurations of the PID controller. A handle to the directory with all SMs is returned by SESMoPy to the EC, referred to as *smHandle*. The EC extends the configuration file with simulation data, such as the solver to use or simulation start and stop time. The EC calls the tool SESEuPy and passes the smHandle as the link to the SMs and the configuration file. In collaboration with the target simulation environment, SESEuPy controls the execution of an SM. An SME can be executed directly, whereas during execution of an SMR an SME is built. Finally, SESEuPy returns the simulation results to the EC.

In case the results meet the experimental goals, the overall results are calculated and returned by the EC. In case the goals are not reached, the second system structure with the additional feedforward controller by the SESvar assignment $feedforward = 1$ is set and a new model configuration and generation is started.

If the experimental goals have been achieved, the overall results of the experiment are the necessary control structure and the appropriate PID controller parameter settings. Otherwise the failure to achieve the objectives may also be established.

In addition simulation with another simulator can be tested. In the SESvar *mysim* another simulator is set and the model generation and simulation process is started over with the structure variant without feedforward controller. In this way, model by model validation is achieved using different simulators.

3 Conclusion

In this paper the extension for working with FMI of some Python-based software tools for variant modeling are presented. The entire process of variability modeling beginning with the system specification with an SES up to automatic variant derivation, model building, and execution is described. The proposed eSES/MB infrastructure makes it possible to model and simulate engineering problems using different target simulation environments.

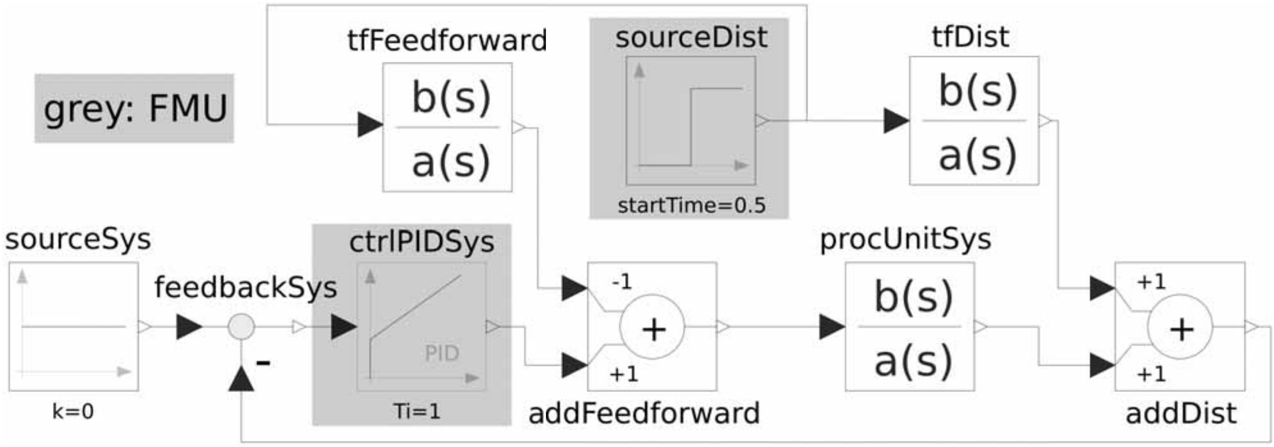


Figure 6: OpenModelica SM of the feedback control system with feedforward control.

References

[1] Folkerts H, Deatcu C, Pawletta T, Hartmann S. Python-based eSES/MB Framework: Model Specification and Automatic Model Generation for Multiple Simulators. *SNE Simulation Notes Europe.* 2019; 29: 207–215. doi: 10.11128/sne.29.tn.10497.

[2] Folkerts H, Deatcu C, Pawletta T, Hartmann S. A Python Framework for Model Specification and Automatic Model Generation for Multiple Simulators. *2019 International Interdisciplinary PhD Workshop*; 2019 May; Wismar. IEEE. doi: 10.1109/IIPHDW.2019.8755423.

[3] Zeigler BP, Kim TG, Praehofer H. *Theory of Modeling and Simulation.* 2nd ed. Cambridge: Academic Pr.; 2000. 510 p.

[4] Schmidt A, Durak U, Pawletta T. Model-Based Testing Methodology Using System Entity Structures for MATLAB/Simulink Models. *SIMULATION: Transactions of The Society for Modeling and Simulation International.* 2016; 92(8): 729–746.

[5] Schmidt, A. *Variant Management in Modeling and Simulation Using the SES/MB Framework* [dissertation]. University of Rostock; 2019.

[6] Blochwitz T, Otter M, Arnold M, Bausch C, Clauß C, Elmqvist H, Junghanns A, Mauss J, Monteiro M, Neidhold T, Neumerkel D, Olsson H, Peetz JV, Wolf S. The Functional Mockup Interface for Tool independent Exchange of Simulation Models. In Proc. of the 8th International Modelica Conference. *Modelica Conference*; 2011 March; Dresden, Germany. p 105–114. doi: 10.3384/ecp11063.

[7] Blochwitz T, Otter M, Akesson J, Arnold M, Clauß C, Elmqvist H, Friedrich M, Junghanns A, Mauss J, Neumerkel D, Olsson H, Viel A. Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models. In Proc. of the 9th International Modelica Conference. *Modelica Conference*; 2012 Sept; Munich, Germany. p 173–184. doi: 10.3384/ecp12076173.

[8] Pawletta T, Pascheka D, Schmidt A, Pawletta S. Ontology-Assisted System Modeling and Simulation within MATLAB/Simulink. *SNE Simulation Notes Europe.* 2014; 24: 59–68. doi: 10.11128/sne.24.tn.102241.

[9] Research Group CEA. *Python-Based eSES/MB Infrastructure.* 2019. www.github.com/hendrikfolkerts, last accessed 2020/09/10.

[10] Modelica Association Project "FMI". *Functional Mock-up Interface for Model Exchange and Co-Simulation.* 2014. https://svn.modelica.org/fmi/branches/public/specifications/v2.0/FMI_for_ModelExchange_and_CoSimulation_v2.0.pdf, last accessed 2019/08/06.

[11] Modelica Association Project "FMI". *Functional mock-up interface for model exchange and co-simulation.* 2019. https://fmi-standard.org/tools/, last accessed 2020/02/21.

[12] Deatcu C, Folkerts H, Pawletta T, Durak U. Design Patterns for Variability Modeling Using SES Ontology. In Proc. of Spring Simulation Multi-Conference 2018. *Spring Simulation Multi-Conference*; 2018 Apr; Baltimore/MD, USA. SCS Int. p 3:1–3:12.

Solving ARGESIM Benchmark C21 'State Events and Structural-dynamic Systems' with Simulink

Rebecca Heitmann, Peter Junglas*, Lisa-Kristin Petrucha

Department of Engineering "Dr. Jürgen Ulderup", PHWT Vechta/Diepholz, Schlesierstr. 13a, 49356 Diepholz, Germany; **peter@peter-junglas.de*

SNE 31(1), 2021, 33-42, DOI: 10.11128/sne.31.bn21.10556
Received: July 14, 2020; Revised: October 10, 2020;
Accepted: October 20, 2020
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. To pinpoint the problems that come with the modeling and simulation of hybrid systems, the ARGESIM C21 benchmark 'State Events and Structural-dynamic Systems' describes three such systems and a lot of corresponding tasks. It is solved here using the well-known simulation environment Simulink, the solutions are based on a direct modeling of the ODEs or DAEs describing the systems. To this end, special schemes have been necessary sometimes, some of which are already provided by Simulink, others had to be modeled explicitely.

Introduction

The Argesim C21 benchmark [1] requires to study three different examples of hybrid systems: a bouncing ball, an RLC circuit with a diode and a rotating pendulum with a free flight phase. A complete solution has been published before that is based on Modelica components [2]. The results presented in the following use the well-known Simulink simulation environment from Mathworks [3] without relying on additional packages for discrete system modeling such as Stateflow [4] or SimEvents [5]. Unlike in [2] modeling doesn't start with the physical components, but from the differential equations that are used to describe the systems.

The complete definition of the example systems and the tasks can be found in the benchmark definition [1]. For conciseness we will not reproduce results that are identical to those in [2], but simply quote the corresponding plots and tables. Instead we will concentrate mainly on the different implementation methods.

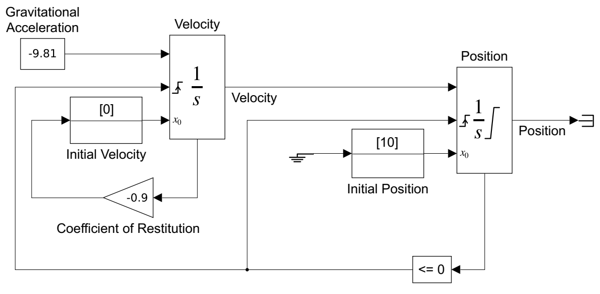
All models and scripts necessary to reproduce the results presented here are available from [6]. They have been prepared using Simulink Version 10.1 (R2020a) under Kubuntu 18.04.

1 Case Study Bouncing Ball

The first example describes a mass falling under gravity and air drag, which bounces off the ground. The bouncing process is described either as a simple event or as a continuous process, using a basic material model.

1.1 Event Contact Model

Description of model implementation. The bouncing ball model (without air drag) and the difficulties it poses for zero-crossing detection, have been widely studied by Mathworks [7]. In fact it is used as an example in Simulink that is discussed at length in the documentation. It is implemented there in two different ways. The first one with two separate integrators (and parameters adapted to the benchmark) is shown in Figure 1. It uses a lower saturation limit of the x integrator and external resets of x and v integrators to create the bounce event.



Copyright 1990-2013 The MathWorks, Inc.

Figure 1: Bouncing ball with event contact and two integrators ([8], parameters adapted).

With standard solver parameters it stops with an error ("too many consecutive zero crossing events"), when approaching the Zenon point. To overcome this problem one can use the optional adaptive algorithm for zero-crossing. It stops bracketing the event after too many events in a short time or if the function variation becomes too small. This works in principle, but the velocity results show a lot of remaining chatter. Changing the relative tolerance of the solver from 1e-3 to 1e-6 and manually adapting the signal threshold of the zero-crossing algorithm, the chatter amplitude is reduced by a factor of 20.

To get rid of the chatter the Simulink library contains a second-order integrator, which is mainly the combination of two connected integrators. The external reset of v, when x reaches 0, is done internally, and a consistent behaviour of x and v at saturation is enforced. With this block the benchmark model including air drag becomes very simple (cf. Figure 2). It uses the standard zero-crossing algorithm, but works fine nevertheless: x and v are exactly zero after the Zenon point.

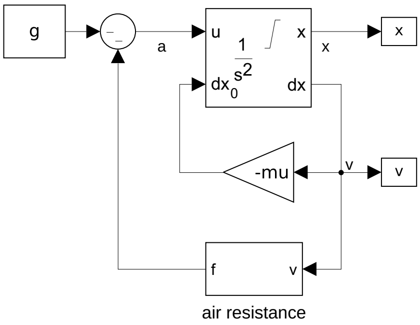


Figure 2: Bouncing ball with event contact and second-order integrator.

Simulation until last bounce – scattering prevention. The Zenon point in the free fall case (without air drag) is easily computed from [1, eq (16)] giving $t_{B,\infty}$ = 27.129019 s. A workaround to prevent event scattering is not necessary, the standard model shown in Figure 2 just works fine. It uses the ode23 solver and a relative and absolute tolerance of 10^{-6}. A simple Matlab script that finds the start time of the final zero values of x gives the results $t_{B,\infty}$ = 27.129019 s for the free fall case and $t_{B,\infty}$ = 25.589465 s when adding air resistance.

Testing accuracy of event handling. To determine the bounce times the standard model exports the simulated values of x and a Matlab script extracts the times where x equals 0.0. This is possible easily in spite of the usual floating point problems, since the saturation event enforces the exact value of 0. Figure 3 shows the difference between the theoretical values and the simulation results for a model without air resistance.

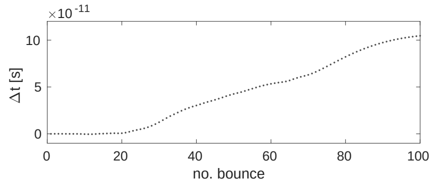


Figure 3: Accuracy of bounce times.

Compensation of linear model deviation. This task asks to introduce an initial velocity v_0 to the linear model so that the same Zenon point is reached as in the nonlinear model. As has been shown in [2] this is not possible, but one has to a add a velocity to the nonlinear model instead. Its value has been computed there as $v_0 = 4.39563\,\mathrm{m/s}$. Using this value in the standard model, the simulation results are the same as in [2, Fig. 3].

1.2 Model with Continuous Contact

Description of model implementation. The bouncing ball model with continuous contact is implemented by directly reproducing the differential equations, where the changing of the right hand sides according to the phase is done with `Switch` components. The complete model is shown in Figure 4, where the equation of the ball deformation is hidden inside a subsystem (cf. Figure 5).

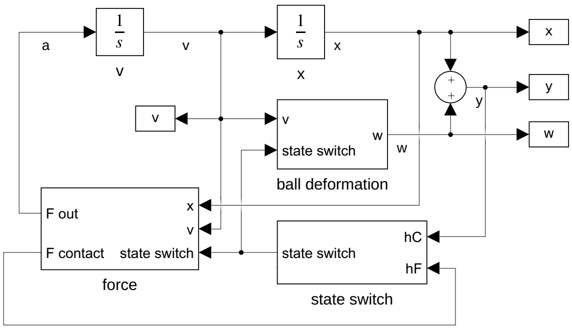


Figure 4: Bouncing ball with continuous contact.

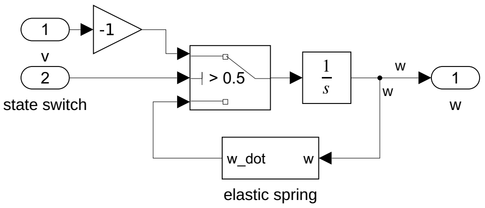


Figure 5: Ball deformation.

The interesting part is the state switch (cf. Figure 6), which toggles between 0 and 1, when the active event function – h_F or h_C according to phase – becomes negative. The actual toggle switch is implemented in a typical way as a triggered subsystem containing a `Unit Delay` with a feedback loop.

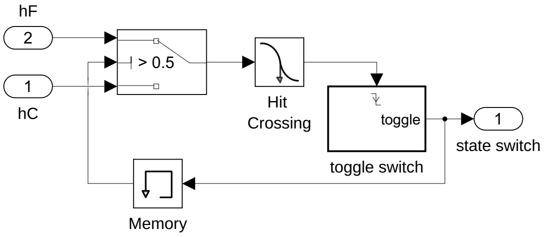


Figure 6: State switch.

The additional `Memory` component is necessary to break an algebraic loop, without it the simulation produces the error “Ambiguous sorted order”. Usually one tries do avoid such a block, because it introduces an additional delay. Here it does no harm, since a few solver steps will always occur, until the phase changes again. Quite odd is the appearance of a `Hit Crossing` component directly before the trigger input: One would expect that the toggle switch could be triggered directly by the event function (i.e. the output of the `Switch`). But without the help from the `Hit Crossing` block, the corresponding events are missed. Either the authors are lacking the necessary understanding of the inner workings of Simulink here – or it is simply a bug!

Since all three state variables are used throughout and only the forces change according to the phase, one could call this implementation a “switching model parts” approach.

Dependency of results from algorithms. Simulink offers seven adaptive ODE solvers for initial value problems, among them the Dormand-Prince solver `ode45` and the NDF-based `ode15s`, which is recommended as standard solver for stiff problems. All solvers are well-known and extensively documented [9].

To compare them, the standard model has been simulated with output values at fixed steps of 1e-3 s using ε_{abs} = 1e-12, ε_{rel} = 1e-12 for a reference solution with `ode45` and ε_{abs} = 1e-6, ε_{rel} = 1e-6 for the actual comparison. Using other solvers for the reference solution leads to almost identical results. A typical error plot is shown in Figure 7. The large spikes for the velocity errors are due to small timing differences of the bouncing phases.

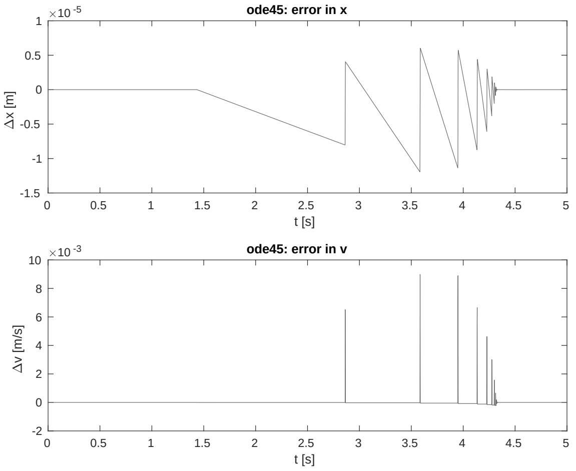


Figure 7: Errors for solver `ode45`.

The maximal errors against the reference solution are given in Table 1. The best one by far is the standard solver `ode45`, the next best `ode23` being far off with a tenfold error. All other solvers, among them all stiff solvers, are much worse with errors 60 to 80 times as large.

	s [mm]	**v [m/s]**
ode45	0.0119	0.0090
ode23	0.1146	0.0858
ode113	0.7299	0.5678
ode15s	0.6518	0.4722
ode23s	0.8247	0.6440
ode23t	0.8210	0.6410
ode23tb	0.6442	0.4999

Table 1: Absolute errors for different solvers.

The effects of changing two very special solver parameters have been examined, using the `ode45` solver and normal tolerances: Switching on the *Shape preservation* leads to better use of derivative information, which – according to the documentation – should increase accuracy for models with strongly changing derivatives. Looking at the velocities this is the case here, and the errors decrease by 25% accordingly. The other parameter is called `MinimalZcImpactIntegration` and should "reduce the impact of zero-crossing on the integration of continuous states" [10]. Though this sounds promising, it has no effects for our model.

Investigation of contact phase. For this task an additional `HitCrossing` block has been added to the standard model that outputs a signal whenever the velocity becomes zero (in either direction). This allows to extract the maximal and minimal heights easily. To get plots of the state and output variables during the three phases the high accuracy of $\varepsilon_{abs} = \varepsilon_{rel} =$ 1e-10 (as in the corresponding task of [2]) has been used. Figure 8 shows the first contact phase, it and the plots of the other two phases are identical to [2, Fig.7 - Fig.9].

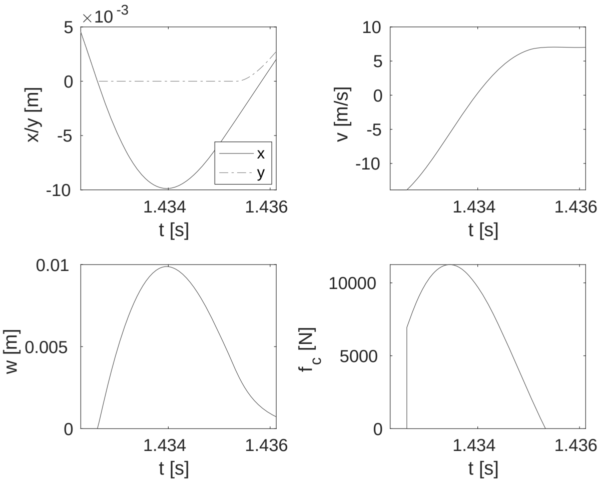


Figure 8: First contact phase.

The first ten values for the maximal height and maximal depression are shown in Table 2. Additional 21 values describe vibrations during the final contact phase.

In spite of the high accuracy the values given here differ from those in [2] by up to 0.05%. To find out, which results are better, additional values have been computed with an even higher accuracy of 1e-12 in Simulink and Maplesim. While the relative error between both Simulink results is only 1e-6, the Maplesim results differ by 1e-4. Furthermore the difference between the higher accuracy results in Simulink and Maplesim stays the same as before. This suggests that the Simulink results have the higher accuracy. This is confirmed by the general behaviour of the solvers that are used here: While `ode45` is known to be generally quite accurate, the Maplesim computations have been done with one of the stiff solvers, which are generally less accurate.

n	h_{max} [m]	w_{max} [mm]
1	10.00000000000	9.87399975
2	2.49475864884	4.97196356
3	0.63279737764	2.51186760
4	0.16059402954	1.26915027
5	0.04046743537	0.64034298
6	0.01002956662	0.32202875
7	0.00239944058	0.16086111
8	0.00053018975	0.07923061
9	0.00009492532	0.03784056
10	0.00000588511	0.01701582

Table 2: Maximal heights and depressions.

Parameter studies. Changing the values of k or d in the standard model reproduces the results from [2, Fig.10, Fig.11]. The only interesting observation here is the behaviour of the stiff version (k = 1e8, d = 500): With the standard solver `ode45` and accuracy 1e-6 the simulation produces a fall-through behaviour. Decreasing the tolerance to 1e-8, the model works fine. Of course, for this model one would prefer a stiff solver anyhow in order to decrease the computation time.

Bouncing ball on Mars. The standard model reproduces the results from [2, Fig.12].

2 Case Study RLC Circuit with Diode

The second test case of the benchmark is an RLC circuit containing a diode, where several models for the diode have to be studied: a simple shortcut model, the well-known Shockley model and an approximation thereof.

Description of model implementations. The model of the shortcut diode implements the differential equations directly. It computes the event function and uses the Simulink `Switch` component to change between the locking and the conducting phase. Therefore it is a classical “switching model parts” approach (cf. Figure 9).

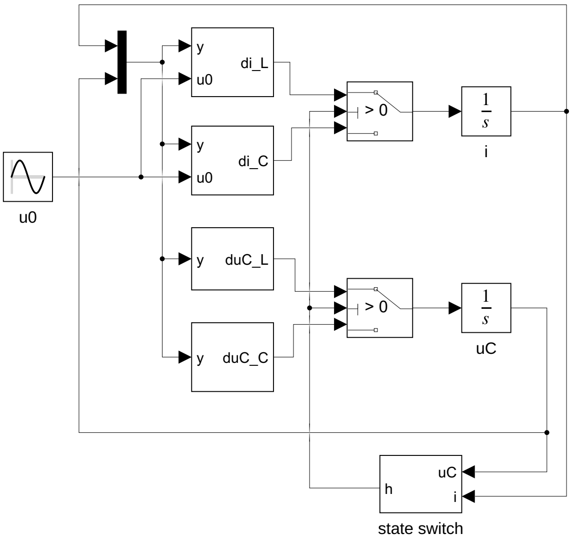


Figure 9: Short cut diode.

To model the Shockley diode one can eliminate the variable i_D from [1, eq. (28)] using the defining relation [1, eq. (34)] to get a semi-explicit DAE system of index 1 for the state variables u_C and i and the algebraic variable u_D. This can be modeled in Simulink in a standard way [11] by using algebraic loops to solve the constraint equation $g(u_C,i,u_D) = 0$ (cf. Figure 10). The discontinuous constraint can be written with the Heaviside step function and easily implemented with Simulink’s standard `Signum` component (cf. Figure 11).

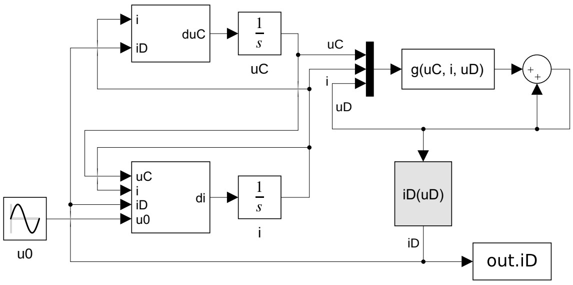


Figure 10: Shockley diode with implicit computation of uD.

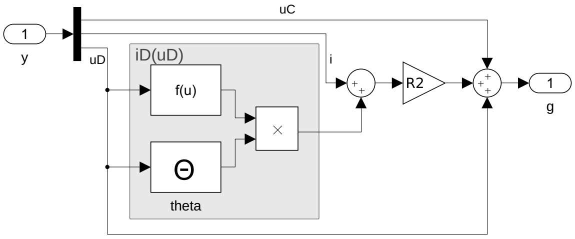


Figure 11: Algebraic constraint of Shockley diode.

So instead of the “switching model parts” approach used before, this is just an ordinary DAE system with a discontinuous constraint equation. It works fine in spite of a warning ("Discontinuities detected within algebraic loop(s), may have trouble solving") and produces the expected results. But: If one deletes the output block `out.iD`, the warning becomes an error ("2 zero crossing signal(s) identified below caused 1000 consecutive zero crossing events..."). Though this can be cured by changing the zero-crossing control algorithm to “adaptive”, for the user it is impossible to understand what is going on in detail, why one model works while the other doesn’t.

Alternatively one could write the equations of the Shockley diode using an implicit computation of i_D and adopt the same approach as before. The constraint equation is now implemented with a `Switch` block to set $g(u_C,i,i_D) = i_D \overset{!}{=} 0$ in the locking phase.

The implementation of the interpolated Shockley diode looks exactly like Figure 10, only the `Fcn` block that computes the $i_D(u_D)$ function in Figure 11 is replaced by a `Lookup Table`. The necessary table values are computed in the `Init Fcn` callback. The model doesn’t run with the `ode45` solver, but needs the stiff solvers `ode23t` or `ode23tb`.

One way to cope with DAEs is to simply differentiate the constraint equation, giving the “explicit Shockley diode” model that again can be implemented with the “switching model parts” approach (cf. Figure 12).

The problem here is that the variable u_D only becomes a state variable in conducting phase, while it still is an algebraic variable in locking phase. Therefore the `uD_conducting` block has an additional integrator block, while the `uD_locking` subsystem only contains algebraic computations. In spite of this strange

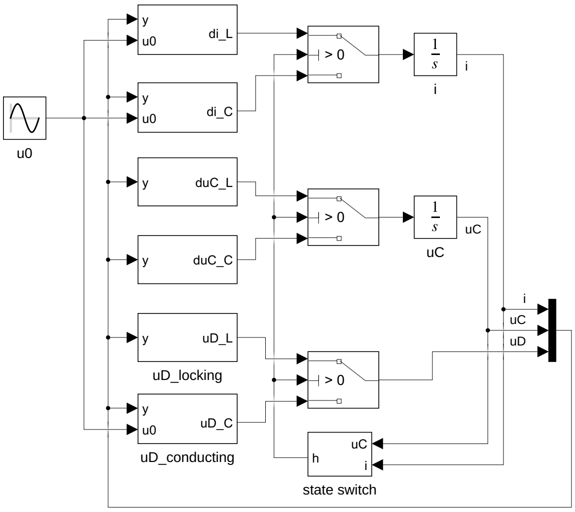


Figure 12: Explicit Shockley diode.

construction the model works without problems.

Dependency of results from algorithms. Following the same procedure as in Section 1.2 the results for the interesting variables i_L, u_C, i_D and u_D have been computed using the seven Simulink solvers and are shown in Table 3 for the shortcut diode and in Table 4 for the Shockley diode.

	ε_{iL}	ε_{uC}	ε_{iD}	ε_{uD}
ode45	0.0435	0.0000	0.0435	0.0011
ode23	30.5122	0.0173	30.5123	0.8522
ode113	50.6304	0.0239	50.6305	0.5859
ode15s	444.0654	181.7101	444.0668	137.8873
ode23s	57.0346	3.2628	57.0348	4.9717
ode23t	283.6237	17.7122	283.6245	13.2222
ode23tb	57.1190	4.0521	57.1192	2.8067

Table 3: Shortcut diode: Relative errors [in 1e-6].

According to these results the solvers can be grouped as follows:

- `ode45` is by far the most accurate solver,
- `ode23`, `ode113` have medium errors for i, i_D and u_D, but small errors for u_C,
- `ode23s`, `ode23tb` have medium errors throughout,

	ε_{iL}	ε_{uC}	ε_{iD}	ε_{uD}
ode45	0.0470	0.0006	0.0146	0.0286
ode23	31.5010	0.3780	9.7677	19.1429
ode113	51.7119	0.6202	15.9115	31.4251
ode15s	454.1598	1045.878	765.7054	651.4087
ode23s	57.4423	3.7010	17.8112	34.9072
ode23t	565.4285	59.3439	177.6517	350.7907
ode23tb	57.5270	3.5943	17.8380	34.9595

Table 4: Shockley diode: Relative errors [in 1e-6].

- `ode15s`, `ode23t` have large errors throughout.

Usually one would choose a stiff solver for a DAE system, therefore this behaviour comes unexpected. Apparently, the combination of `ode45` with a Newton solver for the algebraic loops does a good job here.

The time behaviour of the errors is quite different for different solvers and variables. Figure 13 shows a few examples for the shortcut diode and the variable i_d.

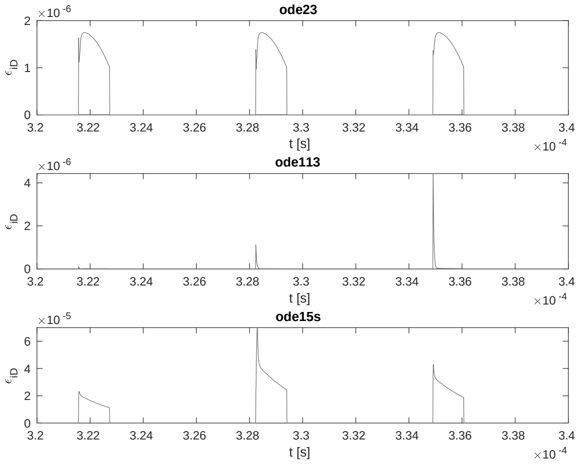


Figure 13: Shortcut diode: Relative errors for i_D

Comparison of shortcut and Shockley diode model. Comparing the results of the shortcut and Shockley diode model leads to the same results as in [2, Fig. 15].

To get accurate simulation times, both models have been run seven times and the mean value of the last five values has been computed. The corresponding results are shown in Table 5.

	shortcut [s]	**Shockley [s]**	**Sh / sc [%]**
ode45	1.0377	2.6004	250.59
ode23	0.7594	1.7771	234.02
ode113	0.6529	1.4567	223.10
ode15s	1.2696	2.9795	234.67
ode23s	7.4267	2.4208	32.60
ode23t	0.8030	1.6715	208.16
ode23tb	0.9082	2.0373	224.33

Table 5: Simulation times for different solvers.

The differences between the solvers are astonishingly small (with the exception of `ode23s`) – one would expect that the stiff solvers like `ode15s` or `ode23s` are much faster for the implicit Shockley model. This is even more surprising in view of the much better accuracy of the `ode45` results.

Approximation of Shockley diode model. The comparison of the Shockley and approximated Shockley diodes (cf. Figure 14) almost reproduces the results shown in [2, Fig. 16]. Only the errors differ: They again have the form of spikes, but they are much more pronounced here.

The real problem is to get the approximated model to run at all: Trying different solvers and precisions and all three values for the number N of breakpoints, one very often gets an error. The proposed solution to use a higher accuracy or switch to a LineSearch-based algorithm usually makes things worse. The only solvers that are working for all N with standard parameters are `ode23t` and `ode23tb`. Since `ode23t` has shown to be very inaccurate, `ode23tb` is used for the task.

Relevance of choice of algebraic state. The Shockley models using u_D or i_D as algebraic variable produce the same results, with differences much lower than the solver tolerance. The runtimes are different, though: The i_D model is 2.4 times slower than the u_D version.

Quite interesting is the warning that appears, when using the Runge-Kutta solvers `ode45` or `ode23` for the i_D version: It states a convergence problem when solving the algebraic loop, and falls back to a strategy from an older Simulink version.

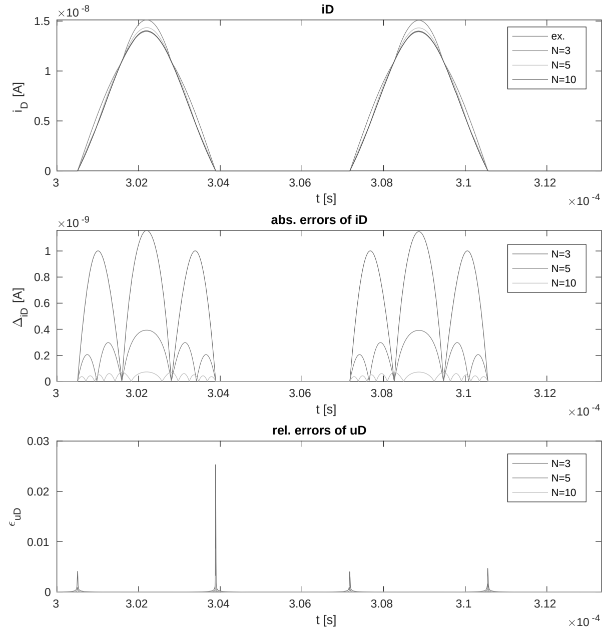


Figure 14: Comparison of Shockley and approx. Shockley diode.

Investigation for real-time simulation. Differentiating the constraint equation of u_D for the Shockley diode and replacing the appearing time derivatives, one gets

$$\begin{aligned}\dot{u}_D \;=\;& \left[\left(\frac{(R_1+R_2)R_2}{L}-\frac{1}{C}\right)i+\frac{R_2}{L}u_C\right.\\ &\left.+\left(\frac{R_2^2}{L}-\frac{1}{C}\right)I_S\left(e^{u_D/U_T}-1\right)-\frac{R_2}{L}u_0\right]\\ &\cdot\left(\frac{R_2 I_S}{U_T}e^{u_D/U_T}+1\right)^{-1}\end{aligned}$$

This equation is hidden inside the `uD_conducting` subsystem in Figure 12.

The three models of the shortcut, approximated and explicite Shockley diode have been run each with a variable step solver (`ode45`, ε_{rel} = 1e-6) and a fixed step solver (`ode4`, ε_{rel} = 1e-8). As before the approximated model needed special attention: The algebraic loop solver had to use the TrustRegion algorithm with `ode45` and the LineSearch algorithm with `ode4`. The differences between both runs are smaller than the solver tolerances for all models.

In addition, the explicit model using u_D as state variable and its alternative version based on i_D have been compared to the results of the implicit Shockley model. Again all results agree, especially there is no drift of the diode current as has been seen in [2].

3 Case Study Rotating Pendulum With Free Flight Phase

The third test model is a point mass swinging on a rope of fixed length. According to the forces acting on the mass, the movement can switch between swinging and falling phases.

Description of model implementations. For the implementation of the swinging mass a classic hybrid decomposition method is applied (cf. Figure 15). The two subblocks `Swinging` and `Falling` model the corresponding differential equations, they both output the current state values and have control inputs to enable or disable them. When enabled, new initial values are supplied by another input.

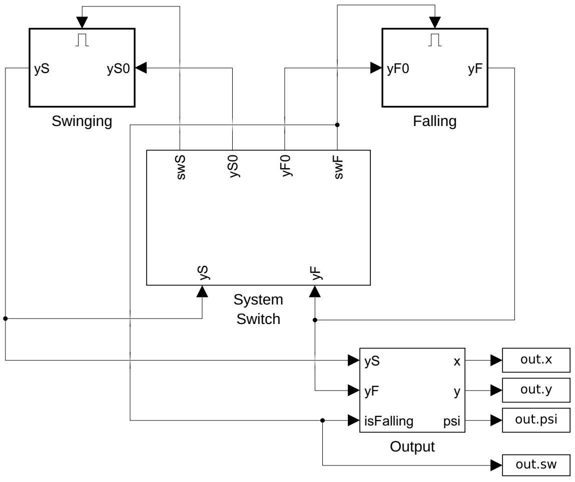


Figure 15: Rotating pendulum model.

The interesting part is the `System Switch` component (cf. Figure 16). It gets the current values of the state variables and outputs the control signals and new initial values for the two “physical” subsystems. To this end it computes the values of the event functions h_F (the rope force) and h_S (the rope slack) and uses them to trigger a state switch, which stores the current state. Such a component has already been used in the bouncing ball model (cf. Figure 6). Again `Memory` blocks are necessary to break the algebraic loops.

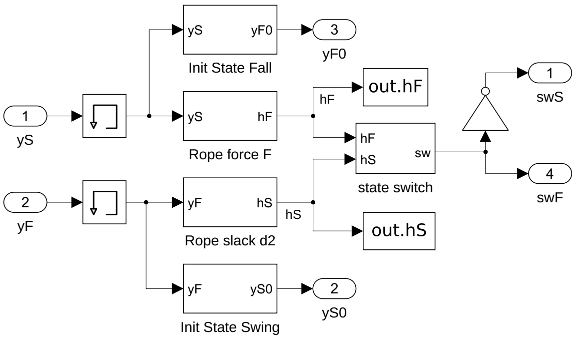


Figure 16: System switch.

Basic simulation of phases. The basic model is extended by a stop mechanism in the `Output` component: A small subsystem stops the simulation, when the angle ψ (measured against the lowest point) is below $\pi/10$. This subsystem is triggered, when the angular velocity goes through zero (at either direction). Running this model produces the results shown in Figure 17, which coincide with those in [2, Fig. 19]. The stop time differs slightly, here it is t = 7.5965376 s, compared to the value t = 7.5962714 s in [2].

Dependency of results from algorithms. The standard procedure for comparing solver precisions has been used again, the results are shown in Table 6. All solvers are very precise here with the notable exception of `ode15s`. Figure 18 displays some plots showing the error over time.

	x [1e-6 m]	y [1e-6 m]	ψ [1e-6]
ode45	0.0000	0.0000	0.0000
ode23	0.0001	0.0000	0.0001
ode113	0.0287	0.0080	0.0298
ode15s	35.3295	9.8355	36.6730
ode23s	0.0681	0.0504	0.0707
ode23t	0.5685	0.4418	0.5901
ode23tb	0.0535	0.0445	0.0555

Table 6: Absolute errors (compared to reference solution).

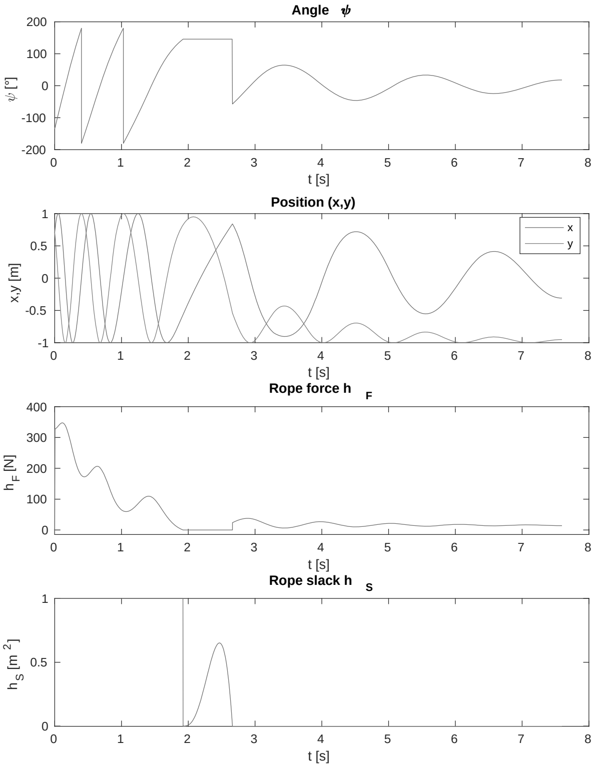


Figure 17: Results of pendulum model.

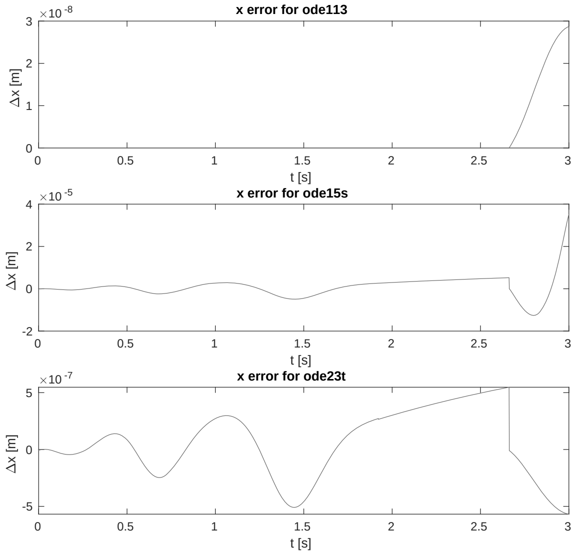


Figure 18: Absolute errors in x for three solvers.

Additionally, the impact of the tuning parameters “Shape preservation” and “MinimalZcImpactIntegration” has been studied. While the second one has no consequences, the first one has interesting effects: The results of `ode113` get better by more than a factor of 100, while those of `ode15s` get even worse by a factor of 10.

External energy supply. The kick factor γ is determined with a backwards running pendulum model, which gives the same results as in [2, Table 6].

The real challenge here is the implementation of the kick mechanism with its chain of interdepending events. Since there are several very different mechanisms in Simulink to create events, there are often special tricks which make models simpler - but which are hard to come up with. Instead we will try to design the model in an as simple and straightforward manner as possible.

The kick is realized inside the `Swinging` block by using the reset mechanism of the `omega` integrator and providing two different initial values: one that is defined by the `System Switch` when changing back to the “swinging” phase, the other one is used at the kick (cf. Figure 19). The signal `isKick` is used here in two ways: Its value (0 or 1) switches between the two different initial values, its change ($0 \rightarrow 1$) triggers the kick. To make this work, the signal has to return to 0 quickly to make way for the usual system switching mechanism.

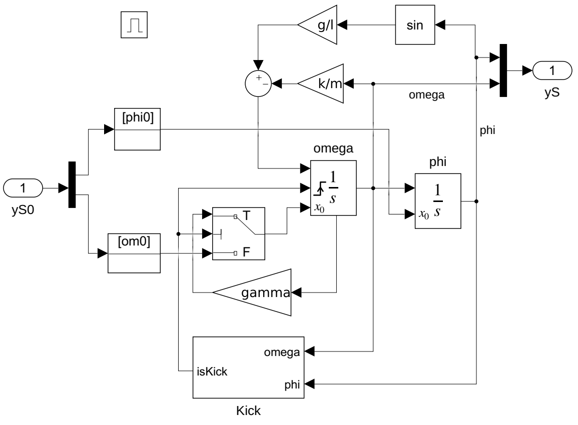


Figure 19: `Swinging` component with kick.

The chain of events is handled inside the `Kick` subsystem (cf. Figure 20): As a first step the `awaitKick` block outputs 1 as soon as the ψ amplitude gets small enough, and enables the `doKick` block, which outputs 1 when ψ reaches 0 afterwards. Both blocks contain only a `Constant` 1 and their output is initialised to 0. To guarantee that `isKick` returns to zero as fast as possible a `Hit Crossing` block is added that creates an "infinitely short" signal at the time of the 0 → 1 transition.

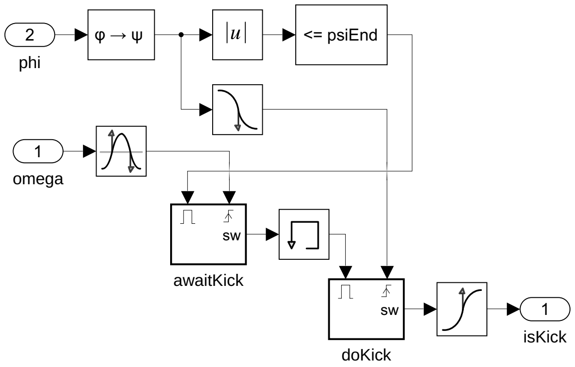


Figure 20: `Kick` subsystem.

4 Conclusions

Simulink provides a lot of different tools to create events and react to them together with special tricks such as the second-order integrator component or tuning parameters for the zero-crossing algorithm. But sometimes they behaved inconsistently, without any reason apparent to the user. Even worse, a few models only ran after playing with different workarounds, like adding `Hit Crossing` components or using very special implementations of simple subtasks.

In every simulation environment one runs into a set of typical problems, when the implementation of simple ideas collides with basic patterns or paradigms within the environment. Usually one builds up a collection of workarounds for common problems. One goal of the ARGESIM benchmarks is to provide a collection of such solutions, maybe in the spirit of the *Design Patterns* of object-oriented programming [12]. But unlike in the OO world, the solutions found in modeling usually are bound to the specific simulation environment – or even to a specific version.

Comparing the mathematical description of the models and their implementation in Simulink, one finds a lot of components that have no direct mathematical counterpart. It would be much better if basic mathematical ideas could be implemented in standard ways, independent of the concrete environment used. But this still remains a task for future simulation programs.

References

[1] Körner A, Breitenecker F. State Events and Structural-dynamic Systems: Definition of ARGESIM Benchmark C21. *SNE Simulation News Europe*. 2016; 26(2):117–122. doi: 10.11128/sne.26.bn21.10339.

[2] Disselkamp JP, Junglas P, Niehüser A, Schönfelder P. A Solution to ARGESIM Benchmark C21 'State Events and Structural-dynamic Systems' based on Modelica Components. *SNE Simulation News Europe*. 2018; 28(2):39–48. doi: 10.11128/sne.28.bn21.10411.

[3] The MathWorks. *Simulink: Simulation and Model-Based Design*. `http://www.mathworks.com/products/simulink/`.

[4] The MathWorks. *Stateflow: Model and simulate decision logic using state machines and flow charts*. `http://www.mathworks.com/products/stateflow/`.

[5] Clune MI, Mosterman PJ, Cassandras CG. Discrete Event and Hybrid System Simulation with SimEvents. In: *8th International Workshop on Discrete Event Systems*. Ann Arbor. 2006; pp. 386–387.

[6] Junglas P. *Argesim C21 models and scripts*. URL `http://www.peter-junglas.de/fh/simulation/argesimc21.html`

[7] Zhang F, Yeddanapudi M, Mosterman PJ. Zero-crossing location and detection algorithms for hybrid system simulation. *IFAC Proceedings Volumes*. 2008; 41(2):7967–7972.

[8] The MathWorks. *Simulink: Simulation of a Bouncing Ball*. `https://www.mathworks.com/help/simulink/slref/simulation-of-a-bouncing-ball.html`.

[9] Shampine LF, Gladwell I, Thompson S. *Solving ODEs with matlab*. New York: Cambridge university press. 2003.

[10] The MathWorks. *Simulink: Enable minimal zero-crossing impact integration*. `https://www.mathworks.com/help/simulink/gui/enable-minimal-zero-crossing-impact-integration.html`.

[11] Shampine LF, Reichelt MW, Kierzenka JA. Solving index-1 DAEs in MATLAB and Simulink. *SIAM review*. 1999;41(3):538–552.

[12] Gamma E, Helm R, Johnson R, Vlissides J. *Design patterns: elements of reusable object-oriented software*. Reading, Mass.: Addison-Wesley. 1995.

ARGESIM Benchmark C11 'SCARA Robot’ with Extended Trajectory Tracking Control: Comparison of Model Approaches and Simulation Results in MATLAB, Simulink and SimMechanics

Martin Batliner[2], Felix Breitenecker[1], Andreas Körner[1]*, Horst Ecker[2]

[1]Inst. of Analysis and Scientific Computing, [2]Inst. of Mechanics and Mechatronics, TU Wien, Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria; **Andreas.Koerner@tuwien.ac.at*

SNE 31(1), 2021, 43 - 51, DOI: 10.11128/sne.31.bne11.10557
Received: October 25, 2018; Revised: July 15, 2019;
Revised: December 15, 2020; Accepted: January 20, 2021
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. Modelling of mechatronic systems often results in implicit model description. Simulation systems provide different strategies to deal with this kind of problem, resulting in different modelling approaches. For this purpose, Mathwork’s MATLAB system offers in its basic MATLAB system at programming level implicit ODE solvers, and in its toolboxes Simulink and SimMechanics (on basis of the Modelica-like system SimScape) graphical modelling environment. This Benchmark Study compares these modelling approaches and the simulation efficiency and results on basis of the ARGESIM Benchmark C11 ‘SCARA Robot’. Additionally, the contribution’s investigations present a tuning of the PID control for point-to-point movement with and without collision prevention and introduce a trajectory tracking control with collision prevention, which improves the performance essentially.

Introduction

Classical derivation of models for mechanic systems results first in implicit models for the generalized coordinates $\vec{q}$, with mass matrix $M(\vec{q},\dot{\vec{q}})$, and with generalized forces $\vec{f}(\vec{q},\dot{\vec{q}},\vec{u})$ with input (feedforward control $\vec{u}$):

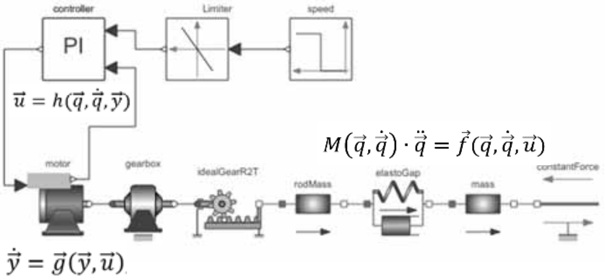
$$M(\vec{q},\dot{\vec{q}})\cdot\ddot{\vec{q}}=\vec{f}(\vec{q},\dot{\vec{q}},\vec{u})$$

Second, feedback control with actuators adds (state) equations for the actuators and feedback functions for control: $\dot{\vec{y}}=\vec{g}(\vec{y},\vec{u}),\;\; \vec{u}=h(\vec{q},\dot{\vec{q}},\vec{y})$

And third, these implicit equations have to be transformed into at least semi-linear state space form with ‘combined’ state vector $\vec{x}$, to be understood by an ODE solver of a ‘basic’ simulation system:

$$\widehat{M}(\vec{x},\vec{u})\cdot\dot{x}=\vec{\hat{f}}(\vec{x},\vec{u}),\;\; \vec{x}=(\vec{q},\dot{\vec{q}},\vec{u})^T$$

On the other side, today simulation environments offer graphical physical modelling with mechanical elements for the mechanical part following the Modelica concept [1] and graphical signal-oriented block modelling for the control part, as sketched in the following figure:


controller
PI
Limiter
speed
$\vec{u}=h(\vec{q},\dot{\vec{q}},\vec{y})$
$M(\vec{q},\dot{\vec{q}})\cdot\ddot{\vec{q}}=\vec{f}(\vec{q},\dot{\vec{q}},\vec{u})$
motor
gearbox
idealGearR2T
rodMass
elastoGap
mass
constantForce
$\dot{\vec{y}}=\vec{g}(\vec{y},\vec{u})$

1 Mechatronics in MATLAB

MATLAB [7] offers modelling and simulation on all these above cited levels, from basic use of ODE solvers for the semi-linear state space form in basic *MATLAB* directly, via signal-oriented graphical modelling for the control part as well as for the mechanical part in *Simulink*, up until to use of *SimMechanics* for the mechanicals in combination with *Simulink* for the control part.

The user clearly expects that all approaches give the same simulation results. But as ‘higher’ modelling techniques generate equations ‘automatically’ (a big comfort for the user), the models may differ in comparison with a ‘laborious manually’ model setup. Consequently, sub-results may differ, especially if discrete elements and/or boundaries are implemented into the overall system.

The following investigations discuss alle three *MATLAB* modelling techniques in a case study with a specific SCARA robot and its environment, as defined in *ARGESIM Benchmark C11* 'SCARA Robot', dealing with mechanics, control, and collision prevention for such type of robot [2]. Additionally, control is extended from point-to-point control to trajectory tracking control, increasing the complexity of both control and mechanical model.

2 SCARA Robot Benchmark

2.1 System Definition

Following the definition of the *ARGESIM Benchmark C11* 'SCARA Robot', the three axis SCARA robot type as shown in Figure 1 has two vertical revolute joints and one vertical prismatic joint. The axes of all three joints are parallel to the *z*-axis.

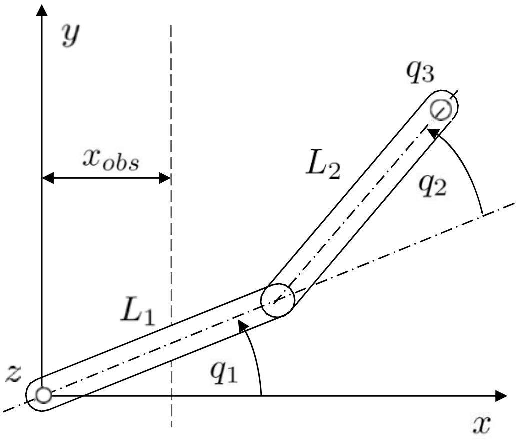


Figure 1: Kinematic structure of a SCARA robot.

The joint vector $\vec{q}$ consists of the joint angles q_1 and q_2 and the joint distance q_3. The equations of motion are given in the form

$$M(\vec{q}) \cdot \ddot{\vec{q}} = \vec{b}(\vec{q}, \dot{\vec{q}})$$

The calculation of the moments of inertia is based on the assumption that the rods have a homogeneous mass distribution. The right hand side of the dynamic equation includes the joint torques and joint forces. The electric relationship of the armature of a robot servo motor is given by the first order differential equation

$$I = \frac{1}{L_{ai}} \cdot (U_{ai} - k_{Ti} \cdot u_i \cdot \dot{q}_i - R_{ai} \cdot I_{ai})$$

where U_{ai} is the applied armature voltage. The resulting armature current I_i is limited to maximum values I_{max}.

The joint torques (forces) of a motor are proportional to the armature current I_{ai} and given by

$$T_i = u_i \frac{\sqrt{3}}{2} k_{Ti} \cdot I_{ai}$$

The detailed equations and numerical data are taken from the definition *ARGESIM Benchmark* C11, see [2].

2.2 Implementation

First, the simulators and their respective solvers are shortly portrayed and their handling of implicit systems. Then the basic model is implemented with the three simulators Simulink, SimMechanics and in *MATLAB* with solver ode15i.

2.3 Point-to-point motion

In order to control the point-to-point motion of the robot a single-axis PD-controller is employed for the control voltage:

$$U_i = P_i \cdot (\hat{q}_i - q_i) - D_i \cdot \dot{q}_i$$

Initial values are $q_1 = q_2 = q_3 = 0$, and the target values are $q_1 = q_2 = 2$ and $q_3 = 0.3$. The velocities should be zero at start and end.

2.4 Collision avoidance

Based on the point-to-point motion now an obstacle has to be avoided. The obstacle is given by an elevated area with the height h_{obs} and the borderline x_{obs} (Figure 1). The border represents an obstacle for the end-effector of the robot arm. Possible contact has to be avoided and must be detected during robot motion.

Therefore, an obstacle sensor should measure the distance between end-effector and obstacle and if that falls below a critical value d_{crit} the controls of rotational drives must be changed until the tool tip has cleared the obstacle height. Maximum voltage may be used in this situation for motor 1 and motor 2 to obtain maximum deceleration.

2.5 Trajectory tracking control

Additionally to the tasks presented in *ARGESIM Benchmark C11*, we now introduce another task: a trajectory tracking control should be implemented to keep the end-effector on a desired trajectory. The robot is assumed to be in the same environment with the obstacle as before.

The foreseen trajectory should not only execute the point-to-point motion, but also bypass the obstacle providing a path which is planned from desired accelerations. Both a PD and a PID controlled are implemented and compared.

3 Implementation

3.1 Simulink

Simulink is a widely used tool for modelling and simulating multidomain dynamic systems. It is integrated in the *MATLAB* environment and can be scripted from it. *MATLAB* is used here to parameterise the model and analyse data.

Simulink provides as primary interface a graphical block diagram tool. The blocks used treating this problem can be found in the model library. The chosen ODE solver is the variable step size, one step solver ode45, which is based on the Dormand-Prince pair. It is based on an explicit Runge-Kutta formula and calculates the 4th and 5th order accurate solution and takes the difference to be the fourth order error to estimate the adaptive step size [8].

Implicit systems cannot be solved directly. Nevertheless, the implicit equation $M(\vec{q})\cdot\ddot{\vec{q}}=\vec{b}(\vec{q},\dot{\vec{q}})$ was implemented directly, but an algebraic loop break had to be added (Figure 2), which finds a solution for the system for every integration step. This is also possible in the presence of a hit-crossing block which is used in Section 5 and works similar to the zero-crossing in the event function discussed in Section 3.3 [3].

Additionally, the model was implemented in an explicit form: the mass matrix *M* was inverted manually using the symbolic computation in *MATLAB* and consequently the system $\ddot{\vec{q}}=M^{-1}(\vec{q})\cdot\vec{b}(\vec{q},\dot{\vec{q}})$ was set up in Simulink using the graphical block diagrams (Figure 3).

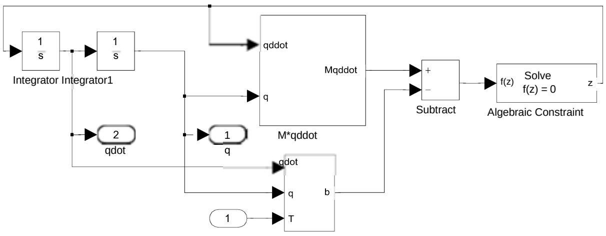


Figure 2: Implicit model description with *Simulink*.

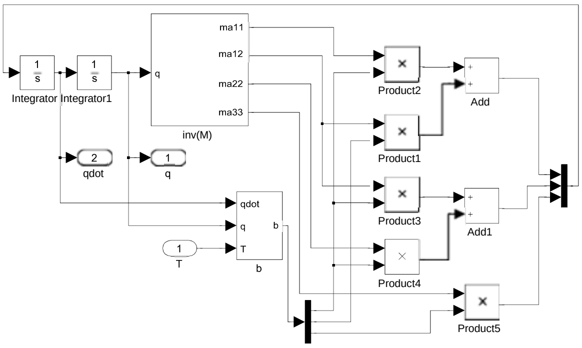


Figure 3: Explicit model description with *Simulink*.

3.2 SimMechanics

SimMechanics extends *Simulink* with tools for modelling three-dimensional mechanical systems within *Simulink* – being based on the general Modelica-like modelling system *SimScape*. Instead of deriving and programming equations this multibody simulation tool can build models composed of bodies, joints, constraints and force elements which reflect the structure of the system. It is integrated in *Simulink* and can be treated as a *Simulink* block, enabling physical model and control environment in one single system [6].

Consequently, simulating a SimMechanics model is a cooperative effort and consists of four steps:

- model validation
- machine initialisation
- force analysis and motion integration
- stiction mode iteration

whereby although the last step being negligible for the given problem as stiction is not treated.

The first two steps occur before the machine motion actually starts and checks all data entries, connections, assembly tolerances and validates the geometries and model topology. Moreover, *machine initialisation* cuts every closed loop and replaces it with a cut joint, constraint or driver block and checks all constraints and drivers for mutual consistency and eliminates redundant constraints. In *force analysis mode and motion integration*, *Simulink* steps up in simulation time and solves the system for every step while *SimMechanics* imposes assembly tolerances and a constraint solver [5]. Unfortunately, an analytic model cannot be derived from the *SimMechanics* block diagram.

The problem was implemented by setting up the block model in Figure 4. Rigid bodies are connected through joints which can be selected from the model library. The model properties such as mass, inertia and geometry can be set in the model blocks. The blocks, however, must be steered and measured with actuators and joint sensors which are connected to the free docks in Figure 4. These can be integrated into the Simulink workflow.

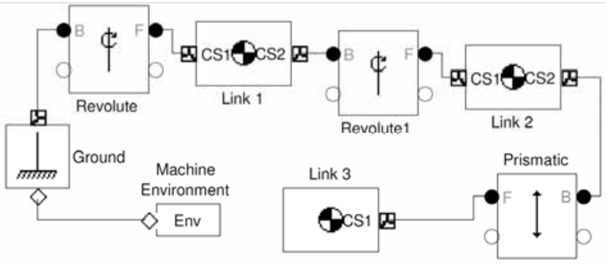


Figure 4: *SimMechanics* model.

3.3 MATLAB

For comparison a model in *MATLAB* was implemented. The solver ode15i in *MATLAB* can solve fully implicit differential and differential algebraic equations of the form $F(t, y(t), y'(t)) = 0$.
Additionally, to the initial values the solver needs the initial derivatives and the initial conditions must be consistent, meaning $F(t_0, y_0, {y_0}') = 0$. If needed they can be computed using the function decic in *MATLAB*.

The solver uses a fixed leading coefficient implementation of the BDFs (backward differentiation formulas) in a Lagrangian form for the polynomial interpolation [6]. This leads to simple expressions, evaluated efficiently in *MATLAB* and convenient for event localisation.

For the detection of state events the 'Events' property, in options set to function events, solves the system while also finding where user defined functions, called event functions, are zero. The event functions used for this problem are displayed in Section 5.2. At this point the system can come to a halt or is terminated. The event function is defined over a zero crossing, thus enabling the detection of the crossing through a change of sign [7].

Integrating the equations the solver passes the zero-crossing, consequently missing the exact value of the zero-crossing. But the solver automatically goes back and iterates back and forth to find the exact whereabouts of the zero-crossing within a set tolerance.

The system had to be reduced to a set of first order differential equations. Both sides of the equations have to be one as shown below. Hereby, *T* and *O* denote the joint torque (force) and the non-linear terms of the right-hand side of the equation:

```
y=[q(4)-qdot (1);…
q(5)-qdot(2);…
q(6)-qdot(3); …
ma11*qdot (4)+ma12*qdot (5)-T(1)-O(1);…
ma21*qdot (4)+ma22*qdot (5)-T(2)-O( 2 );...
ma33*qdot (6)-T(3)-O(3) ]
```

4 Point-to-point Motion

4.1 Simulink/SimMechanics

For the point-to-point movement (Figure 5) both *Simulink* and *SimMechanics* use the same controller, as the SimMechanics model is treated as a Simulink block. The model with controller and plant, which consists of the electrical and the mechanical model, is displayed in Figure 6. A target vector is the input for submodel control model which contains the PD-controller.

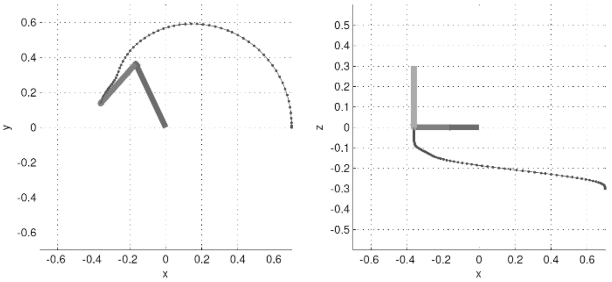


Figure 5: Point-to-point motion.

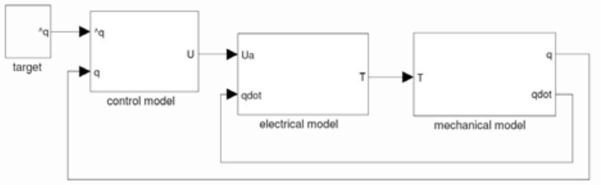


Figure 6: Control model in Simulink.

This was realised using the corresponding block from the Simulink library.

The output voltage *U* is fed into submodel electric model (Figure 6), which models the servo motor. The resulting torque *T* drives the mechanical system.

To incorporate the boundaries of voltages and currents and subsequently torque, *Simulink* offers the option to comfortably set the saturation limits as displayed in the electrical model in Figure 7 by a saturation block.

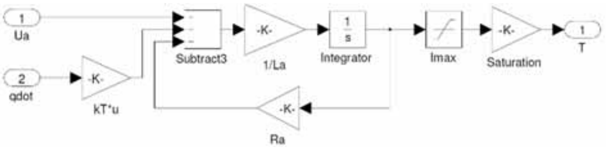


Figure 7: Electrical model.

4.2 MATLAB

For the point-to-point motion the model had to be extended by a control and servo motor model. These were integrated in the function deSCARA containing the robot equations. Moreover, armature current limits had to be implemented explicitly.

This was put into effect using simple if - clauses in the function containing the system equations. The function is called by

```
[ t q ] = ode15i (@deSCARA,…
        …[ t0 tend ], q0, qdot0 );
```

where the timespan, initial values and derivatives have to be implemented in consistent form.

5 Collision Avoidance

5.1 Simulink/SimMechanics

For collision avoidance (Figure 8) a block called state control was added to the submodel control model in Figure 9.

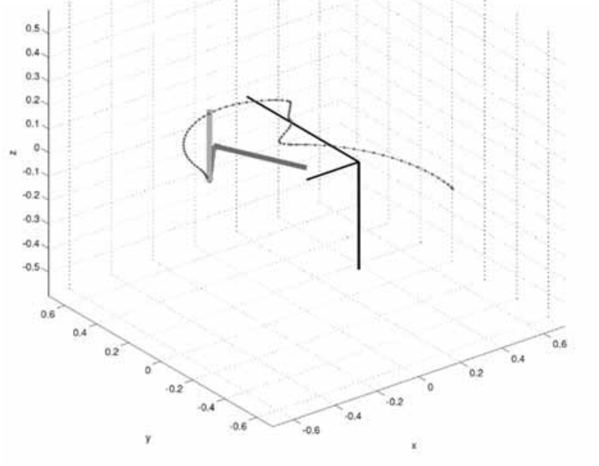


Figure 8: Collision avoidance.

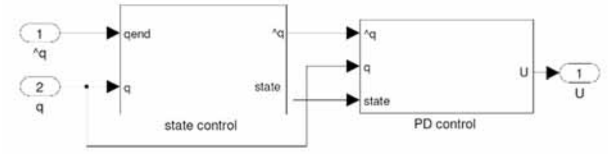


Figure 9: Extended control model.

The distance between the obstacle and the tool tip is permanently checked. If it falls under the critical distance the target positions for q_1 and q_2 are changed to the current position and the emergency maximum voltages are allowed. This is realised through a logic circuit which checks two conditions: One, is the robot in critical zone, and two, is the robot under h_{obs}. If both conditions apply new set-points for q_1 and q_2 are enforced. For robot arm 1 and arm 2 maximum voltage may be used to slow down and return to the position where the danger and subsequently a zero-crossing has been detected (as denoted in Section 3.1). Just after the tool tip of the robot has reached the height of the obstacle, at which only one of the two conditions applies, original target positions are reactivated for the arm 1 and arm 2.

How this is realised displays Figure 10 which shows in detail the state control from Figure 9. The block state control checks if one or both conditions mentioned before apply. If they do, they change their respective value from 0 to 1 and add up in block Add2. This sum now is the reference for the blocks threshold and Switch. Block threshold checks if this sum adds up to 2. If so, the switch forwards new set-points to the PD controller.

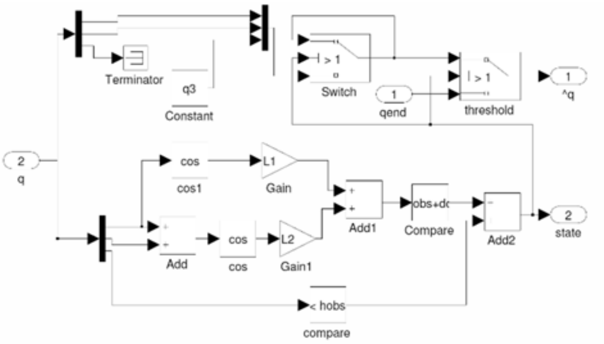


Figure 10: State control.

This point in time is determined accurately as the block enables zero-crossing detection.

The new set-points forwarded to the block threshold do not have to be specified by the user, but are determined through another circuit which localises the joint position of arm 1 and arm 2 when they enter critical zone. This is done by the block Switch. It has as input the current position and via a loop the last position. In critical zone, Switch forwards the looped values for q_1 and q_2 while looping the same value. If one of the two conditions is no longer fulfilled, the block threshold switches back to the original set-points.

For the saturation limits in the controller, the state of the system is forwarded trough the output state and the change of saturation limit in the PD-controller is subsequently realised with a switch block. This way the user is not required to specify way-points for the case that the robot enters the critical zone.

5.2 MATLAB

Additionally to the PD-controller in Section 4.2, collision avoidance has to be implemented employing the event property in the options of the solver shown in code below and implementing two event functions (cf. Section 3.3). The event functions for the given problem are:

$$x_{obs} + d_{crit} - L_1 \cdot \cos(q_1) - L_2 \cdot \cos(q_1 + q_2)$$
$$h_{obs} - q_3 = 0$$

Event specific parameters are returned from the function such as time and solution at a zero crossing and the corresponding type of event. This allows to stop the simulation at the occurrence of an event and restart the simulation with new setpoints.

Other than in the models implemented with *Simulink* and *SimMechanics,* these new set-points were added manually. Changing of set-point occurs twice: for entering critical zone and subsequent to reaching the admissible height, setting the setpoints back to the initial set-points.

The simulation runs until t_{end} when the counter equals 1 and the while-loop stops. Time and output vector are concatenated respectively every time the simulation is restarted:

```
options = odeset ( ' Events ' , @eventFun ) ;
while count==1
[ t q te xv ereig ]=ode15i ( @deSCARA,…
… [ t(end) tend ], q0, qdot0, options ) ;
qout=[ qout; q( : , 1 ) ];
tout=[ tout; t ];
if ( t( end)>=tend )
    count=0;
else
    telen=length( te );
    if ereig( telen )==1
        q1=1; q2=1;
    else ereig ( telen )==2;
        q1=2; q2=2;
    end;
end; end
```

6 Trajectory Tracking

6.1 Trajectory planning

The trajectory control problem in the joint space consists of following a given time-varying trajectory $q_d(t)$ and its successive derivatives which respectively describe the desired velocity and acceleration [9]. The planned path (Figure 11) will execute the point-to-point movement treated earlier with a way-point at $q_1 = q_2 = 1$ and $q_3 = 0.3$ to avoid collision with the obstacle described in the problem definition.

The velocities for start and end should be zero and naturally, the trajectories for the positions have to be smooth. For robot arm 1 two quadratic splines were chosen. These lead to linear velocities and constant accelerations. Robot arm 2 should have constant velocity, which leads to linear motion. To ensure smooth transition cubic splines were chosen for the path. Arm 3 has one cubic spline to the way-point and then stops. This results in linear acceleration. These variations in velocities and accelerations should unveil the capabilities and limits of the controller. For that purpose the *SimMechanics* model from the point-to-point motion is extended where the voltage restrictions still apply.

6.2 PD-Control

Based on PD-controller used earlier the tracking controller has the form

$$U_i = q_{i,d} + P \cdot \left(\ddot{q}_{i,d} - q_i\right) + D \cdot \dot{q}_i$$

where *d* marks the desired terms and the equations computes the control voltage from the position and velocity errors while feeding forward the desired acceleration.

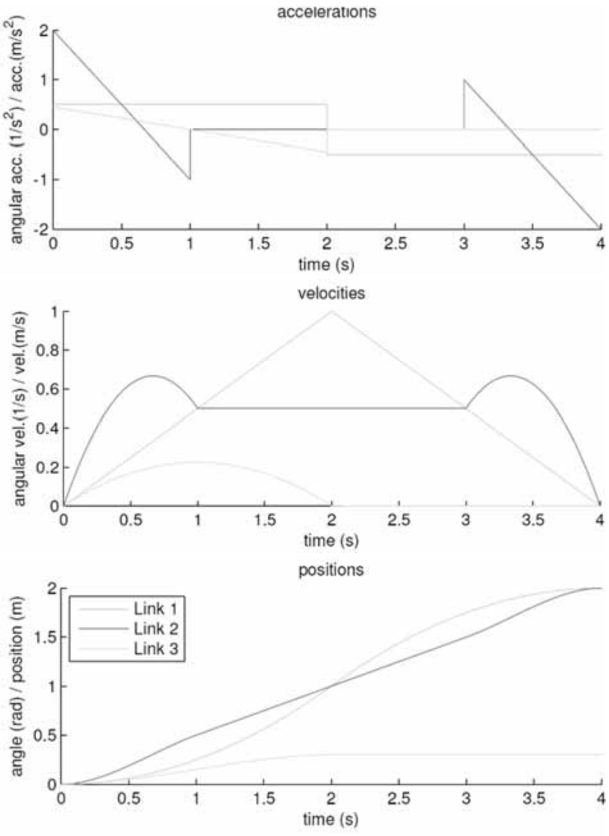


Figure 11: Planned trajectories.

Apart from the feed-forward acceleration it is the same closed loop behaviour as seen earlier and consequently the well-tuned parameters P_i and D_i from the point-to-point motion are used.

6.3 PID-control

The given problem with PD-control has global integral behaviour. This can be seen from the open-loop transfer function e.g. for arm 3 (gravity is unaccounted for)

$$\mathrm{G}_0 = \frac{\frac{\sqrt{3}}{2 \cdot k_{T3} \cdot u_3}}{s \cdot \left(s^2 \cdot ma_{33} \cdot L_{a3} + s \cdot ma_{33} \cdot R_3 + k_{T3} \cdot u_3\right)}$$

or can be seen in the results of the point-to point movement in Section 7.2 as no steady-state errors remains.

To zero the velocity error for constant velocity [10] and consequently increase the overall performance a PID controller is implemented in the form

$$U_i = q_{i,d} + P \cdot \left(\ddot{q}_{i,d} - q_i\right) + I \int \left(\ddot{q}_{i,d} - q_i\right) dt + D \cdot \dot{q}_i$$

where the coefficients given in Table 1 were found through pole placement and subsequently were tuned manually.

Gains	q_1	q_2	q_3
Proportional	1500	1000	15000
Integral	10000	7500	5000
Dervivative	10	25	10

Table 1: Coeffcients for PID controller.

7 Results

7.1 Implementation

The three implementations differ considerably in their modelling approach. In the *Simulink* block diagram modelling approach the user has always the overview of the problem, even bigger problems can be well arranged, although, setting up equations by blocks is comparably cumbersome. Even more, *Simulink* unveils its strength in handling the control model, as it offers out of the box blocks for most applications like saturation limits, and ready controller blocks. Zero-crossing for state events are detected automatically and offer the user a rather comfortable experience.

For modelling a mechanical system from scratch, *SimMechanics* offers the quickest and most intuitive approach to model the physical system as there is not even the need to derive equations. However, other than the other simulators one has to deal with the geometric property of the model, as the geometric relations of the joints and bodies have to be specified explicitly in the coordinate system. In recent *MATLAB* versions *SimMechanics* has been based on *SimScape*, a general denominator for ‘physical’ modelling in mechatronics, electrical engineering and other domains, following the Modelica approach; as consequence, coordinate systems can be avoided. A big advantage is that SimMechanics is fully integrated over the actuators and sensors in the *Simulink* control environment where it is treated as another block. It therefore uses all the *Simulink* functionality regarding state detection, but it does in turn not offer full insight in the process running in the background setting up the mechanical model.

For a given problem with given equations basic *MATLAB* offers with solver ode15i a reliable and reasonably quick implementation for the experienced user to directly compute the given fully implicit equations. The state event property is well integrated in the solver, which is specified over the option set.

However, out of all three approaches the user probably spends the most time debugging in *MATLAB*, as one could easily lose track in the vast amount of variables to be defined. The control law and saturation limits have to be integrated additionally into the functions containing the set of equations which enlarge the formulas and lessens the overview.

7.2 Point-to-point motion

All three approaches execute the point-to-point motion as expected. The result for the Simulink model can be seen in Figure 12.

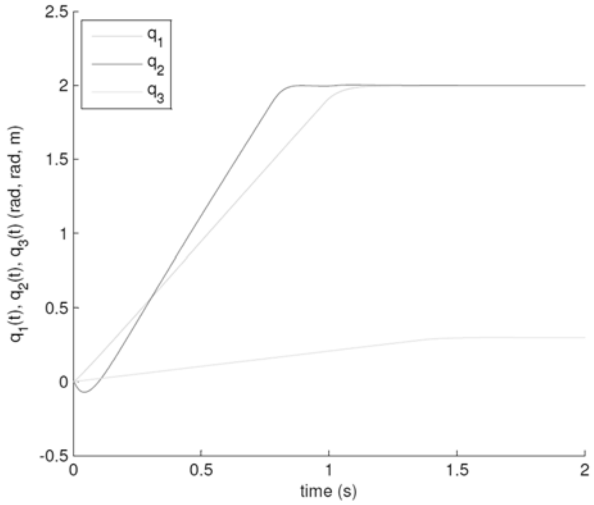


Figure 12: Point-to-point motion in Simulink.

Model description	*Norm. CPU-time*
Simulink explicit	1
Simulink implicit	3.7
MATLAB ode15i	4.6
SimMechanics	6.5 (4.2)

Table 2: Computation time.

Table 2 shows the normed computation time of the approaches for this task. The processing times were measured from *MATLAB* using the tic and toc command. Naturally, the explicit model description with *Simulink* with inverted mass matrix takes by far the least computation time. The implicit description from *Simulink* with algebraic loop breaking and *MATLAB* with implicit solver ode15i are comparable. However, the model implemented with *SimMechanics* almost doubles the time of the *Simulink* implicit model although it uses the same solver and no model manipulation was done beforehand. Recent developments with the *SimScape* basis have done significant improvement and increased the speed (time in parenthesis).

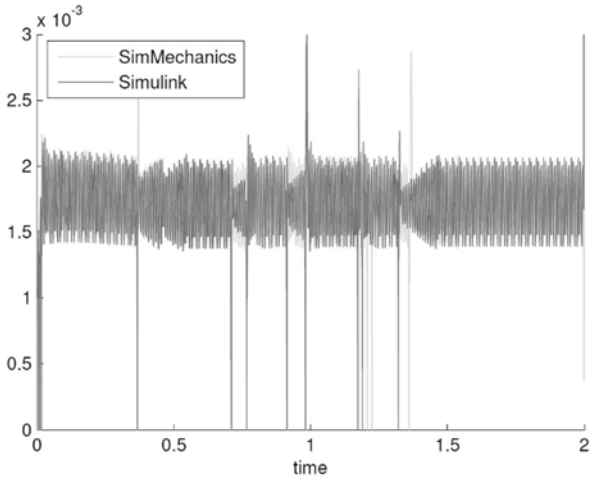


Figure 13: Comparison of stepsize.

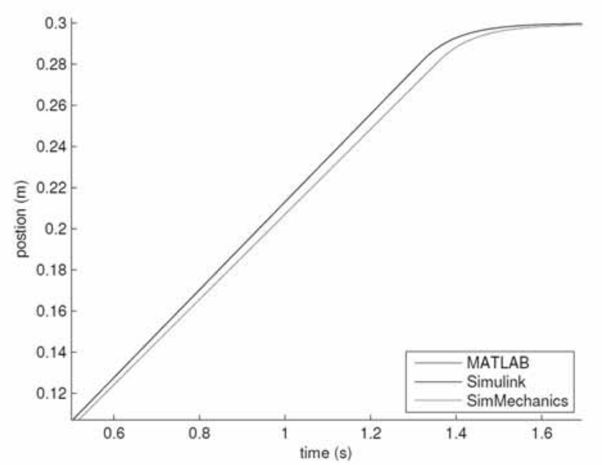


Figure 14: Comparison detail plot.

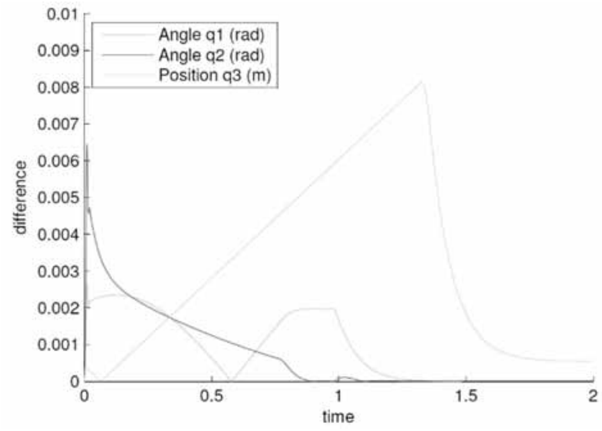


Figure 15: Differences Simulink - SimMechanics

Figure 13 shows the comparison of adaptive stepsizes the ODE solver ode45 used for the *Simulink* implicit model and the *SimMechanics* model. They not only use the same default solver but also have roughly the same step size.

Therefore, the higher computation time results from the premotion model validation and the cooperative solving effort discussed in Section 3.2.

Additionally, in Figure 14 a detail plot of coordinate q_3 for the point-to-point motion in *Simulink*, *SimMechanics* and *MATLAB* ode15i is shown. One can easily see that Simulink and *MATLAB* results are matching; interestingly, the *SimMechanics* result is a bit off – reason may be a difference in the equations fixed before or automatically derived, resp. A comparison of the three angles between Simulink and SimMechanics is depicted in Figure 15: Again, it shows the difference in q_3, which gives a hint for a different integration of the controller in the S*imMechanics* model.

7.3 Collision avoidance

Both *Simulink* and *MATLAB* deliver the same results within a numerical tolerance. This comes as no surprise for the implicit *Simulink* model and the implementation with *MATLAB* ode15i as they solve the same set of equations, but it verifies the *SimMechanics* model in its accuracy.

Figure 16 shows result of the collision avoidance in *SimMechanics* with the path of the tool-tip in *x*-direction and the overlay of the obstacle. The plot shows particularly well that the end-effector stabilises on the edge of the critical zone while robot arm 3 gains height. The adaptive stepsize of the solver ode45 can be seen from the Figure 17 which is a detail plot of Figure 16 and it clearly shows how the stepsize is reduced at a detection of a state event.

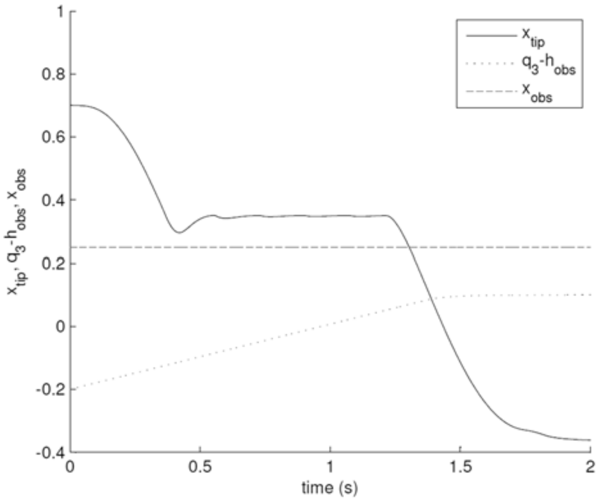


Figure 16: Collision avoidance in SimMechanics.

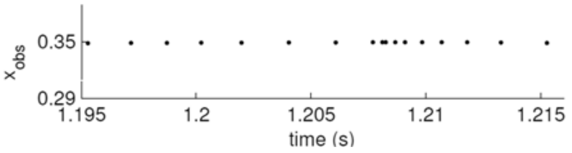


Figure 17: Step size at event detection.

7.4 Trajectory tracking

The deviations of the desired path can be seen in Figure 18 for PD control. It can be seen that for constant velocity for link 2 the deviation is proportional to the velocities. The position error however, decreases with decreasing velocities to zero as the global integral behaviour of the plant suggests.

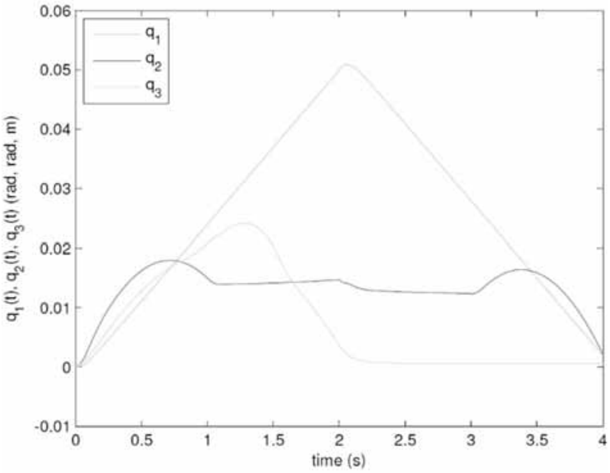


Figure 18: Position error PD-controller.

Figure 19 shows the deviations for the PID controller. When link 2 enters the part with constant velocity at $t = 1$, the position error for link 2 stabilises on zero after an initial overshoot. Here the PID-control comes into effect (as denoted in Section 6.3), zeros the velocity error for constant velocity and consequently the position error.

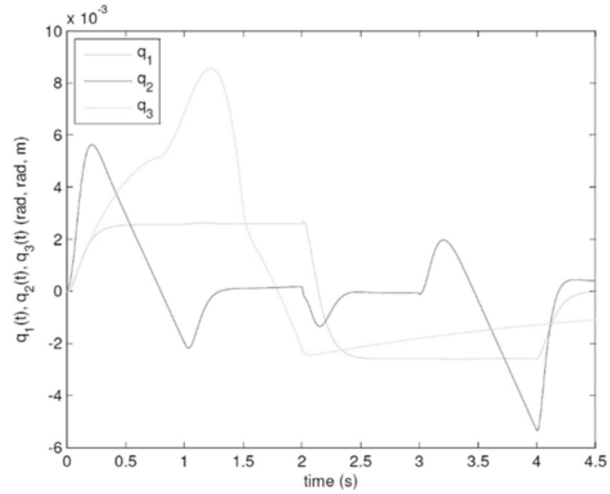


Figure 19: Position error PID-controller.

This can be shown with the response to a ramp input (Figure 20) for e.g. q_2 as the deviation for PID control to the ramp input vanishes [10].

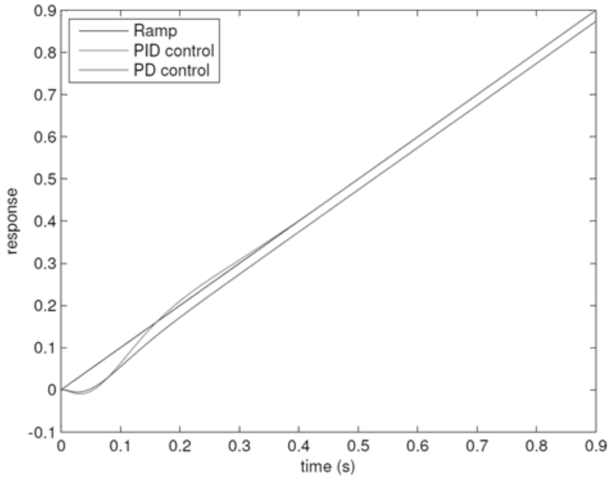


Figure 20: Ramp response.

As the two links are coupled, a little disturbance occurs when link 1 changes the sign of acceleration which is consequently absorbed by the controller. However, the position error of q_3 has an unexpected behaviour, which could result from the unaccounted gravity term. In Figure 19 it can be seen that at $t_{end} = 4$ for PID controller the model deviates from the trajectory, which consequently disappears as the controller integrates the position error and no steady state error remains.

Summing up it can be said that the error polynomials from Figure 18 are reduced one order to Figure 19 and the deviations are reduced approximately one order of magnitude due to the additional integrative term of the PID-controller. Hence, the overall performance has increased significantly.

References

[1] Fritzson P. Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach. ISBN 978-1-118-85912-4, 2014 Wiley-IEEE Press

[2] Ecker H, Breitenecker F. Comparison 11: SCARA Robot – Definition. SNE 8(22), 1998, 30-32.

[3] Simulink R2010/2016 Documentation

[4] SimMechanics User's Guide Version

[5] SimMechanics R2010b Documentation

[6] Shampine L. F. Solving 0 =F(t,y(t),y'(t)) in MATLAB. URL: http://faculty.smu.edu/shampine/cic.pdf, last retrieved Dec. 10, 2020.

[7] MATLAB R2010b/R2016a Documentation

[8] Wikipedia. Dormand-Prince method. Wikipedia, The Free Encyclopedia, 2020. [Online; acc. Dec. 10, 2020]

[9] Canudas de Wit C, Siciliano B. Theory of Robot Control. Springer, 1996.

[10] H.P. Joergl. Repetitorium Regelungstechnik. Band 1. Oldenbourg, 1993.

SNE Simulation News

EUROSIM Data and Quick Info

VESS – Virtual EUROSIM Seminar
Virtual Simulation Presentations, since June 2020 www.eurosim2023.eu

SIMS EUROSIM Conference 2021
Sept. 21-23, 2021, Oulu, Finland www.scansims.org

MATHMOD Vienna 2022
Feb. 16-18, 2022, Vienna, Austria www.mathmod.at

EUROSIM CONGRESS 2023
Spring/Autumn 2023, Amsterdam, The Netherlands www.eurosim2023.eu

Contents

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 Reports Editorial Board

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 modelling and simulation in industry, research, and development – by publication and conferences. → *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 Members* (*), 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*
NSSM	Russian National Simulation Society *Russian Federation*
SIMS	Simulation Society of Scandinavia *Denmark, Finland, Norway, Sweden*
SLOSIM	Slovenian Simulation Society; *Slovenia*
UKSIM	United Kingdom Simulation Society *UK, Ireland*
ALBSIM	Albanian Simulation Society*; *Albania*
ROMSIM	Romanian Society for Modelling and Simulation*; *Romania*
Societies in Re-Organisation:	
CROSSIM	*Croatian Society f. 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, and president, past president, and **SNE** representative. The President is nominated by the society organising the next **EUROSIM** Congress. Secretary, and Treasurer are elected out of members of the board.

President	M. Mujica Mota (DBSS), *m.mujica.mota@hva.nl*
Past President	Emilio Jiménez (CAE-SMSG), *emilio.jimenez@unirioja.es*
Secretary	Niki Popper, *niki.popper@dwh.at*
Treasurer	Felix Breitenecker (ASIM) *felix.breitenecker@tuwien.ac.at*
Webmaster	Irmgard Husinsky, *irmgard.husinsky@tuwien.ac.at*
SNE Editor	F. Breitenecker, *eic@sne-journal.org*

SNE – Simulation Notes Europe. **SNE** is **EUROSIM**’s scientific journal with peer reviewed contributions as well as a membership journal for **EUROSIM** with information from the societies. **EUROSIM** societies distribute **SNE** (electronic or printed) to their members as official membership journal. **SNE** Publishers are **EUROSIM**, **ARGESIM** and **ASIM**.

SNE Editor-in-Chief	Felix Breitenecker *eic@sne-journal.org*

→ *www.sne-journal.org*, ✉ *office@sne-journal.org*

EUROSIM Congress and Conferences.

Each year a major **EUROSIM** event takes place, the **EUROSIM CONGRESS** organised by a member society, **SIMS EUROSIM Conference**, and **MATHMOD Vienna Conference** (**ASIM**).

EUROSIM Congress 2019, the 10^{th} *EUROSIM Congress*, was organised by **CEA-SMSG**, the Spanish Simulation Society, in La Rioja, Logroño, Spain, July 1-5, 2019;

Due to Covid-19 virus in 2020 no **EUROSIM** events take place. To bridge this gap, **EUROSIM** is organising the series **VESS - Virtual EUROSIM Simulation Seminar** – seminars by simulation professionalists (2 hours via web), in preparation for upcoming **EUROSIM** events. → ***www.eurosim2023.eu***

Next main event is **SIMS EUROSIM Conference 2021**, September 21–23, 2021, Oulu, Finland. **SIMS**, the Scandinavian simulation society, extends every third year the annual **SIMS Conference** to the **SIMS EUROSIM Conference**.
→ ***www.scansims.org***

MATHMOD Vienna. This triennial **EUROSIM Conference** is mainly organized by **ASIM**, the German simulation society, and **ARGESIM,** with main co-sponsor **IFAC**.
MATHMOD 2022, the 10^{th} *MATHMOD Vienna Conference on Mathematical Modelling* will take place in Vienna, Februray 16-18, 2022. → ***www.mathmod.at***

EUROSIM Congress 2023, the 11^{th} *EUROSIM Congress*, will be organised by **DBSS**, the Dutch Benelux simulation society, in Amsterdam, Spring/Autumn 2023.
→ ***www.eurosim2023.eu***

Furthermore, **EUROSIM Societies** organize also local conferences, and **EUROSIM** co-operates with the organizers of the **I3M Conference Series.**
→ ***www.liophant.org/conferences/***

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*
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 A*pril 2020*

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/*

✉ *Egils.Ginters@rtu.lv*

⌧ Prof. Egils Ginters, Kirshu Str.13A, Cesis LV-4101, 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 November 2020

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

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*

☞ *bernt.lie@usn.no*

✉ SIMS / Bernt Lie, Faculty of Technology, Univ.College of Southeast Norway, Department of Technology, Kjølnes ring 56, 3914 Porsgrunn, Norway

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

Last data update February 2020

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 KingdomUKSIM Officers

President	David Al-Dabass, *david.al-dabass@ntu.ac.uk*
Secretary	T. Bashford, *tim.bashford@uwtsd.ac.uk*
Treasurer	D. Al-Dabass, *david.al-dabass@ntu.ac.uk*
Membership chair	G. Jenkins, *glenn.l.jenkins@smu.ac.uk*
Local/Venue chair	Richard Cant, *richard.cant@ntu.ac.uk*
Repr. EUROSIM	Dr Taha Osman, taha.osman@ntu.ac.uk
Deputy	T. Bashford, *tim.bashford@uwtsd.ac.uk*
Edit. Board SNE	D. Al-Dabass, *david.al-dabass@ntu.ac.uk*

Last data update March 2020

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

ALBSIM – Albanian Simulation Society

The Albanian Simulation Society has been initiated at the Department of Statistics and Applied Informatics, Faculty of Economy at the University of Tirana, by Prof. Dr. Kozeta Sevrani.

The society is involved in different international and local simulation projects, and is engaged in the organisation of the conference series ISTI - Information Systems and Technology. In July 2019 the society was accepted as 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	
Chairt	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 July 2019

Societies in Re-Organisation / Former Societies

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

- CROSSIM – *Croatian Society for Simulation Modelling*
  Contact: Tarzan Legović, *Tarzan.Legovic@irb.hr*
- FRANCOSIM – Société Francophone de Simulation
- HSS – Hungarian Simulation Society
- ISCS – Italian Society for Computer Simulation

The following societies have been formally terminated:

- MIMOS –Italian Modeling & Simulation Association; terminated end of 2020.

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 A**RGESIM 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
- Support of the Conference Series **MATHMOD Vienna** (triennial, in co-operation with **EUROSIM**, **ASIM**, and **TU Wien**) – www.mathmod.at
- Administration of **ASIM** (German Simulation Society) and administrative support for **EUROSIM** *www.eurosim.info*
- Simulation activities 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

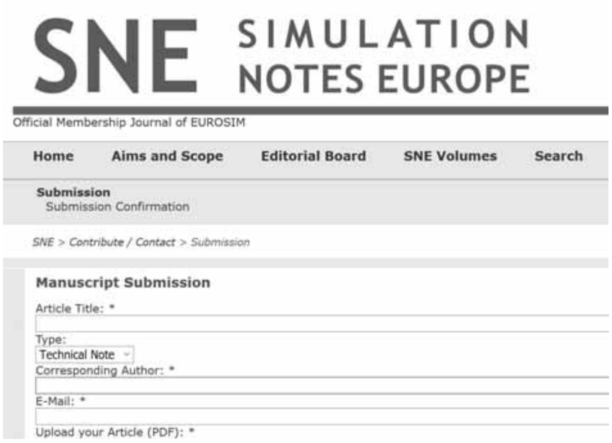
The scientific journal **SNE** – *Simulation Notes Europe* provides an international, high-quality forum for presentation of new ideas and approaches in simulation – from modelling to experiment analysis, from implementation to verification, from validation to identification, from numerics to visualisation – 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 Implemen-tations* 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:


SNE SIMULATION NOTES EUROPE
Official Membership Journal of EUROSIM
Home
Aims and Scope
Editorial Board
SNE Volumes
Search
Submission
Submission Confirmation
SNE > Contribute / Contact > Submission

Manuscript Submission

Article Title: *

Type:
Technical Note

Corresponding Author: *

E-Mail: *

Upload your Article (PDF): *

ASIM ASIM ASIM

ASIM Books – ASIM Book Series – ASIM Buchreihen

Proceedings

Simulation in Production and Logistics 2019 – 18. ASIM Fachtagung Simulation in Produktion und Logistik
18.-20. 9. 2019; M. Putz, A. Schlegel (Hrsg.), Verlag Wissenschaftliche Skripten Auerbach, 2019,
ISBN print 978-3-95735-113-5, ISBN ebook 978-3-95735-114-2; ASIM Mitteilung AM172

Tagungsband ASIM SST 2018 - 24. ASIM Symposium Simulationstechnik, HCU Hamburg, Oktober 2018
C. Deatcu, T. Schramm, K. Zobel (Hrsg.), ARGESIM Verlag Wien, 2018; ISBN print: 978-3-901608-12-4; ISBN ebook: 978-3-901608-17-9; 10.11128/arep.56; ARGESIM Report 56; ASIM Mitteilung AM 168

Simulation in Production and Logistics 2017 – 17. ASIM Fachtagung Simulation in Produktion und Logistik
Sigrid Wenzel, Tim Peter (Hrsg.); ISBN Print 978-3-7376-0192-4, ISBN Online 978-3-7376-0193-1, Kassel university press GmbH, Kassel, 2017; ASIM Mitteilung AM164

Tagungsband ASIM SST 2016 - 23. Symposium Simulationstechnik, HTW Dresden, September 2016
T. Wiedemann (Hrsg.); ARGESIM Verlag Wien, 2016; ISBN ebook 978-3-901608-49-0;
ARGESIM Report 52; ASIM Mitteilung AM 160

Books

Kostensimulation - Grundlagen, Forschungsansätze, Anwendungsbeispiele
T. Claus, F. Herrmann, E. Teich; Springer Gabler, Wiesbaden, 2019; Print ISBN 978-3-658-25167-3;
Online ISBN 978-3-658-25168-0; DOI 10.1007/978-3-658-25168-0; ASIM Mitteilung AM 169

Simulation und Optimierung in Produktion und Logistik – Praxisorientierter Leitfaden mit Fallbeispielen.
L. März, W. Krug, O. Rose, G. Weigert (Hrsg.); ISBN 978-3-642-14535-3, Springer, 2011; AM 130

Book Series Fortschrittsberichte Simulation – Advances in Simulation

Methods for Hybrid Modeling and Simulation-Based Optimization in Energy-Aware Production Planning. B. Heinzl, FBS 37
ISBN ebook 978-3-903347-37-3, DOI 10.11128/fbs.37, ARGESIM Publ. Vienna,2020; ISBN print 978-3-903311-11-4, TUVerlag Wien, 2020

Konforme Abbildungen zur Simulation von Modellen mit verteilten Parametern. Martin Holzinger, FBS 36
ISBN ebook 978-3-903347-36-6, DOI 10.11128/fbs.36, ARGESIM Publ. Vienna, 2020; ISBN print 978-3-903311-10-7, TUVerlag Wien, 2020

Fractional Diffusion by Random Walks on Hierarchical and Fractal Topological Structures. G. Schneckenreither, FBS 35
ISBN ebook 978-3-903347-35-9, DOI 10.11128/fbs.35, ARGESIM Publ. Vienna, 2020

A Framework Including Artificial Neural Networks in Modelling Hybrid Dynamical Systems. Stefanie Winkler. FBS 34
ISBN ebook 978-3-903347-34-2, DOI 10.11128/fbs.34, ARGESIM Publ. Vienna, 2020; ISBN print 978-3-903311-09-1, TUVerlag Wien, 2020

Modelling Synthesis of Lattice Gas Cellular Automata and Random Walk and Application to Gluing of Bulk Material. C. Rößler, FBS 33
ISBN ebook 978-3-903347-33-5, DOI 10.11128/fbs.33, ARGESIM Publ. Vienna, 2020; ISBN print 978-3-903311-08-4, TUVerlag Wien, 2020

Combined Models of Pulse Wave and ECG Analysis for Risk Prediction in End-stage Renal Desease Patients. S. Hagmair, FBS 32
ISBN ebook 978-3-903347-32-8, DOI 10.11128/fbs.32, ARGESIM Publ. Vienna, 2020

Mathematical Models for Pulse Wave Analysis Considering Ventriculo-arterial Coupling in Systolic Heart Failure. S. Parragh, FBS 31
ISBN ebook 978-3-903347-31-1, DOI 10.11128/fbs.31, ARGESIM Publ. Vienna, 2020

Variantenmanagement in der Modellbildung und Simulation unter Verwendung des SES/MB Frameworks. A. Schmidt, FBS 30;
ISBN ebook 978-3-903347-30-4, DOI 10.11128/fbs.30, ARGESIM Verlag, Wien 2019; ISBN print 978-3-903311-03-9, TUVerlag Wien, 2019

Classification of Microscopic Models with Respect to Aggregated System Behaviour. Martin Bicher, FBS 29;
ISBN ebook 978-3-903347-29-8, DOI 10.11128/fbs.29, ARGESIM Publ. Vienna, 2019

Model Based Methods for Early Diagnosis of Cardiovascular Diseases. Martin Bachler, FBS 28;
ISBN ebook 978-3-903347-28-1, DOI 10.11128/fbs.28, ARGESIM Publ. Vienna, 2017; ISBN print 978-3-903024-99-1, TUVerlag Wien, 2019

A Mathematical Characterisation of State Events in Hybrid Modelling. Andreas Körner, FBS 27;
ISBN ebook 978-3-903347-27-4, DOI 10.11128/fbs.27, ARGESIM Publ. Vienna, 2016; ISBN print 978-3-903311-07-7, TUVerlag Wien, 2019

Comparative Modelling and Simulation: A Concept for Modular Modelling and Hybrid Simulation of Complex Systems. N.Popper, FBS 26; ISBN ebook 978-3-903347-26-7, DOI 10.11128/fbs.26, ARGESIM Publ. Vienna, 2016

Rapid Control Prototyping komplexer und flexibler Robotersteuerungen auf Basis des SBE-Ansatzes. Gunnar Maletzki, FBS 25;
ISBN ebook 978-3-903347-25-0, DOI 10.11128/fbs.25, ARGESIM Publ. Vienna, 2019; ISBN Print 978-3-903311-02-2, TUVerlag Wien, 2019

A Comparative Analysis of System Dynamics and Agent-Based Modelling for Health Care Reimbursement Systems. P. Einzinger, FBS 24; ISBN ebook 978-3-903347-24-3, DOI 10.11128/fbs.24, ARGESIM Publ. Vienna, 2016

Agentenbasierte Simulation von Personenströmen mit unterschiedlichen Charakteristiken. Martin Bruckner, FBS 23;
ISBN ebook Online 978-3-903347-23-6, DOI 10.11128/fbs.23, ARGESIM Verlag Wien, 2016

Deployment of Mathematical Simulation Models for Space Management. Stefan Emrich, FBS 22;
ISBN ebook 978-3-903347-22-9, DOI 10.11128/fbs.22, ARGESIM Publisher Vienna, 2016

Lattice Boltzmann Modeling and Simulation of Incompressible Flows in Distensible Tubes for Applications in Hemodynamics.
X. Descovich, FBS 21; ISBN ebook 978-3-903347-21-2, DOI 10.11128/fbs.21, ARGESIM, 2016; ISBN Print 978-3-903024-98-4, TUVerlag 2019

Mathematical Modeling for New Insights into Epidemics by Herd Immunity and Serotype Shift. Florian Miksch, FBS 20;
ISBN ebook 978-3-903347-20-5, DOI 10.11128/fbs.20, ARGESIM Publ. Vienna, 2016; ISBN Print 978-3-903024-21-2, TUVerlag Wien, 2016

Integration of Agent Based Modelling in DEVS for Utilisation Analysis:The MoreSpace Project at TU Vienna. S.Tauböck: FBS19;
ISBN ebook 978-3-903347-19-9, DOI 10.11128/fbs.19, ARGESIM Publ., 2016; ISBN Print 978-3-903024-85-4, TUVerlag Wien, 2019

Download via ASIM - **www.asim-gi.org** **Print-on-Demand** via TUVerlag **www.tuverlag.at**

EUROSIM Conferences and Congress
VESS – Virtual EUROSIM Simulation Seminar

Due to Corona Virus, also **EUROSIM** changed the schedule of **EUROSIM Conferences** and the **EUROSIM Congress** – all events will take place one year later. To bridge the 2020 conference gap the **EUROSIM** societies organise virtual conferences, and the **EUROSIM Board** started in June 2020 **VESS** – the **Virtual EUROSIM Seminar**, a series of online presentations discussing trends in modelling and simulation and preparing emphasis of future **EUROSIM** events.

DBSS
Dutch Benelux Simulation Society
www.eurosim2023.eu
VESS - VIRTUAL EUROSIM SIMULATION SEMINAR
free online seminar series

The **EUROSIM Board** and **DBSS** started in June 2020 **VESS** – the **Virtual EUROSIM Seminar**, a series of online presentations discussing trends in modelling and simulation. These international online simulation seminars – monthly or bi-monthly – are open to everybody, via Zoom, lasting 60 minutes (45 minutes presentations, 15 minutes Q & A).
Information and informal registration via website **www.eurosim2023.eu**

SIMS EUROSIM 2021 Conference on Modelling and Simulation
21-23 September 2021 in Oulu, Finland

The First **SIMS EUROSIM Conference** on Modelling and Simulation, **SIMS EUROSIM 2021** takes place in Oulu, Finland, September 21-23, 2021. The 62nd International Conference of Scandinavian Simulation Society, SIMS 2021, is embedded with **SIMS EUROSIM 2021**. The **SIMS EUROSIM** conference will be organized every third year by **SIMS** and **EUROSIM**.
The background of this conference series is in the 60-years history of Scandinavian Simulation Society, **SIMS**.
The program of the **SIMS EUROSIM 2021 Conference** will have a multi-conference structure with several special topics related to methodologies and application areas. The program includes invited talks, parallel, special and poster sessions, exhibition and versatile technical and social tours – info **www.scansims.org**

MATHMOD 2022
Vienna
10th Vienna International Conference
on Mathematical Modelling
February 16 - 18, 2022, Vienna, Austria

MATHMOD organizers continue the conference series one year later, with **10th MATHMOD 2022**, February 16-18, 2022. **MATHMOD 2022**, one of **EUROSIM**'s main events, provides a forum for professionals, researchers, and experts in the field of theoretic and applied aspects of mathematical modelling for systems of dynamic nature.
The scope of the **MATHMOD 2022** conference covers theoretic and applied aspects of various types of mathematical modelling (equations of various types, automata, Petri nets, bond graphs, qualitative and fuzzy models) for systems of dynamic nature (deterministic, stochastic, continuous, discrete or hybrid) – info and details **www.mathmod.at**

EUROSIM 2023, the **11th EUROSIM Congress**, will take place in Amsterdam, The Netherlands, Spring/Autumn 2023. It will be organized by the Dutch Benelux Simulation Society (www.dutchbss.org) supported mainly by their corporate members like TU Delft, Amsterdam University of Applied Sciences, EUROCONTROL and IGAMT (www.igamt.eu).
Due to the growth of Simulation and its relationship with other analytical techniques like Big Data, AI, Machine Learning, Large Scale Simulation and others, the event will be structured, for the first time, in dedicated tracks focused on different areas and applications of Simulation ranging from aviation to health care and humanitarian activities. We have the ambition to attract at the congress participants from Academia, industry and governmental representatives to share the latest developments in Simulation and related activities and applications.
Please follow the news and activities towards the **EUROSIM 2023** at **www.eurosim2023.eu**

www.tuverlag.at
ISBN 978-3-903311-15-2