Politecnico di Milano
Dipartimento di Elettronica, Informazione e Bioingegneria
Doctoral Programme In Information Technology

Automatic Model Simplification for Continuous and Discontinuous Systems

Doctoral Dissertation of:
Alessandro Vittorio Papadopoulos

Advisor:
Prof. Alberto Leva

Tutor:
Prof. Marco Lovera

Supervisor of the Doctoral Program:
Prof. Carlo Fiorini

2013 - XXVI

Acknowledgements

Many people helped and encouraged me in the last years, and if I managed to reach this goal, it is only thanks to them.

First of all, I would like to thank my advisor Alberto. He has been a great supervisor, mentor and man. The most important thing that he taught me is to be a good researcher.

I would like also to thank Francesco. His help and his intelligence have been of fundamental importance for the development of the core parts of this work, especially the long discussion that we had in Vienna, which put the basis of the first contribution of the thesis.

My stay in Lund totally changed my life, and made me grow up as a researcher, also thanks to Johan, who not only gave me many opportunities and fruitful contacts, but also great ideas and advices.

A special thanks goes also to Maria. She helped me a lot during those last years, and she worked side by side with me to some parts of this dissertation, continuously inspiring me with her great talent and precision. Moreover, she has become more than a colleague, I would say a very good friend.

Speaking about friends, I would like to thank all the postdocs, PhD student, and colleagues that I met in the last years, especially (in alphabetical order) Andrea M., for all the laughs we had together, Daniela, for the deep discussions about life, future and whatever, Fabio, who has become one of my best friends, Marcello, for the long discussions at lunch time, Marco B., for the videos that we used to exchange, last but not least, Martina, who has been (*and will be*) my PR, and who has been really inspiring for me, apart from an undoubtedly good friend.

I would like to thanks also my friends Bea, Chiara and Carlotta, that have been with me for more than ten years, always supporting and loving me. Thanks Martina for always believing in me.

Finally, I would like to thank my family, especially my sister, who encouraged me a lot to make the big step of moving abroad. I am missing all of you, and I will love you forever.

I would really like to thank many other people, and since I am writing these acknowledgements quite in a rush, I will forget someone for sure. Sorry for that.

Alessandro,
Lund (SE), January 31st, 2014

Abstract

In the last years equation-based languages straightened and simplified the way sophisticated models of complex physical systems are built. In particular, the object-oriented modelling paradigm allows to obtain such models in an affordable way by focusing on the development of single "building blocks," i.e., the objects, and connecting them so as to obtain the overall description of the physical system. This often results in large and complicated models, hard to simulate and in general to manage. The aim of the research path to which this work belongs, is to provide methodologies and automatic techniques to cope with said complexity, in order to simplify and streamline the model analysis and simulation.

In the context just sketched, the thesis proposed several contributions. The first one is the development of a simplification framework which includes most of the model manipulation techniques available in the literature and novel ones. Functional to this contribution is the proposal of a technique called "Cycle Analysis", able to perform a structural analysis of a DAE system, and return a dependency graph representing the way the dynamic variables are interacting, associating with each dynamic variable a time scale. Such information can be used to ease and automate the partition of a complex model into decoupled subsystems.

Novel indices are introduced to characterise some structural properties of the system, like its stiffness, and to quantify "how much" a system is suited to be partitioned in the sense above. Also, the results of the Cycle Analysis are used for improving the simulation efficiency by means of mixed-mode integration methods, and of a co-simulation (multi-rate) architecture, showing the effectiveness of the approach on some applications of interest.

In addition, an extension to classical model order reduction techniques conceived for continuous systems is here proposed for hybrid ones, focusing on the switched affine case.

As a result of the above contributions, and moving from the methodological to the technological side of the addressed matter, the integration of the proposed simplification framework in state-of-the-art modelling and simulation tools is here considered, providing a viable and complete solution.

Riassunto

Negli ultimi anni, i linguaggi di modellazione basati su equazioni hanno semplificato il modo di costruire modelli sofisticati di sistemi fisici complessi. In particolare, il paradigma di modellazione object-oriented permette di ottenere tali modelli con uno sforzo accettabile, concentrandosi sullo sviluppo dei singoli "blocchi costitutivi," vale a dire, gli oggetti, e collegandoli in modo da ottenere la descrizione del sistema fisico complessivo. Questo spesso si traduce in modelli di grande dimensione e complessità, difficili da simulare e, in generale, da gestire. L'obiettivo del percorso di ricerca a cui questo lavoro appartiene è quello di definire metodologie e tecniche automatiche per affrontare la suddetta complessità, al fine di semplificare e razionalizzare l'analisi del modello e la sua simulazione.

Nel contesto appena delineato, questa tesi propone diversi contributi. Il primo è lo sviluppo di un framework di semplificazione in grado di comprende la maggior parte delle tecniche di approssimazione disponibili in letteratura oltre ad altre nuove. Funzionale a questo contributo è la proposta di una tecnica chiamata " Cycle Analysis," in grado di eseguire un'analisi strutturale di un sistema algebrico-differenziale e di restituire un grafo di mutue dipendenze che rappresenta il modo in cui le variabili dinamiche interagiscono, associando ad ogni variabile dinamica una scala temporale. Tali informazioni possono essere utilizzate per facilitare e automatizzare la partizione di un modello complesso in sottosistemi disaccoppiati.

Nel lavoro vengono anche introdotti indici innovativi in grado di caratterizzare alcune proprietà strutturali del sistema, come la sua stiffness, e di esprimere "quanto" un sistema si presta ad essere partizionato. Inoltre, i risultati della Cycle Analysis sono utilizzati per migliorare l'efficienza di simulazione mediante metodi di integrazione mixed-mode e di un'architettura co-simulazione (multi-rate) che mostra l' efficacia del metodo in alcune applicazioni significative.

Si propone inoltre un'estensione per tecniche di riduzione d'ordine classiche, concepite per sistemi continui, al caso di sistemi ibridi, concentrandosi sul caso switched affine.

Come risultato dei contributi di cui sopra, e spostando l'attenzione dal lato metodologico dell'argomento a quello tecnologico della questione affrontata, il lavoro tratta l'integrazione del framework di semplificazione proposto in tool di modellazione e simulazione corrispondenti all'attuale stato dell'arte, fornendo una soluzione valida e completa.

Contents

Chapter 1

Introduction and background

Computer simulation is nowadays a fundamental activity in the day by day work of scientists and engineers, who are respectively focused on observing and understanding the world, and on devising and designing technological solutions to improve it. Both attitudes and professions require to turn knowledge, comprehension, and ingenuity into something quantitative, so as to verify the correctness of explanatory theories for the observed phenomena, and to aid the decision processes that permeate any design activity.

In the scientific and technical literature, the task of creating suitable representations of the world for purposes like those just mentioned, is called *modelling*. The task of employing models to provide the required quantitative data is conversely called *simulation*.

> “ The process of modeling concerns itself with the extraction of knowledge from the physical plant to be simulated, organizing that knowledge appropriately, and representing it in some unambiguous fashion.
>
> *Cellier and Kofman [2006]*

This dissertation is part of a long-term research on modelling and simulation. The particular problem addressed herein is that of aiding the scientist/engineer (hereinafter “the modeller” or “the analyst” to better evidence the aspects of her/his activity that are of concern for this work) obtain models with exactly the complexity degree that is adequate for answering the numerous and different questions arising along a scientific or engineering process.

Assuming that the model of the phenomena of interest is initially created at the maximum conceivable complexity degree, obtaining from this single model all those required to answer the numerous questions above, is a matter of *simplification*. We could thus say, at least at the introductory level of this chapter, that this thesis is concerned with *model simplification*.

Quite intuitively, however, the simplification problem has a number of facets. Some of these are related to the nature of the phenomena to be handled. Others conversely refer to the technology and tools used for modelling and simulation. Others, finally, come from the necessity of making the power of modelling and simulation, which ultimately relies on very sophisticated mathematical concepts, available to analysts who are experts of the addressed physical domain, not of the mentioned mathematical theories. We could further specify the scope of this thesis as devising simplification techniques that are general with respect to the physical domain, easy to integrate in state-of-the-art tools, and as transparent as possible for the analyst.

The rest of this chapter is devoted to giving a more precise and formally qualified idea of the concepts exposed above in a very abstracted way. To this end, it is first necessary to introduce the fundamental concepts of dynamic modelling and simulation, and then describe how modern modelling and simulation tools are conceived and what are their present technological development trends. This will lead us to discuss the life cycle of a dynamic simulation model in the context of scientific and engineering projects, and as a consequence, to evidence the parts of that life cycle that naturally appear of interest for the application of model simplification techniques. Once, this introductory *excursus* is completed, the chapter ends by outlining which of the mentioned aspects are the subject of the presented research, and consequently by describing how the rest of the thesis is organised.

1.1 Dynamic modelling and simulation

In extreme synthesis, modelling means writing the equations that rule the phenomena of interest. Some of these equations will invariantly be differential. As a consequence, the most natural form of a model for our purposes is that of a Differential and Algebraic Equation (DAE) system. Also, given the presence of differential equations, the solution of such a system for given initial conditions and exogenous *stimuli*, the computation of which is dynamic simulation, will invariantly require integration, most frequently to be performed numerically given the complexity of the encountered problems. The adjective “dynamic” attributed to the considered systems indicates that their condition at any particular moment in time depends on the past history of exogenous *stimuli*, and on the *initial* condition of the system itself. For brevity, we shall sometimes omit the adjective dynamic, but the reader should bare in mind that many system of scientific and engineering interest, enjoys that property.

Modelling and simulation are important in a huge number of contexts.

Just to give some examples, much of climate science is wholly based on simulation models, that given the impossibility of obtaining certain experimental data are sometimes the only available decision aid. Simulation methods are also common in various disciplines of social sciences, management [Pidd and Carvalho, 2006], and what is more important for the purpose of this work, in engineering applications. These range from energy systems [Casella and Leva, 2003, Ordys et al., 1994], through robotics [Hast et al., 2009, Žlajpah, 2008], up to automotive [Arnold et al., 2011, Schmitt et al., 2009, Zimmer and Otter, 2010] and virtually any field of modern engineering [Mattsson et al., 1998].

Focusing for a moment on engineering, simulation models are nowadays used to take decisions at virtually any stage of a project, and even to stipulate and mutually assess the behaviour of parts being created by different manufacturers before they are assembled, see, e.g., Blochwitz et al. [2012].

This evidences another relevant fact for this work. While in scientific applications one has to observe, model and simulate existing objects, this is very often not true in engineering, where modelling and simulation are frequently used exactly to determine how something not yet existing has to be. This is called "virtual prototyping", and nowadays viewed as a powerful means to achieve competitive advantages in the traditional product development process. Specifically, for example, one may desire certainties on the behaviour of (parts of) the system being designed as soon as possible, so as to reduce – and ideally eliminate – iterative experimental tests interleaved with the consequently decided and product modifications. Of course, real-life product development cycles will always need some physical prototypes for final evaluations, but by using a combination of virtual prototypes of mathematical models, the development time can potentially be dramatically shortened.

Moreover, virtual prototyping is often enormously faster and less expensive than physical prototyping, and opens the additional possibility to conduct tests that would be simply financially unacceptable or even physical impossible on the real plant or product. For example, when developing a control system for a landing gear of an aircraft, several engineers can test their control system simultaneously by simulating a model of the landing gear, instead of using direct access of a physical prototype. Sometimes, physical experiments can be even dangerous; for example, when testing "what-if" *scenarii* on a power plant, the necessity of considering extreme operating conditions makes it apparently preferable to use a simulation model.

Simulation can also significantly reduce the time-to-market of virtually any product, providing a great industrial competitive advantage [Norton, 2001]. In addition, at their most basic level modelling and simulation tools enable engineers, designers and product developers to work together concurrently within a virtual environment to solve design, manufacturing and maintainability issues at the earliest stage of product development. Finally, the mentioned tools often provide also a virtual reality, or more generally, 3D environment, capable to visualise the "real" behaviour of the product and of its functionalities [Ferretti et al., 1999, Kunze et al., 2009].

1.2 Modelling and simulation technologies

Given the importance of modelling and simulation, that is testified by a huge literature of which the previous section has just reported a few examples, a great effort has been spent in the last decades in both the research on mathematical methodologies and the development of effective technologies.

1.2.1 General concepts

When projecting the importance and the expected advantages of modelling and simulation onto the work of scientists and engineers, in fact, one can immediately notice that the mentioned evolution has essentially two consequences, that are somehow conflicting with one another.

On one side there is the modelling part. Model creation and management tools have been dramatically improving, and at present provide a lot of different functionalities, and – more important – allow to construct extremely complex models on lightweight computational platforms, like, e.g., a laptop, in an affordable way; just to give some examples of said tools, Matlab and Modelica-based ones are among the most widely used in this direction, both in academic and industrial contexts.

On the other side there is simulation, not only in the strict sense of "integrating a dynamical system" stated above, but including also the set of manipulation and solution techniques that "transform" the model into an *equivalent* one suitable for the numerical solution. The need has become more and more strong for such manipulation and solution techniques that can run on the same platforms used for the modelling phase, and handle the resulting large and complex models efficiently enough.

In fact, whereas modelling tools and languages ease the work of constructing complex models, there is hardly an adequate counterpart – in terms of symbolic manipulation and simulation techniques – able to cope with the resulting complexity in an effective way. In other words, in the present *scenario*, the available computational resources often become the bottleneck of simulation-based studies, and therefore achieving an efficient integration of complex models becomes even more important.

Avoiding a detailed review that would stray from the scope of this introductory chapter, we could synthetically say that the situation just sketched has given rise to various technological development lines, three of which are worth mentioning in the context of this work.

The first line has been taking care of simplifying the creation of complex models by allowing the analyst to aggregate models of individual components, written independently of their possibly different roles in the various aggregate systems that will contain them. Focusing on this aspect, has led to the principles of *equation-based object-oriented modelling* [Mattsson et al., 1998].

The second development line has been concerned with the creation or reformulation of models that are in a form that is suitable for the numerical solution, thus defining *a priori* which are the input and the output variables—typically ODE form. This approach thus deals with "oriented" or "causal" models (all the mentioned terms will be extensively discussed later on) and has given rise to *block-oriented modelling* [Danby and Harman, 2003].

Finally, the third line has been addressing the hardware and software environments to host the solution of dynamic models, in a view to maximising simulation efficiency from the architectural point of view. The most notable product of such developments is the number of co-simulation techniques and environments proposed to date [Bastian et al., 2011, Schierz et al., 2012].

1.2.2 A brief overview of current tools

As already mentioned, modern modelling and simulation tools provide a set of functionalities that permit the analyst to build complex models in an affordable way. Such tools gained a lot of attention in the last decades, since the modeller just need to focus on how to create the models rather than on how to simulate them, without requiring her/him to be an expert in numerical analysis.

This has been possible thanks to the advances and to the evolution of high-level programming languages, thanks to the improved compilers technologies and capabilities, and thanks to the software engineers and developers who (still) provide the tools that enabled this shift and abstraction from the computing architecture.

In the early days, indeed, a huge proportion of the effort in a simulation project involved getting a model into a computable form, so as it could be used in programming, testing and refinement, all activities for which the modeller is not necessary an expert.

> “Anyone who has written a reasonably complex simulation program in a language like FORTRAN or C++ knows the feeling of relief when it *seems* to work properly—-and the all too common despair when a bug appears later. Eventually the simulation software will be declared fit to its purpose, but behind this will have been many bugs and development problems sorted out one by one, in a laborious process of debugging and verification.
>
> *Pidd and Carvalho [2006]*

Nowadays, the existence of advanced modelling and simulation languages allows a modeller to take for granted that the simulation model will run once it has been built, and that it is “semantically correct”, i.e., the produced results are the numerical integration of the equations—this concept will be clarified better in the next chapters. Thanks to simulation tools, the emphasis is thus on the conceptualisation and use of a model to think through options for change or to develop insight. In other words, modern simulation tools have shifted the emphasis from programming and software development to modelling and model use.

On the market there is a number of different off-the-shelves modelling and simulation tools, with different features and purposes. Some of them are domain specific, e.g., Adams[1] for multibody systems, or PSpice [Monssen, 2001] for electronic circuits, other are more general purpose and multi-domain, e.g., Matlab [Hanselman and Littlefield, 2005], Simulink [Danby and Harman, 2003], or Modelica-based tools [Fritzson, 2003] like Dymola[2], JModelica[3] and OpenModelica[4], and so forth. Whatever is the case, however some common and general features of such kind of application can be identified.

In particular, a modelling and simulation tool must present the following features.

- Modelling tools, such as
  - A graphical and/or textual editor environment.

[1] `www.mscsoftware.com/product/adams`

[2] `www.3ds.com/products-services/catia/portfolio/dymola`

[3] `www.jmodelica.org`

[4] `www.openmodelica.org`

  - Built-in functionalities and operators, and libraries with defined properties and behaviour.
  - Property sheets and visual controls to enable simulation parameters to be set and varied.
  - Sampling routines and other utilities employed in the model.

- Tools to execute the simulation, such as
  - A simulation engine to run a model.
  - Animated graphics (e.g., plots or 3D visualisations) to allow a user to view the model state as the simulation proceeds.
  - Simulation run control to enable the user to interact safely with the simulation as it runs.
- Tools to support experimentation, such as
  - Experimental frames that define run lengths, outputs and parameters.
  - Analysis tools that enable results to be interpreted and presented.
  - Optimization tools or other add-ons.
- Links to other software such as spreadsheets, databases and corporate systems.

Figure 1.1 sketches out all those features, representing the general software architecture of a generic modelling and simulation tool.

Nonetheless, the resulting simulation software produced by the modelling tool needs fulfilling some important properties, which are typical in the software engineering domain [Ghezzi et al., 2002], and that are nowadays somehow implicitly assumed to hold true, thanks again to the aforementioned advances in the back-end of modelling and simulation tools, that made transparent to the user the model manipulation and the automatic simulation code generation.

First of all, the simulation software should be *dependable*, i.e., it must be able to deliver the intended functionality when requested without causing danger or damages, and handling error conditions. In other words, the simulation software must be reliable, available, safe, secure and robust in the software engineering strict sense.

Second, the simulation software should be *usable*, i.e. it provides a user-friendly interface, allowing different levels of utilisation, detection and recovery from input errors. The simulation software must be easy to learn

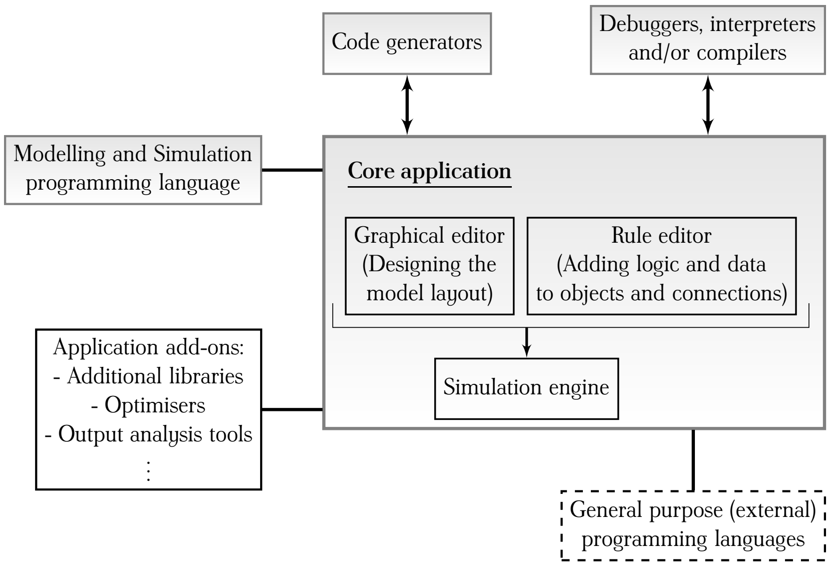

Code generators
Debuggers, interpreters and/or compilers
Modelling and Simulation programming language
Core application
Graphical editor (Designing the model layout)
Rule editor (Adding logic and data to objects and connections)
Simulation engine
Application add-ons:
- Additional libraries
- Optimisers
- Output analysis tools
⋮
General purpose (external) programming languages

Figure 1.1: A typical modelling and simulation tool consists of a core application, a wide range of add-ons, and the possibility of interfacing to some external programming languages or data structures.

and operate, adaptable to specific purposes, and recoverable from user errors. Even if this property may seem among the less important, it is among the main factors determining the success/failure of any software tool.

Another important feature is the *modularity*, i.e., the ability to incrementally build by composition functions, procedures, modules and components, and to allow easy extensions to the produced software. Modularity is quite important when dealing with complex engineering projects.

Finally, *reusability*, i.e., the ability to run on different configurations, be composed with other programs and to communicate with packages of different vendors. This last feature has been underestimated for many years, but recently gained a lot of importance, especially when dealing with co-simulation environments. In fact, in this case defining communication and exchange standards becomes crucial to compose different simulation software to achieve a unique co-simulation environment.

While the first feature is strictly related to the modelling and simulation tool back-end, the others are more related to the modelling language and to the modelling and simulation front-end. In our opinion all those aspects are important to evaluate the quality of a tool, and must not be underestimated.

Apparently enough, all the presented features are desirable in any modelling and simulation tool, and under this viewpoint it seems that there is little space for differentiation and customisation of one tool with respect to the others. This is, however, far from being true. As will be shown in the next sections, there exist a number of different modelling languages and paradigms suited for simulation software accommodating different modelling needs and contexts.

1.2.3 A brief review of modelling languages and paradigms

Even if there are some common features in the available tools a key role in the diversification of the tools plays the modelling and simulation language. In fact, there are different modelling paradigms induced by the programming language of the specific tool.

The possible paradigms can be broadly classified into two classes

- Algorithm-based,
- Equation-based.

Algorithm-based are imperative programming languages, i.e., approaches that consist in describing exactly which are the steps to achieve a desired result from the input data. This kind of approach is the same paradigm used for most of the programming languages, where programs are described with looping structures and if-then-else statements.

Notable examples of this kind of languages are Matlab, Simulink (with the corresponding open source alternatives Octave[5], Scilab[6], and XCos) and LabVIEW[7], apart from classical programming languages like, e.g., FORTRAN and C++.

Another worth mentioning language is Python, which in the last years offered many high performance open source libraries for numerical and scientific computing. Among those libraries, NumPy and SciPy[8] are the most widely adopted and well-established. In particular, those libraries provide high level "Matlab-like" functionalities, offering a very nice alternative to Matlab, while additionally providing all the advantages of a general purpose programming language, like Python. On the basis of those libraries, a lot of very interesting and high-performance tools have been developed like, e.g., PySimulator [Pfeiffer et al., 2012] and Assimulo [Andersson et al., 2011, 2012] for dynamic simulation, and Pyomo [Hart et al., 2012] for optimisation.

[5] `www.gnu.org/software/octave`
[6] `www.scilab.org`
[7] `www.ni.com/labview`
[8] `www.scipy.org/`

Equation-based languages, on the other hand, are essentially functional (or declarative) approaches. In this case, modellers just have to write the equations and do not care how from those equations a simulation will be carried out. In fact, in the computer science domain, functional programming seeks to describe *what* one wants to achieve rather than specify *how* to achieve it. In classical programming languages, this approach may seem like the more confusing way to go about things, and that is the reason why Lisp, Scheme, and Haskell have never really surpassed C, C++, Java and COBOL in commercial popularity. But in the specific context of modelling and simulation tools, this approach is nearer to the classical activity of the modeller, which is more used to write the equations in terms of mass, energy or momentum balance rather than on putting them in a computable form. In addition, functional programming usually requires orders of magnitude less code than imperative programming. That means fewer points of failure, less code to test, and a more productive – and, many would say, happier – programming life. As systems get bigger, this has become more and more important.

Notable examples of this kind of languages are gProms Oh and Pantelides [1996] for chemical engineering, Modelica for multi-domain physical modelling, Modelyze [Broman and Siek, 2012], VHDL-AMS [Christen and Bakalar, 1999] a hardware description language (HDL) with analog and mixed-signal extensions.

A specific yet important class of equation-based languages are the equation-based object-oriented languages, which gained a lot of attention in the last years. The term object-oriented in equation-based object-oriented is not used with exactly the same meaning as for the common object-oriented programming languages. In fact, it is essentially used as opposite to a different modelling paradigm, i.e., the block-oriented one, which is related to imperative paradigms.

To better explain the difference between the equation-based object-oriented and the block-oriented paradigms, a fairly simple example is in order. Consider the simple mechanical system with a mass, a spring and a damper as represented in Figure 1.2, where u is the force applied to the mass M, y is its position, while K and D are respectively the spring constant and the damping factor.

The model is a second order dynamical system

$$M\ddot{y}(t) = u(t) - Ky(t) - D\dot{y}(t)$$

and is obtained by simply applying the Newton's law. If one wants to use an equation-based object-oriented language, it is sufficient for having your

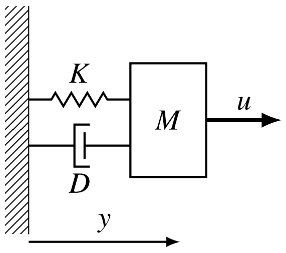


FIGURE 1.2: Mass-spring-damper system.

model to simulate to write this equation with the syntax of the language of choice, and you are done. For example, if Modelica is used, the model can be written as in Listing 1.1.

LISTING 1.1: Mass-spring-damper system in Modelica.

```
model massSpringDamper
  // ...
equation
  der(y) = ydot;
  M*der(ydot) = u -K*y -D*ydot;
end massSpringDamper;
```

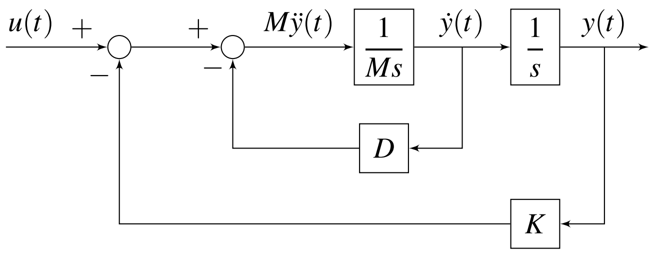
This kind of approach works on "acausal" systems, i.e., there is no direct specification when writing the Newton's law, or any other balance equation of which is the input and which is the output.

On the other hand, if a block-oriented approach is to be taken, then inputs and outputs become important. In the example, we can naturally take the force u applied to the body as the input and the position y as the output. Thus, in this simple, linear example, we can write the transfer function from u to y as

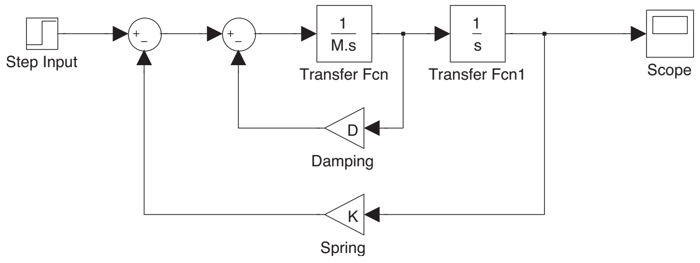
$$G(s) = \frac{Y(s)}{U(s)} = \frac{1}{Ms^2 + Ds + K}$$

that can be represented, for example, by the block diagram in Figure 1.3a, while the Simulink implementation of this block diagram is represented in Figure 1.3b.

Apparently, the Simulink implementation is practically identical to the block-diagram representation, and this is why Simulink is considered to be a Block-oriented language.



(a) Block diagram



(b) Simulink scheme

Figure 1.3: Mass-spring-damper block diagram with Simulink implementation.

Even in this very simple example, this way of modelling is somehow counter-intuitive, at least with respect to equation-based languages, which is in general closer to the way of thinking of a modeller.

Things become more difficult when a (slightly) more complex system is taken. Consider, for example, an electrical circuit with a current source, a capacitor, an inductor and a nonlinear (static) component connected as shown in Figure 2.3; the nonlinear component is ruled by the static constitutive law $v_{\text{nl}} = \imath^2$.

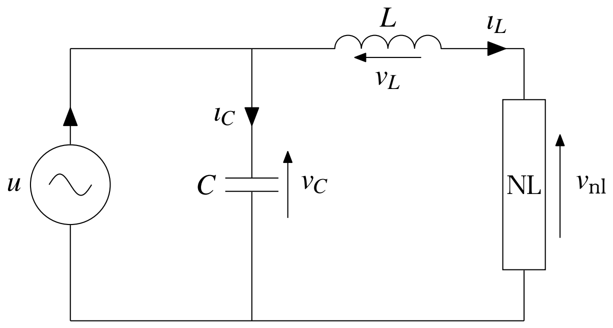


Figure 1.4: Electrical circuit.

The same model can be easily obtained by using the equation-based object-oriented approach [Fritzson, 2003], i.e., each electrical component corresponds to an object made of its constitutive laws, and writing the balancing equations. Starting from those objects, the components can be connected as in an analogous way as in Figure 2.3. To this end, many libraries are available, e.g., the Modelica Standard Library (MSL) contains a package for electrical components. However, one can also want to write its own model, by following the classical modelling approach, thus writing the equation.

The model associated with this circuit can be easily written by applying the Kirchhoff's circuit laws and using the constitutive laws of the components, yielding

$$\begin{aligned}
\imath_C &= C\frac{\mathrm{d}v_C}{dt} \\
v_L &= L\frac{\mathrm{d}\imath_L}{dt} \\
v_{\mathrm{nl}} &= \imath_L^2 \\
u - \imath_L - \imath_C &= 0 \\
v_C - v_L - v_{\mathrm{nl}} &= 0
\end{aligned} \tag{1.1}$$

If an equation-based language is used to model this circuit, it is sufficient to write the set of equations (1.1) with the syntax of the language of choice, and the back-end of the tool will be in charge to manipulate the set of equations to put it in a suitable form for simulation. For example, if Modelica is used, the model can be written as in Listing 1.2.

LISTING 1.2: Electrical circuit in Modelica.

```
model electricalCircuit
  // ...
equation
  // Constituitve laws
  iC  = C*der(vC);
  vL  = L*der(iL);
  vNL = iL^2;

  // Kirchhoff's circuit laws
  u -iL - iC   = 0;
  vC -vL - vNL = 0;
end electricalCircuit;
```

On the other hand, however, if a block-oriented approach is taken, the system must be "causalised", i.e., an input and an output must be chosen, then everything must be turned into a block diagram representation. A Simulink scheme of model (1.1) is shown in Figure 1.5.

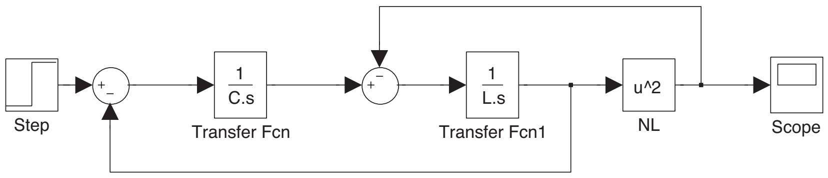


Figure 1.5: Simulink scheme of the electrical circuit.

Once again the block-oriented approach seems not to be the most intuitive for modelling purposes, and definitively different from the original electrical scheme of Figure 2.3. It is quite difficult to see the physical counterpart of the model, and hardly any person except of the modeller itself would understand exactly the model of what is represented in the scheme.

On the other hand, however, it is worth saying that the block-oriented paradigm is the most common when dealing with control systems, where everything is naturally conceived as a block diagram, yet living on a more abstract level than the physical modelling domain.

1.3 The life cycle of a simulation model

It should be evident that in modern engineering, simulation models are not created sparingly to last for only some spot activity, but are important actors along the life of a project. It is therefore legitimate to talk about the life cycle of a dynamic model, as analysing this particular matter provides insight on where and how simplifying techniques are best applicable and most effective for the enhancements of the achievable advantages.

A first important remark is that model complexity is strictly related to its purpose, i.e., to the type of decisions to be taken. Most likely, if one asks an engineer what does the life cycle of a simulation model look like, the answer will be that there is no such thing as a unique model traversing a life cycle, but rather a plurality of models created to answer different questions on the same object.

In the opinion of the author, the importance of modelling throughout a project and the typical view on purpose-specific models just sketched, are apparently dichotomic, or better, evidences a gap between the way modelling is viewed *in abstracto* in the context of a project, and the way models are implemented to fulfil the emerging necessities. More specifically, in the sentence above "modelling" is viewed as the conceptual activity of defining which questions need to be answered by means of simulation, as the project progresses toward maturity. On the other hand, in the same sentence, for "models" we mean the software programs that have to be run to actually obtain the mentioned answers [Sargent et al., 2006].

If we look at the matter from the abstract side of modelling – in the sense above – a life cycle can in fact be outlined, and this is the purpose of this section. It is however worth anticipating that with the technology available to date, creating the models for each stage of that life cycle more or less means starting their development from scratch. It should be thus evident that there is a strong need for some "vertical" modelling framework by which the analyst can maintain a single for the entire project, and automatically obtain from it all the specialised models for the required simulation studies.

Fulfilling this need is a formidable task, extending far beyond the scope of this dissertation. However, as will be explained in the following, model simplification techniques are an enabling factor for solving the problem.

Let us therefore disregard for this section the way individual models are realised, and concentrate on the *modelling* life cycle.

In general, the typical project life cycle for simulation models can be synthesised as presented in Robinson [2001], where the author describes simulation as facilitation. In doing so a life cycle model is proposed based on the work of Lane and Oliva [1998] in system dynamics (see Figure 1.6).

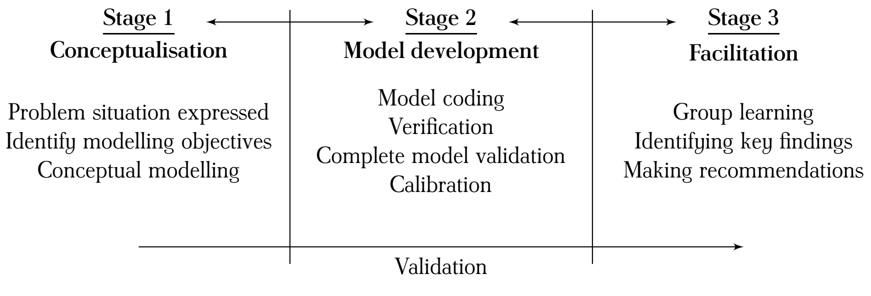


FIGURE 1.6: Life-Cycle Model for Simulation as Facilitation as described in Robinson [2001].

The key processes identified in the modelling and simulation life cycle are: conceptualisation, model development and facilitation. Under each of those, there is a number of sub-processes which require different levels of model abstraction. Iteration between the stages is shown through the double arrows. Validation is identified as a continuous process that is carried out throughout the life cycle, albeit that there is a specific phase where validation of the complete model takes place.

To provide an example of how the exposed concepts can appear in practice, suppose that the addressed project is the design of a controller. The use of simulation models in typical control engineering applications is depicted in Figure 1.7. Iteration among the different stages are possible, and quite common, for example due to requirements or design refinement.

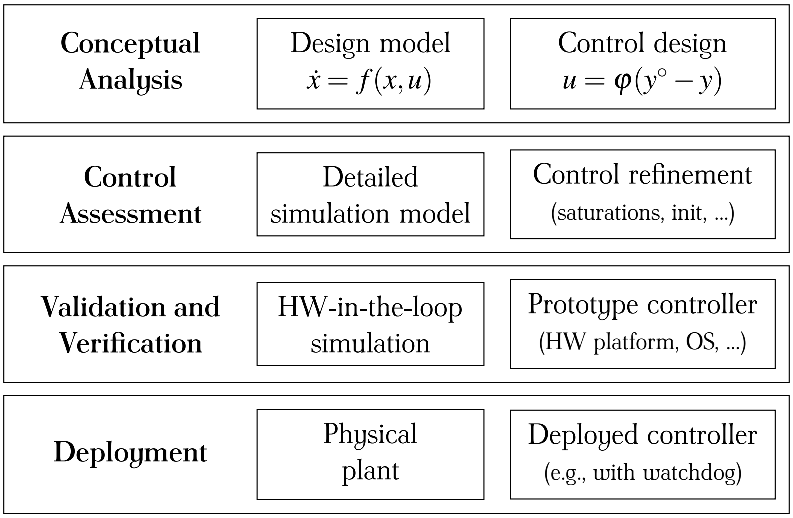

Conceptual Analysis
Design model $\dot{x} = f(x,u)$
Control design $u = \varphi(y^\circ - y)$
Control Assessment
Detailed simulation model
Control refinement (saturations, init, ...)
Validation and Verification
HW-in-the-loop simulation
Prototype controller (HW platform, OS, ...)
Deployment
Physical plant
Deployed controller (e.g., with watchdog)

Figure 1.7: Typical control engineering simulation model life cycle.

The first stage is the *conceptual analysis*, in which quite a simple model is required for control design purposes, i.e., a *design model*, possibly having some specific structure or family of models, and able to capture the main dynamics of the physical process to be control. This is usually obtained by means of black- or grey-box identification techniques on the basis of some experimental data [Ljung, 2001]. Based the obtained model, a controller is thus designed.

When a controller is designed, a more accurate model is thus needed for *control assessment*, so as to test the control system robustness *in silico*, i.e., by means of simulations using different *scenarii*, e.g., set-point following or disturbance rejection. The simulation model must be as accurate as possible, also accounting for most of the unmodelled dynamics of the one used in the design phase. This accurate model is usually a first principle one, and it is obtained on the basis of physical considerations and of conservation laws, possibly with the support of some modelling tools like, e.g., Modelica [Fritzson, 2003]. In addition, in this phase also the controller is complicated by considering saturations, initialization problems, and so on. Due to the complexity of this simulation model, and to the non-criticality in terms of time of this phase, it is acceptable to spend a significant amount of simulation time to test the different *scenarii*. Moreover, the design and assessment phases can be iterated many times until all the project specifications are fulfilled.

The following phase is thus the *validation and verification* one, which not only accounts for all the unmodelled dynamics or behaviours in the design phase, but also some other implementation details. In this case a hardware-in-the-loop simulation must be performed, including all the limitations and introduced by the controller hardware, and computational limits, to test the feasibility of the project before installing the controller on the real plant.

Finally, the designed and validated controller can be deployed on the physical plant, which can be for example endowed with a watchdog monitor—typical in safety-critical applications in order to prevent faults and damages to the plant and to the rest of the environment [Cellier and Kofman, 2006]. In fact, a watchdog monitor of a nuclear power station reasons about the sanity of the plant. It has some knowledge of how the plant is supposed to operate, and looks out for significant discrepancies between expected and observed plant behaviour. To this end, the watchdog monitor maintains a model of the power plant that it runs in parallel with the real plant, comparing its outputs to the measurement data extracted from the real plant. The watchdog monitor thus contains a real-time simulation of a model of the correctly working power plant. Once it discovers a significant aberration in real plant behaviour, it kicks off a fault discriminator program that, again in real time, tries to narrow down the source of the fault, i.e., seeks to determine, which of the subsystems of the real plant is malfunctioning. It maintains real-time simulations of abstractions of models of all subsystems that permit it to localize errors to a particular subsystem. Once this has been accomplished, a fault isolation program is kicked off that invokes a real-time simulation of a more refined model of the faulty subsystem including models of faulty behaviour with the aim of identifying the kind of error that is most likely to have occurred within the faulty subsystem.

Apparently, in this last phase, the simulation model needs fulfilling real-time simulation constraints and to be somehow an intermediate representation between the control design and the validation model. In this case, indeed, the simulation is, as defined in Cellier and Kofman [2006], “a race against time”, and it is crucial both to guarantee the termination of the simulation within a deadline, and to maintain a certain level of accuracy so as to avoid fake fault detections.

The described situation is quite typical in control engineering practice, and it is representative of how the modelling life cycle is far at present poorly connected with the creation of simulation models along a typical project life cycle.

In describing the above three stages of modelling practice, and on the basis of the presented control engineering example, it is apparent that *one* model is unlikely to be appropriate to all the project phases. Moreover, even within a single stage a range of different models may apply. Indeed, the identified stages are not meant to be seen as discrete, but part of a continuum of practices from software engineering to facilitation. As a result, a range of different models should exist, and, further, their relevance to a stage should be explicitly identified.

1.4 The enabling power of simplification

The reader may have noticed that the previous section looks more like the statement of a series of problems than the description of a life cycle. It should however be clear that most of these problems would vanish if one could use a single model for all the activities.

To evidence the paramount role of model simplification in this context, let us now review the model life cycle from a different standpoint, somehow connected to the typical "V-shape" diagram used to described a design, implementation and verification iterative process, summarised in Figure 1.8.

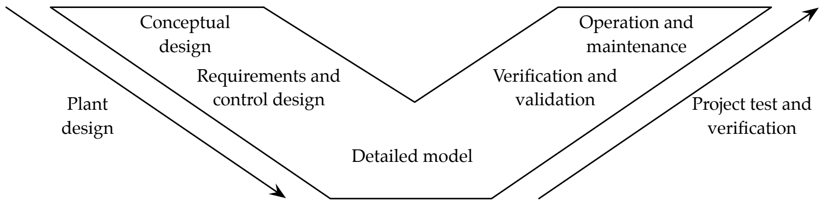

Conceptual design
Operation and maintenance
Requirements and control design
Verification and validation
Plant design
Project test and verification
Detailed model

Figure 1.8: Typical V-shape diagram.

To avoid lengthy abstract treatise, we consider as an example the design of a power plant. Typically, one starts out by outlining the main components of the plant, such as the steam generator, the turbine, and the alternator. For first-cut evaluations on the sizing of those components, very simple models are created, typically composed of less than ten equations per component. subsequently, one (or possibly more than one team in parallel) starts detailing the internals of each component, proceeding by refining steps and using simulation results at each step to provide the additional parameters required by the more detailed model of the following step. At the end of this process, by assembling all the full-detail models obtained so far, the full-detail model of the entire plant is thus available. For this part of the activity, which corresponds to traversing the left side of the "V" in Figure 1.8 downward, support is already provided by object-oriented modelling tools, that allow to preserve component connectivity by means of the abstraction of connectors [Fritzson, 2003]. At this moment, one therefore possesses the full-detail plant model, and also the numerous component models – at various level of detail – that were created along the process. If the object-oriented paradigm was correctly followed, one *should* thus be capable of creating a variety of models that describe some parts of the plant in full-detail, and the others with any of the traversed detail levels, thereby being capable of performing simulation studies concentrating on any specific part of the problem.

There are, however, some subtle issues still open, and this is why in the previous sentence we wrote "should". First, when creating some intermediate level of some component, the analyst has apparently had to make some simplification assumptions on both the internal behaviour of the component, and its role in the overall plant. In principle, there is no guarantee that the subsequent developments of the project did not take a path somehow invalidating those assumptions. The correct behaviour of the full-detail model of the complete plant does guarantee that the outcome of the entire design is consistent, but ensuring that combining component models of different detail level as they were created in the past yields equally consistent results, is a completely different matter. Crudely speaking, a correct use of the object-oriented paradigm can only ensure that the so obtained "mixed-detail" models will compile, but the significance and correctness of the results they produce may be highly questionable.

Second and most important, consider the right part of the "V". When some modifications are introduced at a certain level of detail, apart from the issues above, this also invalidates all the models of lower detail. If some iteration on the V-shape diagram requires the availability of those models, they simply need rewriting from scratch.

To avoid all the problems just mentioned, one should have the possibility of descending the left side of the "V" only once, introduce possible subsequent modifications on the full-detail model, and have any less detailed model that she/he can subsequently need generated by the full-detail one transparently.

Quite apparently, a matter of automatic model simplification.

1.5 Motivation and contributions of the thesis

The motivation for research on automatic model simplification techniques suitable for integration in modelling and simulation tools is just a direct consequence of the enabling role of simplification evidenced in the previous section.

In such a complex and articulated *scenario*, this thesis focuses from the methodological standpoint on simplification techniques aimed at increasing simulation speed, and from the technological standpoint on equation-based object-oriented modelling and simulation tools. Of course thus, it is not the intention of this work to claim any exhaustiveness. Many other simplification problems can be considered, e.g., having the "simplified model" enjoy some properties that the original one does not, and many other paradigm exists other than the object-oriented one. However, specific care is here taken on one side to clearly separate the methodological and the technological aspects of the addressed problems, and on the other side to prove the viability of all the proposed (general) methodological solutions by outlining their implementation in the considered (specific) technological paradigm.

Specifically, the organization of this thesis, and thereby its contributions can be summarised as follows. Chapter 2 reviews some literature on model simplification, providing some general definitions and concepts useful for the development of the presented approximation framework. Chapter 3 focuses on an approximation technique named *dynamic decoupling*, that improves simulation speed by partitioning a model based on the time scales of the contained dynamics. In this respect,

- a novel analysis technique is proposed, termed *cycle analysis*, for the automatic determination and clustering of the mentioned time scales;
- based on said analysis, suitable *separability indices* are introduced to quantify the keenness of a model to be partitioned, in accordance with the contained time scales and the desired degree of approximation;
- a technique consequently presented to automatically exploit the so detected separability in a way that is easily interpreted by the analyst.

Chapter 4 presents and discusses some examples to demonstrate the validity and the practical usefulness of the results described in Chapter 3. Chapter 5, deals with the extension of classical model order reduction techniques in the context of hybrid systems focusing on switched affine ones, illustrating the proposed approach throughout a representative example. Chapter 6 moves from the methodological to the technological side of the overall problem, by showing how the proposed techniques can be integrated in the typical manipulation model toolchain of an equation-based object-oriented modelling and simulation environment. In this respect, a complete solution is presented. Chapter 7 draws some conclusions, evidences some relevant open problems, and outlines future research.

Chapter 2

Related Work and Problem Statement

As stated in the introductory Chapter 1, this dissertation deals with dynamic model approximation, with the main goal of obtaining simplified models, better suited for their intended use.

In this chapter we present some preliminary concepts that will be used in the rest of the thesis, including a brief preliminary literature review to provide the *panorama* of the research context. More detailed references, specific to the problems considered in the rest of the thesis, can be found in Chapter 3 and Chapter 5. For the purpose of this part of the work, a taxonomy of approximation techniques is here given, so as to better clarify the goal of the presented research. Also, some novel definitions of model distance are introduced, posing the basis of the unifying approximation framework that constitutes a very relevant part of the overall proposal.

2.1 Literature review

To contextualise the presented research, some words need spending on efficiency-targeted approximation techniques, with specific emphasis on their applicability and convenience in Equation-based Object-Oriented (EOO) Modelling and Simulation (M&S) tools. We start with a few general remarks on the typical M&S toolchain, thereby also motivating some statements of Chapter 1, and introducing some fundamental concepts and the terminology that will be used in the following. In the rest of the thesis, Modelica is taken as a representative example of EOO languages, but most of the presented concepts are totally general, and can be applied to other modelling environments.

2.1.1 The Modelica compilation and simulation process

Figure 2.1 outlines the typical compilation and simulation process of a Modelica-based software tool. The input (top of the figure) to the process is a Modelica

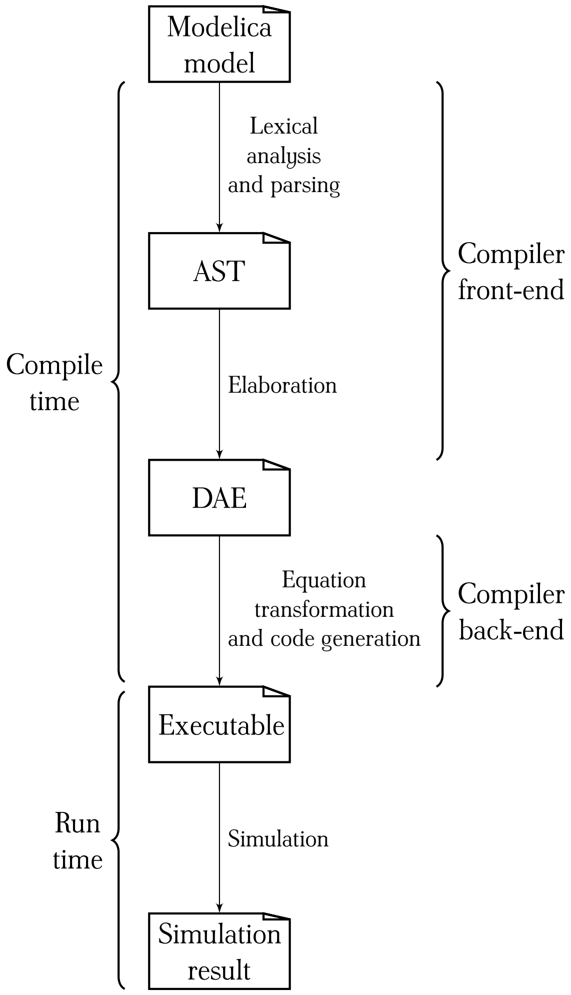


Figure 2.1: Outline of a typical compilation and simulation process for a Modelica language tool.

model, which typically aggregates and references a potentially very large set of other models.

The first phase that is carried out is a standard lexical analysis and parsing, and the output from this step is an Abstract Syntax Tree (AST). Depending on the implementation, this phase can be performed in several stages, where each stage simplifies and normalizes the form of the AST [Fritzson, 2003].

The second phase of the process is the elaboration, that transforms the AST into a Differential Algebraic Equation (DAE) system. A DAE consists of variable declarations, the DAE system (for continuous-time behaviour), algorithm sections, and `when`-clauses for triggering discrete-time behaviour (for hybrid systems). During this phase the model is also checked for errors, such as those related to conformance of types.

The two first phases, lexical analysis and parsing followed by elaboration, are often collectively referred to as the compiler front-end. Many works on this matter can be found in the literature, since the considered part is among the most critical ones for an EOO M&S tool [Broman, 2010], as already specified in the introductory chapter.

The next phase of the compilation process, known as the back-end, consists of first transforming and manipulating the DAE system to make it treatable. Key aspects of this process are the use of the Pantelides algorithm [Pantelides, 1988], the Block Lower Triangular (BLT) transformation [Duff and Reid, 1978], the dummy derivatives [Mattsson and Söderlind, 1993], and tearing [Elmqvist and Otter, 1994]. All these operations are essentially devoted to manipulating the DAE system so as to simplify and reduce its complexity, without altering its "semantics"—this concept will be better clarified in the following. In fact, typically the DAE is reduced to an index 1 problem, and then solved with a DAE solver such as DASSL [Petzold, 1982] or the IDA solver within the SUNDIALS suite [Hindmarsh et al., 2005]. The equation system could also be translated and sorted to form an ODE, to be solved with a numerical integration method, such as Runge-Kutta, Backward Differentiation Formulas (BDF), or any other method for ODE integration. Typically, the right-hand side of the equation system (for an ODE) or the residual function (for an DAE) is translated to executable code, where the typical target language is C. Finally, these generated functions together with a main program are linked together with a numerical solver and then compiled into an executable file as represented in Figure 2.1.

This thesis concentrates on the compiler back-end, in a view to improving simulation efficiency in as transparent as possible a manner for the end user.

2.1.2 The manipulation toolchain: a novel view

In the context of this work, models are natively created in the form of acausal DAE systems. As synthetically anticipated in the previous section, the typical chain of operations of a modelling and simulation environment, that starts from said native model description and ends with the simulation code, can be broadly divided into two parts.

The first part, which we call *acting on the continuous-time equations*, converts the a-causal DAE system into a causal Ordinary Differential Equations (ODE) one. This is done without altering the equations' semantic, by resorting to techniques such as the Tarjan algorithm, alias elimination, index reduction, and so forth Cellier and Kofman [2006]. The same operation can also be done by accepting some semantic alteration – i.e., by altering the continuous-time equations – in exchange for an efficiency improvement. The major techniques for such a purpose are Model Order Reduction (MOR) ones [Antoulas, 2005] and scenario-based approximations [Mikelsons and Brandt, 2009, 2011, Papadopoulos and Prandini, 2014].

The second part, which we call *acting on the discrete-time solution*, consists of taking the ODE model as the basis to generate routines that – once linked to the numeric solver of choice – provide the simulation code. Assuming that acting on the discrete-time solution is done "correctly", i.e., preserving numerical stability, also in this case two ways of operating can be distinguished. The first one does not alter the solution semantic, applying the chosen discretisation method as is. In this case, errors in the solution only come from the inherent imperfection of that method. The second way conversely alters the semantic, by deliberately deviating from the natural application of the discretisation method. Notice that most co-simulation techniques naturally fall into the second class (see, e.g., Arnold and Schiehlen [2009], Bastian et al. [2011], González et al. [2011]).

In Chapter 3 and Chapter 4 we concentrate on the latter type of operation, for which Dynamic Decoupling (DD) is a powerful technique, albeit not fully exploited in a structured (thus possibly automated) manner, see Bartolini et al. [1998]. For the purpose of this section, suffice to say that this technique aims at partitioning a model into submodels, based on time-scale separation. The method is particularly of interest – as will be better detailed in Section 3.2 – because it can be divided into two well separated phases: an analysis part performed on the overall model, and a simulation part that can either be monolithic or make use of co-simulation.

On the other hand, also the former kind of operation mentioned above, is really important from the modeller viewpoint, and to date, to the best of the authors knowledge, no EOO M&S tool allow to automate and integrate in the modelling environment such kind of techniques, e.g., MOR ones. Some words are spent on that in Chapter 6, while Chapter 5 investigate how to extend classical MOR techniques, conceived for continuous-time systems, in the context of hybrid systems [Papadopoulos and Prandini, 2014].

To motivate the choice of focusing on DD in the first part of the thesis, a brief discussion on the major possible alternatives is in order. This discussion is also functional to justifying and supporting the choice of focusing on the extension of classical MOR techniques to hybrid systems, which is the subject of the second part of the thesis.

2.1.3 Alternatives Approaches

As already stated, among the techniques that act on the continuous-time equations, MOR ones are the most adopted, and there exists a vast literature on the matter. MOR is based on the idea of approximating a certain part of the high-dimensional state space of the original model with a lower-dimensional state space, by performing a projection. Roughly speaking, the main differences among MOR techniques come from the way the projection is performed. In any case, most MOR techniques were developed for linear systems Antoulas [2005], and this hampers their application to complex physical cases, where high dimension often appear in conjunction with nonlinearities.

In fact, developing effective MOR strategies for large nonlinear systems is quite a challenging and relatively open problem [Gu, 2011]. Some proposals can be found in the literature, based, e.g., on linearisation or Taylor expansion [Chen et al., 2004], bilinearisation [Phillips, 2000], or functional Volterra series expansion [Innocent et al., 2003], followed by a suitable projection. Other proposals worth mentioning are those based on Proper Orthogonal Decomposition (POD) [Chen and Kang, 2001], to produce approximate truncated balanced realisations for nonlinear systems [Scherpen, 1993], often to find approximate Gramians [Lall et al., 2002], and (for switched systems) generalized [Shaker and Wisniewski, 2012] Gramians. However, when addressing the nonlinear case, the former type of MOR extensions are in practice stuck to quadratic expansions, which strongly limits their applicability. As for the latter type, the cost of evaluating the projected nonlinear operator is often quite high, which reduces computational performance.

Recently, Mikelsons and Brandt [2009, 2011], and Mikelsons et al. [2011] proposed methods specifically dealing with the reduction of EOO models, with the goal of integrating such techniques in the Open Modelica Compiler (OMC) as soon as possible. The main idea behind the quoted works, is that one can define some operation to be performed on the nonlinear system –

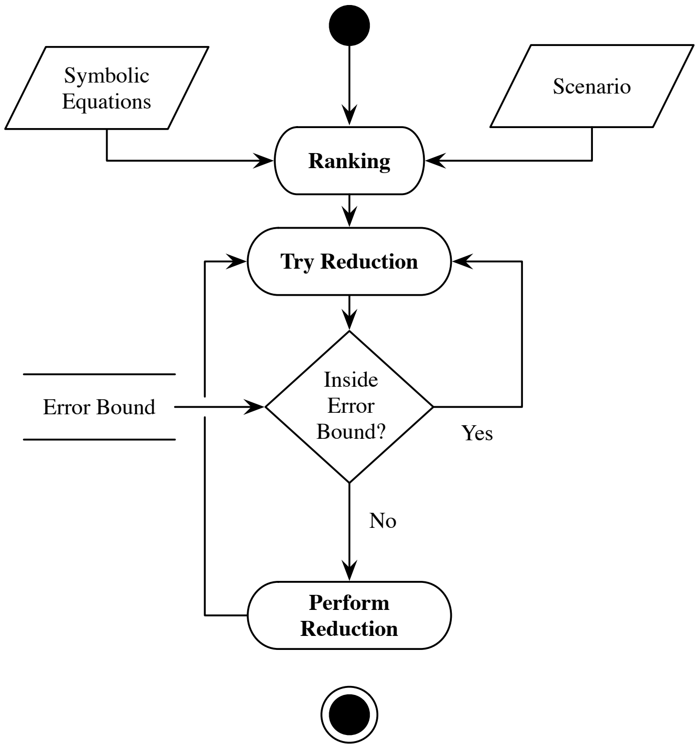


Figure 2.2: Scheme of the reduction algorithm proposed in [Mikelsons and Brandt, 2011].

e.g., “neglect a term”, “linearise a part of the model”, or any other kind of projection that can be performed – and use some ranking metrics to identify *a priori* which is the “best” (single) manipulation that can be performed on the model.

At a first instance, the authors consider only a single manipulation at a time (i.e., the impact on the error of manipulating with a single operation). Based on this information, they rank the manipulations, and consequently try to perform them in sequence accordingly, until the considered error measurement is inside a given bound, as described in Figure 2.2.

It is worth noticing that the error is computed *a posteriori* by comparing the reference solution of the overall system to the numerical solution of the approximated model.

The number of possible manipulations increases in an uncontrollable manner with the dimension of the problem, and with its nonlinearity. The authors thus proposed a more efficient approach, considering the manipulations in clusters, and trying to apply a cluster at a time, until the approximated model fulfils the error bound. Therefore, the last cluster of operations is partitioned in a binary-search-like manner, so as to reach the maximum number of manipulations fulfilling the error bound.

Apparently, the limit of this approach is that ranking all the possible manipulation combinations is not feasible—in fact, the authors try to find out some other heuristics, such as the mentioned clustering techniques, to reduce the combinatorial part of the approach. Moreover, there is no guarantee that performing the manipulations in the ranked order – even in clusters – will eventually lead to the optimal manipulation, since they are considered one at a time.

Another problem is the high cost of generating the reduced order models, due to necessity of computing "snapshots" in the time domain, i.e., simulations of the reduced model to check whether error bound is fulfilled, which in turn requires performing numerous simulations of the original nonlinear system. Furthermore, this approach is scenario-based, i.e., the simplified model is guaranteed to be good – and the error within the error bound – only for a set of initial conditions, a set of inputs and a time span. If the *scenario* is changed, the overall manipulation must be performed again, limiting again the applicability of the method. Moreover, the considered *scenarii* is generally not even stochastic, accounting for possible disturbances or randomness, thus not evaluating the robustness of the approximation.

The quite old idea of DD has thus been recently reconsidered, for example by the Transmission Line Modelling (TLM) approach of Sjölund [2012], Sjölund et al. [2010]. TLM is based on modelling the propagation of a signal which is limited by the time it takes to travel across a medium. By utilizing this information it is possible to partition the DAE system into independent blocks that may be simulated in parallel. This leads to improved simulation efficiency since it enables full performance of multi-core CPUs. This, however requires that the analyst explicitly introduces the transmission model, i.e., the decoupling part, by introducing some additional components, based on his/her intuition.

The approach proposed in this dissertation conversely aims at having decoupling emerge from an automated analysis of the model.

2.1.4 A Brief Comparison

Based on the previous discussion, we now spend some additional words on the advantages of the technique proposed in this work, and sketched out in the introduction, with respect to the analysed alternatives.

In comparison with MOR, our proposal does not alter the state vector, nor does it involve base changes in the state space, thereby preserving the physical meaning of dynamic variables. Also, instead of attempting to simplify the model in a view to monolithic solution, we go exactly in the opposite direction, as the model is not reduced but *partitioned*, allowing for parallel simulation, with the same *rationale* of Sjölund et al. [2010].

Of course, our proposal is not the only way to partition a system. As an alternative, for example, one may neglect or approximate in some way the subspace spanned by the eigenvectors associated with its fast eigenvalues. However, this is possible only in the linear case, while extensions to nonlinear models require local linearisation. This does preserve the dimension of the state space, but to recover the native dynamic variables of the model, a coordinate transformation is necessary at each integration step, to the apparent detriment of simulation efficiency.

No matter how the partition is obtained, then, it can be exploited in two ways. One is to ease a monolithic solution, in some sense adapting the model to the used architecture (single solver with a unique integration step). The other is to conversely tailor the solution architecture to the model *as analysed and partitioned by the method*; this can be used to fruitfully employ parallel simulation, or co-simulation. If the latter route is taken, eigenvalue-based partitioning reveals however another problem, as the properties of a so obtained partition may change in time, while decoupled integration, let alone co-simulation, require the same partition to be specified *a priori*.

As a consequence, for the specific purpose of this work, state selection criteria are preferable to eigenvalue-based ones, also in accordance with Schiela and Olsson [2000], and in this context, the proposed method exhibits the further advantage of being naturally keen to a nonlinear context.

With respect to scenario-based approximations, the most computing-intensive part of the proposal (as will be explained later on) is simply not scenario-based: information related to the considered *scenarii* come into play only at a later stage, and this separation results in lightening the computing effort. Furthermore, the proposal does not alter the model equations, thus being less exposed to the possible unpredictable effects of local modifications at the overall system level.

Finally, contrary to the TLM approach, this work aims at having decoupling emerge from an automated analysis of the model, and not introduced by the analyst, still having the advantage of exploiting full multi-core CPUs performances, by parallel simulation.

2.1.5 "Simpler models" from the modeller viewpoint

One of the heaviest tasks for the analyst, is to face model complexity, performing – as anticipated – the necessary simplifications to obtain the descriptions of one object for all the envisaged uses of its model, both by means of heterogeneous tools, e.g., for MOR techniques, and by hand, explicitly introducing approximations in the model, as in the case of TLM. The main issue, however, is that said uses may change during the life-cycle of the simulation model, and within different stages of the project, and some simplifications may not hold true anymore for new *scenarii*, thus requiring to start from scratch with the model simplification. Therefore, a great improvement from the analyst's viewpoint would be the availability of an automatic tool which produces the required simplifications accordingly to the *scenario* of interest, or equivalently, that produces approximations that are totally independent of the scenario at hand.

Focusing for a moment on DD, it can be immediately observed that it is a technique of the second type, i.e., it is based on structural properties of the considered system, instead of basing the approximation on a given scenario. In addition, such a kind of operation could bring some benefits also for linear system, where MOR techniques are usually considered the only way to deal with complexity.

For example, let's consider the electrical circuit in Figure 2.3.

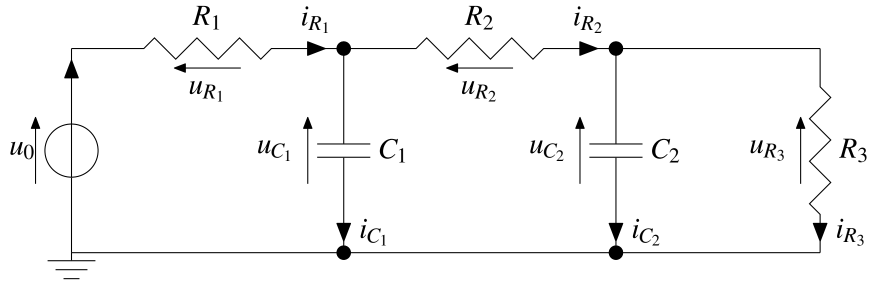


FIGURE 2.3: Electrical circuit.

For a given set of parameters, e.g., $C_1 = kC$, and $C_2 = C$, with $k \gg 1$, the system can be quite easily approximated with a first-order one with any MOR technique. On the other hand, however, the physical interpretation of the considered dynamical variables is lost – MOR techniques only preserve the input-output relationship – while in some relevant cases, especially for control purposes, it may be really important. The considered circuit can be considered composed of two subsystems, decoupled by C_1, and can be simulated with a parallel configuration. Depending on the purpose of the model, one or the other choice can be taken.

In synthesis, what is generally sought in this context is a "simpler model", whatever is meant for that, and however it is obtained. Apparently, hardly any clear definition of "simpler model" has been given in the literature, and not even a taxonomy of what is meant for that is present. Nonetheless, very often "simpler" is – *a priori* – connected with "reduced order", thus falling in the class of MOR techniques, and as a consequence leaving out many other possibilities. In the opinion of the author, this connection is not general. Indeed, the concept of "simpler" must be associated with the intended use of the model, i.e., we can say that we seek a model "better suited to its intended use".

A taxonomy of possible intended uses needs thus defining. In general, when dealing with EOO M&S tools of the Object-Oriented Modelling (OOM) type, a quite detailed description of the complete system is somehow desired. In other words, the manipulations that an analyst would perform on the model strictly depend on its purpose, and aim at obtaining purpose-oriented results without approximating *too much* its semantics. As such, some classes of purposes can be identified as follows.

- Structural purposes:
  - the manipulated model has some structural properties of interest, which the original model has not;
  - the dimension of the manipulated model should be as low as possible;
  - some variables are not of interest for simulation.
- Computational purposes:
  - the manipulated model should be simulated as fast as possible;
  - the manipulated model should consume as few memory as possible;
  - it is acceptable that the solution does not show dynamics with a time scale that is smaller than a given threshold.

In particular, starting from the original model M there are different axes in which simple model $\widehat{M}$ can be sought:

- The simplified model $\widehat{M}$ only deals with a part of the system M, and must represent this part with a high detail; the other parts of M can be roughly simplified, but have to reproduce reliable boundary conditions to $\widehat{M}$.

- The simplified model $\widehat{M}$ has to enjoy a set of properties $\mathscr{P}$, which M has not.

The set of properties $\mathscr{P}$ is in general heterogeneous, due to the different desires related to the *scenario* of interest. As mentioned before, one may seek *structural* properties concerning the model itself: e.g., the model should be linear, or it should belong to the class of Linear Parameter Varying (LPV) models, or Linear Fractional Transformation (LFT) and so forth. On the other hand, one desires the model to enjoy some *computational* properties: e.g., the model should be simulated as fast as possible, minimise the memory for needed its simulation, and so on. Besides the mentioned classes of properties, other classes may be found, but for the purpose of this work the sketched *panorama* is sufficient.

Just to give a first possible "recipe book" that associates each problem with some viable solutions, Table 2.1 summarises which are some of the techniques that can be adopted for a specific purpose.

It is worth noticing that the mentioned purposes are not mutually independent. For example, order reduction *usually* yields models that simulate faster, while DD – which is basically used to speed up simulation – has also the side effect of partitioning the model in reduced order ones, decreasing also the memory needed for the simulation.

Apparently, there is a wide variety of techniques to achieve the same goal, but in very few cases the modeller can use those methods in an affordable way, without being an expert of the matter. The long-term goal of the research to which this thesis belongs is to "make the modeller's life easy", in other words, to find a way to automatise all the mentioned techniques in a unique manipulation framework, in as transparent as possible a way for the end user of EOO M&S tools.

2.2 Model simplification from a general viewpoint

To make a manipulation system capable of simplifying, in the sense suggested so far, the following is needed:

1. a definition of *quasi-equivalence* or model distance,

2. a taxonomy of interventions on the *model* and on its *solution*,

Objective		Technique
Structural properties	Given structure	Automatic generation of LFT [Casella et al., 2009, Varga et al., 1998].
	Low order	Classical MOR [Antoulas, 2005], or reduction scenario-based techniques [Mikelsons and Brandt, 2009].
	Variable selection	Reduction scenario-based techniques [Mikelsons and Brandt, 2009], or DD
Computational properties	Fast	MOR, reduction scenario-based techniques, TLM [Sjölund et al., 2010], or DD
	Few Memory	MOR, reduction scenario-based techniques, Mixed-mode integration [Schiela and Olsson, 2000], or DD
	Time scale selection	Mixed-mode integration, or DD

Table 2.1: Summary of the techniques that can be adopted for a given intended use.

3. the formalisation of said interventions in terms of manipulation functionalities,

4. and finally the design of an extended toolchain capable of hosting the so devised functionalities.

This section provides some model equivalence definitions, and sketches out how they can possibly be used to intervene on the model. In the next chapters, the proposed techniques will be analysed in detail providing significant examples, and Chapter 6 defines a prospective way of modifying the standard toolchain in a view to integrate the proposed techniques in the compiler back-end.

2.2.1 Terminology and preliminary definitions

In order to better clarify the framework we are developing, some preliminary definitions are needed. This is also needed because in the literature it is quite difficult to find a unique and general way to define the meaning of equivalence between two models. This is definitely of interest since many of the manipulation techniques present in the literature, e.g., MOR and also DD, can be viewed as "elementary operations" on the model in this framework.

The general case of nonlinear systems of DAEs of the form

$$M : \begin{cases} \mathbf{F}(t, y, \dot{y}, u, p) = 0 \\ \mathbf{F} : \mathbb{I} \times \mathbb{D}_y \times \mathbb{D}_{\dot{y}} \times \mathbb{D}_u \times \mathbb{D}_p \mapsto \mathbb{R}^m \end{cases}$$

is considered here, where $\mathbb{I} \subseteq \mathbb{R}$ is a (compact) interval and $\mathbb{D}_y$, $\mathbb{D}_{\dot{y}} \subseteq \mathbb{R}^n$ are open, m, $n \in \mathbb{N}$.

A first widely used but seldom defined concept is that of *scenario*. A discussion on the matter can be found in Mikelsons and Brandt [2011], and based on the ideas reported therein, a formal definition can be here given as follows.

Definition 2.2.1 (Scenario). A *scenario* is a set of a vector field defined on the interval $\mathbb{I}$ for the system inputs, the initial values and the parameters.

It is thus possible to define the concept of "semantic equivalence" between a model M_1 and M_2, denoted by the symbol $\equiv$

Definition 2.2.2 (Semantic equivalence). $M_1 \equiv M_2$ iff solved with *the same* integration method $\mathscr{N}(\cdot)$ with the same configuration (e.g., the same tolerances) produce solutions $\mathscr{N}(M_1) = \mathscr{N}(M_2)$, i.e., acceptable within the tolerances. This is true for a set of prescribed *scenarii*.

The last definition is quite restrictive, and holds only for models that – crudely speaking for brevity – produce almost the same numerical results. From the definition directly follows a theorem, the proof of which is trivial and omitted here for brevity.

Theorem 2.2.1. *If M_1 and M_2 have the same equations, then $M_1 \equiv M_2$.*

Such definitions, however, become of hardly any use as soon as approximation is brought into play, thus definitions for "approximated equivalence" must be provided. Definitions 2.2.3-2.2.5 provide different level of approximations, from the most to the least conservative.

Definition 2.2.3 (Quasi-equivalence). $M_1 \cong M_2$ iff they have different sets of variables and of equations, but $m_1 \subseteq M_1$ and $m_2 \subseteq M_2$ have the same set of variables and $m_1 \equiv m_2$.

Definition 2.2.4 (ε-equivalence). $M_1 \equiv_\varepsilon M_2$ iff they have the same set of variables, different sets of equations, $M_1 \not\equiv M_2$ with the default tolerances, but $M_1 \equiv M_2$ with tolerances of ε.

Definition 2.2.5 (ε-quasi-equivalence). $M_1 \cong_\varepsilon M_2$ iff they have different sets of variables and of equations, but $m_1 \subseteq M_1$ and $m_2 \subseteq M_2$ have the same set of variables and $m_1 \equiv_\varepsilon m_2$.

Quasi-equivalence and ε-quasi-equivalence are particularly useful when dealing with MOR and reduction scenario-based techniques, since they generally produce different state variables and different equations, but are conceived to produce the similar input-output relationship (possibly in a prescribed bandwidth) with respect to the original model.

On the other side, ε-equivalence can be used for evaluating the approximation performance of DD, given that it do not alters the state variables, but rather the equations, i.e., how the system is solved. In particular, in the following, the term "accuracy" related to the DD numerical solution will be exactly the ε of Definition 2.2.4, indicating the maximum value of the tolerances (absolute and relative) needed to make the approximation acceptable.

Notice that the proposed definitions can be applied also in the case of hybrid systems, since they are based on the numerical solution instead of the mathematical structure of the dynamical system. In Chapter 5, however, a different distance measurement is used, since it is better suited for reachability and verification, typical analysis in the context of stochastic hybrid systems Abate and Prandini [2011], Amin et al. [2006], Girard et al. [2008].

The provided definitions may be difficult to be computed, but are general and totally unrelated to the manipulation technique. It is also worth stressing that all the proposed definitions are related to the integration method. In the authors' opinion, this is necessary because even if two models are analytically *near* – whatever is meant for that – when simulated may produce very different results due to integration approximations. Thus, the definition of the semantic equivalence relation – and of all the derived definitions – must take this aspect into account.

Chapter 3

Dynamic Decoupling

Most of the results presented in this chapter come from Papadopoulos et al. [2013], and Papadopoulos and Leva [2013a,b,c,d], Papadopoulos et al. [2014].

3.1 Introduction

This chapter is aimed at investigating how to manage model *complexity*, by devising model analysis, manipulation, simplification and solution techniques that can be made part of modern modelling and simulation environments, in a view to achieve efficient integration as transparently as possible for the user.

Complexity – in the sense considered in the entirety of this research – can have different sources, the major ones being model dimension, nonlinearities, necessity of different modelling paradigms (e.g., equation- or algorithm-based), and presence of different time scales (i.e., stiffness).

In the literature, those kinds of complexity are addressed with different approaches, as discussed in Chapter 2. Large-scale systems are typically handled by means of MOR techniques [Antoulas, 2005]. These are however essentially limited to the linear case, while nonlinear extensions are basically heuristic, domain specific, or scenario-based [Mikelsons and Brandt, 2011]. As for multi-paradigm models, advanced tools – typically object-oriented modelling languages – are inherently conceived to handle them, allowing for example to combine equation and algorithm models [Cellier and Kofman, 2006]; also, co-simulation environments are available to cooperatively employ specialised simulation tools [Arnold and Schiehlen, 2009, González et al., 2011]. Finally, the integration of systems with different time scales can be made more efficient by means of approximation techniques, such as the so called DD [Bartolini et al., 1998].

Whatever the source of complexity is, here we take as the main goal of model simplification that of improving computational performance while respecting convenient precision/accuracy constraints for the specific simulation study at hand. In this respect, it is worth noticing that modern tools already allow to apply some simplification techniques in quite an easy way. For example, environments like Matlab provide many well-established functions for linear MOR, e.g., `balred`. However, to the best of the authors' knowledge, for virtually all the techniques mentioned in Chapter 2 only problem specific solutions are available, and their full integration in M&S environments is still an open problem.

This chapter deals with the exploitation of the aforementioned DD. The proposed methodology is grounded on an analysis technique, described in the following, which is somehow analogous to eigenvalue analysis but applicable also to nonlinear systems.

Said technique, called Cycle Analysis (CA), is the first contribution of this work. A second contribution, building on the CA idea, is the proposal of some indices to quantify the "separability" of a model into submodels, based on DD. By jointly exploiting said contributions, the following main advances are obtained beyond the state of the art:

1. if a monolithic solution (i.e., no co-simulation) is required, CA provides evidence of possible internal weak couplings among dynamic variables, which can be exploited to ease the numerical integration; in the presence of a parallel computing architecture, this is apparently useful also for selecting the simulation threads;

2. if one (further) wants to apply integration schemes tailored to decoupled systems, these can be applied and configured on an objective basis, according to structural properties of the model at hand;

3. if a co-simulation setting is considered and some degrees of freedom are available as for the model partitioning, these can be exploited automatically;

4. whatever solution setting is adopted, the proposed indices allow to take any decision concerning its configuration based on quantities that are easily interpreted by the analyst.

Reference is here made to EOO M&S tools because they are particularly keen to be complemented with the proposed functionalities, but the proposed ideas are completely general, and applicable also in different contexts.

More specifically, on the basis of the discussion of Section 2.1.4 and on the statements above, the contributions of this chapter can now be better qualified as follows.

- CA quantitatively characterises the dynamics of the addressed system, including the numerical integration algorithm, without resorting to eigenvalue-based techniques, therefore applying to both the linear and the nonlinear case.

- Some "separability indices" are defined, whose information content extends beyond that of previously introduced quantities, like stiffness coefficients. The proposed indices thus complement traditional "stiffness" measures in basically two senses:

  1. they are not tied to the sole idea of "fast" and "slow" dynamics, and
  2. they apply also to nonlinear systems.

- The two ideas above are suitably joined to demonstrate, with a proof-of-concept application and some examples, that they can be used to achieve an automatic application of DD, i.e., to build a tool that partitions a model requiring the analyst to provide only information that pertains to the physics of the simulated object.

The rest of the chapter is organised as follows. In Section 3.2, the concept of DD is reviewed under a novel viewpoint, while Section 3.3 describes the proposed procedure for structural analysis, i.e., the Cycle Analysis. Based on that method, Section 3.4 describes some new synthetic indices to characterise and quantify structural properties of the system, e.g., how much stiff or "separable" a system is, relating those quantities (when possible) to quantities already present in the literature. Section 3.5 describes how to exploit the results coming from the Cycle Analysis in a mixed-mode integration scheme. Some application-oriented remarks and more general discussion on the proposed method are reported in Section 3.6.

3.2 Dynamic decoupling

Multi-physics models are often made of parts evolving within different time scales, and the core idea of DD is to exploit this partition to enhance simulation efficiency.

In some cases, figuring out how to partition a model can be quite straightforward, but this is not general at all. For example, in mechatronic systems, a "slow" mechanical part is often driven by "fast" electric circuits. However, even if this is the case, characterising the found time scales quantitatively – e.g., to determine whether or not it is really convenient to partition the model, and how to do it – may not be equally simple, since the actual evidence of multiple time scales may not only come from the presence of multiple physical domains, but also strongly depend on parameter values. Furthermore, there are cases in which multiple time scales are not originated by multiple physical contexts, but emerge from some structural characteristics of the model that are virtually impossible for the analyst to detect *a priori*, especially for large models.

As a result, DD is formally based on some characteristics of the mutual relationships among the model state variables, that are formulated in an abstracted manner with respect to the underlying physical domain(s). For a short explanation of the DD *rationale*, consider the generic state equation of a continuous-time ODE model, and write it as

$$\phi_i(\mathbf{x})\frac{\mathrm{d}x_i(t)}{dt} = \gamma_i(\mathbf{x},\mathbf{u}) \tag{3.1}$$

where function ϕ_i plays the role of a time-varying "capacitance" associated with the state variable x_i, while function γ_i conveys the contributions of all states (including x_i) and inputs (variables $\mathbf{u}$) to its variation. Given this, DD can be synthetically expressed as the following two principles.

1. If, in a certain region of the state and input space, some γ_i/ϕ_i ratio is "small", then in the discrete-time solution it can be acceptable to use the value of x_i computed at the previous integration step, given its "slow" variation;

2. If, in a certain region of the state and input space, the contribution of a certain x_j to γ_i is "small", then in the discrete-time solution it can be acceptable to use the value of x_j at the previous integration step, given the "small" error introduced in the computation of the new x_i.

The two principles above take different forms in various contexts (see, e.g., Bartolini et al. [1998] for a thermo-hydraulic application) but are *per se* general. From an operational viewpoint, DD can be thought of as composed of two subsequent phases, termed here *structural analysis* and *decoupled integration*. The former is an *offline* activity, and consists of identifying in the model possible occurrences of the two principles above. The latter consists of exploiting the analysis outcome to select and suitably configure an integration scheme so as to improve simulation efficiency.

Both phases can be carried out with multiple techniques. For the structural analysis phase, we propose here a novel method, called Cycle Analysis (CA), described in the following, that is particularly suited to investigate mutual relationships among dynamic variables independently of the structure of the individual state equations, and therefore carries most of the merit for the applicability of the entire technique to the nonlinear case. For the decoupled integration phase, we conversely resort to mixed-mode integration similar to the one proposed in Schiela and Olsson [2000], but any co-simulation framework can be used, e.g., the one proposed in González et al. [2011].

A very important point to keep in mind is that pursuing an automatic application of DD is a twofold problem. On one side, the analysis phase needs to be performed by an automatic procedure rather than manually. On the other side, the outcome of said phase must take a form that is readable for the analyst, who is typically an expert of the addressed physical domain, not of simulation. Such an output is carried out by means of a set of *separability indices*. The following sections thus deal, in this order, with CA, with the correspondingly obtained separability indices, and with the use of both for decoupled integration.

3.3 Cycle analysis

3.3.1 Preliminaries and definitions

In the last paragraphs DAE systems are considered. This is actually the scope of this work, but in the rest of the chapter we are using ODE systems. This is not limiting the applicability of the proposed method, since DAE systems are usually manipulated both symbolically and numerically so as to obtain an ODE system, that will be simulated. This operation is typical in EOO tools, and how to plug the proposed methodology into the toolchain of manipulations will be discussed in more details in Chapter 6.

Consider the generic ODE system

$$\dot{\mathbf{x}}_{CT}(t) = \mathbf{f}(\mathbf{x}_{CT}(t), \mathbf{u}_{CT}(t)) \tag{3.2}$$

where $\mathbf{x} \in \mathbb{R}^{n_{CT}}$ is the vector of state (i.e., dynamic) variables, and $\mathbf{u} \in \mathbb{R}^{m_{CT}}$ the vector of input variables. Generally speaking, the idea of CA is to obtain from the discretisation of (3.2) a directed graph representing the mutual influence among the dynamic variables along the integration steps, and then to compute quantities that generalise – in a sense that will be explained later – the idea of "time constants" for the linear case.

To this end, discretise (3.2) with an *explicit* method with fixed single-step h[1]. It is important to notice right from now that the method used in this

[1]It is known that any multi-step method can be reduced to a single-step method with an increased state space vector. Thus, in this dissertation we only focus only on the case of single-step methods without loss of generality.

phase is a "probe method", i.e., just functional to the analysis technique, while the successive simulation phase is in no sense tied to it. The corresponding discrete-time system can be thus written as

$$\mathbf{x}_{k+1} = \mathbf{F}_{\mathscr{N}}(\mathbf{x}_k, \mathbf{u}_k, h) \tag{3.3}$$

where $\mathbf{x}_k^T \in \mathbb{R}^n$, with $n = n_{CT}$ is the discrete-time state, while the form of function $\mathbf{F}_{\mathscr{N}}(\cdot,\cdot,\cdot)$ depends on the particular numerical integration method $\mathscr{N}$.

The required dependency directed graph (or digraph) G is formally defined as

$$G = (N, E), \qquad N = \{1, \dots, n\}, \quad E = \{e^{i,j}\} \subseteq N \times N. \tag{3.4}$$

The nodes of G are associated with the discrete-time model dynamic variables, while its edges are characterised by a source node, a destination node, and a weight, defined by the operators

$$\varsigma\left[e^{i,j}\right] := i, \quad \delta\left[e^{i,j}\right] := j, \quad \rho\left[e^{i,j}\right] := \frac{\partial F_i}{\partial x_j}. \tag{3.5}$$

Notice that the construction of G is straightforward based on the structure of system (3.3).

Definition 3.3.1. A *path p* of length L in a digraph $G = (N, E)$ is an ordered sequence of L edges, where the destination node of each edge is the source node of the following one in the sequence. Formally,

$$\begin{aligned} &p := \langle e_1, e_2, \dots, e_L \rangle, \quad \text{with } e_i \in E, \quad \forall i \in \{1, \dots, L\}, \\ &\text{with } \delta[e_i] = \varsigma[e_{i+1}], \quad \forall i \in \{1, \dots, L-1\}. \end{aligned}$$

A path can be also denoted by means of the ordered sequence of touched nodes, i.e.

$$p = \langle \varsigma[e_1], \varsigma[e_2], \dots, \varsigma[e_L], \delta[e_L] \rangle.$$

Definition 3.3.2. A path with no repeated nodes is called a *simple path* (or *walk*).

Definition 3.3.3. A *simple cycle c* of length L exists in a digraph $G = (N, E)$ iff

1. there exists a simple path $\langle e_1, e_2, \dots, e_{L-1} \rangle$,

2. there exists one edge e_L from $\delta[e_{L-1}]$ to $\varsigma[e_1]$.

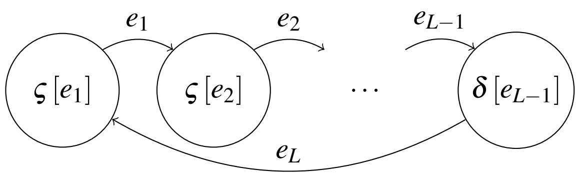


FIGURE 3.1: Graphical representation of a simple cycle.

For the sake of clarity, a simple cycle can be graphically represented as shown in Figure 3.1.

Adopting the same notation used for paths, a simple cycle can be denoted as

$$c = \langle \varsigma[e_1], \varsigma[e_2], \ldots, \varsigma[e_{L-1}], \delta[e_{L-1}], \varsigma[e_1] \rangle,$$

i.e., by listing the ordered sequence of the touched nodes.

Notice that the definition of a simple cycle in terms of edges is unique up to a circular permutation, while the definition in terms of touched nodes varies according to which of them is (conventionally) taken as the "first" one in the cycle. This is why we prefer to use the definition in terms of edges.

In the following we shall make reference only to simple cycles, thus "cycle" and "simple cycle" will be used interchangeably.

Definition 3.3.4. The *cycle gain* $\mu_c(h)$ of a cycle c is defined as

$$\mu_c(h) = \prod_{e_i \in c} \rho[e_i]. \tag{3.6}$$

3.3.1.1 An explanatory example

Let us consider the continuous-time linear time-invariant dynamic system

$$\dot{\mathbf{x}}_{CT} = A\mathbf{x}_{CT} = \begin{bmatrix} -1 & 0.5 & 0 \\ 0.5 & -1.5 & 0.5 \\ 0 & 0.5 & -1 \end{bmatrix} \mathbf{x}_{CT},$$

Suppose that the discretisation of choice for the analysis part is the Heun's algorithm Cellier and Kofman [2006]. Thus, the corresponding discrete-time system (3.3) becomes

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{F}_{\text{Heun}}(\mathbf{x}_k, h), \\ \mathbf{F}_{\text{Heun}}(\mathbf{x}_k, h) &= \left(I_{3\times 3} + Ah + \frac{(Ah)^2}{2} \right) \mathbf{x}_k, \end{aligned} \tag{3.7}$$

where $I_{3\times 3}$ is a 3×3 identity matrix, and $\mathbf{x} = \mathbf{x}_{CT}$. Therefore, the dependency graph G associated to the system has a weight matrix

$$\begin{aligned} W =& \frac{\partial \mathbf{F}_{\text{Heun}}}{\partial \mathbf{x}} = I_{3\times 3} + Ah + \frac{(Ah)^2}{2} = \\ =& I_{3\times 3} + \frac{h}{8}\begin{bmatrix} 5h-8 & 4-5h & h \\ 4-5h & 11h-12 & 4-5h \\ h & 4-5h & 5h-8 \end{bmatrix}, \end{aligned} \tag{3.8}$$

yielding a completely connected graph, represented in Figure 3.2.

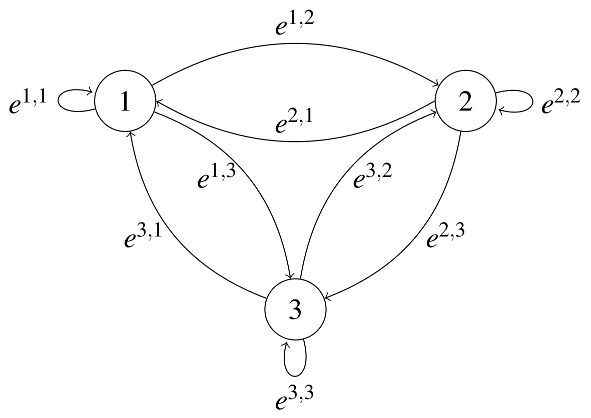


Figure 3.2: Dependency graph associated with the discretised system (3.7).

In this case, the set of simple cycles $\mathscr{C}$ present in the graph G, and the corresponding cycle gains are

$$
\begin{aligned}
c_1 &= \langle e^{1,1} \rangle & \mu_{c_1}(h) &= \frac{5h^2}{8} - h + 1, \\
c_2 &= \langle e^{2,2} \rangle & \mu_{c_2}(h) &= \frac{h}{8}(11h - 12), \\
c_3 &= \langle e^{3,3} \rangle & \mu_{c_3}(h) &= \frac{h}{8}(5h - 8), \\
c_4 &= \langle e^{1,2}, e^{2,1} \rangle & \mu_{c_4}(h) &= \frac{h^2}{64}(4 - 5h)^2, \\
c_5 &= \langle e^{1,3}, e^{3,1} \rangle & \mu_{c_5}(h) &= \frac{h^4}{64}, \\
c_6 &= \langle e^{2,3}, e^{3,2} \rangle & \mu_{c_6}(h) &= \frac{h^2}{64}(4 - 5h)^2, \\
c_7 &= \langle e^{1,2}, e^{2,3}, e^{3,1} \rangle & \mu_{c_7}(h) &= \frac{h^4}{512}(4 - 5h)^2, \\
c_8 &= \langle e^{1,3}, e^{3,2}, e^{3,3} \rangle & \mu_{c_8}(h) &= \frac{h^4}{512}(4 - 5h)^2 .
\end{aligned}
$$

Notice that if the matrix W is symmetric, it is sufficient to consider only its lower triangular part (including the diagonal).

3.3.2 The analysis technique

As anticipated, the ultimate goal of the analysis is to (automatically) recognise the presence in the model of different time scales, and cluster the dynamic variables accordingly. The underlying *rationale* of the approach is based on a convenient interpretation of the cycle gains of Definition 3.3.4.

To provide this interpretation, let us consider system (3.3) at an asymptotic stable equilibrium, i.e., $\mathbf{x}_{k+1} = \mathbf{x}_k$. Suppose to apply a small impulsive perturbation to one state variable x_i. A transient will then occur, and two things may happen:

- the perturbation affects the other state variables, without re-affecting x_i, i.e., in the associated model digraph G, there is no cycle involving node i;
- the perturbation, after some integration steps, re-affects x_i, i.e., there exists at least one cycle involving node i.

In the first case, no numerical instability can be introduced by the integration method. This is conversely possible in the second case, and occurs if the perturbation undergoes a sufficient amplification along at least one of the involved cycles. Since that amplification is quantified by the corresponding cycle gain, we can conjecture that the perturbation vanishes if all the gains of the involved cycles are in magnitude less than a certain $\underline{\mu}$, while instability arises if at least one of said gains is larger in magnitude than a certain $\overline{\mu} > \underline{\mu}$.

It is now worth recalling that, considering an ODE system at a certain stable operating point, in the vicinity of said point (i.e., near enough to it for the linearisation of the original system to be sufficiently precise) there exists one value of h that constitutes the boundary between a stable and an unstable behaviour of the discrete time solution.

It is also well known that with explicit methods, instability originates from model dynamics that have too fast a time scale with respect to the employed integration step. Since the cycle gains depend on h, if an unstable behaviour is observed, it is legitimate to state that the dynamic variables involved in the cycles that provide the excessive amplification are evolving with a time scale that is "fast" with respect to h.

Based on the discussion above, we can now describe the analysis procedure as follows.

1. Select an explicit fixed-step integration method. It is worth stressing that this method is only functional to the analysis, and in no sense constrains the choice of the method(s) used for the subsequent decoupled integration.

2. Discretise the system.

3. Construct the digraph.

4. Perform a topological analysis to find the set $\mathscr{C}$ of all the (simple) cycles. The potential complexity of the cycle search operation will be discussed later on.

5. Express the cycle gains as per (3.6).

6. Construct a set of inequalities in the form

$$|\mu_c(h)| \leq \alpha, \qquad \forall c \in \mathscr{C},$$

   where α is the single real parameter of the analysis, to be discussed in the following.

7. Solve each inequality individually for h, thereby associating with each cycle a value for the integration step that produces low enough a magnitude of the corresponding gain.

8. Associate with each dynamic variable x_i the lowest h value, called here $\overline{h}_{x_i}$, among those found at the previous point for all the cycles in which x_i appears. Formally, associate with x_i the most restrictive constraint on $\overline{h}_{x_i}$ among the set of cycles $\mathfrak{C}_{x_i} = \{c \in \mathscr{C} | x_i \in c\}$, i.e.,

$$\begin{aligned} \overline{h}_{x_i} = \max \quad & h \\ \text{s.t.} \quad & h > 0, \\ & |\mu_c(h)| < \alpha, \quad \forall c \in \mathfrak{C}_{x_i}. \end{aligned}$$

The final result of the analysis is thus having each dynamic variable associated with a time scale. More precisely, if α was correctly chosen (in a sense to be discussed), it is guaranteed that if the integration step is set below a certain $\overline{h}_i$, then the discretised ODE equation that computes $x_{i,k+1}$ cannot be responsible for possible instabilities.

Ranking the dynamic variables by $\overline{h}_i$ will provide the basis for the subsequent decoupled integration. Before that, however, it is convenient to show how the procedure just sketched can be specialised and implemented with an integration method of the considered class. For the sake of simplicity we here select the Explicit Euler one.

3.3.3 A possible analysis implementation

Taking the Explicit Euler (EE) as the “probe” integration method – see the remark before (3.3) – the discretised system of the same equation specialises to

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h \cdot f(\mathbf{x}_k, \mathbf{u}_k). \tag{3.9}$$

Thus, the edge weights of the associated digraph take the form

$$\rho\left[e^{i,j}\right](h) = \begin{cases} 1 + h \cdot \dfrac{\partial f_i}{\partial x_i} & \text{if } i = j, \\ h \cdot \dfrac{\partial f_i}{\partial x_j} & \text{if } i \neq j. \end{cases}$$

As a consequence the cycle gains (3.6) can be computed as

$$\mu_c(h) = \begin{cases} 1 + h \cdot \dfrac{\partial f_i}{\partial x_i} & \text{if } L = 1 \text{ and } \dfrac{\partial f_i}{\partial x_i} < 0, \\ h^L \prod\limits_{e^{i,j} \in c} \dfrac{\partial f_i}{\partial x_j} & \text{otherwise,} \end{cases} \tag{3.10}$$

resulting in a set of constraints

$$|\mu_c(h)| \le \alpha \quad \Rightarrow \quad \begin{cases} 0 < h \le (1+\alpha)\left|\dfrac{\partial f_i}{\partial x_i}\right|^{-1} & \text{if } L = 1 \text{ and } \dfrac{\partial f_i}{\partial x_i} < 0, \\ 0 < h \le \sqrt[L]{\alpha} \cdot \left|\displaystyle\prod_{e^{i,j} \in c} \dfrac{\partial f_i}{\partial x_j}\right|^{-\frac{1}{L}} & \text{otherwise.} \end{cases} \tag{3.11}$$

Notice that the set of constraints (3.11) can be solved analytically in a closed form. This is one of the advantages of adopting EE instead of a more complex integration method for the analysis part. The presented CA implementation can be summarised by the pseudo-code presented in listing 1.

As a final remark, observe that the analysis technique could be based on *any* explicit method, i.e., not limited to the EE one. However, the adopted choice has two advantages. First, it results in explicit constraint expressions having α as the sole parameter. This allows to analyse the model at hand for different values of α in a computationally affordable manner, that is, to conduct a *parametric* separability analysis as exemplified later on in Chapter 4. Then, the higher the order of the used explicit method, the less the sparsity degree of the discrete-time model dynamic matrix, and clearly a sparse matrix is in favour of an efficient detection of the system cycles. To witness this, reconsider the example of Section 3.3.1.1: should one use the EE instead of the Heun's method, the resulting dynamic matrix would be

$$W = \frac{\partial \mathbf{F}_{\mathrm{EE}}}{\partial \mathbf{x}} = I_{3\times 3} + Ah = I_{3\times 3} + \frac{h}{2}\begin{bmatrix} -2 & 1 & 0 \\ 1 & -3 & 1 \\ 0 & 1 & -2 \end{bmatrix},$$

which is remarkably more sparse than (3.8), and apparently much easier to solve with respect to h. The EE method has thus the nice property of providing reasonably conservative stability regions with a light computational burden, whence its choice as the probe one.

3.3.4 Cycle analysis and eigenvalue analysis

In the literature, two are the major techniques to serve an analogous purpose, concerning time scale analysis, as that of this dissertation: eigenvalue Schiela and Olsson [2000] and Lyapunov exponent analysis Kuznetsov [2004], Wolf et al. [1985]. This section compares our technique to eigenvalue analysis, spending also some words on the Lyapunov exponent subject, as for the problem of guaranteeing the stability of the discrete-time solution. Doing

Algorithm 1 Algorithm to detect all the cycles in the dependency digraph.

```
function GET_PATH_FROM_A_TO_B(graph, a, b)
    paths = $\emptyset$;                                  //Initialise two empty lists
    q = $\emptyset$;
    q.append(a);
    while q $\neq \emptyset$ do
        path = q.pop();                                   //Get the first element of q
        final = q(end);
        if final == b and length(path)>1 then
            paths.append(path);
        end if
        for e $\in$ graph.successors(final) do
            if e $\notin$ path(2:end) then
                next = path;
                next.append(e);
                q.append(next);
            end if
        end for
    end while
    return paths;
end function

function GET_CYCLES(graph)
    cycles = $\emptyset$;                 //Initialise an empty list of dependency cycles
    nodes = graph.nodes();                //Get the list of all the nodes in the graph
    while length(nodes)$\neq \emptyset$ do
        n = nodes.pop();                                  //Get the first node in the list
        paths = get_path_from_a_to_b(graph,n,n);
        if paths is not empty then
            cycles.append(paths);
        end if
        graph.remove_node(n);             //All the cycles involving n are detected
    end while
    return cycles
end function
```

so, we also provide the background for the subsequent discussion 3.4 on how the analysed techniques can lead to a suitable partition of the system, in a view to a decoupled solution. To this end, we first go through a representative example, and then draw the necessary general conclusions.

3.3.4.1 An example: loosely damped models and stability issues

Consider the linear, time-invariant, autonomous system

$$\dot{\mathbf{x}} = \begin{bmatrix} -\omega_n\xi & -\omega_n\sqrt{1-\xi^2} \\ \omega_n\sqrt{1-\xi^2} & -\omega_n\xi \end{bmatrix}\mathbf{x}, \tag{3.12}$$

that has the two complex conjugate eigenvalues

$$\lambda_{1,2} = -\omega_n \cdot \left(\xi \pm \imath\sqrt{1-\xi^2}\right),$$

with natural frequency $\omega_n > 0$ and damping factor $0 < \xi \leq 1$, thus being asymptotically stable. If (3.12) is discretised with the EE method, the eigenvalues of the corresponding discrete-time system provide the stability condition

$$h < 2\frac{\xi}{\omega_n} := \overline{h}_s. \tag{3.13}$$

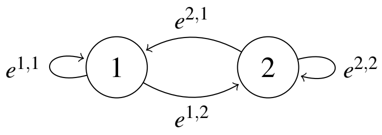


Figure 3.3: Dependency graph associated with system (3.12) discretised with EE.

Applying CA, the digraph of Figure 3.3 is readily built, and the cycle gains turn out to be

$$\begin{aligned}
\mu_{\langle e^{1,1}\rangle} &= \rho\left[e^{1,1}\right] = 1 + h\frac{\partial f_1}{\partial x_1} = 1 - h\omega_n\xi \\
\mu_{\langle e^{2,2}\rangle} &= \rho\left[e^{2,2}\right] = 1 + h\frac{\partial f_2}{\partial x_2} = 1 - h\omega_n\xi \\
\mu_{\langle e^{1,2},e^{2,1}\rangle} &= \rho\left[e^{1,2}\right]\cdot\rho\left[e^{2,1}\right] = h^2\cdot\frac{\partial f_1}{\partial x_2}\cdot\frac{\partial f_2}{\partial x_1} = -h^2\omega_n^2(1-\xi^2)
\end{aligned}$$

leading to the α-dependent constraints

$$\begin{aligned}
\left|\mu_{\langle e^{1,1}\rangle}\right| < \alpha \quad &\Rightarrow \quad h \leq \frac{1+\alpha}{\omega_n \xi} := \overline{h}_{c,1} \\
\left|\mu_{\langle e^{2,2}\rangle}\right| < \alpha \quad &\Rightarrow \quad h \leq \frac{1+\alpha}{\omega_n \xi} := \overline{h}_{c,2} \\
\left|\mu_{\langle e^{1,2}, e^{2,1}\rangle}\right| < \alpha \quad &\Rightarrow \quad h \leq \frac{1}{\omega_n} \cdot \sqrt{\frac{\alpha}{1-\xi^2}} := \overline{h}_{c,3}
\end{aligned} \tag{3.14}$$

It is then interesting to compare the stability bounds on h provided by eigenvalue analysis, and those that limit the magnitude of the cycle gains provided by CA. In particular, the CA bounds on h are looser than the eigenvalue-related bounds (thus CA does not guarantee discrete-time stability) if $\overline{h}_s \leq \overline{h}_{c,i}$, i.e.,

$$\begin{cases} 2\dfrac{\xi}{\omega_n} \leq \dfrac{1+\alpha}{\omega_n \xi} \\ 2\dfrac{\xi}{\omega_n} \leq \dfrac{1}{\omega_n} \cdot \sqrt{\dfrac{\alpha}{1-\xi^2}} \end{cases} \quad \Rightarrow \quad \begin{cases} \alpha \geq 2\xi^2 - 1 \\ \alpha \geq 4\xi^2\left(1-\xi^2\right) \end{cases} \tag{3.15}$$

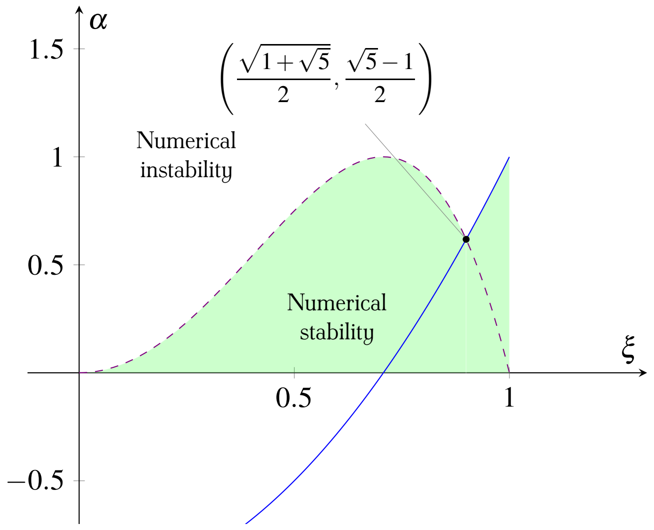


FIGURE 3.4: Stability conditions on the parameter α w.r.t. ξ.

3.3.4.2 Discussion

In the example – but this is intuitively general – a value of α can be found so that the CA constraints also guarantee stability, as the eigenvalue ones do.

In particular, there exists an α that makes the two upper bounds on h coincident. Below said value, α however provides to CA an additional degree of freedom with respect to eigenvalue analysis, and this degree of freedom can be exploited to attenuate the effects of mutual dependencies among the discrete-time dynamic variables—a purpose that is apparently decoupling-related, and not natural to pursue with the eigenvalue-based approach.

Coming back to the example, we can notice that the value of α that makes the two bounds on h coincide, depends only on ξ and not on ω_n. This is an important fact since the stability properties of the system are essentially determined by the damping factor.

Furthermore, analysing Figure 3.4, one can observe that depending on the value of ξ, the active constraint changes. In particular, it is worth noticing that for high damping factors, the constraint related to the cycles with cycle length $L = 1$ – i.e., relating each dynamic variable to itself – dominates, while for low damping factors the dominant constraint becomes the one related to the cycles with cycle length $L = 2$— i.e., involving two dynamic variables. This can be easily interpreted as the importance of the couplings among the variables increases as long as the damping factor – i.e., the oscillatory behaviour of the involved dynamic variables – decreases, and vice versa; this fact can be intuitively generalised also to more complex cases.

In other words, while a reduction of ξ – viewed from the eigenvalue standpoint – appears just as a stability degree reduction, the same fact – observed conversely by CA – reveals its nature of a stronger coupling between parts of the system. In this sense, therefore, CA provides stability-related information in a way that is particularly keen to be used for system partitioning in a view to decoupled integration.

To see the same matter from another viewpoint, one can notice that for a (linear) system of order n, eigenvalue analysis provides n constraints on h, one per each of the system modes, while CA provides *at least n* constraints, one constraint per system cycle. In other words, with eigenvalue analysis one observes the system mode by mode, implicitly considering a state space where all those modes are decoupled (the examples showed this only for a couple of complex modes, but the generalisation is straightforward). With CA, on the contrary, the same information is split in such a way to explicitly evidence the couplings that eigenvalue analysis – in the sense above – conceals.

Incidentally, in the linear case, CA provides exactly n constraints in the case of a triangular system system with real eigenvalues, as in such a case said eigenvalues appear in the diagonal of the dynamic matrix; in this case, quite obviously, the value of α discriminating stability from instability is the unity.

As a consequence of the remarks above, the value of α is quite hard to be chosen *a priori*, but must be limited to the compact $(0,1)$, and must be related to the damping factor of the system. However, when choosing as a probe method EE, CA can be performed independently of the chosen value of α, and analytically. This allows for other kind of analysis, based on some indices that are presented in the next section, letting the modeller get more insight on structural properties of the system, and thus perform the partition irrespective of the value of α.

3.4 Separability indices

The result of CA is to associate each dynamic variable with an upper bound of the integration step, thus with a quantity related to its time-scale. The variables can then be ordered – and possibly clustered – by increasing value of $\overline{h}_{x_i}$. Based on this, some synthetic indices will now be defined, useful for deciding how to partition the original model in weakly coupled submodels. It will also be shown how such indices extend the idea of "stiffness", like CA was shown to evidence more decoupling-related information than eigenvalue analysis.

To start, consider the classical stiffness indicator based on eigenvalues analysis, i.e., the *stiffness ratio*.

Definition 3.4.1 (Stiffness ratio). The *stiffness ratio* σ_R Cellier and Kofman [2006] is defined as the ratio between the absolute largest real part and the absolute smallest real part of any eigenvalue, i.e.,

$$\sigma_R = \frac{\max_i |\Re\{\lambda_i\}|}{\min_i |\Re\{\lambda_i\}|}.$$

Highly stiff systems are associated with high values of σ_R. This definition, however, cannot be applied to any stiff system, e.g., either stiff systems of order 1, or systems with eigenvalues on the imaginary axis. Moreover, since it just considers the real part of the eigenvalue, the index may suggest that the system is highly stiff, even if it is not oscillatory.

Apparently, the stiffness ratio is defined for a linear (or linearised) system, and indicates how much the smaller time scale differs from the larger one. It is thus a good index for understanding whether or not to use an integration method for stiff systems on the entire model, but gives no information on

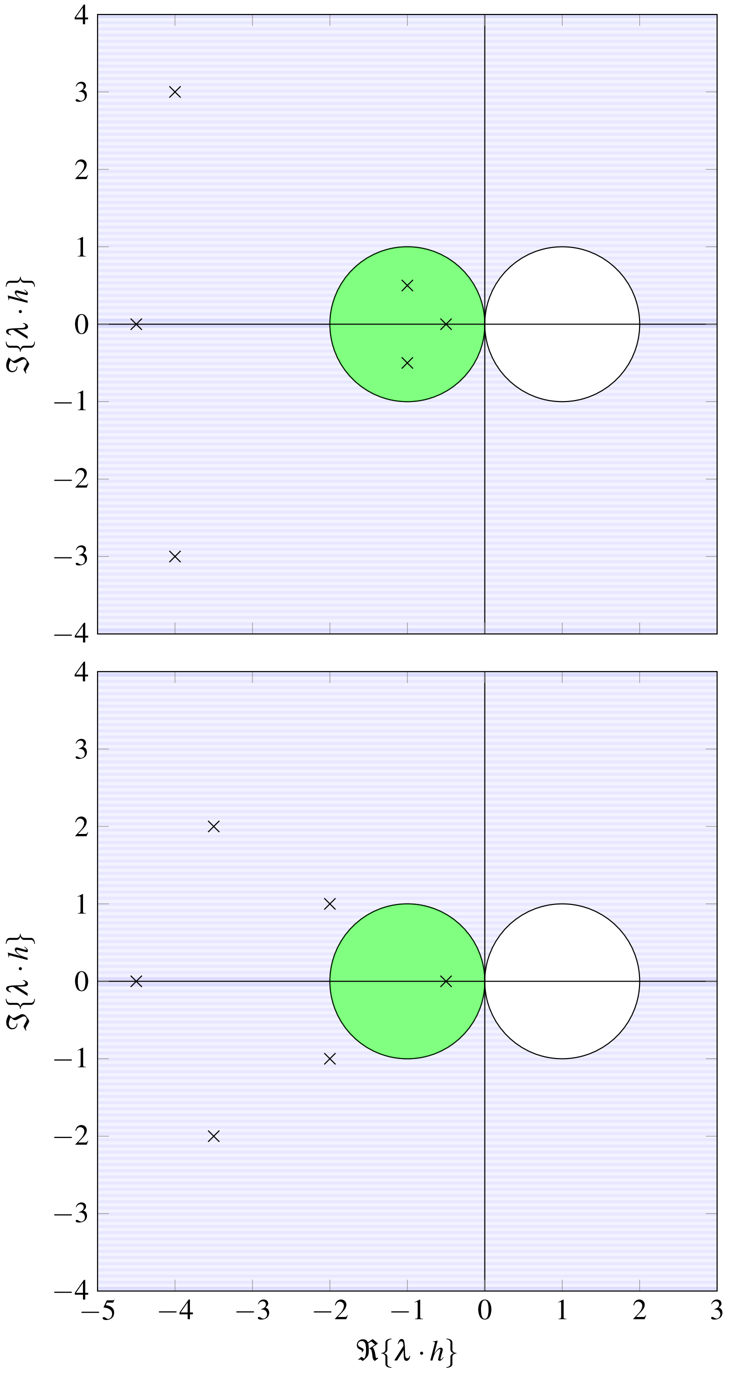


Figure 3.5: Two cases of linear systems with the same stiff ratio.

how many “clusters of time scales” are present in it, nor on which dynamic variable belongs to which cluster.

To exemplify, let us limit to the linear case, and consider Figure 3.5. In the left graph, the continuous-time eigenvalues of the system (indicated with the cross) are not equally spaced in the left-half-plane, and can be divided into two clusters: those that are close to the origin are associated with “slow dynamics”, while the others are associated with “fast dynamics”. The presence of the two different time scales is also evidenced by computing the stiffness ratio of Definition 3.4.1. Let us now consider the right graph of the same figure. In this case, the stiffness ratio is the same, since the closest and the farthest eigenvalues from the origin are the same, while the eigenvalues of the system are almost equally distributed in the left-half-plane. This feature of the system is strictly related to how much the system can be “separable” and is not evidenced in any way by the stiffness ratio.

Coming back to the CA approach, two different indices based on it can be defined. One (the *stiffness index*, see Definition 3.4.2) quantifies the span of the time scales in the model, analogously to the one of Definition 3.4.1. The other (the *separability index*, see Definition 3.4.4) indicates to what extent the clusters of dynamic variables corresponding to those time scales the system can be computed in a decoupled manner. Both indices are function of α, and being based on CA, they can be computed also for nonlinear systems.

Denote by $\mathscr{H}$ the set of integration steps h_{x_i} associated with each dynamic variable, and assume $\mathscr{H}$ ordered by ascending values of h, i.e.,

$$\mathscr{H} = \{h_1 \leq h_2 \leq \ldots \leq h_N\}.$$

Based on that, the following definitions can be given.

Definition 3.4.2 (Stiffness index). The *stiffness index* for a given α is the ratio between the minimal and the maximal integration step found with the cycle analysis, i.e.,

$$\sigma(\alpha) = \frac{h_{\max}(\alpha)}{h_{\min}(\alpha)}. \tag{3.16}$$

Analogously to the stiffness ratio σ_R, also for the stiffness index highly stiff systems are associated with high values of σ.

Definition 3.4.3 (Separability term). The *separability term* for a given α, and for a given couple of variables x_i and x_j is

$$s_\alpha(i,j) = \frac{|h_i(\alpha) - h_j(\alpha)|}{\max_m (h_{m+1}(\alpha) - h_m(\alpha))}, \qquad h_i, h_j \in \mathscr{H}.$$

Definition 3.4.4 (Separability index). The *separability index* for a given α is the one minus the ratio between the maximal and the average difference among two subsequent values of the time scales, i.e.,

$$s(\alpha) = 1 - \frac{\frac{1}{N-1}\sum_{i=1}^{N-1} h_{i+1}(\alpha) - h_i(\alpha)}{\max_i \left(h_{i+1}(\alpha) - h_i(\alpha)\right)} = 1 - \frac{1}{N-1}\sum_{i=1}^{N-1} s_\alpha(i+1,i).$$

Apparently, high values of $s(\alpha) \in (0,1)$ indicate that the time scales involved in the system are different enough to be effectively separated.

In Chapter 4, the presented indices will be used to evaluate the level of stiffness and separability of the considered examples. Summarising, the stiffness ratio and index are comparable and synthetic descriptions of the separation between the maximal and the minimal model time scales, not suited however for understanding whether said model can be partitioned. The separability index is another synthetic one, but is specifically targeted at quantifying the possibility of such a separation. The separability term is a local index to a couple of adjacent time scales, and an analysis of its behaviour can easily suggest possible separation points.

3.4.1 Separability analysis

The proposed separability index (3.4.4) is a synthetic description of a structural property of the overall system, but additional information can be extracted from CA, providing also suggestions on how the system can be partitioned. However, CA – thus the computation of the proposed indices – usually requires the choice of a value of α, which has been already discussed above.

On the basis of those remarks, a *parametric separability analysis* can be performed:

1. perform a parametric CA, and express the time scales associated with each dynamic variable as a function of $\alpha \in (0,1)$;
2. for each value of $\alpha \in (0,1)$, order the time scales obtaining a set of values of the integration steps $\mathscr{H} = \{h_1 \leq h_2 \leq \ldots \leq h_N\}$;
3. for each value of $\alpha \in (0,1)$, compute the separability terms $s_\alpha(i+1,i)$ for all $i = 1,\ldots,N-1$;

4. plot the obtained $s_\alpha(i+1,i)$ as a function of α and $i = 1,\ldots,N-1$, possibly as a colormap.

The result of this kind of analysis is that whenever a couple of dynamic variables (identified in the set $\mathscr{H}$ with their indices i and $i+1$), the plot will highlight a peak—examples of those kind of plots are presented in the next chapter.

This kind of analysis provides information that is twofold and immediately interpretable by the modeller. First, considering only a single peak for simplicity, the variables on one side of the peak may be considered as coupled, and highly decoupled in terms of time scale from the variables on the other side of the peak. This can be exploited for designing a partition of the system, or multiple in the case of many peaks. In addition, since the variables are ordered by time scales, those with lower indices are associated with fast time scales, the others with slow ones.

3.4.2 Exploiting the partition

The information coming from the separability analysis can be exploited in different ways. A first possibility is to split the model into two submodels, and use suitable mixed-mode integration methods, as discussed in the next section. On the other hand, the identified time scales can be used to structure more complex (co-)simulation architectures, splitting the system into many subsystems.

In particular, if the time scales present in the considered model can be clustered into more than two sets, an iterative approach can be used so as to improve simulation efficiency, extending the mixed-mode integration method to a multi-rate mixed-mode integration. The simulation structure can be obtained as follows

1. identify the time scales by means of the separability analysis proposed herein;

2. according to the time scale of interest, the system can be split into two subsystems, one slow that will be simulated with an explicit method, one fast that will be integrated with implicit method(s);

3. if the faster dynamics cannot be split into other subsystems according to the time scales, then the algorithm terminates;

4. otherwise, the fast dynamics are split into two subsystems, one faster, and one slower, that will be integrated with a multi-rate implicit method;

5. go to step (3).

This approach automatically builds the simulation architecture, exploiting the system structure without any added effort on the part of the modeller. The resulting faster partitions are smaller reducing the computational complexity of solving them with implicit algorithms.

3.5 Mixed-mode integration

This section deals with how the information coming from CA can be exploited to improve simulation efficiency. Broadly speaking, since a complete treatise of this matter will require more than one future work, we can say that such an exploitation takes place along two fundamental axes. One refers to the used integration methods, the other to the adopted simulation architecture.

3.5.1 Exploiting by integration methods

Having clustered the dynamic variables by time scale, one can use explicit integration methods for the slow ones, and implicit methods for the fast ones. This, in some cases, may result in loosing a precise representation of fast phenomena, but apparently improves efficiency. Moreover, the type of information provided by CA facilitates the modeller, as he/she has just to decide which is the smallest time scale of interest for the simulation study at hand.

We now illustrate some applications of this idea, limiting the scope to a system partitioned in two subsystems. This is done for simplicity and without loss of generality, since after a first partition one could simple re-apply the proposed technique to one or more of the obtained subsystems. We also use different pair of integration methods, to show that the proposed exploitation is applicable for diverse methods, e.g., Ascher et al. [1997] present some examples of joint use of pairs of Runge-Kutta integration methods. Incidentally, this further highlights the neat separation between the analysis and the simulation part, thus between the probe method used for CA, and the integration methods. Finally, we show that once the order of accuracy of the used methods is chosen, the technique leads to the same mixed-mode integration algorithm whatever the particular methods are. This allows the modeller to select – among those of the desired accuracy – the integration methods that are most efficient for the particular application.

Coming to the exploitation technique, consider the generic autonomous nonlinear ODE system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) \tag{3.17}$$

and assume it to be partitioned into two subsystem: one with slow dynamics, the other with fast dynamics. Following an approach similar to the one presented in Schiela and Olsson [2000], we can left-multiply the state vector by a projection matrix $P = \text{diag}\{p_1, p_2, \ldots, p_n\}$, with $p_i \in \{0,1\}$ to select the slow part, and by $\overline{P} = I - P$ to select the fast part. Therefore (3.17) can be written as

$$\begin{cases} \dot{\mathbf{x}}^S = P\dot{\mathbf{x}} = P\mathbf{f}\left(\mathbf{x}^S, \mathbf{x}^F\right) \\ \dot{\mathbf{x}}^F = \overline{P}\dot{\mathbf{x}} = \overline{P}\mathbf{f}\left(\mathbf{x}^S, \mathbf{x}^F\right) \end{cases} \tag{3.18}$$

where $\mathbf{x}^S$ represents the slow variables, and $\mathbf{x}^F$ the fast ones.

To qualify the used methods, we adopt the classical notation used for Runge-Kutta ones, i.e., a method of order s is expressed in the general form

$$x_{k+1} = x_k + h\sum_{i=1}^{s} b_i \kappa_i, \quad \text{with} \quad \kappa_i = \mathbf{f}\left(x_k + h\sum_{j=1}^{s} a_{ij}\kappa_j, t_k + c_i h\right), \tag{3.19}$$

where the coefficients a_{ij}, b_i and c_i are usually expressed by means of the so-called Butcher tableau

$$\begin{array}{c|c} c & A \\ \hline & b \end{array} = \begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & \cdots & a_{ss} \\ \hline & b_1 & \cdots & b_s \end{array}.$$

Recall for convenience that an explicit Runge-Kutta method is characterised by $a_{ij} = 0$ for all $j \geq i$, which is not true for implicit ones.

3.5.1.1 Explicit-Implicit Euler

First of all we consider the simplest exploitation case, i.e., using Explicit Euler (EE) for the slow part, and Implicit Euler (IE) for the fast part. This approach has already been presented in Schiela and Olsson [2000] for the linear case.

The Butcher tableaux of the two methods are respectively

$$\text{EE:} \quad \begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array} \qquad\qquad \text{IE:} \quad \begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

Correspondingly, equation (3.18) can be expressed as

$$\begin{aligned}\mathbf{x}_{k+1}^S &= P\mathbf{x}_{k+1} = P\mathbf{x}_k + hP\mathbf{f}\left(\mathbf{x}_k^S, \mathbf{x}_k^F, t_k\right)\\ \mathbf{x}_{k+1}^F &= \overline{P}\mathbf{x}_{k+1} = \overline{P}\mathbf{x}_k + h\overline{P}\mathbf{f}\left(\mathbf{x}_{k+1}^S, \mathbf{x}_{k+1}^F, t_k\right),\end{aligned}$$

which in the linear case becomes

$$\begin{aligned}\mathbf{x}_{k+1}^S &= P\mathbf{x}_k + hPA\mathbf{x}_k\\ \mathbf{x}_{k+1}^F &= \overline{P}\mathbf{x}_k + h\overline{P}A\mathbf{x}_{k+1}.\end{aligned}$$

Composing those two equations, and solving for $\mathbf{x}_{k+1}$, we can obtain

$$\mathbf{x}_{k+1} = \left(I - h\overline{P}A\right)^{-1}(I + hPA)\,\mathbf{x}_k.$$

3.5.1.2 Explicit-Implicit midpoint

In this section we consider the Explicit Midpoint (EM) and Implicit Midpoint (IM), two second-order accurate integration methods. The corresponding Butcher tableaux are

$$\text{EM:}\quad \begin{array}{c|cc} 0 & 0 & 0\\ 1/2 & 1/2 & 0\\ \hline & 0 & 1\end{array} \qquad\qquad \text{IM:}\quad \begin{array}{c|c} 1/2 & 1/2\\ \hline & 1\end{array}$$

According to (3.19), the slow part of equation (3.18) becomes

$$\mathbf{x}_{k+1}^S = P\mathbf{x}_{k+1} = P\mathbf{x}_k + h\kappa_2 \qquad \text{with} \quad \begin{aligned}\kappa_1 &= P\mathbf{f}\left(\mathbf{x}_k^S, \mathbf{x}_k^F, t_k\right)\\ \kappa_2 &= P\mathbf{f}\left(\mathbf{x}_k^S + \frac{h}{2}\kappa_1, \mathbf{x}_k^F, t_k + \frac{h}{2}\right),\end{aligned}$$

leading to the more compact equation

$$\mathbf{x}_{k+1}^S = P\mathbf{x}_k + hP\mathbf{f}\left(\mathbf{x}_k^S + P\frac{h}{2}\mathbf{f}_k, \mathbf{x}_k^F, t_k + \frac{h}{2}\right)$$

where

$$\mathbf{f}_k := \mathbf{f}\left(\mathbf{x}_k^S, \mathbf{x}_k^F, t_k\right).$$

As for the fast part, the IM method can be used. Thus, using the numerical solution of the slow part as an input for the fast part, leads to

$$\mathbf{x}_{k+1}^F = \overline{P}\mathbf{x}_{k+1} = \overline{P}\mathbf{x}_k + h\overline{P}\kappa_1^F \qquad \text{with} \quad \kappa_1^F = \mathbf{f}\left(\mathbf{x}_k + \frac{h}{2}\kappa_1, t_k + \frac{h}{2}\right). \tag{3.20}$$

Observing that $\kappa_1^F = (\mathbf{x}_{k+1} - \mathbf{x}_k)/h$, equation (3.20) can be written in a more compact form

$$\mathbf{x}_{k+1}^F = \overline{P}\mathbf{x}_k + h\overline{P}\mathbf{f}\left(\frac{\mathbf{x}_{k+1}^S + \mathbf{x}_k^S}{2}, \frac{\mathbf{x}_{k+1}^F + \mathbf{x}_k^F}{2}, t_k + \frac{h}{2}\right).$$

In the linear case, the two expressions become

$$\begin{aligned} \mathbf{x}_{k+1}^S &= P\mathbf{x}_k + hPA\mathbf{x}_k + \frac{h^2}{2}(PA)^2\mathbf{x}_k \\ \mathbf{x}_{k+1}^F &= \overline{P}\mathbf{x}_k + \frac{h}{2}\overline{P}A\mathbf{x}_k + \frac{h}{2}\overline{P}A\mathbf{x}_{k+1} \end{aligned}.$$

Composing the two expressions, the overall dynamics can be computed as

$$\mathbf{x}_{k+1} = \left(I - \frac{h}{2}(I-P)A\right)^{-1}\left(I + \frac{h}{2}(I+P)A + \frac{h}{2}(PA)^2\right)\mathbf{x}_k.$$

3.5.1.3 Heun-Lobatto mixed-mode

Analogous computations can be done with different integration methods. For example the Heun's method can be taken as the explicit method, and the second-order Lobatto IIIA one for the implicit part. The Butcher tableaux of the two methods are respectively

$$\text{Heun:}\quad \begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & 1/2 & 1/2 \end{array} \qquad\qquad \text{Lobatto IIIA:}\quad \begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \end{array}$$

With similar computations to the ones performed for the midpoints methods, it is easy to show that the slow part dynamics are ruled by

$$\begin{aligned} \mathbf{x}_{k+1}^S &= P\mathbf{x}_k + h\mathbf{f}_k\frac{h^2}{2}\mathbf{f}\left(\mathbf{x}_k^S + hP\mathbf{f}_k, \mathbf{x}_k^F, t_k + h\right) \\ \mathbf{x}_{k+1}^F &= \overline{P}\mathbf{x}_k + \frac{h}{2}\overline{P}\left(\mathbf{f}_k + \mathbf{f}_{k+1}\right) \end{aligned}$$

Which in the linear case become

$$\begin{aligned} \mathbf{x}_{k+1}^S &= P\mathbf{x}_k + hPA\mathbf{x}_k + \frac{h^2}{2}(PA)^2\mathbf{x}_k \\ \mathbf{x}_{k+1}^F &= \overline{P}\mathbf{x}_k + \frac{h}{2}\overline{P}A\mathbf{x}_k + \frac{h}{2}\overline{P}A\mathbf{x}_{k+1} \end{aligned}$$

Leading to the overall dynamics

$$\mathbf{x}_{k+1} = \left(I - \frac{h}{2}(I-P)A\right)^{-1}\left(I + \frac{h}{2}(I+P)A + \frac{h^2}{2}(PA)^2\right)\mathbf{x}_k$$

which is exactly the same dynamics obtained by the combination of any second-order accurate couple of Runge-Kutta methods.

3.5.1.4 General application

Alternatively to the proposed techniques, other pairs of Explicit-Implicit Runge-Kutta methods can be used, for example all the ones proposed in Ascher et al. [1997]. In principle one could also use completely different methods.

Summarising, exploiting CA by integration method leads to join the best of implicit and explicit integration in a knowledgeable manner for the case under question. The resulting integration scheme is represented in Figure 3.6.

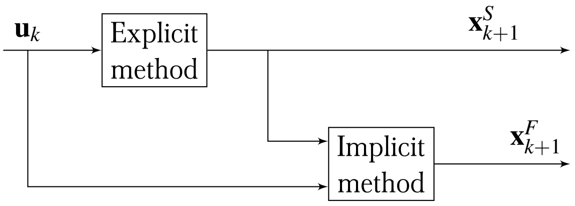


Figure 3.6: Mixed-mode integration scheme.

This kind of scheme will be used in Chapter 4 to evaluate the performance of the proposed methods.

3.5.2 Exploiting by simulation architecture

The last sections presented a possible way of exploiting the information coming from CA to speed up simulation, by exploiting appropriately numerical integration methods. However, there is an additional possibility that we propose in this work. Broadly speaking, we can say that CA can be exploited along two fundamental axes. One – treated above – refers to the used integration methods, the other to the adopted simulation architecture.

To enhance simulation efficiency, it is useful to identify which parts of a model can be simulated in parallel. To this end, the dependency digraph used for CA can be further exploited by detecting Parallelisable Cycle Sets (PCS), as briefly explained in this section.

A PCS is defined in the simplest manner as a set of cycles in the digraph that share a single node and have no other nodes in common. Extensions can be given considering *sets* of common nodes instead of a single one, or "weak" absence of other common nodes, for example based on the dominance of some cycle gains over others, but these are not necessary for the purpose of this section and cannot be treated in this work for space limitations. The interested reader is referred to Fortunato [2010] for some details on how to formally define and detect PCS-like structures – usually defined as community in the network analysis theory – on the same digraph used here for CA.

The key idea motivating the search for PCS is that it is not infrequent to encounter situations in which fast parts of an overall model are made mutually dependent only by slower ones. This happens, for example, when several heat networks are connected to a large central energy storage. Another similar situation is when the presence of some controls eliminates high-frequency variabilities and thus confines the coupling of some parts of the model to low frequency only. This could be the case when branches of a grid are connected to a central strong node, which is tightly controlled. A possible example of PCS as seen on the model digraph is shown in Figure 3.7. If the model parameters actually make the PCS emerge, node C would be the common one, and the four subsystems corresponding to the cycles in the PCS would be composed of nodes {T}, {R}, {B}, and {L, LT, LB}.

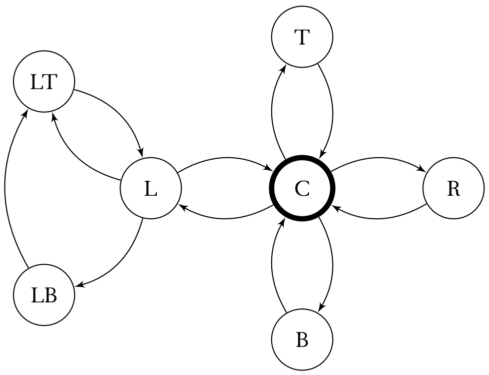


Figure 3.7: An example of PCS as seen on the model digraph.

The usefulness of PCS comes by simply observing that they evidence situations like those just mentioned, and that in such cases the fast parts of the system can not only be dynamically decoupled from the slow ones, but also simulated in parallel. Furthermore, given the variety of the encountered time scales, the same model can give rise to different partitions into parallelisable models, depending on how the analyst chooses to split the time scales. Based on this idea, PCS can be exploited in at least two ways.

First, they can be detected on the entire digraph, i.e., before possibly selecting the time scale splits. Even in the case of a monolithic solution, and independently of the integration method, doing so provides an automatic selection of which parts of the system can be parallelised, e.g., by acting as discussed in Casella [2013].

Second, one can perform CA as described in the previous sections, and then detect PCS only for those parts for which implicit methods are to be used, so as to combine the improvements coming from DD with an efficiency enhancement of the most computationally intensive part of the simulation code.

From a more technological standpoint, one can then just employ parallel computing architectures, or even use the so obtained information to structure a co-simulation setup. In the latter case, the proposed technique provides more formally grounded an alternative to heuristics based e.g. on the minimisation of the number of signals exchanged among the co-simulation units [Hendrickson and Devine, 2000, Kernighan and Lin, 1970]. Of course such optimisations are not possible when the structure of the simulation setup is dictated by the used software tools, but in the last years formalisms and standards have been emerging to provide designers with more freedom in this respect, see e.g. [Andersson et al., 2011, Blochwitz et al., 2012, Papadopoulos and Leva, 2013a].

As a final but important remark, the proposed approach allows to obtain a co-simulation setup starting from a monolithic model. This can be extremely useful to solve the initialisation problem, as doing so in a centralised manner generally yields improved convergence guarantees with respect to a distributed approach [Burrage, 1993].

3.6 Application-oriented remarks

After presenting the proposed DD-based technique in its entirety, a few words are in order to motivate some of the adopted choices, and discuss its practical use.

Starting from CA, its application requires to select the "probe" discretisation method. The choice made in this work is the EE one, and some motivation for that is in order.

In fact, after describing the CA technique, we could observe that the ranking of the dynamic variables by time scale was obtained by exploiting a known weakness of explicit fixed-step integration (probe) methods, i.e., their fairly small region of numerical stability—see, e.g., Figure 3.8 for the Explicit Runge-Kutta (ERK) family Cellier and Kofman [2006].

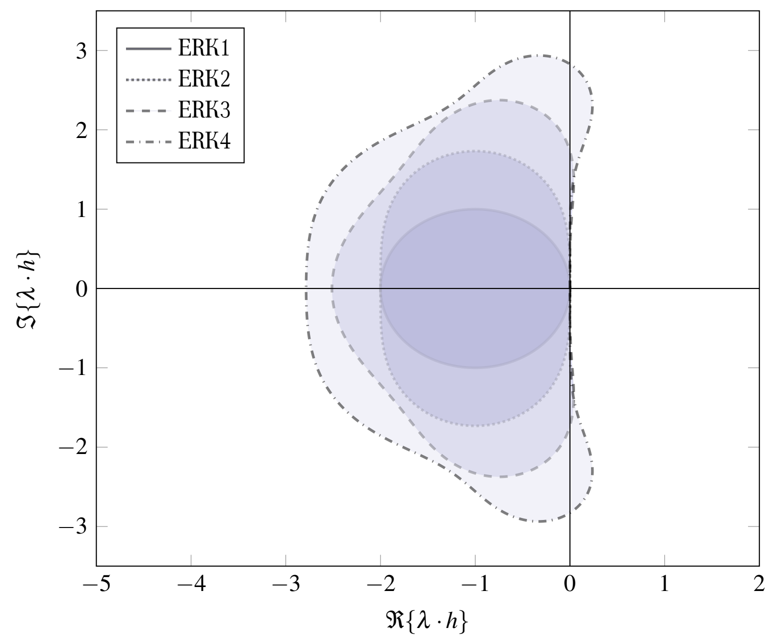


FIGURE 3.8: Numerical stability domains of ERK (interior of the curves).

The natural method selection guidelines are therefore the conservatism of the obtained ranking, and the ease in writing and solving the constraint inequalities on h. By the way, the second guideline is the major reason why explicit methods are considered, since in the opposite case it would be necessary to solve those inequalities numerically.

Having so motivated the choice for explicit methods, and given that CA was shown to be applicable to any of them, the method selection problem is reduced to its core. Among all the methods that are still candidates at this point, the EE one has the advantage of always permitting an analytical closed-form solution of the constraint inequalities, while exhibiting small enough a stability region to provide a conservative ranking (see again Figure 3.8 recalling that EE coincides with ERK1). From a practical standpoint, the author cannot see any reason for the use of different methods, except possibly for those Adams-Bashforth type in the case of extremely loosely damped dynamics, and in general the EE method performed satisfactorily in all the numerous applications considered so far.

A possible issue with CA is the computational complexity of the method. Unfortunately the problem of finding all the cycles in a digraph has complexity $\mathcal{O}\left(2^{|E|-|N|+1}\right)$, and is a well-known and studied problem in the operations research community Goldberg and Ann [2009], Johnson [1975], Tarjan [1971, 1972]. This is of course a limitation with strongly connected digraphs, which are however seldom encountered when modelling physical systems, especially in the multibody case.

Apart from the last remark, in the first place CA is an offline activity with respect to simulations, and needs to be performed only once for a given model. Then, optimisations are possible for the search procedure so as to make the required computation time well acceptable, achieving a detection rate of thousands of cycle per second (see the remark in the example of Section 4.3). Describing the software implementation of CA is not within the scope of this dissertation; it is however worth mentioning that the used one is still a proof-of-concept prototype. See Papadopoulos and Leva [2013a] for some ideas and ongoing research on CA performance improvement, and software details.

To conclude this point, it is worth evidencing that the possibly incurred computational complexity is paid back, as anticipated, in terms of the information coming from CA. In particular, CA dictates not only the time scales associated with each state variable, but also which are the variables that are mutually interacting. This information can be used to identify independent components in the model – the strongly connected components of the dependency graph – to make the simulation code parallel, possibly combining this work with Casella [2013] (a matter deferred to future research).

Another point to discuss is the choice of α, which is the only design parameter of the method, and controls the tradeoff between the accuracy of the resulting simulation and the achieved degree of decoupling. Specifically, lower values of α result in a higher simulation accuracy, but also in a reduced capability to detect weakly coupled components.

At this stage of the research, in the choice of α some heuristics is still required. According to experience, we could say that a reasonable default choice for α is the unity in the presence of systems exhibiting only over-damped dynamics, while things can be more critical, requiring lower values, in the presence of loosely damped behaviours. Further investigation of this matter is devoted to future works, but it can already be stated that suitable guidelines for the choice of α, possibly problem-specific as just suggested, can be devised quite easily. It is also worth noticing that the computationally intensive part of the method is the analysis of the model digraph, which does not depend on α: if needed, performing multiple analysis runs with different values for that parameter, until a reasonable accuracy/separability compromise is found, is therefore an affordable task. Even more specifically, if EE is chosen as the probe discretisation method, stiffness index, separability terms and index can be computed as a function of α, allowing for a parametric analysis as performed in the previous section.

On the same front, we could thus better qualify the statement made in Chapter 2, that the presented technique is not scenario-based. In fact the *result* of the technique – i.e., the model partition – does depend on the considered operating point, but (again, if a convenient probe method like EE is used) this dependence just means that the ranking of the dynamic variables may need to be re-computed, while the analysis is done only once. This is not true for other scenario-based techniques – see, e.g., Mikelsons and Brandt [2011] – where the entire procedure has to be repeated.

As a final remark, although the matter rigorously strays from the scope of this work, the very relevant problem of model initialisation in a co-simulation context [Arnold and Schiehlen, 2009] is tendentiously easier to handle if one *first* obtains and initialises a monolithic model, and *then* partitions it. The advantages of the presented technique in this respect should be quite evident.

Chapter 4

Dynamic Decoupling: simulation examples

In this chapter, some representative examples coming from different physical domains are described and analysed. In particular, a parametric separability analysis is presented (so as to analyse the system behaviour independently of the choice of α), and a mixed-mode integration method is used to evaluate the simulation performance.

The obtained results are compared with those obtained by applying (without DD) other integration methods, namely two first-order fixed step integration methods – Explicit Euler (EE) and Implicit Euler (IE) – and other more sophisticated multistep methods – Backward Differentiation Formulas (BDF) and LSODAR (short for Livermore Solver for Ordinary Differential equations, with Automatic method switching for stiff and nonstiff problems, and with Root-finding) – are used as baseline to compute the accuracy of the computed solution. All the simulation results were obtained using jModelica and Assimulo (see Chapter 1 for more detail).

Most of the results presented in this chapter come from Papadopoulos et al. [2013], and Papadopoulos and Leva [2013a,b,c,d]. More examples can be found in Appendix A.

4.1 DC motor

Let us start from a very simple physical system: a DC motor. This example is just to apply the proposed methodology on a system that can be easily analysed also manually, without the help of automatic tools. In this case, it is quite easy to understand which are the fast – the electrical variables – and which are the slow ones – the mechanical variables, also without performing any analysis.

The motor can be represented by a third order model of the form:

$$\begin{cases} L \cdot \dot{I} = -R \cdot I - k_m \cdot \omega + u(t) \\ J \cdot \dot{\omega} = k_m \cdot I - b \cdot \omega - \tau(t) \\ \dot{\varphi} = \omega \end{cases} \tag{4.1}$$

where $L = 3\,\mathrm{mH}$ is the armature inductance, $R = 50\,\mathrm{m}\Omega$ is the armature resistance, $J = 1500\,\mathrm{kg\,m^2}$ is the inertia, $b = 0.001\,\mathrm{kg\,m^2/s}$ is the friction coefficient, and $k_m = 6.785\,\mathrm{V\,s}$ is the electro-motorical force (EMF) constant of the motor. These parameter values correspond to those of a real system. The armature voltage, $u(t)$, and the torque load, $\tau(t)$, can be taken as inputs of the system. In the given example, $u(t)$ is 500 V, and the torque is of 2500 N m.

In this preliminary example, CA is able to find 4 cycles, and the parametric separability analysis result is shown in Figure 4.1.

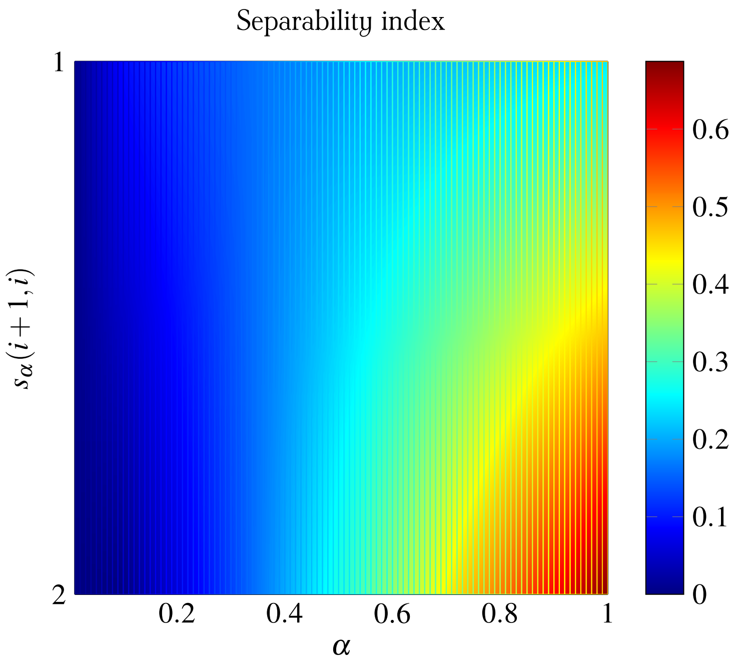


Figure 4.1: Separability parametric analysis of the DC motor.

Choosing $\alpha = 0.5$, CA leads to the following constraints:

$$\begin{aligned} I:&\quad h \le 0.032 \\ \omega:&\quad h \le 0.221 \\ \varphi:&\quad h \le 0.500 \end{aligned} \tag{4.2}$$

hence, choosing an integration step $h = 0.2$ leads to a partition of the system that is natural, separating the electrical components from the mechanical variables. Figure 4.2 shows the simulation results.

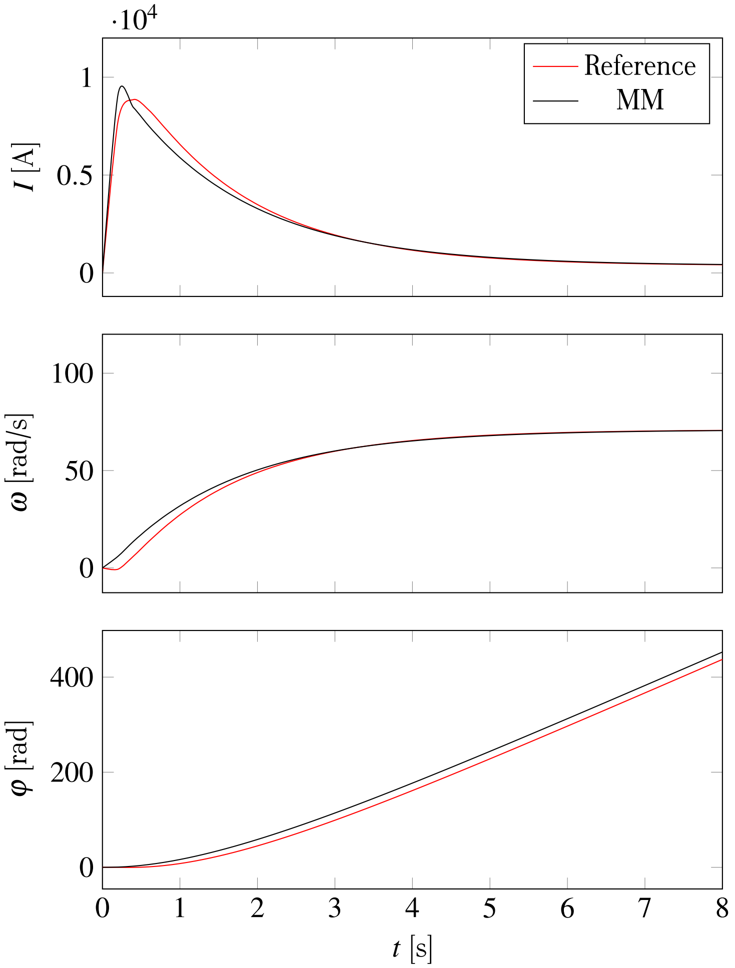


FIGURE 4.2: Simulation results of Model (4.1). Black lines represent the reference solution of the trajectories, while the red lines the mixed-mode ones.

Table 4.1 shows the simulation statistics for different integration methods. It is worth noticing that the dimension of the system the Newton iteration has to solve is reduced from 3 to 1 in the mixed-mode method. Notice, also that the EE method needs a smaller step size ($h = 0.01$) for numerical stability reasons. Apparently, the mixed-mode method performs better or in a comparable way than the others in terms of simulation statistics also in this very simple case.

Table 4.1: Simulation statistics for Model (4.1).

	Mixed-mode	**BDF**	**IE**	**EE**
# Steps	**40**	255	**40**	800
# Function ev.	123	283	**122**	–
# Jacobian ev.	**2**	5	**2**	–
# Fun. ev. in Jac. ev.	**4**	15	8	–
# Newton iterations	83	279	**82**	–
# Newton fail	0	0	0	–
Accuracy	**1.139**	–	1.531	1.876
Sim time	**0.03s**	0.09s	0.05s	0.22s

To complete the example, the proposed indices proposed in Section 3.4 are here computed, yielding the following indices—notice that since there is an eigenvalue in the origin, σ_R cannot be computed, so the stiff ratio is not defined.

$$\sigma(0.5) = 15.667, \quad s(0.5) = 0.161.$$

The stiffness $\sigma(\alpha)$ index shows that the system is highly stiff, while the separability one shows that this kind of system, with the given set of parameters is not very suited for separation. This is due to many factors, the main of which is the simplicity and low-dimensionality of the system. In the following examples, it will be shown that in more complex and realistic cases things are separability analysis is more keen to be applied and give more insight on the model at hand.

4.2 Mechanical system with brake

This example considers a nonlinear and more complex system. The system shown in Figure 4.3 is considered. A body of mass M moves on a horizontal guide subject to an exogenous motor torque command $\tau(t)^o = 10\sin(2\pi t/5)$ and to friction, acting on the wheels. The motor is not modelled for simplicity, and the relationship between the torque command and

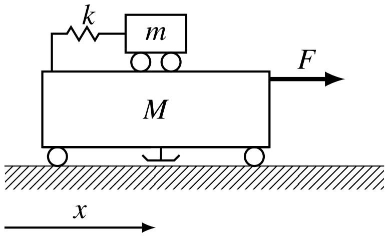


FIGURE 4.3: Mechanical system with brake.

the actual torque $\tau(t)$ is simply represented by a unity-gain, first-order continuous-time system. Also, the motor-wheel system compliance is lumped in a single rotational elasticity, $\delta\varphi$ indicating the angle difference between its sides. Another body of mass m is connected to the first one by a spring, and is also subject to friction with the former. The system also contains a brake, mounted on mass M and acting on the guide, thus introducing an input-by-state nonlinearity.

In the following x_M denotes the position of mass M, x_m that of mass m, φ the angle of the wheels, and ω their angular velocity.

Similarly to what have been done for the previous example, a preliminary analysis keeping α as a parameter is needed to understand if the system at hand exhibits quite different time scales. The result of this parametric analysis is presented in Figure 4.4.

In this case, CA detects 19 cycles. The highest separability term depends on the choice of α, since for values close to 1 the separability terms assume higher values for $i = 5$, i.e., partitioning the system between the 5-th and the 6-th variables, while for lower values of α, the highest separability term is obtained for $i = 6$. Assuming that it is preferable to keep the fast subsystem as small as possible, in this case we proceed with $\alpha = 1.0$.

$$\begin{array}{rlrl}
\dot{x}_M: & h \leq 1.000 \times 10^{-6} & \delta\varphi: & h \leq 0.111 \\
x_m: & h \leq 0.006 & \varphi: & h \leq 0.150 \\
\dot{x}_m: & h \leq 0.006 & \omega: & h \leq 0.150 \\
x_M: & h \leq 0.014 & & \\
\tau_t: & h \leq 0.053 & &
\end{array}$$

Partitioning the system as suggested by the parametric analysis means, for example, to choose an integration step $h = 0.06$ for the mixed-mode integration method, obtaining as fast variables the set of the ones on the left, and as slow variables the set of the ones on the right.

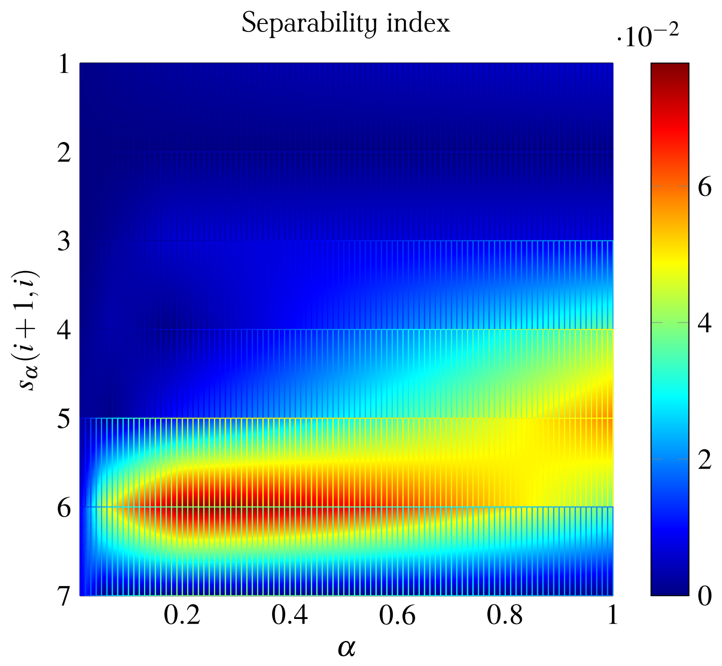


Figure 4.4: Separability analysis of the mechanical system with brake.

Figure 4.5 shows the numerical results of the mixed-mode integration method, while Table 4.2 presents some comparative simulation statistics. Notice that in this case a very tiny integration step ($h = 10^{-4}$) has been chosen for EE, due to numerical stability reasons.

Table 4.2: Simulation statistics for the mechanical system with brake.

	Mixed-mode	**LSODAR**	**IE**	**EE**
# Steps	**168**	8363	**168**	10^5
# Function ev.	1296	18816	**1293**	–
# Jacobian ev.	**103**	928	**103**	–
# Fun. ev. in Jac. ev.	**618**	–	927	–
# Newton iterations	1128	–	**1125**	–
# Newton fail	**102**	–	**102**	–
Accuracy	9.962	–	**1.480**	10.567
Sim time	**0.45s**	2.17s	0.50s	23.9s

Also in this case, the mixed-mode integration method is able to capture

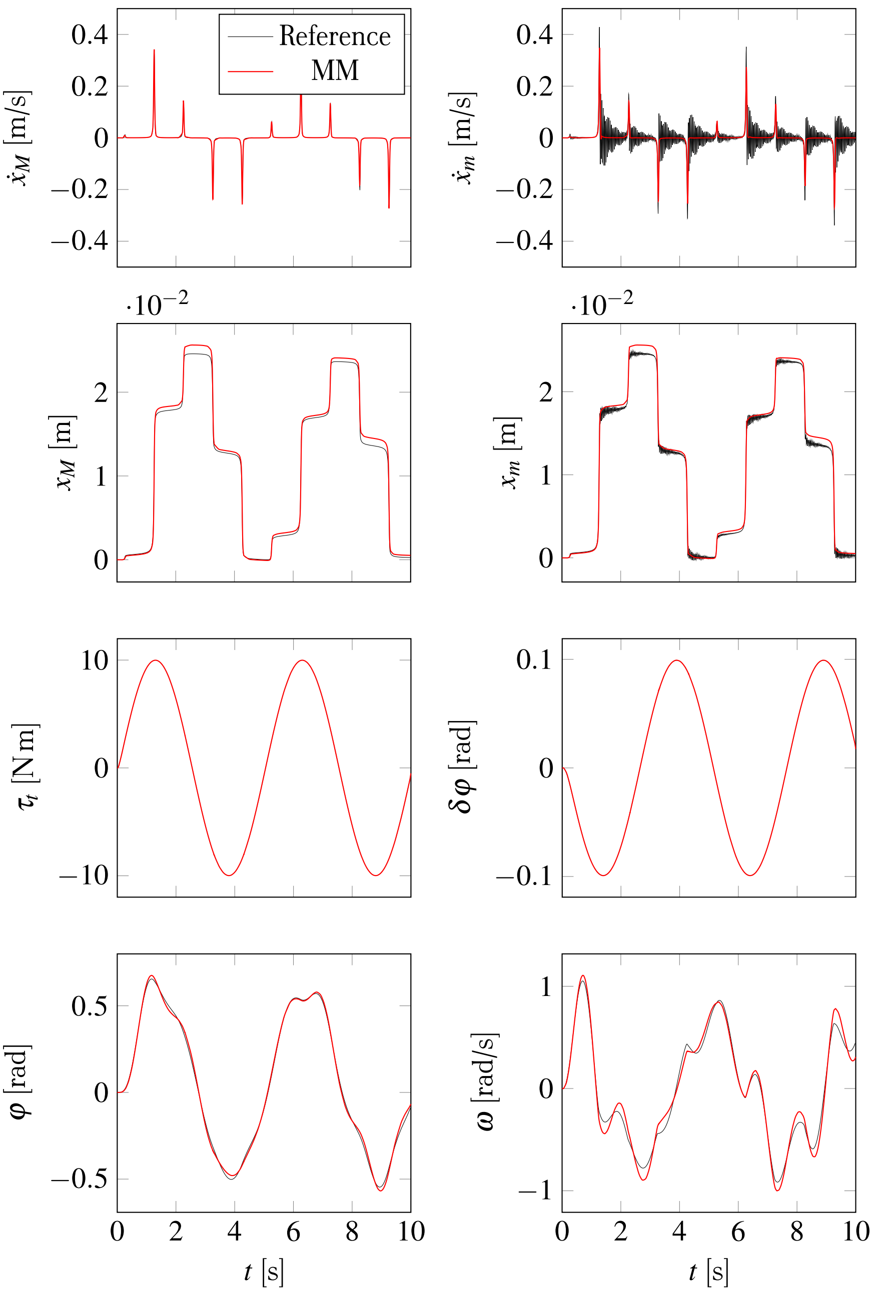


Figure 4.5: Simulation results of the mechanical system with brake.

the main dynamics in accordance with the chosen separation time scale. Furthermore, simulation statistics show that also in this nonlinear example, there is an improvement in terms of performance with respect to other methods, especially for LSODAR, which is a variable-step one.

To complete the example, the proposed indices proposed in Section 3.4 are here computed, yielding the following indices—notice that due to the nonlinearity of the system, σ_R cannot be computed.

$$\sigma(1.0) = 1495.528, \quad s(1.0) = 0.635.$$

The stiffness $\sigma(\alpha)$ index shows that the system is highly stiff, while the separability one shows that it is also well suited to be partitioned.

4.3 Triangle of masses

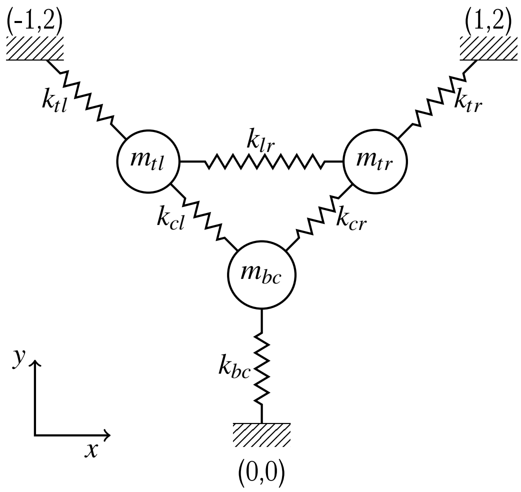


Figure 4.6: The “triangle of masses” system (dampers are not represented to simplify the drawing).

In this example, a highly nonlinear system is considered. The system is composed of three masses, moving in a vertical plane subject to gravity and to the action of six spring-damper elements, as shown in Figure 4.6 (a two-dimensional model was created for simplicity). Notice that this model is strongly nonlinear, and the eigenvalue analysis is not applicable.

In the following, x_i and y_i represent the horizontal and vertical displacement of the three masses, while the indices b, c, l, r and t indicate respectively bottom, center, left, right and top. The spring elasticity coefficients k_i

and the damping factors d_i are reported in Table 4.3, while the three masses are $m_{bc} = m_{tl} = m_{tr} = 1\,\mathrm{kg}$.

Table 4.3: The "triangle of masses" system parameters.

Parameters			
k_{tl}	$1\,\mathrm{N/m}$	d_{tl}	$1\,\mathrm{N\,s/m}$
k_{tr}	$1\,\mathrm{N/m}$	d_{tr}	$1\,\mathrm{N\,s/m}$
k_{bc}	$1\,\mathrm{N/m}$	d_{bc}	$1\,\mathrm{N\,s/m}$
k_{lr}	$1\,\mathrm{N/m}$	d_{lr}	$2\,\mathrm{N\,s/m}$
k_{cl}	$10\,\mathrm{N/m}$	d_{cl}	$1\,\mathrm{N\,s/m}$
k_{cr}	$1\,\mathrm{N/m}$	d_{cr}	$1\,\mathrm{N\,s/m}$

Also in this case, a parametric CA is performed so as to analyse the structure of the system, and Figure 4.7 shows its result.

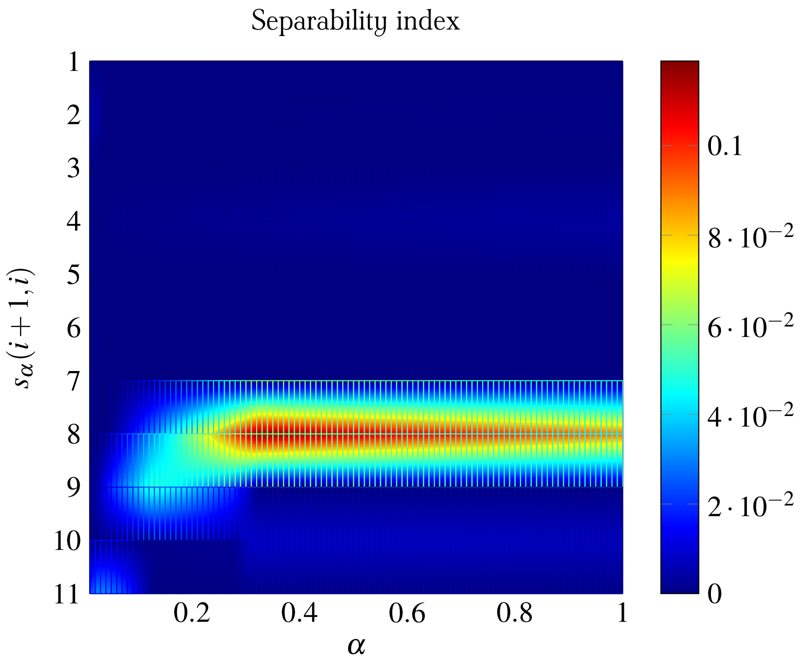


Figure 4.7: Separability analysis of the triangle of masses system.

In the "triangle of masses" system, the computed separability terms evidence that there is a neat separation between the 8-*th* and the 9-*th* variable, suggesting this as a good point for the partition.

Table 4.4: Simulation statistics for the masses triangle.

	Mixed-mode	**LSODAR**	**IE**	**EE**
# Steps	**200**	596	**200**	2000
# Function ev.	1236	1232	**857**	–
# Jacobian ev.	**13**	38	**13**	–
# Fun. ev. in Jac. ev.	**117**	–	169	–
# Newton iterations	1036	–	**657**	–
# Newton fail	5	–	**4**	–
Accuracy	0.871	–	0.780	1.337
Sim time	**0.38s**	0.51s	0.4s	0.48s

The CA detected 3984 cycles, evidencing how significant the impact of the system's degrees of freedom can be on the number of cycles. Anyway, all cycles are found in less than 1 s. The choice of α can be made on the basis of Figure 4.7, trying to maximise the separability term, e.g., by choosing $\alpha = 0.5$. The resulting constraints are thus

$$\begin{array}{rlrl}
\dot{y}_{bc}: & h \leq 0.083 & \dot{x}_{tl}: & h \leq 0.207 \\
y_{bc}: & h \leq 0.083 & \dot{y}_{tl}: & h \leq 0.207 \\
\dot{x}_{bc}: & h \leq 0.084 & x_{tl}: & h \leq 0.207 \\
x_{bc}: & h \leq 0.084 & y_{tl}: & h \leq 0.207 \\
\dot{x}_{tr}: & h \leq 0.087 & & \\
x_{tr}: & h \leq 0.087 & & \\
\dot{y}_{tr}: & h \leq 0.087 & & \\
y_{tr}: & h \leq 0.087 & &
\end{array}$$

As in the other examples, the variables on the left are the fast ones, and those on the right the slow ones. The choice of $h = 0.1$ partitions the model in those two subsystems for the mixed-mode integration, obtaining the numerical simulation represented in Figure 4.10 (simulation statistics are reported in Table 4.4). Notice that for EE a smaller integration step ($h = 0.01$) was used, for numerical stability reasons.

To complete the example, the indices proposed in Section 3.4 are here computed, yielding the following results—notice that also in this case, due to the nonlinearity of the system, σ_R cannot be computed.

$$\sigma(0.5) = 2.491, \quad s(0.5) = 0.899.$$

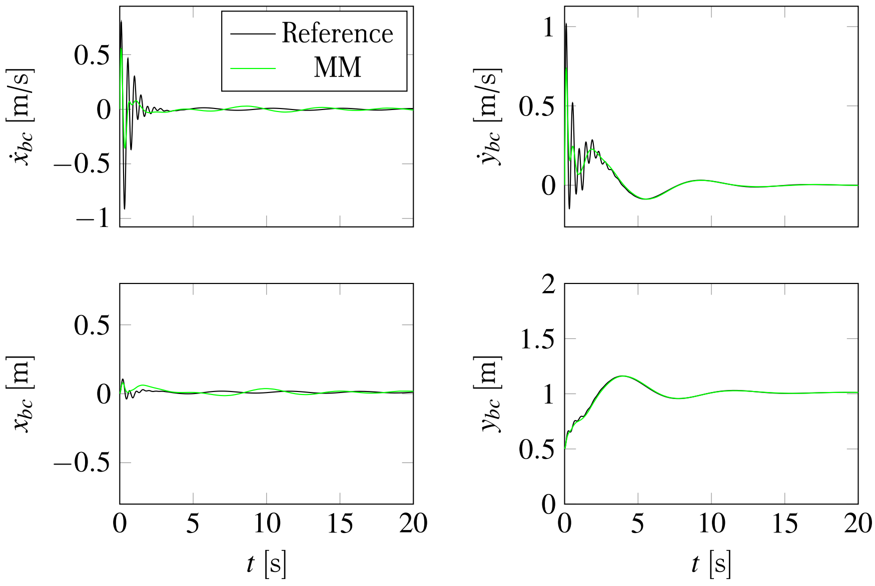


FIGURE 4.8: Simulation results for the triangle of masses system, for the bottom-center (bc) mass.

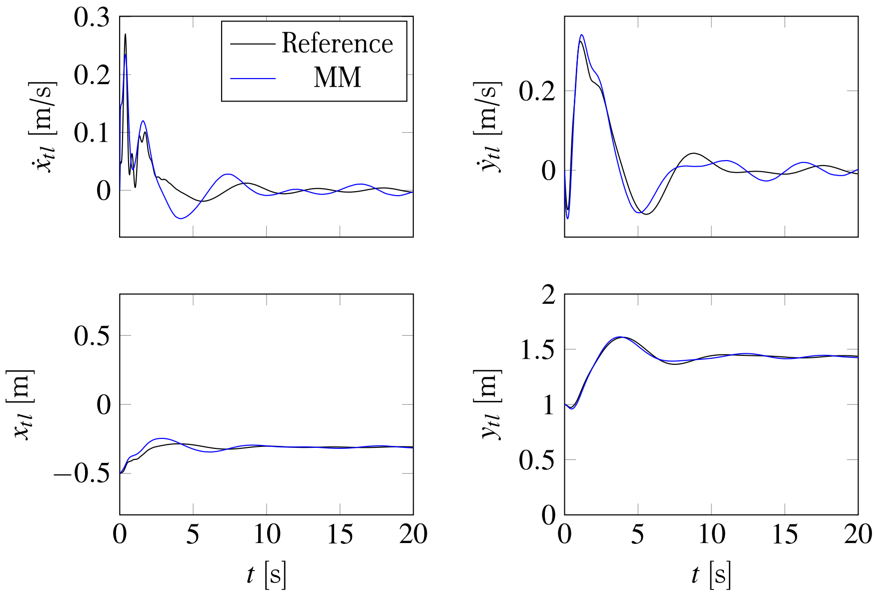


FIGURE 4.9: Simulation results for the triangle of masses system, for the top-left (tl) mass.

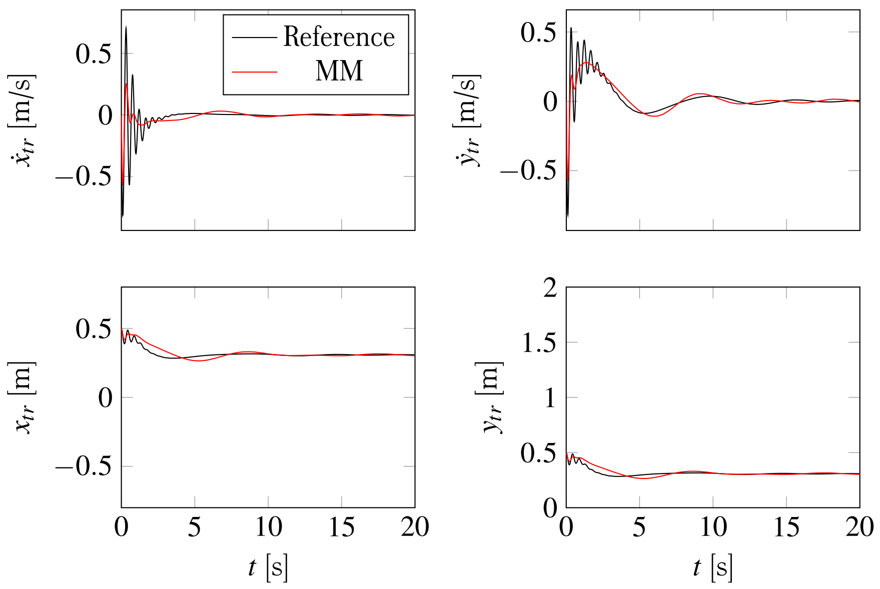


Figure 4.10: Simulation results for the triangle of masses system, for the top-right (tr) mass.

The stiffness $\sigma(\alpha)$ index shows that the considered system is sufficiently stiff, but nonetheless the separability one shows that it is suited for the partition, since its time scales are well separated.

4.4 Counterflow heat exchanger

In this example we consider a typical nonlinearity that is encountered in thermo-hydraulic systems. In addition, this example shows that DD scales well with the system dimension, and that CA is able to identify the same structural properties of the considered system in different conditions.

In particular, this example refers to a counterflow heat exchanger with two incompressible streams (Figure 4.11). Both streams and the interposed

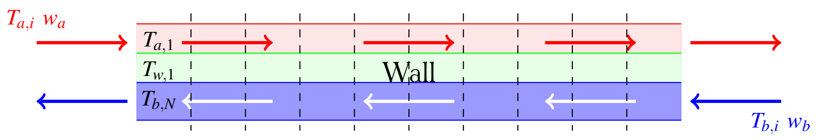


Figure 4.11: Counterflow heat exchanger scheme.

wall are spatially discretised with the finite volume approach, neglecting axial diffusion in the wall – as is common practice – and also in the streams, as zero-flow operation is not considered for simplicity. Taking ten volumes for both streams and the wall, with the same spatial division (again, for simplicity) leads to a nonlinear dynamic system of order 30, having as boundary conditions the four pressures at the stream inlets and outlets, and the two temperatures at the inlets. More precisely, the system is given by

$$\begin{cases} c_a \frac{M_a}{N} \dot{T}_{a,i} = w_a c_a \cdot (T_{a,i-1} - T_{a,i}) + \frac{G_a}{N} \cdot (T_{w,i} - T_{a,i}) \\ c_w \frac{M_w}{N} \dot{T}_{w,i} = -\frac{G_a}{N} \cdot (T_{w,i} - T_{a,i}) - \frac{G_b}{N} \cdot (T_{w,i} - T_{b,N-i+1}) \\ c_b \frac{M_b}{N} \dot{T}_{b,i} = w_b c_b \cdot (T_{b,i-1} - T_{b,i}) + \frac{G_b}{N} \cdot (T_{w,N-i+1} - T_{b,i}) \end{cases} \tag{4.3}$$

where T stands for temperature, w for mass flowrate, c for (constant) specific heat, M for mass, and G for thermal conductance; the a, b and w subscripts denote respectively the two streams and the wall, while $i \in [1,N]$ ($i = 0$ for boundary conditions) is the volume index, counted for both streams from inlet to outlet, the wall being enumerated like stream a.

The parameter values used in the example are reported in Table 4.5.

TABLE 4.5: Parameter values of Model (4.3).

Parameters					
N	10	M_b	1 kg	c_w	3500 J/(kg K)
$T_{a,\text{in}}$	323.15 K	M_w	10 kg	G_a	8000 W/K
$T_{b,\text{in}}$	288.15 K	c_a	4200 J/(kg K)	G_b	8000 W/K
M_a	0.1 kg	c_b	3500 J/(kg K)		

A parametric CA is performed so as to analyse the structure of the system, and Figure 4.12 shows its result. In particular, 95 cycles are present in the system.

The separability analysis evidences that there are at least a couple of points where the system can be separated. However, for $\alpha = 1.0$ there is only one point in which the system can be split, and it is between the 10-*th* and the 11-*th* variable. It is worth noticing that in this example, there is no neat physical separation between the dynamics, since they all belong to the same physical domain, and also to the same physical object. Separability analysis, however, can detect those structural properties of the system independently of its nature.

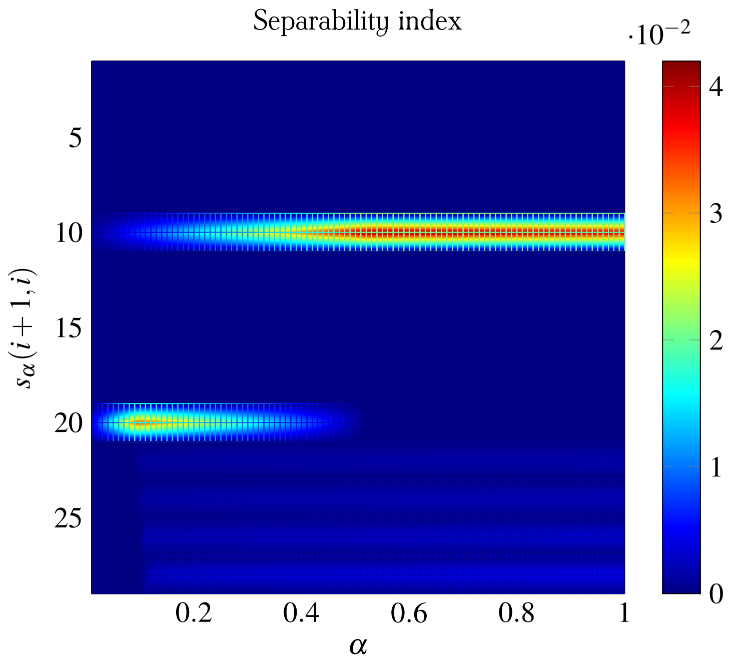


Figure 4.12: Separability analysis of the heat exchanger (4.3) with $N = 10$.

Hence, choosing $\alpha = 1.0$ CA leads to the following constraints:

$$T_{a,i}: \quad h \leq 0.008 \qquad \begin{aligned} T_{b,i}: &\quad h \leq 0.048 \\ T_{w,1,10}: &\quad h \leq 0.048 \\ T_{w,2,9}: &\quad h \leq 0.050 \\ T_{w,3,8}: &\quad h \leq 0.052 \\ T_{w,4,7}: &\quad h \leq 0.056 \\ T_{w,5,6}: &\quad h \leq 0.061 \end{aligned}$$

Choosing an integration step $h = 0.04$ yields a partition of the system that considers the $T_{a,i}$ as the fast while the $T_{b,i}$ and $T_{w,i}$ as the slow states. Figure 4.13 shows the simulation results — notice that the temperatures are reported with different scales.

Table 4.6 shows the simulation statistics for different integration methods. It is worth noticing that, since the complexity of IE is $\mathcal{O}(n^3)$, where n is the dimension of the model, integrating implicitly only $T_{a,i}$ instead of the whole model, reduces the computations from $30^3 = 27000$ to $10^3 = 1000$, leading to a significant improvement in terms of simulation efficiency. No-

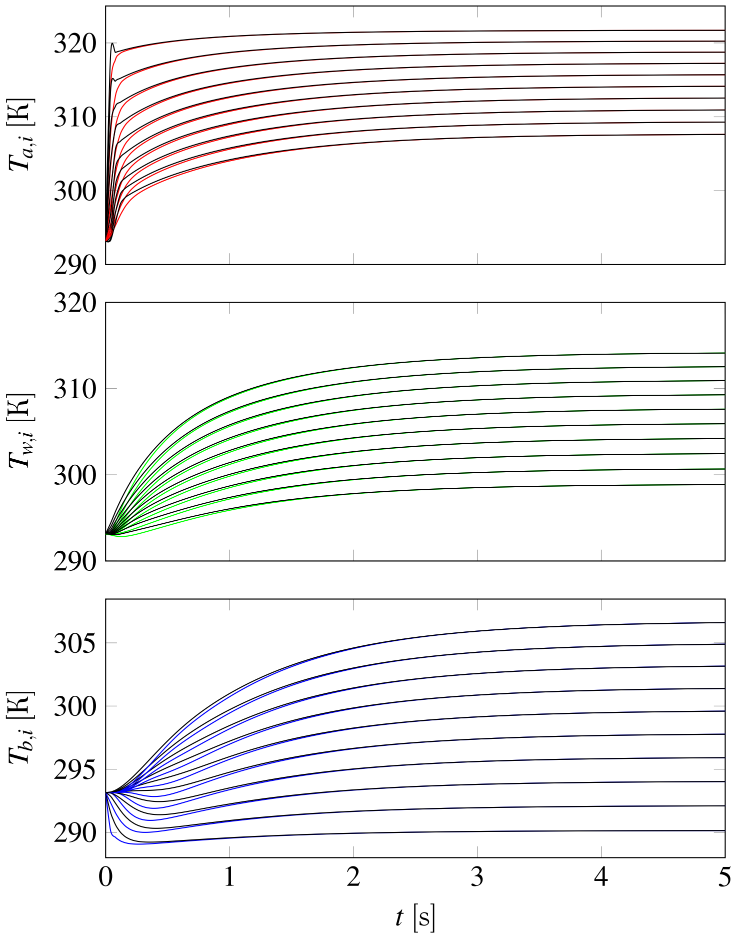


FIGURE 4.13: Simulation results of (4.3). Black lines represent the reference solution of the trajectories, while the coloured lines the mixed-mode ones.

tice also that the EE method needs a smaller step size ($h = 0.01$) for numerical stability reasons.

Table 4.6: Simulation statistics for Model (4.3) ($h = 0.04$).

	Mixed-mode	**BDF**	**IE**	**EE**
# Steps	**125**	260	**125**	500
# Function ev.	375	**296**	375	–
# Jacobian ev.	6	**5**	6	–
# Fun. ev. in Jac. ev.	**66**	150	186	–
# Newton iterations	**250**	292	**250**	–
# Newton fail	0	0	0	–
Accuracy	0.019	–	**0.018**	0.107
Sim time	**0.06s**	0.21s	0.12s	0.15s

Now, the indices proposed in Section 3.4 are here computed, yielding the following results—notice that also in this case, due to the nonlinearity of the system, σ_R cannot be computed.

$$\sigma(0.5) = 13.612, \quad s(0.5) = 0.954.$$

The stiffness $\sigma(\alpha)$ index shows that the considered system is sufficiently stiff, but the more interesting aspect is that the separability one shows that it is very suited for the partition, since its time scales are very well separated.

The presented examples have been kept as small as possible in order to improve results readability, but the method can be applied to larger models as well. For example, by changing in (4.3) the parameter N to 30, the model becomes of order 90. Thus, CA detects 585 cycles and the same parametric separability analysis can be performed.

In this case, the chosen integration step here is $h = 0.01$. The obtained simulation results are summarised in Table 4.7.

In this case the indices become

$$\sigma(0.5) = 9.771, \quad s(0.5) = 0.986,$$

thus, even if the order of the system is changed, but the system is actually the same, those structural indices have the same order of magnitude, and, what is more important are not changed too much, providing the same information.

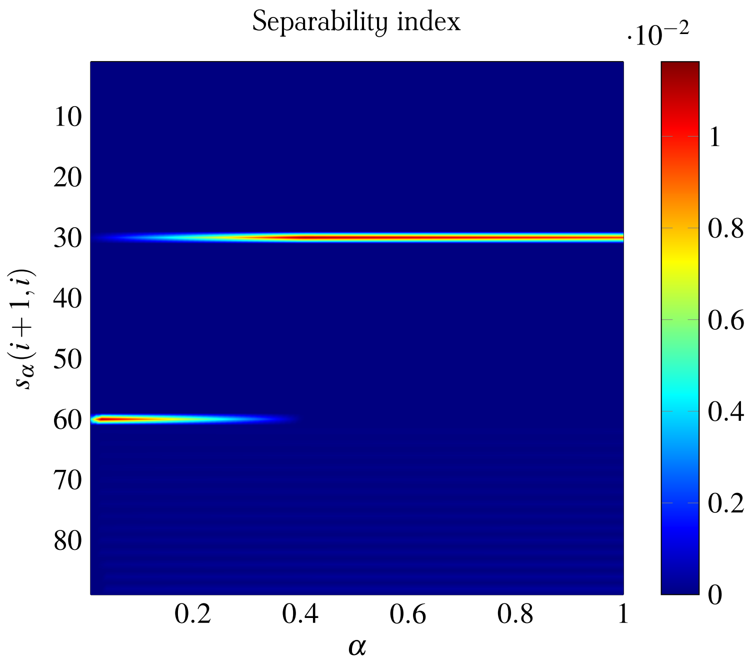


FIGURE 4.14: Separability analysis of the heat exchanger (4.3) with $N = 30$.

TABLE 4.7: Simulation statistics for Model (4.3) with $N = 30$.

	Mixed-mode	**BDF**	**IE**	**EE**
# Steps	500	**290**	500	5000
# Function ev.	1321	**323**	1348	–
# Jacobian ev.	24	**6**	24	–
# Fun. ev. in Jac. ev.	744	**540**	2184	–
# Newton iterations	820	**319**	847	–
# Newton fail	0	0	0	–
Accuracy	6.586	–	**6.566**	24.151
Sim time	**0.76s**	0.96s	1.34s	1.76s

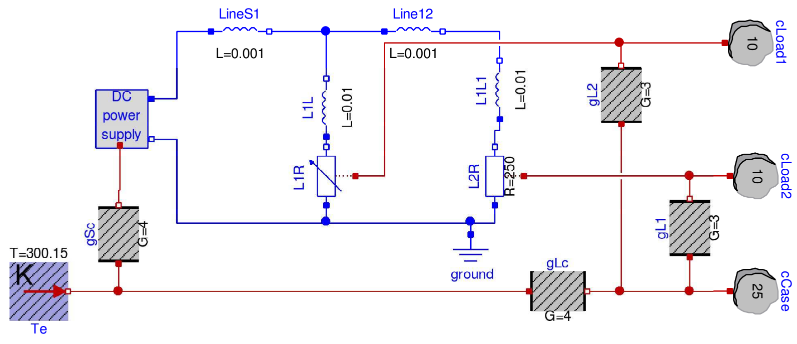


Figure 4.15: Modelica scheme of the model for the power supply with electric loads example.

4.5 Power supply with electric loads

This last example propose a realistic model of a power supply with electric loads. It is a representative example of typical uses of OOM in which different physical domains are involved – thus also different time scales – for which DD, and specifically CA, show more their effectiveness.

In this example we consider a system composed of a rectifier-based DC supply, modelled with enough detail to represent the output voltage ripple. This power supply is connected to a small network of electric loads; both the supply and the load components are heated by electrical phenomena, are characterised by convenient thermal capacities, and disperse heat toward an external environment with a prescribed temperature. The Modelica scheme of the model, that and around 150 scalar equations and after the manipulation contains 10 state variables, is shown in Figure 4.15.

More precisely, the example intends to show that one can use the single detailed model of the system to perform simulations focusing either on the electrical or thermal phenomena, which are apparently of interest for different control problems. Thanks to the proposed technique, in fact, the analyst has just to select the proper finest time scale for the study at hand, and the simulation environment will automatically refrain from wasting simulation effort in the representation of dynamics that are too fast to be relevant.

In the considered system there are apparently two well distinguished time scales—a fast one for electrical phenomena and a slow one for thermal phenomena. Applying CA, 18 dependency cycle are detected, leading to the

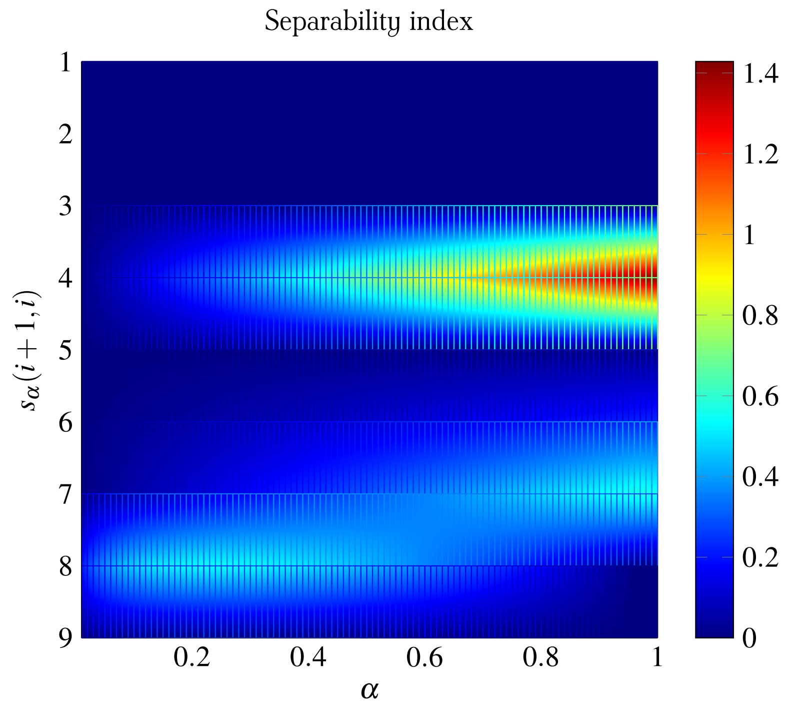


FIGURE 4.16: Separability analysis of the power supply with electric loads example.

following time scales (ordered by increasing time scale)

$$\begin{aligned}
&I_{L_1}: \quad h \le 2.183 \times 10^{-5} && T_{\text{load1}}: h \le 1.429 \\
&I_{L_2}: \quad h \le 4.764 \times 10^{-5} && T_{\text{load2}}: h \le 1.429 \\
&V_{C_1,\text{supply}}: \quad h \le 5.000 \times 10^{-5} && T_{\text{case}}: h \le 1.667 \\
&V_{C_2,\text{supply}}: \quad h \le 5.000 \times 10^{-5} && T_{\text{supply}}: h \le 2.236 \\
& && T_{\text{R1,supply}}: h \le 2.236 \\
& && T_{\text{R2,supply}}: h \le 2.236
\end{aligned} \tag{4.4}$$

Apparently, the state variables can be separated into two sets of slow and fast ones, as expected from physical considerations. Thus, by choosing an integration step of $h = 0.1$, we induce a partition of the two subsystems that can be integrated with the mixed-mode integration described in Section 3.5. The numerical results are represented in Figure 4.17, and compared with a more sophisticated integration method, i.e., LSODAR (short for Livermore Solver for Ordinary Differential equations, with Automatic method switch-

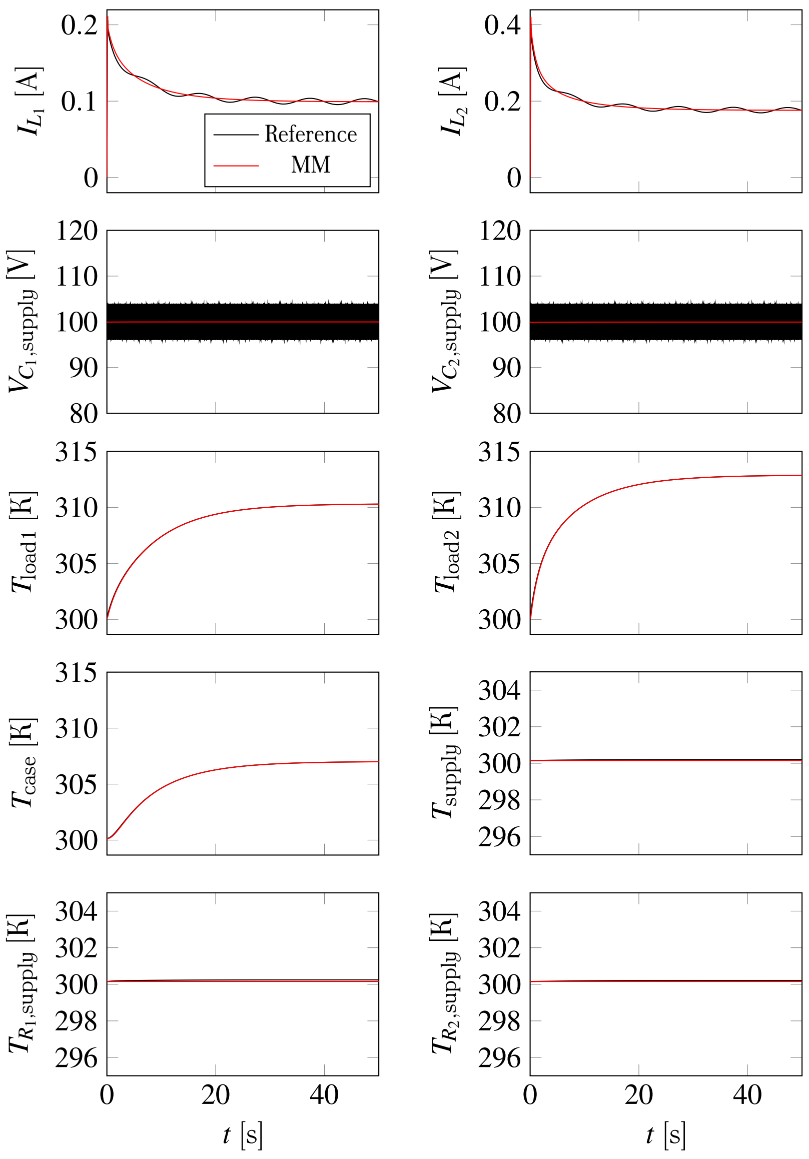


Figure 4.17: Results of the power supply with electric loads example.

ing for stiff and nonstiff problems, and with Root-finding). Other methods first-order methods (that are more comparable to the mixed-mode one) are here not considered since Explicit Euler, with the chosen integration step presents numerical instability, while Implicit Euler do not converge.

Simulation statistics of the considered integration methods are reported in Table 4.8, showing a significant improvement in terms of all the indices, and particularly for simulation time.

Table 4.8: Simulation statistics for the power supply with electric loads example.

	Mixed-mode	**LSODAR**
# Steps	500	128037
# Function ev.	1543	336217
# Jacobian ev.	25	15754
# Fun. ev. in Jac. ev.	125	–
# Newton iterations	1043	–
Accuracy	**0.047**	–
Sim time	**0.56s**	64.88s

In this example the structural indices become

$$\sigma(1.0) = 102413.091, \quad s(1.0) = 0.826,$$

showing that the considered system is highly stiff – fact that is also apparent from the time scales (4.4) identified by CA – thus indicating that a purely explicit method is not suitable for simulating efficiently the system. On the other hand, the separability index also shows that the system is suited to be partitioned, also accordingly to the separability analysis reported in Figure 4.16.

4.6 Discussion

After showing the examples, their collective outcome can be summarised as follows. First, when there is an evident dynamic separation in the system, the proposed technique finds it out without requiring *a priori* information on the part of the user. In other words, the validity and the usefulness of the technique are backed up by observing that the produced results are in accordance with intuition, when intuition can figure them out.

Also, and in some sense as a complement, the proposed indices allow to synthetically appreciate the possible internal model couplings that can be exploited via DD, even when these are not apparent at all.

Moreover, and specific to the use of the technique for mixed-mode integration, its characteristics are very suited to the typical studies that are required to really have simulation follow the life-cycle of the project, as envisaged in the introduction. On this final point, however, some more words are in order.

When a simulation study is required to answer a specific question, most frequently the focus is on part of the system, or – somehow equivalently – on part of the phenomenon occurring in it. In such a very frequent case, the rest of the system does not need to be simulated accurately, provided that the boundary conditions presented to the part that is relevant for the study, allow for a precise evaluation of the investigated quantities. In many situations of the type just mentioned, the interest of the analyst is on certain time scales on the system phenomena, and provided these are well reproduced, loosing faster behaviours is not only acceptable, but in fact necessary to achieve the desired performance. In fact, the same remark holds also for almost the totality of MOR techniques, where low-frequency approximations of the original model are the typical result, and the quality of a reduction is not evaluated in terms of time error – which is typically large due to the transients – but rather in terms of the H_∞-norm of the difference between the original and the reduced model, i.e., in the frequency domain. This is totally analogous to the proposed approach, where a good approximation is not strictly related to a small simulation error, but to a good representation of the time scales of interest.

Chapter 5

Model Order Reduction for Hybrid Systems

This chapter presents an approach to build a reduced order model for a Switched Affine (SA) system. The main idea is to reduce the SA system to an equivalent Switched Linear (SL) system with state reset, and then apply balanced truncation to the linear dynamic associated to each mode and appropriately redefine the reset maps so as to best reproduce the free evolution of the system output. A randomised approach is proposed to choose the order of the resulting reduced SL system in the case when the input is stochastic and one is interested in reproducing the output of the original SA system over a finite time-horizon. The performance of the approach is shown on a benchmark example. The results presented in this chapter come from [Papadopoulos and Prandini, 2014].

More specifically, this chapter deals with the problem of approximating a hybrid system by means of some simpler model, see e.g. [Girard and Pappas, 2007, Girard et al., 2010, Julius and Pappas, 2009, Mazzi et al., 2008, Petreczky and Vidal, 2007, Shaker and Wisniewski, 2012] to cite a few. Hybrid systems are characterised by intertwined continuous and discrete dynamics, and are suitable for modelling complex, large scale systems, see e.g. [Lunze and Lamnabhi-Lagarrigue, 2009] for an overview of applications of hybrid models to various domains. The study of hybrid systems is more challenging than for other classes of systems, and many problems still lack an effective solution. In particular, this is the case for the design of simple models approximating a hybrid system.

In this chapter we focus on the design of an approximate model for a switched affine system. More specifically, the goal is to obtain a simpler model of the system which can be effectively used for system verification over some finite horizon T.

Verification of properties related to the hybrid system evolution, like, e.g., safety and reach/avoid properties, are typically addressed through numerical methods that scales badly with the state-space dimension, [Abate et al., 2007, 2010, Frehse, 2005, Girard and Guernic, 2008, Kurzhanski and Varaiya, 2005, Mitchell, 2002, Prandini and Hu, 2006, Tomlin et al., 2003]. The aim of the approximation is then to build a model that mimics the behaviour of the original system and that can be used in place of the system to scale-up numerical methods for the verification of the property of interest. When the hybrid system input is stochastic, the notion of approximate simulation introduced in [Julius and Pappas, 2009] can be used to quantify the model performance.

The approach proposed in this chapter is inspired by [Mazzi et al., 2008], where a balanced truncation is adopted for reducing the order of the linear dynamics governing the evolution of the continuous component of an hybrid system. The main advances with respect to [Mazzi et al., 2008] are

- the extension to the class of switched affine systems;
- the introduction of a novel method for defining the state reset map that provides better performance than the one adopted in Mazzi et al. [2008]; and
- the introduction of a procedure to select the order of the reduced model based on a randomised approach, when the input is stochastic.

Note that, differently from most of the works on switched affine/linear system reduction, [Petreczky et al., 2012, Shaker and Wisniewski, 2012], the transitions between discrete modes in the considered switched affine system class are determined by an endogenous signal that depends on the continuous state evolution, which makes the approximation problem more challenging.

The rest of the chapter is organised as follows. We start with a brief review of balanced truncation for linear systems in Section 5.1. We then describe the considered switched affine system class (Section 5.2) and the proposed model reduction method (Section 5.3). The randomised approach to model order selection is illustrated in Section 5.4, whilst a numerical example showing the performance of the approach is presented in Section 5.5.

5.1 Balanced truncation for linear systems: a brief review

There is a vast literature on model order reduction for linear systems (see e.g. [Antoulas, 2005, Gugercin and Antoulas, 2004, Moore, 1981]). In particular, balanced truncation is one of the more popular techniques, and the one adopted here for reducing the order of the continuous dynamics within each mode. The balanced truncation method rests on the representation of the system in the balanced realization form, which is recalled next for the purpose of self-containedness.

Let $\mathscr{S}$ be a continuous-time linear time-invariant dynamic system described in state-space form through a 4-tuple of matrices $(\mathscr{A},\mathscr{B},\mathscr{C},\mathscr{D})$:

$$\mathscr{S}:\begin{pmatrix}\mathscr{A} & \mathscr{B}\\ \mathscr{C} & \mathscr{D}\end{pmatrix}.$$

Suppose that $\mathscr{S}$ is controllable, observable and asymptotically stable.

Definition 5.1.1 (Balanced system). System $\mathscr{S}$ is *balanced* if $\mathscr{W}_c=\mathscr{W}_o$, where

$$\mathscr{W}_c=\int_0^\infty e^{\mathscr{A}\tau}\mathscr{B}\mathscr{B}^T e^{\mathscr{A}^T\tau}\,\mathrm{d}\tau$$
$$\mathscr{W}_o=\int_0^\infty e^{\mathscr{A}^T\tau}\mathscr{C}^T\mathscr{C}e^{\mathscr{A}\tau}\,\mathrm{d}\tau$$

are, respectively, the infinite *controllability* and *observability Gramians* of $\mathscr{S}$. Furthermore, $\mathscr{S}$ is *principal-axis balanced* if $\mathscr{W}_c=\mathscr{W}_o=\Sigma$, with

$$\Sigma=\operatorname{diag}\{\sigma_1,\sigma_2,\dots,\sigma_n\},$$

where σ_i are the Hankel singular values of $\mathscr{S}$, listed in decreasing order.

The problem of finding the balanced realization of a system is equivalent to that of determining a balancing transformation matrix T such that

$$\begin{cases}W_c=T\mathscr{W}_cT^*\\ W_o=T^{-*}\mathscr{W}_oT^{-1}\end{cases}\Rightarrow\ W_cW_o=T\left(\mathscr{W}_c\mathscr{W}_o\right)T^{-1}=\Sigma^2,$$

where T^* denotes the Hermitian adjoint of T, which, in turn, reduces to solving the following minimization problem [Antoulas, 2005]

$$\min_T \operatorname{tr}\left[T\mathscr{W}_cT^*+T^{-*}\mathscr{W}_oT^{-1}\right]=2\operatorname{tr}\{\Sigma\}. \tag{5.1}$$

The system in the balanced state-space form is then obtained by applying the transformation matrix T, i.e.,

$$S:\begin{pmatrix}A & B\\ C & D\end{pmatrix}=\begin{pmatrix}T\mathscr{A}T^{-1} & T\mathscr{B}\\ \mathscr{C}T^{-1} & \mathscr{D}\end{pmatrix}.$$

The idea of the balanced truncation method is that in the balanced realization the state variables are ordered by decreasing importance as for their contribution to the input/output map, so that one can decompose the state vector (and the system) into two parts and neglect that with lowest importance. Formally, vector x is separated into two components

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad S : \begin{pmatrix} A_{11} & A_{12} & B_1 \\ A_{21} & A_{22} & B_2 \\ C_1 & C_2 & D \end{pmatrix}.$$

with $x_1 \in \mathbb{R}^{n_r}$ and $x_2 \in \mathbb{R}^{n-n_r}$. Correspondingly,

$$\Sigma = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix},$$

and if Σ_1 and Σ_2 do not contain any common element, then, the matrices A_{ii} $(i = 1, 2)$ are asymptotically stable [Liu and Anderson, 1989].

A reduced order model S_r of the system can then be obtained by setting $x_2 = 0$ and eliminating its contribution, thus getting:

$$S_r : \begin{pmatrix} A_r & B_r \\ C_r & D_r \end{pmatrix} = \begin{pmatrix} A_{11} & B_1 \\ C_1 & D \end{pmatrix}.$$

Alternatively, one can set $\dot{x}_2 = 0$, thus obtaining

$$S_r : \begin{pmatrix} A_r & B_r \\ C_r & D_r \end{pmatrix} = \begin{pmatrix} A_{11} - A_{12}A_{22}^{-1}A_{21} & B_1 - A_{12}A_{22}^{-1}B_2 \\ C_1 - C_2A_{22}^{-1}A_{21} & D - C_2A_{22}^{-1}B_2 \end{pmatrix}. \tag{5.2}$$

An estimate of the neglected state x_2 is then given by

$$\widehat{x}_2 = -A_{22}^{-1}A_{21}x_1 - A_{22}^{-1}B_2u, \tag{5.3}$$

which corresponds to the condition $\dot{x}_2 = 0$. If Σ_1 and Σ_2 do not contain any common element, then, S_r is asymptotically stable, controllable and observable [Liu and Anderson, 1989]. Moreover, the static gain of S_r is equal to that of the original system S.

In order to select the order of the reduced model, one can choose $\gamma \in [0, 1]$ and set

$$n_r = \min\{i \in \{1, 2, \dots, n\} : \psi(i) < \gamma\},$$

where $\psi : \{1, 2, \dots n\} \to [0, 1)$ is defined based on the Hankel singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$ of system S as follows:

$$\psi(i) = 1 - \frac{\sum_{j=1}^{i} \sigma_j}{\sum_{j=1}^{n} \sigma_j}. \tag{5.4}$$

The bound γ can be used as a knob to control the tradeoff between the dimension of the reduced state and the quality of the approximation.

Approximation by balanced truncation preserves stability and the difference between system S and its reduced model S_r has its $\mathscr{H}_\infty$-norm bounded by the sum of the neglected Hankel singular values as follows:

$$\|S - S_r\|_{\mathscr{H}_\infty} \leq 2\,\mathrm{tr}\{\Sigma_2\}\,. \tag{5.5}$$

5.2 Modeling framework

We consider the class of Switched Affine (SA) systems, whose evolution is characterised through a discrete state component q_a taking values in $Q = \{1, 2, \dots, m\}$ and a continuous component $\xi_a \in \Xi_a = \mathbb{R}^n$ evolving according to an affine dynamics that depends on the operating mode q_a. Correspondingly, the output $y_a \in Y_a = \mathbb{R}^p$ is an affine function of the state and the input $u \in U = \mathbb{R}^m$ that depends on q_a as well. In formulas:

$$\begin{cases} \dot{\xi}_a(t) = \mathscr{A}_{q_a}\xi_a(t) + \mathscr{B}_{q_a}u(t) + f_{q_a} \\ y_a(t) = \mathscr{C}_{q_a}\xi_a(t) + g_{q_a}. \end{cases} \tag{5.6}$$

A collection of polyhedra $\{Dom_{a,i} \subseteq Y_a \times U,\, i \in Q\}$ is given, which covers the whole set $Y_a \times U$[1]. Each polyhedron $Dom_{a,i}$ is defined through a system of r_i linear inequalities:

$$Dom_{a,i} = \{(y_a, u) \in Y_a \times U : G_i^{y_a} y_a + G_i^u u \leq G_i\},$$

with $G_i^{y_a} \in \mathbb{R}^{r_i \times p}$, $G_i^u \in \mathbb{R}^{r_i \times m}$ and $G_i \in \mathbb{R}^{r_i}$.
The system evolves according to the dynamics associated with mode i as long as (ξ_a, u) is such that (y_a, u) keeps evolving within $Dom_{a,i}$ and commute to the dynamics associated with $j \in Q$ as soon as (y_a, u) exits $Dom_{a,i}$ and enters into $Dom_{a,j}$.

Remark. $Dom_{a,i}$ appears to be a function of both y_a and u. However, if $G_i^u = 0$, then, the dependence on u is not present. Furthermore, those cases when the transition condition depends on the whole state ξ_a can be reframed in our setting by including ξ_a in the output variables.

Remark. Note that if $\{Dom_{a,i}, i \in Q\}$ is a polyhedral subdivision of $Y_a \times U$ (i.e., a finite collection of polyhedra on $Y_a \times U$ such that $\cup_{i \in Q} Dom_{a,i} = Y_a \times U$, each polyhedron $Dom_{a,i}$ is of dimension $p + m$, and the intersection $Dom_{a,i} \cap Dom_{a,j}$, $i \neq j$, is either empty or a common proper face of both polyhedra), then, the SA system reduces to a piecewise affine system.

[1] $\cup_{i \in Q} Dom_{a,i} = Y_a \times U$

5.3 System reduction

In this section, we introduce a procedure for designing a reduced order model of the SA system (5.6) that tries to best reproduce its output y_a. The proposed procedure rests on Assumption 1 below, and is based on the following key steps:

- reformulation of the SA system as a Switched Linear (SL) system with state reset;
- model reduction of the SL system through balanced truncation of the continuous dynamics and definition of appropriate state reset maps when a mode transition occurs;
- reconstruction of the output of the SA system based on the reduced SL system.

Assumption 1. For any $i \in Q$, matrix $\mathscr{A}_i$ is Hurwitz, $(\mathscr{A}_i, \mathscr{B}_i)$ is controllable, and $(\mathscr{A}_i, \mathscr{C}_i)$ is observable.

In Section 5.4, a randomised method is then described for selecting the order of the reduced order model of the SA system when the input u is stochastic and the goal is verifying a finite horizon property that depends on the behaviour of the SA system output y_a along the time horizon T.

5.3.1 Reformulation of the SA system as a SL system with state reset

We next build a SL system with state reset that is equivalent to the original SA system, in that (ξ_a, q_a) and y_a can be recovered exactly from the state and output variables of such a system.

Let $\xi \in \Xi = \Xi_a$ evolve according to a linear dynamics that depends on the operating mode $q \in Q$ as follows:

$$\begin{cases} \dot{\xi}(t) = \mathscr{A}_q \xi(t) + \mathscr{B}_q u(t) \\ y(t) = \mathscr{C}_q \xi(t) \end{cases} \tag{5.7}$$

where $y \in Y = Y_a$.

Set $\bar{y}_{a,q} = \mathscr{C}_q \bar{\xi}_{a,q} + g_q$, where $\bar{\xi}_{a,q} = -\mathscr{A}_q^{-1} f_q$, with $\mathscr{A}_q$ invertible by Assumption 1. A transition from mode $i \in Q$ to mode $j \in Q$ occurs as soon as $(y + \bar{y}_{a,i}, u)$ exits Dom_i and enters Dom_j, where $Dom_q = Dom_{a,q}$, $q \in Q$.

When a discrete transition from mode $i \in Q$ to mode $j \in Q$ occurs at time t^-, then, ξ is reset as follows

$$\xi(t) = \xi(t^-) + \bar{\xi}_{a,i} - \bar{\xi}_{a,j}. \tag{5.8}$$

Proposition 5.3.1. Suppose that the SA and SL systems are initialised with $\xi_a(0) = \xi_{a,0}$, $q_a(0) = q_{a,0}$, and $\xi(0) = \xi_{a,0} - \bar{\xi}_{a,q_{a,0}}$, $q(0) = q_{a,0}$, respectively, and are both fed by the same input $u(t)$, $t \in [0,T]$. Then, the execution of ξ_a, q_a and y_a over $[0,T]$ can be recovered from those of ξ, q and y as follows:

$$\begin{aligned} q_a(t) &= q(t) \\ \xi_a(t) &= \xi(t) + \bar{\xi}_{a,q(t)} \\ y_a(t) &= y(t) + \bar{y}_{a,q(t)}. \end{aligned} \tag{5.9}$$

Proof. The result immediately follows by observing that $\bar{\xi}_{a,q}$ and $\bar{y}_{a,q}$ are the state and output equilibria of system (5.6) associated with $u = 0$. □

Remark. Note that the reset condition in (5.8) is such that variable ξ_a reconstructed from ξ according to (5.9) is continuous. Continuity of ξ_a is generally not guaranteed if ξ is approximated through a reduced order model of the SL system.

5.3.2 Reduction of the SL system

A reduced order model of the SL system with reset defined before can be obtained by applying balanced truncation (5.2) to each single linear dynamics in (5.7). This is is in order to best reproduce the evolution of the output y within a fixed mode, and also the discrete transitions between modes, since they are defined through a condition involving y.
We associate to each mode $q_r \in Q$ a reduced model of order $n_{r,q} \le n$:

$$\begin{cases} \dot{x}_{r,q_r}(t) = A_{r,q_r} x_{r,q_r}(t) + B_{r,q_r} u(t) \\ \hat{y}(t) = C_{r,q_r} x_{r,q_r}(t) + D_{r,q_r} u(t) \end{cases} \tag{5.10}$$

and define transitions between modes, say from mode i to mode j, by evaluating when $(\hat{y} + \bar{y}_{a,i}, u)$ exits from domain Dom_i and enters into Dom_j. As for the state reset map (5.8) associated with a transition from mode $i \in Q$ to mode $j \in Q$, we shall reformulate it in the following form

$$x_{r,j}(t) = L_{ji} x_{r,i}(t^-) + M_{ji} u(t^-) + N_{ji}. \tag{5.11}$$

where $x_{r,i}(t^-) \in \mathbb{R}^{n_{r,i}}$, $x_{r,j}(t) \in \mathbb{R}^{n_{r,j}}$, and L_{ji}, M_{ji}, N_{ji} are matrices of appropriate dimensions.

We shall present next two methods to define matrices L_{ji}, M_{ji}, N_{ji}. In both of them we shall refer to the following variables:

1. the estimate $\hat{x}_i$ of the state of the SL system dynamics associated with mode $i \in Q$ in balanced form. $\hat{x}_i$ is reconstructed from the reduced state $x_{r,i}$ according to:

$$\begin{aligned}\hat{x}_i &= \begin{bmatrix} x_{r,i} \\ -A_{i,22}^{-1}A_{i,21}x_{r,i} - A_{i,22}^{-1}B_{i,2}u \end{bmatrix} \\ &= \begin{bmatrix} I_{n_{r,i}\times n_{r,i}} \\ -A_{i,22}^{-1}A_{i,21} \end{bmatrix} x_{r,i} + \begin{bmatrix} \mathbf{0}_{n_{r,i}\times 1} \\ -A_{i,22}^{-1}B_{i,2} \end{bmatrix} u \end{aligned} \tag{5.12}$$

   Expression (5.12) can be rewritten in compact form as

$$\hat{x}_i = H_i x_{r,i} + K_i u, \tag{5.13}$$

   with

$$H_i = \begin{bmatrix} I_{n_{r,i}\times n_{r,i}} \\ -A_{i,22}^{-1}A_{i,21} \end{bmatrix} \qquad K_i = \begin{bmatrix} \mathbf{0}_{n_{r,i}\times 1} \\ -A_{i,22}^{-1}B_{i,2} \end{bmatrix}$$

   where $I_{n_{r,i}\times n_{r,i}}$ is an identity matrix of dimension $n_{r,i} \times n_{r,i}$, and $\mathbf{0}_{n_{r,i}\times 1}$ is a zero vector of $n_{r,i}$ elements;

2. the estimate $\hat{\xi}_i$ of the state of the SL system associated with mode $i \in Q$:

$$\hat{\xi}_i = T_i^{-1}\hat{x}_i, \tag{5.14}$$

   obtained from $\hat{x}_i$ through the balanced transformation matrix T_i.

We are now in a position to defined the reduced state reset maps for a transition from $i \in Q$ at time t^- to $j \in Q$ at time t.

a) reset map proposed in [Mazzi et al., 2008]:

We start setting

$$x_{r,j}(t) = E_{n_{r,j}}\hat{x}_j(t)$$

where $E_{n_{r,j}}$ is a matrix that extracts the first $n_{r,j}$ rows from $\hat{x}_j(t)$, being $n_{r,j}$ the dimension of $x_{r,j}$ in mode j. Now,

$$\begin{aligned}\hat{x}_j(t) &= T_j\hat{\xi}_j(t) = T_j\left(\hat{\xi}_i(t^-) + \bar{\xi}_{a,i} - \bar{\xi}_{a,j}\right) \\ &= T_j\left(T_i^{-1}\hat{x}_i(t^-) + \bar{\xi}_{a,i} - \bar{\xi}_{a,j}\right) \\ &= T_j\left(T_i^{-1}H_i x_{r,i}(t^-) + T_i^{-1}K_i u(t^-) + \bar{\xi}_{a,i} - \bar{\xi}_{a,j}\right),\end{aligned}$$

so that

$$x_{r,j}(t) = E_{n_{r,j}} T_j \left(T_i^{-1} H_i x_{r,i}(t^-) + T_i^{-1} K_i u(t^-) + \bar{\xi}_{a,i} - \bar{\xi}_{a,j} \right). \tag{5.15}$$

By direct comparison of this expression with (5.11), we get the reset matrices:

$$\begin{aligned} L_{ji} &= E_{n_{r,j}} T_j T_i^{-1} H_i \\ M_{ji} &= E_{n_{r,j}} T_j T_i^{-1} K_i \\ N_{ji} &= E_{n_{r,j}} T_j \left(\bar{\xi}_{a,i} - \bar{\xi}_{a,j} \right). \end{aligned}$$

According to a similar reasoning, the system is initialised as follows

$$\begin{aligned} q_r(0) &= q_a(0) = q_0 \\ x_{r,q_0}(0) &= E_{n_{r,q_0}} T_{q_0} \left(\xi_a(0) - \bar{\xi}_{a,q_0} \right), \end{aligned}$$

with the understanding that $(y_a(0), u(0))$ is an interior point of Dom_{a,q_0} for any admissible $u(0)$.

b) reset map best reproducing the output free evolution:

Model reduction techniques for asymptotically stable linear systems aim at finding a model that best reproduce the forced response of the system, while neglecting the free evolution. This motivates the introduction of an alternative reset map that minimises the norm-2 error when reproducing the free evolution of the output y. More precisely, we set

$$x_{r,j} = \Psi_j \hat{\xi}_j$$

and choose Ψ_j so as to minimise

$$J = \int_0^{+\infty} \| y_{fr,j}(t) - \hat{y}_{fr,j}(t) \|^2 \, \mathrm{d}t, \tag{5.16}$$

where $y_{fr,j}$ and $\hat{y}_{fr,j}$ respectively denote the free evolution of the original linear dynamics (5.7) initialised with $\hat{\xi}_j$ and that of the reduced order dynamics (5.10) initialised with $x_{r,j} = \Psi_j \hat{\xi}_j$. The solution to this optimization problem can be found analytically as shown in Proposition 5.3.2.

Proposition 5.3.2. Matrix Ψ_j minimizing (5.16) for any $\hat{\xi}_j$ is given by

$$\Psi_j = \mathscr{W}_{r,o,j}^{-1} \mathscr{W}_{\times,j}.$$

where

$$\mathscr{W}_{r,o,j} = \int_0^{+\infty} (e^{A_{r,j}t})^T C_{r,j}^T C_{r,j} e^{A_{r,j}t} \, \mathrm{d}t \tag{5.17}$$

$$\mathscr{W}_{\times,j} = \int_0^{+\infty} (e^{A_j t})^T C_j^T C_{r,j} e^{A_{r,j}t} \, \mathrm{d}t \tag{5.18}$$

and invertibility of the infinite observability Gramian $\mathscr{W}_{r,o,j}$ is guaranteed by the observability of the reduced order model (5.10) with $q = j$.

Proof. The cost function J can be written as

$$\begin{aligned} J = & \int_0^{+\infty} (C_j e^{A_j t} \hat{\xi}_j - C_{r,j} e^{A_{r,j} t} x_{r,j})^T (C_j e^{A_j t} \hat{\xi}_j - C_{r,j} e^{A_{r,j} t} x_{r,j}) \, \mathrm{d}t \\ = & x_{r,j}^T \mathscr{W}_{r,o,j} x_{r,j} - 2 x_{r,j} \mathscr{W}_{\times,j} \hat{\xi}_j + \hat{\xi}_j^T \mathscr{W}_{o,j} \hat{\xi}, \end{aligned}$$

where we set

$$\mathscr{W}_{o,j} = \int_0^{+\infty} (e^{A_j t})^T C_j^T C_j e^{A_j t} \, \mathrm{d}t.$$

Then, the minimum of J as a function of $x_{r,j}$ satisfies

$$\frac{\partial J}{\partial x_{r,j}} = 2 \mathscr{W}_{r,o,j} x_{r,j} - 2 \mathscr{W}_{\times,q'} \hat{\xi}_j = 0$$

yielding the reset map

$$x_{r,j} = \mathscr{W}_{r,o,j}^{-1} \mathscr{W}_{\times,j} \hat{\xi}_j.$$

□

Note that the quantity (5.17) is the solution of the Lyapunov equation

$$A_{r,j} \mathscr{W}_{r,o,j} + \mathscr{W}_{r,o,j} A_{r,j}^T + C_{r,j}^T C_{r,j} = 0,$$

while quantity (5.18) is the solution of the Sylvester equation

$$A_{r,j}^T \mathscr{W}_{\times,j} + \mathscr{W}_{\times,j} A_j + C_{r,j}^T C_j = 0.$$

Given Ψ_j, the following derivations

$$\begin{aligned} x_{r,j}(t) = & \Psi_j \hat{\xi}_j(t) = \Psi_j \left(\hat{\xi}_i(t^-) + \bar{\xi}_{a,i} - \bar{\xi}_{a,j} \right) = \\ = & \Psi_j \left(T_i^{-1} \hat{x}_i(t^-) + \bar{\xi}_{a,i} - \bar{\xi}_{a,j} \right) \\ = & \Psi_j \left(T_i^{-1} H_i x_{r,i}(t^-) + T_i^{-1} K_i u(t^-) + \bar{\xi}_{a,i} - \bar{\xi}_{a,j} \right) \end{aligned} \tag{5.19}$$

using the reset map (5.8) and equations (5.14) and (5.13) lead to the following definition of the matrices in the reset map (5.11):

$$\begin{aligned} L_{ji} &= \Psi_j T_i^{-1} H_i, \\ M_{ji} &= \Psi_j T_i^{-1} K_i, \\ N_{ji} &= \Psi_j \left(\bar{\xi}_{a,i} - \bar{\xi}_{a,j} \right). \end{aligned}$$

As for the system initialization, we set

$$\begin{aligned} q_r(0) &= q_a(0) = q_0 \\ x_{r,q_0}(0) &= \Psi_j \left(\xi_a(0) - \bar{\xi}_{a,q_0}\right). \end{aligned}$$

A different reset map that accounts for the switching nature of the system can be obtained by considering a finite horizon $[0, \tau]$ for the minimization of the free evolution error:

$$J = \int_0^\tau \|y_{fr,j}(t) - \hat{y}_{fr,j}(t)\|^2 \, \mathrm{d}t.$$

The resulting optimal $\Psi_j^{(\tau)}$ can be computed through the following expression

$$\Psi_j^{(\tau)} = \mathscr{W}_{r,o,j}^{-1}(\tau)\mathscr{W}_{\times,j}(\tau),$$

with

$$\begin{aligned} \mathscr{W}_{r,o,j}(\tau) &= \int_0^\tau (e^{A_{r,j}t})^T C_{r,j}^T C_{r,j} e^{A_{r,j}t} \, \mathrm{d}t \\ \mathscr{W}_{\times,j}(\tau) &= \int_0^\tau (e^{A_j t})^T C_j^T C_{r,j} e^{A_{r,j}t} \, \mathrm{d}t, \end{aligned}$$

the proof being analogous to that in the infinite horizon case. The above finite horizon quantities can be computed as

$$\begin{aligned} \mathscr{W}_{r,o,j}(\tau) &= \mathscr{W}_{r,o,j} - \int_\tau^{+\infty} (e^{A_{r,j}t})^T C_{r,j}^T C_{r,j} e^{A_{r,j}t} \, \mathrm{d}t = \mathscr{W}_{r,o,j} - \mathscr{W}_{r,o,j}^{(\tau,\infty)}, \\ \mathscr{W}_{\times,j}(\tau) &= \mathscr{W}_{\times,j} - \int_\tau^{\infty} (e^{A_j t})^T C_j^T C_{r,j} e^{A_{r,j}t} \, \mathrm{d}t = \mathscr{W}_{\times,j} - W_{\times,j}^{(\tau,\infty)}, \end{aligned}$$

where the quantities $\mathscr{W}_{r,o,j}^{(\tau,\infty)}$ and $\mathscr{W}_{\times,j}^{(\tau,\infty)}$ can be obtained respectively as the solution of the Lyapunov and Sylvester equations

$$\begin{aligned} A_{r,j}\mathscr{W}_{r,o,j}^{(\tau,\infty)} + \mathscr{W}_{r,o,j}^{(\tau,\infty)} A_{r,j}^T + \left(e^{A_{r,j}\tau}\right)^T C_{r,j}^T C_{r,j} e^{A_{r,j}\tau} &= 0, \\ A_{r,j}^T \mathscr{W}_{\times,j}^{(\tau,\infty)} + \mathscr{W}_{\times,j}^{(\tau,\infty)} A_j + \left(e^{A_{r,j}\tau}\right)^T C_{r,j}^T C_j e^{A_j \tau} &= 0, \end{aligned}$$

which are identical to the previous ones except for the fact that C_j and $C_{r,j}$ are replaced by $C_j e^{A_j \tau}$ and $C_{r,j} e^{A_{r,j}\tau}$, respectively. Note that well-posedness of the above equations is guaranteed by the fact that A_j and $A_{r,j}$ are Hurwitz.

The matrices in the reset map (5.11) and the system initialization are given by:

$$\begin{aligned} L_{ji} &= \Psi_j^{(\tau)} T_i^{-1} H_i, \\ M_{ji} &= \Psi_j^{(\tau)} T_i^{-1} K_i, \\ N_{ji} &= \Psi_j^{(\tau)} \left(\bar{\xi}_{a,i} - \bar{\xi}_{a,j}\right) \end{aligned}$$

and

$$q_r(0) = q_a(0) = q_0$$
$$x_{r,q_0}(0) = \Psi_j^{(\tau)}\left(\xi_a(0) - \bar{\xi}_{a,q_0}\right).$$

The choice for τ depends on the settling times of the different mode dynamics. A sensible choice is suggested in the numerical example of Section 5.5.

5.3.3 Reconstruction of the SA system output

The output of the SA system is reconstructed based on (5.9) using the output $\hat{y}$ of the SL reduced system as an estimate of the output y of the SL system:

$$\hat{y}_a(t) = \hat{y}(t) + \bar{y}_{a,q_r(t)}.$$

5.4 A randomised method for model order selection

In this section, a randomised method is described for selecting the order of the reduced order model of the SA system when the input u is stochastic and the goal is verifying a finite horizon property that depends on the behavior of the SA system output y_a along the time horizon T.

The proposed method involves feeding the reduced model and the system with some realizations of the stochastic input. This in practice means that either the distribution of the input is known, or some of its realizations are available as historical time series.

As discussed in Section 5.1, a sensible way of choosing the order of the reduced model for a linear system is setting a threshold value for the ψ function in (5.4) and then define the order accordingly. By following the same logic as in Mazzi et al. [2008], a function $\psi_q : \{1, 2, \dots n\} \to [0, 1)$ can then be considered for each mode $q \in Q$

$$\psi_q(i) = 1 - \frac{\sum_{j=1}^{i} \sigma_{j,q}}{\sum_{j=1}^{n} \sigma_{j,q}},$$

where $\sigma_{1,q} \geq \sigma_{2,q} \geq \cdots \geq \sigma_{n,q}$ are the Hankel singular values of the SL system dynamics (5.7) in mode q, and the order of the model (5.10) defining the reduced SL system can be set according to

$$n_{r,q} = \min\{i \in \{1, 2, \dots, n\} : \psi_q(i) < \gamma\},$$

for each $q \in Q$.

Our goal is now to introduce a method for choosing an appropriate value for γ.
To this purpose, we denote by $\hat{y}_a^\gamma$ the estimate of y_a obtained through the reduced SL system with parameter γ, and by Γ the (finite) set of threshold values for γ, those that result in a different choice for $\{n_{r,q},\ q \in Q\}$.

In order to choose an appropriate order for the reduced dynamics associated to each mode, we quantify the approximation error through some function $d_T(\cdot,\cdot)$ that maps each pair of trajectories $y_a(t)$, $t \in T$, and $\hat{y}_a^\gamma(t)$, $t \in T$, into a positive real number $d_T(y_a,\hat{y}_a^\gamma)$ that represents the extent to which the output y_a of the SA system differs from its estimate $\hat{y}_a^\gamma$ along the time horizon T. Obviously, if we set $\gamma = 0$, then, no reduction is performed and $d_T(y_a,\hat{y}_a^\gamma) = 0$ since $\hat{y}_a^\gamma(t) = y_a(t)$, $t \in T$.

Note that $d_T(y_a,\hat{y}_a^\gamma)$ is a random quantity since it depends on the realization of the stochastic input $u(t)$ and the (possibly) stochastic initialization $\xi_a(0)$ of the SA system.

According to the notion of approximate simulation in [Abate and Prandini, 2011, Garatti and Prandini, 2012, Julius and Pappas, 2009], we assess the approximation quality of the reduced order model with parameter γ through the maximal value $\rho_\gamma^\star$ taken by $d_T(y_a,\hat{y}_a^\gamma)$ over all realizations of the stochastic input and initial state except for a set of probability at most $\varepsilon \in (0,1)$. An 'optimal' value for γ can then be chosen by inspecting the values of $\rho_\gamma^\star$ as a function of γ and selecting the appropriate compromise between quality of the approximation and tractability of the resulting reduced order model.

For each $\gamma \in \Gamma \subset [0,1]$, the approximation quality $\rho_\gamma^\star$ of the reduced order model with parameter γ is the solution to the following chance-constrained optimization problem:

$$
\begin{aligned}
CCP_\gamma : &\min_\rho \rho \\
&\text{subject to: } \mathbb{P}\{d_T(y_a,\hat{y}_a^\gamma) \le \rho\} \ge 1 - \varepsilon.
\end{aligned}
\tag{5.20}
$$

Remark (choice of $d_T(y_a,\hat{y}_a^\gamma)$). As argued in [Abate and Prandini, 2011], the directional Hausdorff distance

$$
d_T(y_a,\hat{y}_a^\gamma) = \sup_{t \in T} \inf_{\tau \in T} \|y_a(t) - \hat{y}_a^\gamma(\tau)\| \tag{5.21}
$$

is a sensible choice for $d_T(y_a,\hat{y}_a^\gamma)$ when performing probabilistic verification such as, e.g., estimating of the probability that y_a will enter some set within the time horizon T. For the verification of more complex reachability properties, such as that of reaching some set only after passing through some region within a given finite time interval, however, this choice for $d_T(y_a, \hat{y}_a^\gamma)$ is not adequate since the timing information is lost, and one can opt for

$$d_T(y_a, \hat{y}_a^\gamma) = \sup_{t \in T} \|y_a(t) - \hat{y}_a^\gamma(t)\|.$$

Irrespectively of the choice for $d_T(y_a, \hat{y}_a^\gamma)$, solving the chance-constrained problem (5.20) is known to be difficult, [Prèkopa, 2003], since it involves determining, among all sets of realizations of the stochastic input and initial state that have a probability $1 - \varepsilon$, the one that provides the best (lowest) value for $d_T(y_a, \hat{y}_a^\gamma)$. We then head for an approximate solution where instead of considering all the possible realizations for the stochastic uncertainty, we consider only a finite number N of them called "scenarios", extracted at random according to their probability distribution, and treat them as if they were the only admissible uncertainty instances. This leads to the formulation of Algorithm 2, where the chance-constrained solution is determined using some empirical violation parameter $\eta \in (0, \varepsilon)$.

Algorithm 2 randomised solution

```
1: extract $N$ realizations of the stochastic input $u^{(i)}(t)$, $t \in T$, $i = 1,2,\dots,N$, and $N$ samples of the initial condition $\xi_a(0)^{(i)}$, $i = 1,2,\dots,N$, and let $k = \lfloor \eta N \rfloor$;
2: for all $\gamma \in \Gamma$ do
    2.1: determine the $N$ realizations of the output signals $y_a^{(i)}(t)$ and $\hat{y}_a^{\gamma,(i)}(t)$, $t \in T$, $i = 1,2,\dots,N$, when the SL system and the reduced order model with parameter $\gamma$ are fed by the extracted uncertainty instances;
    2.2: compute
             $\hat{\rho}^{(i)} := d_T(y_a^{(i)}, \hat{y}_a^{\gamma,(i)}), i = 1,2,\dots,N;$
         and determine the indices $\{h_1, h_2, \dots h_k\} \subset \{1,2,\dots,N\}$ of the $k$ largest values of $\{\hat{\rho}^{(i)}, i = 1,2,\dots,N\}$
    2.3: set
             $\hat{\rho}_\gamma^\star = \max_{i \in \{1,2,\dots,N\} \setminus \{h_1,h_2,\dots,h_k\}} \hat{\rho}^{(i)}.$
```

Notably, if the number N of extractions is appropriately chosen, the obtained estimate of $\rho_\gamma^\star$ is chance-constrained feasible, uniformly with respect to $\gamma \in \Gamma$, with a-priori specified (high) probability. This result is based on the "scenario theory", [Campi et al., 2009], which was first introduced for solving uncertain convex programs via randomization [Calafiore and Campi, 2005] and then extended to chance-constrained optimization problems in [Campi and Garatti, 2011].

Proposition 5.4.1. Select a confidence parameter $\beta \in (0,1)$ and an empirical violation parameter $\eta \in (0,\varepsilon)$. If N is such that

$$\sum_{i=0}^{\lfloor \eta N \rfloor} \binom{N}{i} \varepsilon^i (1-\varepsilon)^{N-i} \leq \frac{\beta}{|\Gamma|}, \tag{5.22}$$

then, the solution $\hat{\rho}^\star_\gamma$, $\gamma \in \Gamma$, to Algorithm 2 satisfies

$$\mathbb{P}\{d_T(y_a, \hat{y}^\gamma_a) \leq \hat{\rho}^\star_\gamma\} \geq 1-\varepsilon, \ \forall \gamma \in \Gamma, \tag{5.23}$$

with probability at least $1-\beta$. □

If we discard the confidence parameter β for a moment, this proposition states that for any $\gamma \in \Gamma$, the randomised solution $\hat{\rho}^\star_\gamma$ obtained through Algorithm 2 is feasible for the chance-constrained problem (5.20). As η tends to ε, $\hat{\rho}^\star_\gamma$ approaches the desired optimal chance constrained solution $\rho^\star_\gamma$. In turn, the computational effort grows unbounded since N scales as $\frac{1}{\varepsilon-\eta}$, [Campi and Garatti, 2011].

As for the confidence parameter β, one should note that $\hat{\rho}^\star_\gamma$ is a random quantity that depends on the randomly extracted input realizations and initial conditions. It may happen that the extracted samples are not representative enough, in which case the size of the violation set will be larger than ε. Parameter β controls the probability that this happens and the final result holds with probability $1-\beta$. N satisfying (5.22) depend logarithmically on $|\Gamma|/\beta$, [Campi and Garatti, 2011], so that β can be chosen as small as 10^{-10} (and, hence, $1-\beta \simeq 1$) without growing significantly N.

(Proposition 5.4.1). Note that the chance-constrained problem (5.20) needs to be solved for a finite number $|\Gamma|$ of values for γ. The application of Theorem 2.1 in [Campi and Garatti, 2011] to the randomised solution obtained with Algorithm 2 for each given $\bar{\gamma} \in \Gamma$, provides the following guarantees on the solution $\hat{\rho}^\star_{\bar{\gamma}}$:

$$\mathbb{P}\{d_T(y_a, \hat{y}^{\bar{\gamma}}_a) \leq \hat{\rho}^\star_{\bar{\gamma}}\} \geq 1-\varepsilon, \text{ with probability at least } 1-\frac{\beta}{|\Gamma|}.$$

As a result, guarantee (5.23) involving all $\gamma \in \Gamma$ holds except for a set whose probability can be upper bounded by $\sum_{i=1}^{|\Gamma|} \frac{\beta}{|\Gamma|} = \beta$, thus proving the thesis. □

Notice that the guarantees provided by Proposition 5.4.1 are valid irrespectively of the underlying probability distribution of the input, which may even not be known explicitly, e.g., when feeding Algorithm 2 with historical time series as realizations of the stochastic input u.

5.5 A numerical example

In this section we present a numerical example to show the performance of the proposed approach for model reduction. The example is inspired by a benchmark for hybrid system verification presented in [Fehnker and Ivancic, 2004].

5.5.1 Model description

The example deals with the heating of a number of rooms in a house. Each room has one single heater, but there is some constraint on the number of "active" heaters that can possibly be on at the same time. The temperature in each room depends on the temperature of the adjacent rooms, on the outside temperature, and on whether a heater is on in the room or not. The heater is controlled by a typical thermostat, i.e., it is switched on if the temperature is below a certain threshold, and off if it is beyond another (higher) threshold. Differently from the original benchmark in [Fehnker and Ivancic, 2004], we model also the dynamic of the heaters.

When the temperature in a room, say room i, falls below a certain level, its heater may become active (and eventually be switched on) if a heater was active in one of the adjacent rooms, say room j, provided that the temperature in room j is significantly higher than that in room i. In this case, we shall say for brevity that the heater is "moved" from room i to room j. The underlying *rationale* of the control policy is that, even if all the rooms have their own heater, the number of heaters that can be on at the same time must be limited, so as to exploit also the heat exchange among the rooms in order to maintain some minimum temperature in all rooms.

Let T_i be the temperature in room i, T_{ext} the outside temperature, and h_i a boolean variable that is 1 when the heater is on in room i, and 0 otherwise.

The heat transfer coefficient between room i and room j is k_{ij}, and the one between room i and the external environment is $k_{e,i}$. We assume that the heat exchange is symmetric, i.e., $k_{ij} = k_{ji}$. We say that rooms i and j

are adjacent if $k_{ij} > 0$. The volume of the room is V_i, and the wall surface between room i and room j is $S_{r,ij}$, while that between room i and the environment is $S_{e,i}$. Air density and heat capacity are $\rho_a = 1.225\,\mathrm{kg/m^3}$ and $c = 1005\,\mathrm{J/(kg\,K)}$, respectively. Letting $\phi_i = \rho_a c V_i$, we can formulate the following dynamic model for room i and its heater:

$$\begin{aligned} \phi_i \dot{T}_i &= \sum_{j \neq i} S_{r,ij} k_{ij} (T_j - T_i) + S_{e_i} k_{e,i} (T_{\text{ext}} - T_i) + \kappa_i \theta_i \\ \tau_{h,i} \dot{\theta}_i &= -\theta_i + h_i \cdot p_i - \chi_i T_{\text{ext}} \end{aligned} \tag{5.24}$$

which is an affine system, with T_i representing the temperature in the i-th room, κ_i representing the maximum heat flow rate that the heater can provide, while $p_i \in \{0, 1\}$ is a binary variable indicating if the heater is active in room i. The heater dynamics is represented by a first-order system with a time constant $\tau_{h,i}$. If we neglect the term $-\chi_i T_{\text{ext}}$ in the heater dynamics and set $h_i = p_i = 1$, the heater state variable θ_i will tend to 1 so that the heater will provide its maximum heat flow rate κ_i to the room when it is active and on. The term $-\chi_i T_{\text{ext}}$ is introduced to account for the influence of the external temperature on the effectiveness of the heating system.

5.5.2 The switching control policy

There is a *room policy*, which decides whether or not to switch on the heater of a single room, and a *building policy* which decides how to “move” the heaters that can be switched on.
As for the room policy, each room has a thermostat that switches the heater on if the measured temperature is below a certain threshold, and off when the temperature reaches a higher temperature. For each room we define thresholds on_i and off_i: the heater in room i is on if $T_i \leq on_i$ and off if $T_i \geq off_i$.

On the other hand, the building policy can be defined as follows. A heater is moved from room j to an adjacent room i if the following holds

- room i has no active heater;
- room j has an active heater;
- temperature $T_i \leq get_i$;
- the difference $T_j - T_i \geq dif_i$.

Notice that the control policy may have non-deterministic behaviours, since a room j may have more than one room, e.g., rooms i_1 and i_2, that is adjacent, and it may happen that conditions for the building policy to move the heater to room i_1 and to room i_2 are satisfied at the same time. To avoid non-deterministic choices in the policy, each room is identified by some integer index, and, in the previously mentioned situation, the heater is always moved to the room with higher index.

Apparently enough, the switching nature of the system originates from the control policy. The complexity of the considered system significantly increases with the number of rooms, thus making the problem particularly suitable for reduction when dealing with realistic cases.

5.5.3 The considered system

In the following we consider four adjacent rooms as represented in Figure 5.1, having each its own heater, but with the constraint that only three heaters can be active at the same time, i.e., $\sum_{i=1}^{4} p_i = 3$.

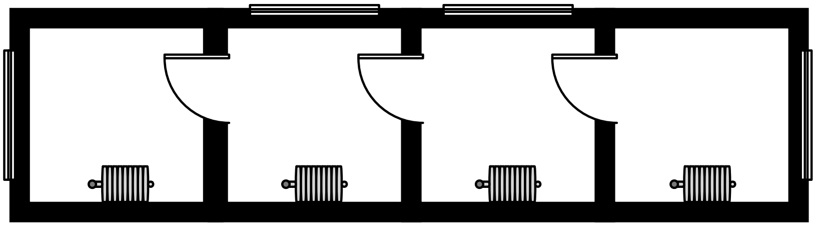


Figure 5.1: Scheme of the four rooms.

The rooms have different heat transfer coefficients among them, but identical geometric characteristics. The considered parameters are reported in Table 5.1.

Parameters			
k_{12}	$2\,\mathrm{W/(m^2K)}$	$S_{r,ij}$	$12\,\mathrm{m^2}$
k_{23}	$5\,\mathrm{W/(m^2K)}$	$S_{e,i}$	$24\,\mathrm{m^2}$
k_{34}	$2\,\mathrm{W/(m^2K)}$	V_i	$48\,\mathrm{m^3}$
$k_{e,i}$	$1\,\mathrm{W/(m^2K)}$	χ_i	10^{-5}

Table 5.1: Four rooms parameters.

The outside temperature is modelled as a sinusoidal source of period 24 hours with an offset of 4°C, affected by a band-limited Gaussian noise with zero mean and variance 4.

We assume that the initial conditions are deterministic and given by

$$T(0) = \begin{bmatrix} 20 \\ 20 \\ 20 \\ 20 \end{bmatrix}, \quad \theta(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad h(0) = p(0) = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

where T is the vector of the 4 rooms temperatures, θ is the vector of the heaters states, h and p are the vectors denoting, respectively, the on/off status and the active/inactive status of the heaters. Obviously, $p(0)$ satisfies the condition that only 3 over the 4 heaters are active. As for the (switching) control policy parameters, we use

$$off = \begin{bmatrix} 21 \\ 21 \\ 21 \\ 21 \end{bmatrix}, \quad on = \begin{bmatrix} 20 \\ 20 \\ 20 \\ 20 \end{bmatrix}, \quad get = \begin{bmatrix} 19 \\ 19 \\ 19 \\ 19 \end{bmatrix}, \quad dif = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}. \tag{5.25}$$

According to the described policy, model (5.24) can be represented as a SA system with continuous state $\xi_a = \begin{bmatrix} T' & \theta' \end{bmatrix}'$, input $u = T_{\text{ext}}$, and output $y_a = T$:

$$\begin{aligned} \dot{\xi}_a &= \mathscr{A}\,\xi_a + \mathscr{B}u + f_{q_a} \\ y_a &= \mathscr{C}\xi_a. \end{aligned} \tag{5.26}$$

As for the mode q_a, it is identified by the value of h and p, which determine the affine term entering the dynamics of ξ_a. The polyhedral sets Dom_{a,q_a} are determined by the building and room control policies through the threshold values (5.25) as described in Section 5.5.2.

Notice that in this example only the affine term f_{q_a} depends on the discrete mode $q_a \in Q$, while the state-space matrices $(\mathscr{A}, \mathscr{B}, \mathscr{C})$ are constant.

As for the choice of the order of the reduced model, the standard approach used in balanced truncation techniques [Mazzi et al., 2008] and resting on classical Hankel Singular Values (HSV) analysis can be applied so as to identify to what extent reducing the system dynamics in each single mode. This analysis is independent of the discrete mode. More importantly, it does not consider the impact of the choice of the order on the switched system approximation, which involves also mode transitions.

Figure 5.2 shows the HSV of system (5.26) sorted by decreasing magnitude. On the basis of the HSV, it seems that most of the dynamics can be caught by reducing the continuous dynamics of the SA system to a first-order one. Indeed, computing the distance (5.4) used in [Mazzi et al., 2008] results in $\psi(1) \cdot 100 = 2.64\%$.

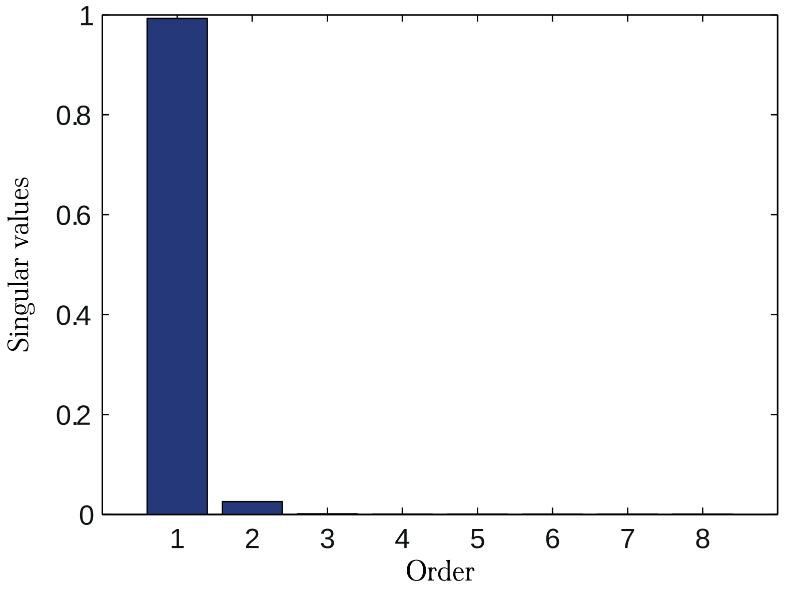


Figure 5.2: Hankel Singular Values sorted by decreasing magnitude.

As anticipated, this evaluation of the quality of the reduced model does not account for the impact of mode transitions, thus care has to be taken when applying it to the context of SA systems. In fact, classical balanced truncation techniques are typically based on the assumption that the free evolution of the system can be neglected since it asymptotically vanishes in an asymptotically stable linear system, fact that notoriously does not hold true when dealing with hybrid behaviours.

5.5.4 Proposed model reduction method

We apply now the proposed model reduction method to the considered system, including the randomised method for order selection based on the directional Hausdorff distance (5.21). In particular, referring to the chance constrained optimization problem (5.20), we choose $\varepsilon = 0.1$, $\beta = 10^{-6}/7$. Thus, setting $\eta = 0.05$, and solving the implicit formula (5.22), the number of experiments to be performed for each possible threshold value for γ is $N = 778$, corresponding to a number $\lfloor \eta N \rfloor = 38$ of realizations to be removed, as described in Algorithm 2.

The randomised order selection is performed with the reset maps (5.15) proposed in Mazzi et al. [2008], map (5.19) proposed here for the first time, both in its finite and infinite horizon versions. As for the choice of the finite horizon, the time constant τ_h of the heater is chosen.

Figure 5.3 shows a realization of the temperatures obtained with the original model and with the reduced models of order 5 implementing the three reset maps.

Notice that there is a discrete map $m_\gamma : \Gamma \to \{1,2,\ldots,n\}$ between the threshold values of γ and the corresponding order n_r of the reduced order model, in formulas

$$n_r = \underset{i=\{1,2,\ldots,n\}}{\arg\min} \left\{ d_T(y_a, \hat{y}_a^\gamma) \leq \hat{\rho}_\gamma^\star \right\}.$$

For the sake of clarity, it is more convenient to express the estimate of $\rho_\gamma^\star$ as a function of the reduced order n_r. The values for $\hat{\rho}_\gamma^\star$ obtained with the different reset methods are presented in Figure 5.4 as a function of n_r.

5.5.5 Discussion

Two facts can be noticed by analysing the results presented in Figure 5.4. First of all, the reset map affects the value of the directional Hausdorff distance, and the novel reset maps exhibit a better performance for any order n_r chosen for the reduction.

Furthermore, the outcome of our analysis through the randomised approach is quite different from that based on the HSV only (see Figure 5.2). In fact, reducing the system to a first-order approximation results in quite bad performance when the goal of the approximation is the analysis of reachability properties for which the directional Hausdorff distance is a suitable accuracy measure. In addition, such a drastic reduction yields discontinuities in the state reset that may possibly produce chattering behaviours. On the other hand, from the randomised based analysis it appears that one can push the reduction up to a fifth order without degrading significantly the accuracy of the model.

5.6 Concluding remarks

In this chapter, we presented a novel approach to model reduction of switched affine systems using balanced truncation for reducing the continuous affine dynamics. The main novel ingredients of the approach are:

- the introduction of suitable state reset maps that serve the purpose of making the reduced model best reproduce the free evolution of the original system; and

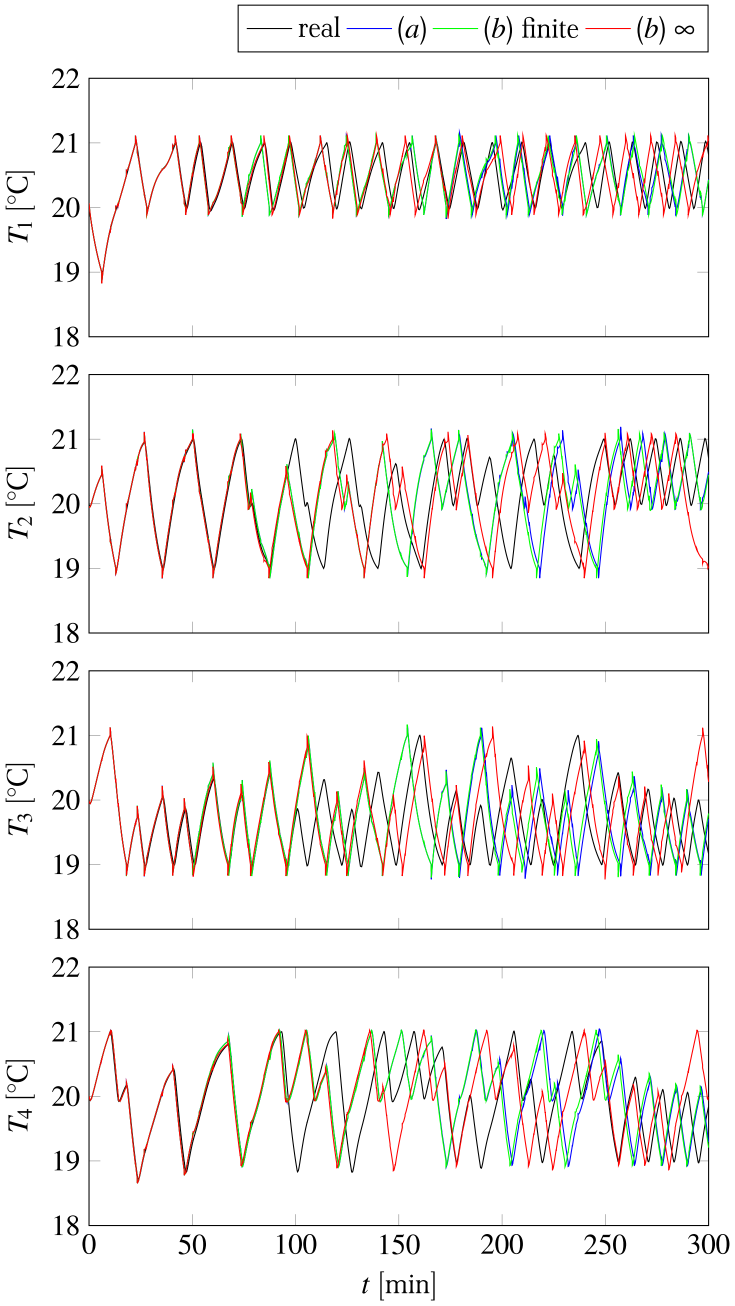


Figure 5.3: Comparison of the temperatures evolution obtained with the original model and with the reduced ones implementing the considered reset maps.

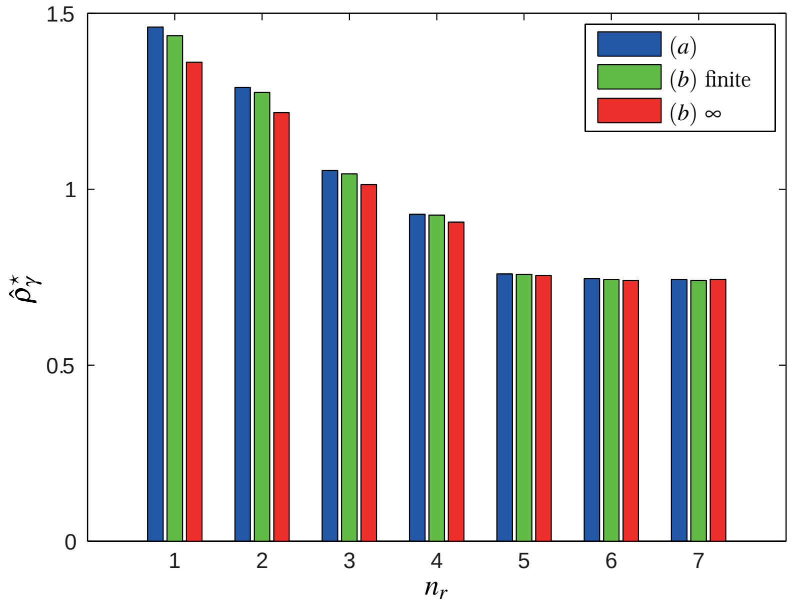


FIGURE 5.4: Performance of different reduced models as a function of the order n_r and of the adopted reset map.

- the integration in the reduced order model design of a randomised procedure for model order selection.

The considered class of switched systems is characterised by an endogenous switching signal, in that the transitions between modes are determined by the evolution of the continuous state component. The method can be applied also to the case when transitions are determined by some exogenous switching signal, possibly probabilistic as in the case of Markov jump linear systems, [Zhang et al., 2003]. In the case when the switching signal is subject to some dwell time τ_D and the approximated dynamics has a settling time smaller than τ_D, then, the approximation error introduced by the state reset will be negligible.

Chapter 6

Model manipulation toolchain

This chapter aims at supporting the idea, presented in Chapter 2, of devising a unifying approximation framework to complement – and thus to be integrated in – the typical manipulation toolchain of EOO M&S tools. Having presented the proposed approximation techniques in the context of continuous-time and hybrid systems, we here present a novel manipulation toolchain able to include most of the available techniques in quite an affordable way. To this end, some words are also spent to clarify the difficulties and the peculiarities of the problem, both from a conceptual and from a technological viewpoint.

6.1 The manipulation process

The task of a modeller typically consists of building a model of a physical system from first-principle laws, apply some transformation, so as to translate this model into a program capable to simulate its dynamic behaviour. Figure 6.1 depicts the main stages of this process, indicating for each of them the mathematical object characterising the stage.

All those stages can be carried out manually, but this obviously limits the complexity that can be handled, and has the additional drawback of possi-

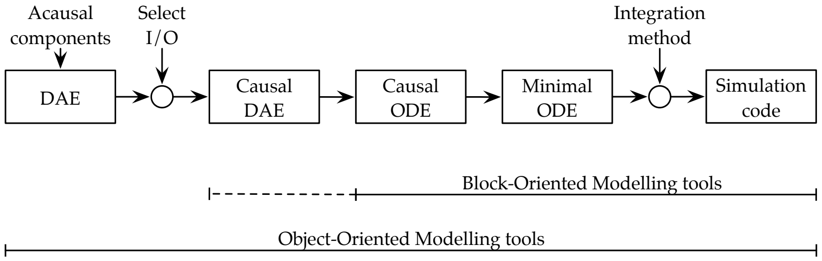


Figure 6.1: Chain of operations from the model construction to the code generation.

bly introducing flaws and bugs in the final implementation of the simulation code. Thus tools capable of supporting this chain of operations have been created, in order to make the overall process – or at least the great part of it – automatic, and as transparent as possible for the end user. It is worth remarking that the underlying assumption of the process is that the simulation code is *semantically equivalent* to the solution of the original model, i.e., the manipulation should not introduce any kind of approximation—except of those that are inherently present in the integration method, but that do not influence the *semantic equivalence* defined in this thesis.

In so doing, different modelling paradigms emerged, in particular the Block-Oriented and Object-Oriented ones, having both of them their respective advantages and disadvantages. Sticking to the manipulation viewpoint, the set of operations that are automatised are represented in Figure 6.1 by the solid lines under the toolchain. In particular, Object-Oriented Modelling tools support a fully automatised manipulation from the composition of the model to the simulation code. On the other hand, Block-Oriented ones – think, e.g., to Simulink – only partially support the chain of operation, leaving to the modeller the task of obtaining a causal ODE system form the physical model of interest. Even if it is a limit of the latter approach – at least for the complexity management –, it can be considered at the same time an advantage, since it allows the modeller to manually (or semi-automatically) manipulate the continuous-time equations with, e.g., MOR techniques.

This last remark is quite important for the scope and purpose of this research. A fully automatised manipulation toolchain may in fact be a double-edged sword, especially if it is conceived as a unique "black-box" part of the tool. For example, using MOR techniques in an Object-Oriented modelling tool is far from being convenient, since the modeller do not have direct (nor indirect, e.g., by means of some options or configuration parameters of the tool) access to the continuous-time ODE equations. On the other hand, both Block- and Object-Oriented paradigms have hard-coded simulation methods, leaving very few possibilities of customising – and possibly adding experimental – numerical solvers.

All the mentioned limitations are essentially due to the impossibility of accessing the data structures needed either for the approximation or to build a simulation architecture like, e.g., a co-simulation one. This is even more difficult in the Object-Oriented paradigm where everything is encapsulated in a unique manipulation toolchain, hindering the possibility of introducing any approximation.

Apparently, the presented problem is more technological than conceptual, but since the complexity of models emerging in the day by day engineering work is steadily increasing, the availability of a general (possibly tool-independent) solution becomes crucial. In the following, we analyse deeper the problem in the context of EOO modelling tools, thus describing the adopted technological solution.

6.2 Advantages and disadvantages of EOO languages

EOO modelling languages are known to possess a number of interesting advantages, and in the context of this work, two are particularly relevant. First, the EOO modelling paradigm is inherently suited for building modular, multi-physic models. Second, the model designer has not to take care of how the system will be simulated, just focusing on how to write the equations of its components. In one word, with EOO Modelling Tools (EOOMT) one handles the complete model by just aggregating components and acting on them. The translator included in typical EOOMT is then in charge of manipulating all the gathered equations, and producing efficient simulation code [Cellier and Kofman, 2006, Fritzson, 2003].

As long as the obtained simulation efficiency is sufficient, the possibility of managing complexity at the component level has practically no cost. However, as widely discussed in the thesis, there are some cases where to achieve the desired simulation efficiency, approximations need introducing, and EOOMT are neither meant nor suited for that. As discussed herein, approximations can be introduced either by altering some equations in some components, or by acting on the numerical solution of the complete model. And while with EOOMT the first action is natural, the second is not at all. This rules out several powerful approximation techniques aimed at enhancing simulation speed, e.g., MOR and DD ones treated herein.

To enter the subject, it is convenient to specify why introducing approximation at the level of the solution is "unnatural" in EOO modelling. The main reason is that the possibility/opportunity of doing so depends on properties of the *whole* model, not of the individual components, and in the typical toolchain of EOOMT no user interaction is envisaged at that point. Moreover, assuming that the use of any approximation technique requires some parameters, it is necessary to provide the user with the necessary information to give them a value, and to accept his/her choices, in a comprehensible manner, manageable by people who are more experts of physics than of simulation theory.

In this work we refer as "EOO Modelling Tool" to a Modelica translator, to allow exemplifying the (more general) presented ideas. For a Modelica translator, the EOO modelling toolchain can be synthetically depicted as in Figure 6.2.

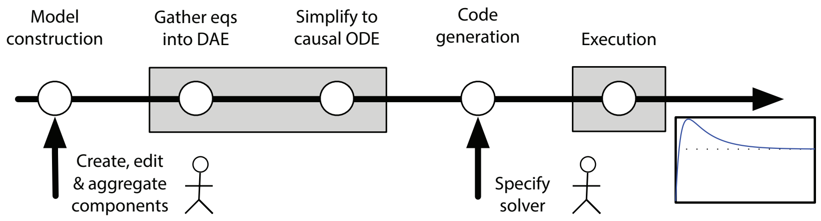


FIGURE 6.2: The typical EOO modelling toolchain.

For our purposes, said toolchain has to be extended – and in some sense "opened" – as suggested in Figure 6.3, introducing some (clearly optional) automatic system-wide analysis, and taking care of having the user interact with simple enough information despite of operating at the whole system level.

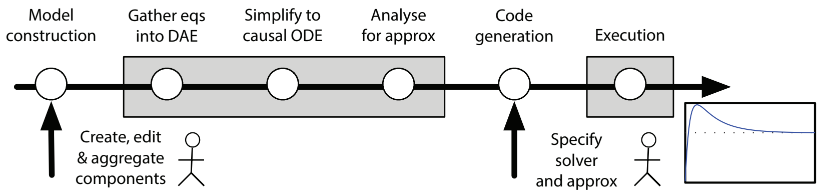


FIGURE 6.3: Extending the EOO modelling toolchain.

We here present a solution assuming that the desired type of approximation is DD (which is the most difficult to integrate in typical manipulation toolchains), therefore tailoring the analysis and the use of the produced information to that case. Nonetheless the way of acting on the toolchain is general with respect to the approximation type, thus a novel manipulation toolchain complementing the functionalities of pre-existent ones is here proposed.

6.3 The technological solution: Functional Mockup Interface

Most of modern M&S tools do not allow to directly access to intermediate formats of the manipulation. Therefore, performing approximations acting either on the continuous-time equations, or on the numerical solution is virtually impossible. The problem is, however, more technological than conceptual since all the information needed for both kind of approximations is present in a certain (non-standard, i.e., tool-dependent) format as internal data structures of the tool at hand.

Blochwitz et al. [2011] proposed a technological solution for tool independent simulation models exchange, creating the Functional Mockup Interface (FMI). In this paper they say

> “ The Functional Mockup Interface (FMI) is a tool independent standard for the exchange of dynamic models and for co-simulation. The development of FMI was initiated and organized by Daimler AG within the ITEA2 project MODELISAR. The primary goal is to support the exchange of simulation models between suppliers and OEMs even if a large variety of different tools are used. The FMI was developed in a close collaboration between simulation tool vendors and research institutes. In this article an overview about FMI is given and technical details about the solution are discussed.
>
> *[Blochwitz et al., 2011]*

Vendors of Modelica tools (AMESim, Dymola, SimulationX) and non Modelica tools (SIMPACK, Silver, Exite), as well as research institutes worked closely together and recently defined the Functional Mockup Interface. This interface covers the aspects of model exchange and of co-simulation, practically providing a standard language between different tools.

This exchange format thus constitutes the technological solution to the aforementioned problem, and all the developed software architecture is based on the idea that a certain model – in this work a Modelica model, but now it is even more clear that the presented concepts can be easily applied to any M&S tool – is exported in the FMI format, i.e., a Functional Mockup Unit (FMU), providing suitable Application Programming Interfaces (API) so as to allow manipulation on both the continuous-time equations and on the numerical solution.

The latter manipulation, as better detailed in the following, can be exploited by means of other tools providing numerical efficient packages for the solution of ODEs in the FMI format, e.g., Assimulo[1].

Assimulo has the nice feature that is written in Python, a high-level programming language, combining a wide variety of different solvers written in

[1] `www.jmodelica.org/assimulo`

FORTRAN, C and wrapped with a Python interface, thus providing good numerical performance. In addition, the construction of experimental solvers, like the mixed-mode one presented in Section 3.5, is quite easy and affordable, and takes benefits of the underlying efficient numerical infrastructure.

6.4 An example toolchain implementation

Thanks to the adoption of such a technological solution, it has been possible to develop a manipulation toolchain capable to perform the desired approximations, in an affordable way, by means of open-source tools.

To prove the feasibility of our extension proposal, in fact, the toolchain has been carried out by using JModelica[2] as the Modelica translator, exporting the model as a Functional Mockup Unit (FMU) [Andersson et al., 2011], and employing Assimulo[3] for the numerical integration, having developed *ad hoc* the mixed-mode integrator, but on the basis of already developed (and well established) explicit and implicit integration methods. Since some of the API needed for the development of the Cycle Analysis and of the mixed-mode integration missed in the first version, FMI 2.0 is needed [Blochwitz et al., 2012]. However, most of co-simulation schemes can be easily implemented on the basis of the FMI standard [Schierz et al., 2012].

More in detail, the toolchain of Figure 6.2 was modified as shown in Figure 6.4: the output of the continuous-time part (the manipulated `model.mo`) is exported by means of the FMI 2.0 standard to `model.fmu`, elaborated by the external python module `jd2.py` that performs CA (i.e., takes care of the "discretisation" and the "solution manipulation" blocks); the partitioned model is then simulated with Assimulo, with the developed mixed-mode method. It is worth noticing that the integration of a new functionality (like DD) into an EOO modelling toolchain was greatly eased by adopting tools that allow for some common interchange format—a feature of great importance indeed.

The developed code, including the reported examples, is available as free software[4], within the terms of the Modelica License v2.

[2]`http://www.jmodelica.org`

[3]`http://www.jmodelica.org/assimulo`

[4]The code is available at `http://home.dei.polimi.it/leva/jd2.html`.

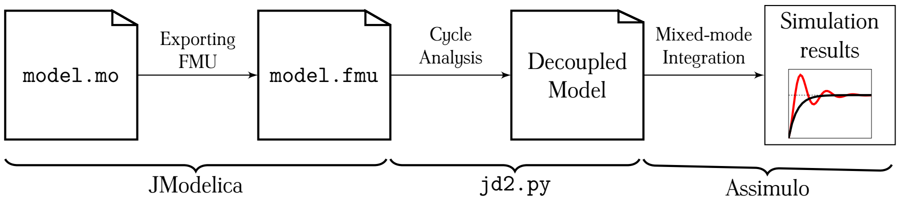


Figure 6.4: Integration of DD in the toolchain of Figure 6.5.

6.5 A Unifying Manipulation Toolchain

Having devised a viable technological solution for the introduction of approximations in the numerical integration – which is incidentally the most difficult part to modify in modern EOO M&S tools – it is now quite easy to think of an complementing extension of the classical manipulation toolchain by also introducing other approximations, such as – but not limiting to – MOR ones. An interesting remark is that functions may appear in the model, thus limiting the scope of the approach. However, Papadopoulos et al. [2012] proposed a method for function inlining of Modelica models, extending the capabilities of the approach. The proposed methodology do not alter the semantic of the original model, thus not introduces any kind of approximation.

An activity diagram of the proposed extended toolchain is presented in Figure 6.5.

It is apparent that typical operations are preserved, but just some decision node have been introduced so as to provide the necessary data structures needed to perform some model manipulation in a view to improve simulation efficiency in the desired direction. Recall, in fact that in Chapter 2 a taxonomy of the intended uses of a model was given, evidencing that in difference context different desires on its properties may arise.

The underlying *rationale* is that one can build in an affordable way a quite sophisticated model by aggregating different (multi-physics) components, thus exploiting all the advantages of the EOO modelling approach. Therefore, the modeller can specify which is its intended use, and leaving to the tool the task of performing the suitable approximations aimed at obtaining the desired properties, possibly specifying some thresholds on the error, similarly to what is done with the integration methods.

In particular, the idea is that if the modeller wants to perform some approximations in order to improve simulation speed, he/she needs to be able to specify high-level properties, e.g., upper bounds on the approximation

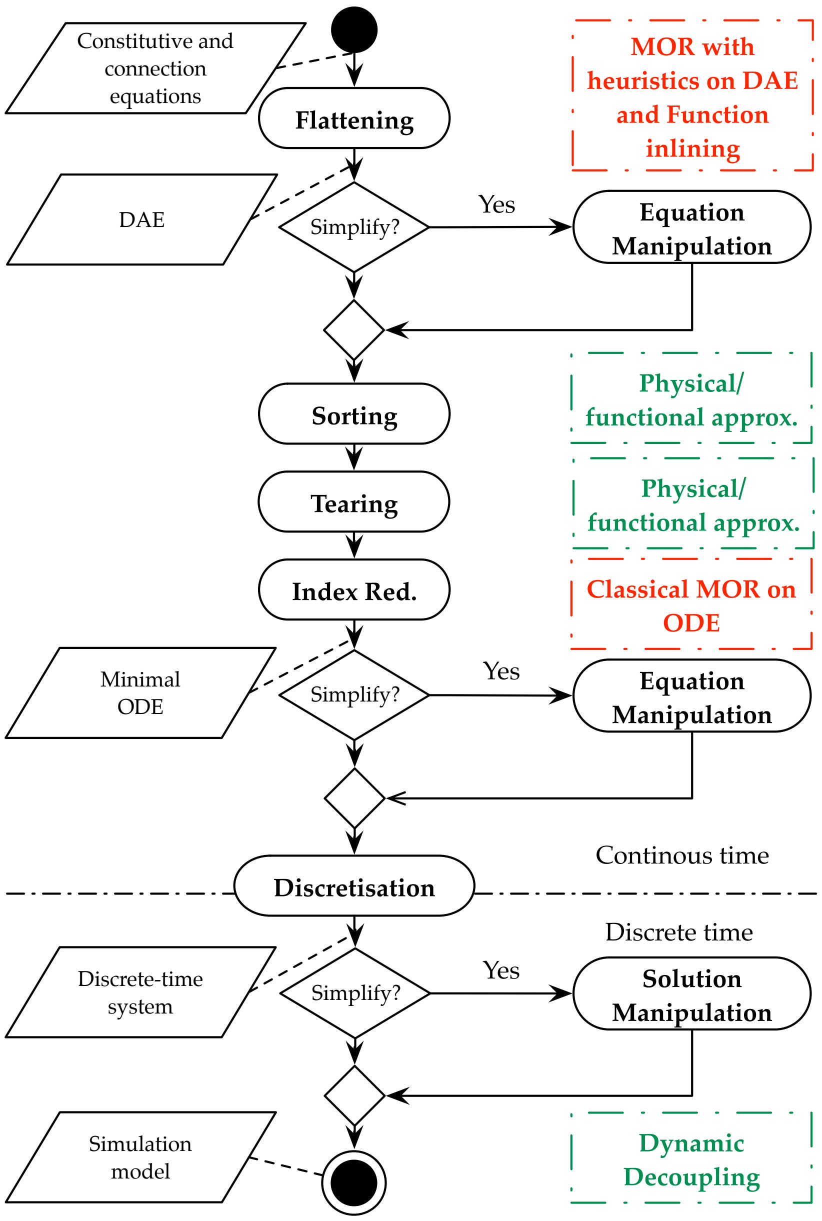


Figure 6.5: Activity diagram of the modified manipulation toolchain.

error, and which technique must be used for it, e.g., the MOR technique as well as whether or not the use of DD is advisable.

Figure 6.5 depicts the proposed toolchain of model manipulations, from the EOO description to the simulation algorithm ready for code generation. The decision nodes (the diamond ones in the diagram) show where additional manipulation for simplification can be performed. If, in every decision node, the simplification is not performed, the classical manipulation toolchain comes out. Otherwise, a simpler model is produced at the end of the toolchain.

The diagram also reports some coloured dashed boxes on the right side. Red boxes stand for already available methodologies that can be automatically applicable at this level of the manipulation, while green ones stand for potential methodologies which may be introduced as automatic procedures, e.g., function inlining composed with some other approximation technique, but to date not exploited in the context of EOO modelling.

Chapter 7

Conclusion and open problems

In this dissertation, some model simplification techniques were presented, as the first nucleus of a unified framework to enhance the capability of existing M&S tools. The matter was thus addressed from both the methodological and technological point of view. From the technological side, the main result is the integration of a simplification technique to improve simulation efficiency in the toolchain of an EOO tool. To achieve this results, several methodological ones were derived whose generality extends far beyond the proposed technological solution.

An analysis technique was proposed to automatically partition complex dynamic models in both the linear and the nonlinear case, based on time scale information that is easy to understand for the modeller. As a useful-by-product, suitable separability indices were also devised, that are apparently useful also for other purposes than the one analysed herein. Also, and again as a contribution functional to the main one, the problem of quantifying the distance between two model was studied, proposing some measures of that distance that are particularly suited for the purpose of model simplification, both in the case of continuous and of discontinuous systems.

Given their importance in several application domains, hybrid systems were also taken into account. The main reason for doing so is that for those systems, MOR techniques are significantly less mature than they are for continuous systems. As such, in a view to comprehending also hybrid systems in the simplification framework, that is the ultimate goal of this research, extensions of MOR techniques to hybrid systems were studied. In this respect, the main results are the proposal of different reinitialisation maps accounting not only for the forced motion of the system, but also for its free one, and a randomised method for the order selection overcoming the inherent limits of classical ones used in MOR techniques for continuous systems. Quite obviously, however, some problems are still open.

7.1 Open problems

The presented DD technique relies on a design parameter – denoted by α in the corresponding chapters – that evidently influences the precision of the obtained solution. However, at present, we are not yet capable of determining that parameter based on some error bound for the same solution. Solving this problem would further enhance the ease of use of the presented technique for the analyst.

As for the exploitation of DD by integration methods, we have been considering only fixed-step ones. Although, in principle we can foresee no reasons preventing the use of variable-step methods, the problem of suitably extending the framework is at present open.

Also, in the case of MOR techniques for hybrid systems, we already mentioned that having the possibility of introducing a dwell time would be beneficial for the quality of the approximation. However, it is not yet clear how to deal with those cases – like the one proposed in the corresponding chapter – in which a dwell time does not have a "physical" motivation, and thus may possibly lead to infeasible or, better, physically meaningless situations—e.g., in the example having the temperature in the rooms reaching values that in any real implementation would ask for a controller intervention on smaller enough a time scale than the resulting dwell time.

Finally, sticking to hybrid systems, we still need to devise reduction techniques for the discrete modes, so as to contain complexity also on this side.

7.2 Future work

Besides the open problems just mentioned, there are some other issues for which a solution can already be envisioned, but was not yet developed.

A first one is related to managing the partitioning of a model in more than two decoupled subsystems. Also in this case, it is possible to ask the analyst to only provide time scale information. However, the way the tool has to interpret the analysts desires to configure the simulation becomes more articulated than it is in the presented examples. Solving this issue does not pose any conceptual problem, requiring nonetheless to suitably extend the exploitation of the proposed analysis.

On a similar front, it is already clear how parallelisable cycle sets (PCS) can be used to further partition the model for better efficiency. Here too, however, a complete implementation of this exploitation needs designing.

Finally, plans are underway to test the generality of the proposed approach by including in the consequently designed framework other techniques than those considered in this work.

Appendix A

Additional Examples on Dynamic Decoupling

A.1 Automotive suspension

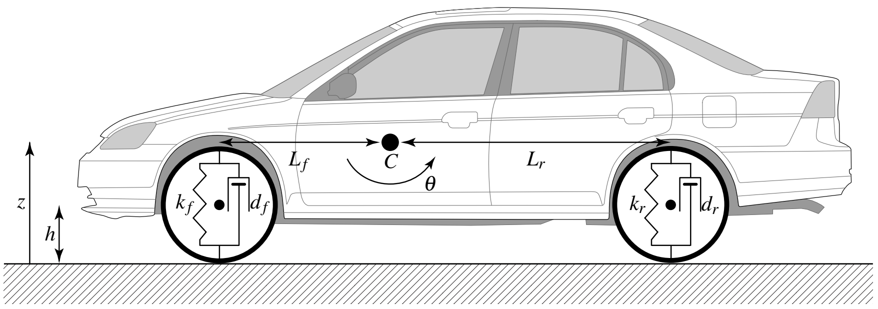


FIGURE A.1: Half-car suspension model.

Figure A.1 illustrates the modelled characteristics of a half-car. The front and rear suspension are modelled as spring/damper systems. A more detailed model would include a tire model, and damper nonlinearities such as velocity-dependent damping (with greater damping during rebound than compression). The vehicle body has pitch and bounce degrees of freedom. They are represented in the model by four dynamic states: vertical displacement z, vertical velocity $\dot{z}$, pitch angular displacement θ, and pitch angular velocity $\dot{\theta}$. A full model with six degrees of freedom can be implemented using vector algebra blocks to perform axis transformations and force/displacement/velocity calculations. The front and rear suspensions influence the bounce (i.e., vertical degree of freedom) according to the equations

$$\begin{aligned} F_f =& 2k_f\left(L_f\theta - z\right) + 2d_f\left(L_f\dot{\theta} - \dot{z}\right) \\ F_r =& 2k_r\left(L_r\theta + z\right) - 2d_r\left(L_r\dot{\theta} + \dot{z}\right) \end{aligned}$$

where F_f and F_r are the upward force on body from front and rear suspension, k_f and k_r are the front and rear suspension spring constant, d_f and d_r are the front and rear damping factors, L_f and L_r are the horizontal distance from the center of gravity (C) to front and rear suspensions.

The pitch contribution to the front and rear suspension is given by

$$\begin{aligned} \tau_f =& - L_f F_f \\ \tau_r =& L_r F_r \end{aligned}$$

where τ_f and τ_r are the pitch torque due to the front and rear suspension.

Hence, using the Newton's law, the forces and moments are balanced as

$$\begin{aligned} M\ddot{z} =& F_f + F_r - mg \\ J\ddot{\theta} =& \tau_f + \tau_r + \tau_a \end{aligned}$$

where M is the body mass, J is the body momentum of inertia about the center of gravity and τ_a is the pitch torque induced by vehicle acceleration. The parameters used for this example are reported in Table A.1.

Table A.1: Half-car suspension parameters.

Parameters			
M	1000 kg	d_f	3500 N s/m
J	2100 kg m^2	d_r	3500 N s/m
k_f	28 000 N/m	L_f	0.9 m
k_r	21 000 N/m	L_r	1.2 m

In this example we initially consider the vehicle accelerating, so $\tau_a = 500\,\mathrm{N\,m}$ for $t \geq 0$ (0 otherwise). Hence, at time $t = 2$, we consider the road height h increased by 0.15 m.

According to CA there are 7 cycles in the model digraph, and choosing $\alpha = 0.5$, the following constraints on the integration step are obtained.

$$\begin{aligned} \dot{z}: \quad h \leq 0.038 \quad z: \quad & h \leq 0.071 \\ \dot{\theta}: \quad & h \leq 0.077 \\ \theta: \quad & h \leq 0.100 \end{aligned} \tag{A.1}$$

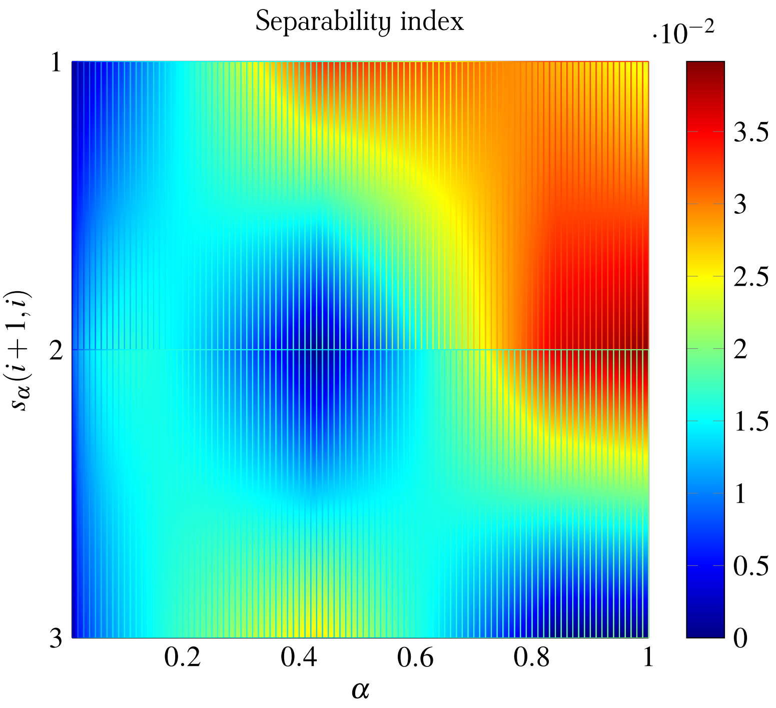


FIGURE A.2: Separability parametric analysis of half-car suspension model.

hence, choosing an integration step $h = 0.04$, we can separate the system accordingly to what suggested by Figure A.2. Figure A.3 shows the simulation results.

Table A.2 shows the simulation statistics for different integration methods. It is worth noticing that the dimension of the system the Newton iteration has to solve is reduced from 4 to 1 in the mixed-mode method. Notice, also that the EE method needs a smaller step size ($h = 0.05$) for numerical stability reasons. Apparently, the mixed-mode method performs better than the others also in this very simple case.

To complete the example, the proposed indices proposed in Section 3.4 are here computed, yielding the following indices

$$\sigma_R = 1.869 \quad \sigma(0.5) = 2.590, \quad s(0.5) = 0.382.$$

The stiffness $\sigma(\alpha)$ index shows that the system is stiff, while the separability one shows that this kind of system is sufficiently suited for separation—recall that high values of s mean that the system is separable.

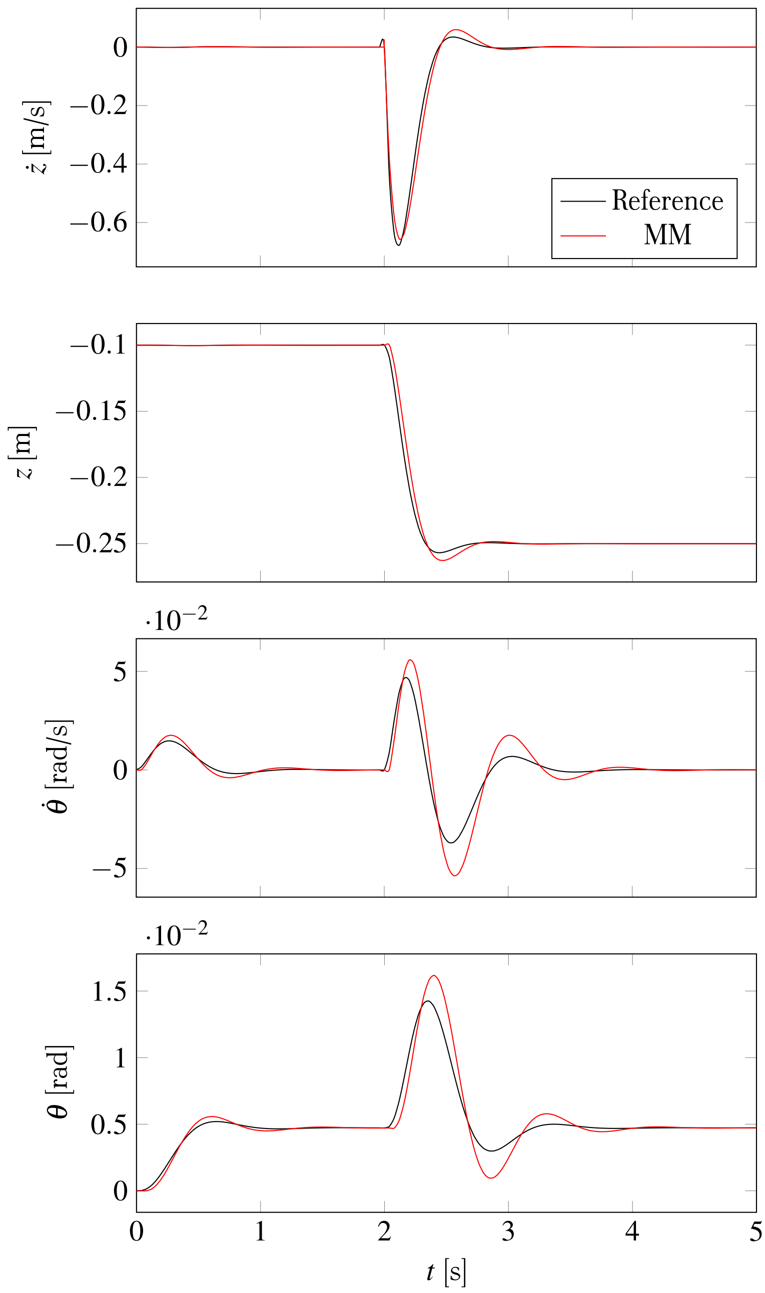


Figure A.3: Simulation results of the half-car suspension model.

TABLE A.2: Simulation statistics for half-car model.

	Mixed-mode	**BDF**	**IE**	**EE**
# Steps	**126**	198	**126**	126
# Function ev.	376	236	**375**	–
# Jacobian ev.	**6**	5	**6**	–
# Fun. ev. in Jac. ev.	**12**	20	30	–
# Newton iterations	250	228	**249**	–
# Newton fail	0	0	0	–
Accuracy	**0.638**	–	**0.638**	0.674
Sim time	0.1s	0.16s	0.13s	**0.04s**

A.2 Double-mass, triple spring-damper

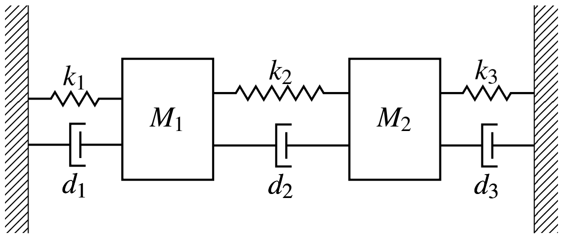


FIGURE A.4: Double-mass, triple spring-damper.

This example refers to a simple test problem, similar to that presented in González et al. [2011]. The considered system is composed of two masses and three parallel spring-damper elements, connected as shown in Figure A.4, and moving in a horizontal plane (i.e., gravity has no effect). Both elasticity and damping friction are assumed to be linear phenomena, so that the couplings between the dynamic variables can be easily determined by acting on the elastic constants k_i and the damping factors d_i. In particular, in the reported test, $M_1 = M_2 = 1\,\text{kg}$, $k_1 = 500\,\text{N/m}$, $d_1 = 5\,\text{N s/m}$, $k_2 = 1\,\text{N/m}$, $d_2 = 1\,\text{N s/m}$, $k_3 = 5\,\text{N/m}$, and $d_3 = 1\,\text{N s/m}$.

Letting x_1 and x_2 the horizontal positions of the two masses represented in Figure A.4, the model can be written as

$$\begin{cases} M_1\ddot{x}_1 = -(d_1+d_2)\dot{x}_1 + d_2\dot{x}_2 - (k_1+k_2)x_1 + k_2x_2 \\ M_2\ddot{x}_2 = d_2\dot{x}_1 - (d_2+d_3)\dot{x}_2 + k_2x_1 - (k_2+k_3)x_2 \end{cases} \tag{A.2}$$

A preliminary analysis is needed so as to understand if the model, with the given set of parameters is suited to be partitioned. This is carried out by means of a parametric CA, where EE is chosen as the probe integration method, i.e., by exploiting the closed-form solution (3.11), and by computing the separability terms of Definition 3.4.3. The result is shown in Figure A.5, where the numbers on the vertical axis index the variables ordered by increasing time scale.

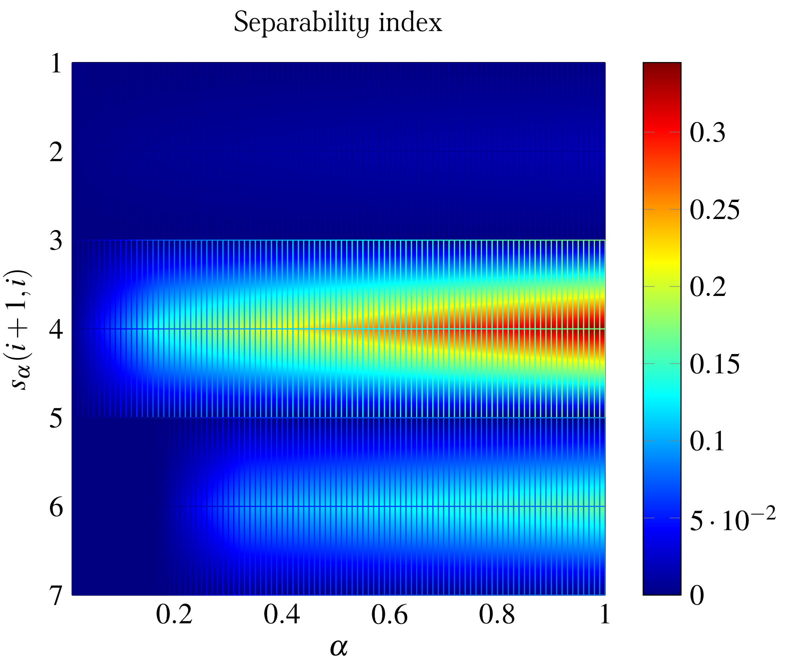


Figure A.5: Separability parametric analysis of the double-mass, triple spring-damper system.

It is apparent from the figure, that the highest separability term is obtained between the 4-*th* and the 5-*th* variable, suggesting where to partition the model for the decoupled integration.

According to CA there are 17 cycles in the model digraph, and choosing $\alpha = 0.5$, the following constraints on the integration step are obtained.

$$\begin{array}{llll} \dot{x}_1: & h \leq 0.032 & \dot{x}_2: & h \leq 0.289 \\ x_1: & h \leq 0.032 & x_2: & h \leq 0.289 \\ \dot{y}_1: & h \leq 0.045 & \dot{y}_2: & h \leq 0.408 \\ y_1: & h \leq 0.045 & y_2: & h \leq 0.408 \end{array} \tag{A.3}$$

Based on the aforementioned analysis, we can partition the model into two submodels by separating the first 4 variables – the set of the ones on the left, that are considered fast – from the other 4 — the set of the ones on the right that are considered slow. Hence, the integration step can be chosen as $h = 0.05$. Notice that incidentally the analysis brought to intuitive indeed a result, i.e., to separate the two set of equations associated to the two masses, without any *a priori* suggestion to the method of the physical structure of the system.

Figure A.6 shows the numerical results of the mixed-mode integration method, while Table A.3 presents some comparative simulation statistics. Notice that for EE a smaller integration step ($h = 0.01$) has been used for numerical stability reasons.

TABLE A.3: Simulation statistics Double-mass, triple spring-damper.

	Mixed-mode	**LSODAR**	**IE**	**EE**
# Steps	**101**	483	**101**	500
# Function ev.	**302**	949	**302**	–
# Jacobian ev.	**5**	28	**5**	–
# Fun. ev. in Jac. ev.	**25**	–	45	–
# Newton iterations	**201**	–	**201**	–
# Newton fail	0	–	0	–
Accuracy	5.969	–	5.964	14.472
Sim time	**0.1s**	0.12s	0.11s	0.13s

The mixed-mode integration method is able to capture the system behaviour, especially for the steady-state, while the choice of a "large" integration step has the effect that the "fast" dynamics, i.e., the transient oscillations, are approximated by a slower dynamics. Furthermore, simulation statistics show that also in this first (linear) example, the performance are slightly improved, with respect to other comparable methods Cellier and Kofman [2006].

To complete the example, the proposed indices proposed in Section 3.4 are here computed. In particular, model (A.2), and CA result (A.3) yield the following indices. Notice that σ_R cannot be computed since there are purely imaginary eigenvalues in the system.

$$\sigma(0.5) = 12.923, \quad s(0.5) = 0.779.$$

The stiffness index $\sigma(\alpha)$ indicates that the system is highly stiff. On the other hand, the separability index $s(\alpha)$ shows that the considered example

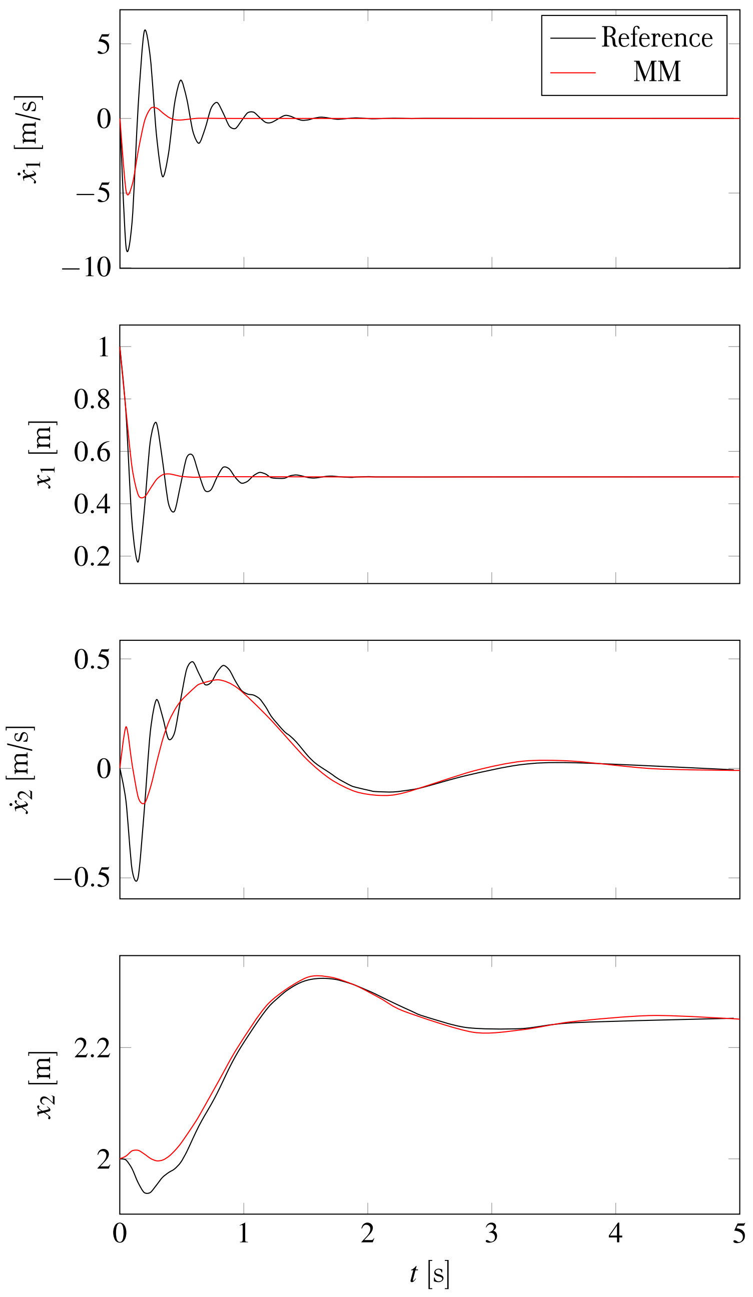


Figure A.6: Simulation results of the double-mass, triple spring-damper system.

has dynamics evolving with quite different time scales, thus making it effective to partition the model into subsystems. Finally, to decide how to obtain that partition, adequate clues are provided by Figure A.5.

A.3 A small smart grid

The system considered in this example is composed of an islanded three-phase generator feeding an inductive load. In modelling this system mechanical phenomena were considered, although the prime mover is idealised as a torque generator applied to the alternator shaft and governed by a combined power/frequency controller, with primary and secondary action. We also consider the load self-heating phenomenon, plus – obviously – electrical ones.

The Modelica scheme of the considered system is represented in Figure A.7.

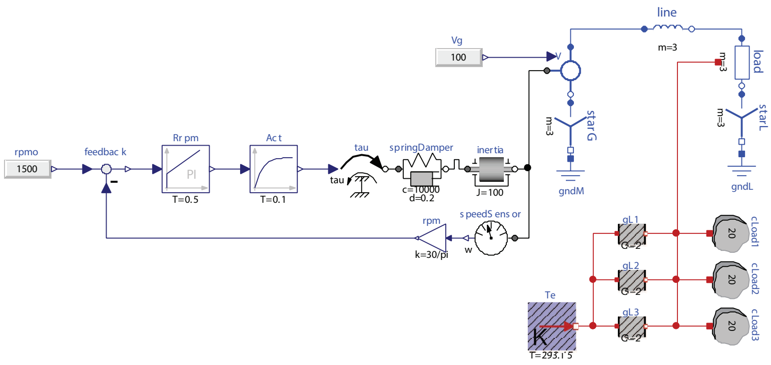


FIGURE A.7: Multiscale smart grid Modelica scheme.

The separability analysis results are represented in Figure A.8. Apparently enough, it is possible to separate the dynamics into two clusters of variables, i.e., from the first variable to the seventh one, and from the eighth to the last one.

The system has 13 cycles, leading to the following constraints by choos-

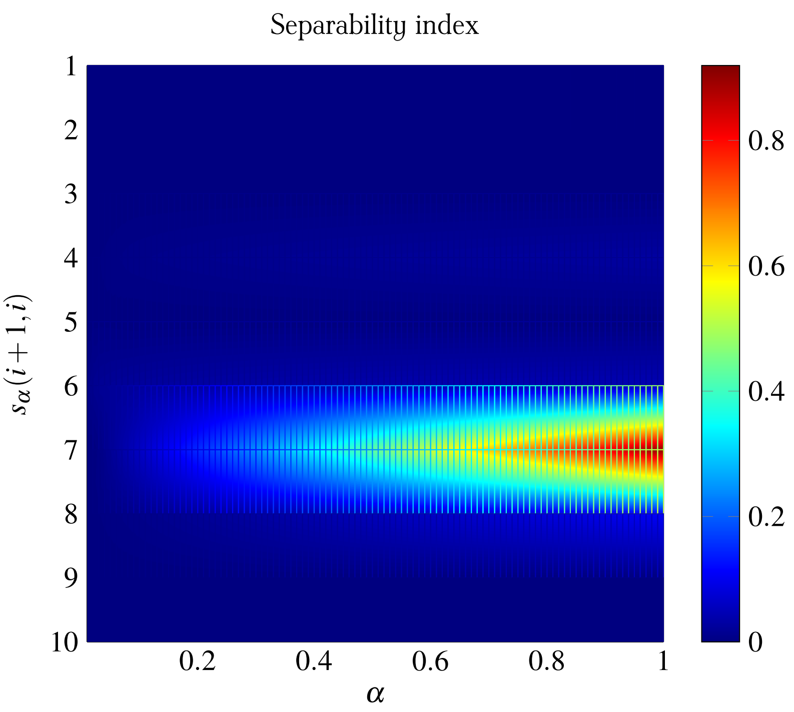


Figure A.8: Separability analysis of the smart grid example.

ing $\alpha = 1.0$

$$\begin{aligned} \phi &: \quad h \leq 2.000 \times 10^{-5} \\ I_1 &: \quad h \leq 0.002 \\ I_2 &: \quad h \leq 0.002 \\ I_3 &: \quad h \leq 0.002 \\ y_{\text{actuator}} &: \quad h \leq 0.032 \\ \omega_{\text{inertia}} &: \quad h \leq 0.032 \\ x_{\text{PI}} &: \quad h \leq 0.081 \end{aligned} \qquad \begin{aligned} \phi_{\text{inertia}} &: \quad h \leq 0.081 \\ T_1 &: \quad h \leq 1.111 \\ T_2 &: \quad h \leq 1.111 \\ T_3 &: \quad h \leq 1.111 \end{aligned} \tag{A.4}$$

Thus, choosing a integration step of $h = 0.1$, we can partition the system according to the separability analysis result. Therefore, on the left there are the fast variables, while on the right there are the slow ones.

The simulation result is represented in Figure A.9, while Table A.4 reports the simulation statistics for the Mixed-mode and LSODAR integration scheme. Purely explicit and implicit Euler, are here not considered since the first needs too a tiny integration step due to the system stiffness, while the latter is not even able to initialize the system, since the nonlinear Newton iteration do not converge.

TABLE A.4: Simulation statistics for the simulation example.

	Mixed-mode	**LSODAR**
# Steps	200	54891
# Function ev.	11088	88775
# Jacobian ev.	120	2762
# Fun. ev. in Jac. ev.	600	–
# Newton iterations	9088	–
Accuracy	**639×10^9**	–
Sim time	**3.61s**	20.23s

In this case the level of accuracy is very low, accordingly to the value. This is essentially due to the fact that the mixed-mode integration is simulating an average behaviour of the system – remember that the integration technique is approximating the dynamics under the chosen time scale – thus leading to a large error when high-frequency oscillating behaviours are present in the system.

In this example the structural indices become

$$\sigma(1.0) = 55554.444, \quad s(1.0) = 0.879,$$

showing that the considered system is highly stiff – fact that is also apparent from the time scales (A.4) identified by CA that can be partitioned in three clusters – thus indicating that a purely explicit method is not suitable for simulating efficiently the system. On the other hand, the separability index also shows that the system is suited to be partitioned, also accordingly to the separability analysis reported in Figure A.8.

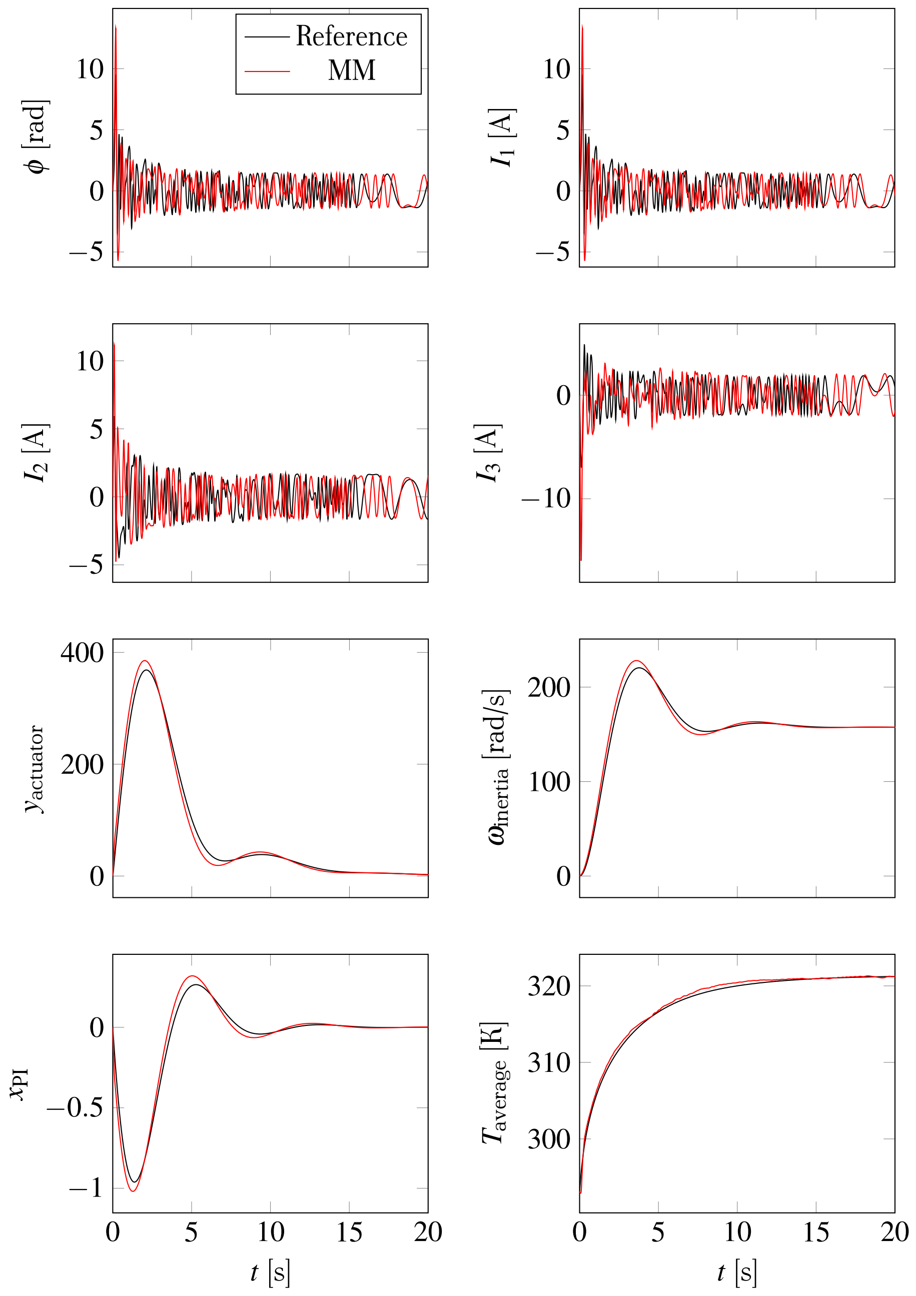


Figure A.9: Multiscale smart grid simulation.

Bibliography

A. Abate and M. Prandini. Approximate abstractions of stochastic systems: a randomized method. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 4861–4866. IEEE, Dec 2011. doi: 10.1109/CDC.2011.6161148.

A. Abate, S. Amin, M. Prandini, J. Lygeros, and S. Sastry. Computational approaches to reachability analysis of stochastic hybrid systems. In A. Bemporad, A. Bicchi, and G. Buttazzo, editors, *Hybrid Systems: Computation and Control*, volume 4416 of *Lecture Notes in Computer Science*, pages 4–17. Springer Berlin Heidelberg, Apr. 2007. ISBN 978-3-540-71492-7. doi: 10.1007/978-3-540-71493-4_4.

A. Abate, J.-P. Katoen, J. Lygeros, and M. Prandini. Approximate model checking of stochastic hybrid systems. *European Journal of Control, special issue on Stochastic hybrid systems*, 16(6):624–641, Dec. 2010.

S. Amin, A. Abate, M. Prandini, J. Lygeros, and S. Sastry. Reachability analysis for controlled discrete time stochastic hybrid systems. In *Hybrid Systems: Computation and Control*, pages 49–63. Springer, 2006.

C. Andersson, J. Åkesson, C. Führer, and M. Gäfvert. Import and export of functional mock-up units in jmodelica.org. In *Proc. of the 8th International Modelica Conference*, 2011.

C. Andersson, J. Andreasson, C. Führer, and J. Åkesson. A workbench for multibody systems ODE and DAE solvers. In *Proceedings of the IMSD2012 - The 2nd Joint International Conference on Multibody System Dynamics*, 2012.

A. Antoulas. *Approximation of large-scale dynamical systems*, volume 6. Society for Industrial Mathematics, 2005.

M. Arnold and W. O. Schiehlen. *Simulation Techniques for Applied Dynamics*, volume 507. Springer, 2009.

M. Arnold, B. Burgermeister, C. Führer, G. Hippmann, and G. Rill. Numerical methods in vehicle system dynamics: state of the art and current developments. *Vehicle System Dynamics*, 49(7):1159–1207, 2011. doi: 10.1080/00423114.2011.582953.

U. M. Ascher, S. J. Ruuth, and R. J. Spiteri. Implicit-explicit runge-kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics*, 25(2–3):151–167, 1997. ISSN 0168-9274. doi: 10.1016/S0168-9274(97)00056-1.

A. Bartolini, A. Leva, and C. Maffezzoni. A process simulation environment based on visual programming and dynamic decoupling. *Simulation*, 71 (3):183–193, 1998.

J. Bastian, C. Clauß, S. Wolf, and P. Schneider. Master for co-simulation using FMI. In *Proc. of the 8th International Modelica Conference*, 2011.

T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauß, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold, D. Neumerkel, H. Olsson, J.-V. Peetz, and S. Wolf. The functional mockup interface for tool independent exchange of simulation models. In *8th International Modelica Conference*, pages 20–22, 2011.

T. Blochwitz, M. Otter, J. Åkesson, M. Arnold, C. Clauß, H. Elmqvist, M. Friedrich, A. Junghanns, J. Mauss, D. Neumerkel, H. Olsson, and A. Viel. Functional Mockup Interface 2.0: The standard for tool independent exchange of simulation models. In *9th International Modelica Conference*, pages 173–184, Münich, Germany, 2012. doi: 10.3384/ecp12076173.

D. Broman. *Meta-Languages and Semantics for Equation-Based Modeling and Simulation*. PhD thesis, Department of Computer and Information Science, Linköping University, Sweden, 2010.

D. Broman and J. G. Siek. Modelyze: a Gradually Typed Host Language for Embedding Equation-Based Modeling Languages. Technical Report UCB/EECS-2012-173, EECS Department, University of California, Berkeley, Jun 2012.

K. Burrage. Parallel methods for initial value problems. *Applied Numerical Mathematics*, 11(1–3):5–25, 1993. ISSN 0168-9274. doi: 10.1016/0168-9274(93)90037-R.

G. Calafiore and M. Campi. Uncertain convex programs: randomized solutions and confidence levels. *Mathematical Programming*, 102(1):25–46, 2005.

M. Campi and S. Garatti. A sampling-and-discarding approach to chance-constrained optimization: Feasibility and optimality. *Journal of Optimization Theory and Applications*, 148(2):257–280, 2011.

M. C. Campi, S. Garatti, and M. Prandini. The scenario approach for systems and control design. *Annual Reviews in Control*, 33(2):149–157, 2009. ISSN 1367-5788. doi: 10.1016/j.arcontrol.2009.07.001.

F. Casella. A strategy for parallel simulation of declarative object-oriented models of generalized physical networks. In *5th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools.*, pages 45–51, 2013.

F. Casella and A. Leva. Modelica open library for power plant simulation: design and experimental validation. In *Proc. of the 3rd Modelica conference, Linköping, Sweden*, 2003.

F. Casella, F. Donida, and M. Lovera. Automatic Generation of LFTs from Object-Oriented Non-Linear Models with Uncertain Parameters. In *Proceedings MATHMOD 09 Vienna, Vienna, Austria*, pages 1359–1367, 2009.

F. E. Cellier and E. Kofman. *Continuous system simulation*. Springer, London, UK, 2006. ISBN 9780387261027.

J. Chen and S.-M. Kang. Model-order reduction of nonlinear MEMS devices through arclength-based Karhunen-Loeve decomposition. In *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE Int. Symposium on*, volume 3, pages 457–460 vol. 2, 2001. doi: 10.1109/ISCAS.2001.921346.

J. Chen, S.-M. Kang, J. Zou, C. Liu, and J. Schutt-Aine. Reduced-order modeling of weakly nonlinear MEMS devices with Taylor-series expansion and Arnoldi approach. *Microelectromechanical Systems, Journal of*, 13 (3):441– 451, 2004. ISSN 1057-7157. doi: 10.1109/JMEMS.2004.828704.

E. Christen and K. Bakalar. VHDL-AMS-a hardware description language for analog and mixed-signal applications. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 46(10):1263–1272, 1999. ISSN 1057-7130. doi: 10.1109/82.799677.

J. B. Danby and T. L. Harman. *Mastering Simulink*. Pearson/Prentice Hall, 2003. ISBN 9780131424777.

I. S. Duff and J. K. Reid. An implementation of tarjan's algorithm for the block triangularization of a matrix. *ACM Trans. Math. Softw.*, 4(2):137–147, Jun 1978. ISSN 0098-3500. doi: 10.1145/355780.355785.

H. Elmqvist and M. Otter. Methods for tearing systems of equations in object-oriented modeling. In *ESM'94 European Simulation Multi- conference*, volume 94, pages 326–332, 1994.

A. Fehnker and F. Ivancic. Benchmarks for hybrid systems verification. In *In Hybrid Systems: Computation and Control (HSCC 2004) (2004*, pages 326–341. Springer, 2004.

G. Ferretti, S. Filippi, C. Maffezzoni, G. Magnani, and P. Rocco. Modular dynamic virtual-reality modeling of robotic systems. *Robotics Automation Magazine, IEEE*, 6(4):13–23, 1999. ISSN 1070-9932. doi: 10.1109/100.813823.

S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

G. Frehse. Phaver: Algorithmic verification of hybrid systems past hytech. In *Hybrid Systems: Computation and Control*, pages 258–273. Springer, 2005.

P. Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Wiley, 2003.

S. Garatti and M. Prandini. A simulation-based approach to the approximation of stochastic hybrid systems. In *Analysis and Design of Hybrid Systems*, pages 406–411, 2012.

C. Ghezzi, M. Jazayeri, and D. Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 2002. ISBN 0133056996.

A. Girard and C. Guernic. Zonotope/hyperplane intersection for hybrid systems reachability analysis. In *Proceedings of the 11th international workshop on Hybrid Systems: Computation and Control*, HSCC '08, pages 215–228, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-78928-4. doi: 10.1007/978-3-540-78929-1_16. URL `http://dx.doi.org/10.1007/978-3-540-78929-1_16`.

A. Girard and G. Pappas. Approximation metrics for discrete and continuous systems. *IEEE Trans. on Automatic Control*, 52(5):782–798, 2007.

A. Girard, A. A. Julius, and G. J. Pappas. Approximate simulation relations for hybrid systems. *Discrete Event Dynamic Systems*, 18(2):163–179, 2008.

A. Girard, G. Pola, and P. Tabuada. Approximately bisimilar symbolic models for incrementally stable switched systems. *IEEE Transactions on Automatic Control*, 55(1):116–126, Jan 2010.

L. Goldberg and G. Ann. *Efficient algorithms for listing combinatorial structures*, volume 5. Cambridge Univ Pr, 2009.

F. González, M. Á. Naya, A. Luaces, and M. González. On the effect of multirate co-simulation techniques in the efficiency and accuracy of multibody system dynamics. *Multibody System Dynamics*, 25(4):461–483, 2011. ISSN 1384-5640. doi: 10.1007/s11044-010-9234-7.

C. Gu. *Model Order Reduction of Nonlinear Dynamical Systems*. PhD thesis, University of California, Berkeley, 2011.

S. Gugercin and A. C. Antoulas. A survey of model reduction by balanced truncation and some new results. *International Journal of Control*, 77(8): 748–766, 2004.

D. Hanselman and B. Littlefield. *Mastering MATLAB Seven*. Pearson/Prentice Hall, 2005. ISBN 9780131857148.

W. E. Hart, C. Laird, J.-P. Watson, and D. L. Woodruff. *Pyomo – Optimization Modeling in Python*. Springer, 2012. ISBN 978-1-4614-3225-8.

M. Hast, J. Åkesson, and A. Robertsson. Optimal robot control using modelica and optimica. In *Proc. of the 7th International Modelica Conference*, 2009.

B. Hendrickson and K. Devine. Dynamic load balancing in computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 184(2–4):485–500, 2000. ISSN 0045-7825. doi: 10.1016/S0045-7825(99) 00241-8.

A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward. Sundials: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):363–396, 2005.

M. Innocent, P. Wambacq, S. Donnay, H. Tilmans, W. Sansen, and H. De Man. An analytic volterra-series-based model for a mems variable capacitor. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Trans. on*, 22(2):124–131, 2003. ISSN 0278-0070. doi: 10.1109/TCAD.2002.806603.

D. Johnson. Finding all the elementary circuits of a directed graph. *SIAM J. Comput.*, 4(1):77–84, 1975.

A. Julius and G. Pappas. Approximations of stochastic hybrid systems. *IEEE Transactions on Automatic Control*, 54(6):1193–1203, 2009.

B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell system technical journal*, 29:291–307, 1970.

G. Kunze, J. Frenkel, C. Schubert, and K. Jankov. Using modelica for interactive simulations of technical systems in a virtual reality environment. In *Proceedings of the 7th International Modelica Conference, Como (Italy)*, 2009.

A. B. Kurzhanski and P. Varaiya. Ellipsoidal techniques for hybrid dynamics: the reachability problem. In *in New Directions and Applications in Control Theory*, pages 193–205. Springer, 2005.

Y. Kuznetsov. *Elements of Applied Bifurcation Theory*, volume 112 of *Applied Mathematical Sciences*. Springer, 2004.

S. Lall, J. Marsden, and S. Glavaski. A subspace approach to balanced truncation for model reduction of nonlinear control systems. *International Journal of Robust and Nonlinear Control*, 12:519–535, 2002.

D. C. Lane and R. Oliva. The greater whole: Towards a synthesis of system dynamics and soft systems methodology. *European Journal of Operational Research*, 107(1):214–235, 1998. ISSN 0377-2217. doi: 10.1016/S0377-2217(97)00205-1.

Y. Liu and B. D. Anderson. Singular perturbation approximation of balanced systems. *International Journal of Control*, 50(4):1379–1405, 1989.

L. Ljung. *System Identification*. John Wiley & Sons, Inc., 2001. ISBN 9780471346081. doi: 10.1002/047134608X.W1046.

J. Lunze and F. Lamnabhi-Lagarrigue, editors. *Handbook of Hybrid Systems Control – Theory, Tools, Applications*. Cambridge University Press, Cambridge, UK, 2009.

S. Mattsson, H. Elmqvist, and M. Otter. Physical system modeling with Modelica. *Control Engineering Practice*, 6(4):501–510, 1998. ISSN 0967-0661. doi: 10.1016/S0967-0661(98)00047-1.

S. E. Mattsson and G. Söderlind. Index reduction in differential-algebraic equations using dummy derivatives. *SIAM Journal on Scientific Computing*, 14(3):677–692, 1993.

E. Mazzi, A. Sangiovanni Vincentelli, A. Balluchi, and A. Bicchi. Hybrid system reduction. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 227–232. IEEE, Dec 2008. doi: 10.1109/CDC.2008.4739350.

L. Mikelsons and T. Brandt. Symbolic model reduction for interval-valued scenarios. In *ASME Conf. Proc.*, volume 49002, pages 263–272. ASME, 2009. doi: 10.1115/DETC2009-86954.

L. Mikelsons and T. Brandt. Generation of continuously adjustable vehicle models using symbolic reduction methods. *Multibody System Dynamics*, 26:153–173, 2011. ISSN 1384-5640.

L. Mikelsons, T. Brandt, and D. Schramm. Real-time vehicle dynamics using equation-based reduction techniques. In *IUTAM Symposium on Dynamics Modeling and Interaction Control in Virtual and Real Environments*, volume 30 of *IUTAM Bookseries*, pages 91–98. Springer Netherlands, 2011. ISBN 978-94-007-1642-1. doi: 10.1007/978-94-007-1643-8_11.

I. Mitchell. *Application of Level Set Methods to Control and Reachability Problems in Continuous and Hybrid Systems*. PhD thesis, Ph.D. Dissertation. Dept. Scientific Computing and Computational Mathematics, Stanford Univ., CA, 2002.

F. J. Monssen. *OrCAD PSpice with Circuit Analysis*. Prentice-Hall PTR, Upper Saddle River, N.J., 2001.

B. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *Automatic Control, IEEE Transactions on*, 26(1):17–32, Feb 1981. ISSN 0018-9286. doi: 10.1109/TAC.1981.1102568.

A. Norton. Utilising rapid product development and late customisation methodologies within manufacturing smes, 2001.

M. Oh and C. Pantelides. A modelling and simulation language for combined lumped and distributed parameter systems. *Computers & Chemical Engineering*, 20(6–7):611–633, 1996. ISSN 0098-1354. doi: 10.1016/0098-1354(95)00196-4.

A. W. Ordys, A. Pike, M. A. Johnson, R. M. Katebi, and M. Grimble. *Modelling and Simulation of Power Generation Plants. Advances in Industrial Control.* Springer-Verlag, 1994. ISBN 978-3-540-19907-6.

C. Pantelides. The consistent initialization of differential-algebraic systems. *SIAM Journal on Scientific and Statistical Computing*, 9(2):213–231, 1988. doi: 10.1137/0909014.

A. V. Papadopoulos and A. Leva. Automating dynamic decoupling in object-oriented modelling and simulation tools. In *5th International workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, pages 37–44, Apr. 2013a. ISBN 978-91-7519-621-3.

A. V. Papadopoulos and A. Leva. A model partitioning method based on dynamic decoupling for the efficient simulation of multibody systems. *Multibody System Dynamics*, 2013b. (accepted).

A. V. Papadopoulos and A. Leva. Automating efficiency-targeted approximations in modelling and simulation tools: dynamic decoupling and mixed-mode integration. *SIMULATION: Transactions of The Society for Modeling and Simulation International*, 2013c. (under review).

A. V. Papadopoulos and A. Leva. Enhancing dynamic simulation performance for models of energy systems and smart grids. *IEEE Transactions on Industrial Informatics*, 2013d. (under review).

A. V. Papadopoulos and M. Prandini. Model reduction of switched affine systems: a method based on balanced truncation and randomized optimization. In *17th International Conference on Hybrid Systems: Computation and Control (HSCC 2014)*, 2014. doi: 10.1145/2562059.2562131.

A. V. Papadopoulos, M. Maggio, F. Casella, and J. Åkesson. Function inlining in modelica models. In *Proceedings of the 7th International Conference of Mathematical Modelling, MATHMOD'12*, volume 7, pages 1091–1094. IFAC, Feb. 2012. doi: 10.3182/20120215-3-AT-3016.00193.

A. V. Papadopoulos, J. Åkesson, F. Casella, and A. Leva. Automatic partitioning and simulation of weakly coupled systems. In *Decision and Control*

(CDC), 2013 IEEE 52nd Annual Conference on, pages 3172–3177, Dec. 2013. ISBN 978-1-4673-5716-6.

A. V. Papadopoulos, F. Casella, and A. Leva. Model separability indices for efficient dynamic simulation. In *19th IFAC World Congress*, 2014. (under review).

M. Petreczky and R. Vidal. Metrics and topology for nonlinear and hybrid systems. In *Proceedings of the 10th International Conference on Hybrid Systems: Computation and Control*, volume 4416 of *Lecture Notes in Computer Sciences*, pages 459–472, 2007.

M. Petreczky, R. Wisniewski, and J. Leth. Theoretical analysis of balanced truncation for linear switched systems. In *Analysis and Design of Hybrid Systems*, pages 240–247, 2012. doi: 10.3182/20120606-3-NL-3011.00039.

L. Petzold. *Description of DASSL: a differential/algebraic system solver*. Sep. 1982.

A. Pfeiffer, M. Hellerer, S. Hartweg, M. Otter, and M. Reiner. Pysimulator–a simulation and analysis environment in python with plugin infrastructure. In *Proceedings of 9th International Modelica Conference, Munich, Germany, Sept*, 2012. doi: 10.3384/ecp12076523.

J. R. Phillips. Projection frameworks for model reduction of weakly nonlinear systems. In *Proc. of the 37th Annual Design Automation Conf.*, DAC '00, pages 184–189, New York, NY, USA, 2000. ACM. ISBN 1-58113-187-9. doi: 10.1145/337292.337380.

M. Pidd and A. Carvalho. Simulation software: not the same yesterday, today or forever. *Journal of Simulation*, 1(1):7–20, 2006.

M. Prandini and J. Hu. Stochastic reachability: Theoretical foundations and numerical approximation. In *Stochastic hybrid systems*, volume 24 of *Control Engineering Series*, pages 107–138. Taylor & Francis Group/CRC Press, 2006.

A. Prèkopa. Probabilistic programming. In A. Ruszczyǹski and A. Shapiro, editors, *Stochastic Programming*, volume 10 of *handbooks in operations research and management science*, London, UK, 2003. Elsevier.

S. Robinson. Soft with a hard centre: Discrete-event simulation in facilitation. *The Journal of the Operational Research Society*, 52(8):905–915, 2001. ISSN 01605682.

R. G. Sargent, R. E. Nance, C. M. Overstreet, S. Robinson, and J. Talbot. The simulation project life-cycle: models and realities. In *Proceedings of the 38th conference on Winter simulation*, WSC '06, pages 863–871. Winter Simulation Conference, 2006. ISBN 1-4244-0501-7.

J. Scherpen. Balancing for nonlinear systems. *Systems & Control Letters*, 21 (2):143–153, 1993.

A. Schiela and H. Olsson. Mixed-mode integration for real-time simulation. In *Modelica Workshop 2000 Proceedings*, pages 69–75, 2000.

T. Schierz, M. Arnold, and C. Clauß. Co-simulation with communication step size control in an FMI compatible master algorithm. In *Proc. of the 9th International Modelica Conference*, 2012.

T. Schmitt, D. Zimmer, and F. E. Cellier. A virtual motorcycle rider based on automatic controller design. In *Proc. 7th Modelica Conference*, volume 3, pages 19–28, 2009.

H. R. Shaker and R. Wisniewski. Model reduction of switched systems based on switching generalized gramians. *International Journal of Innovative Computing, Information and Control*, 8(7(B)):5025–5044, 2012. ISSN 1349-4198.

M. Sjölund. TLM and Parallelization. Technical report, Linköping University, 2012.

M. Sjölund, R. Braun, P. Fritzson, and P. Krus. Towards efficient distributed simulation in modelica using transmission line modeling. In *3rd Int. workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, pages 71–80, 2010.

R. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1971.

R. Tarjan. Enumeration of the elementary circuits of a directed graph. *SIAM J. Comput.*, 2(3):211–216, 1972.

C. Tomlin, I. Mitchell, A. Bayen, and M. Oishi. Computational techniques for the verification of hybrid systems. *Proceedings of the IEEE*, 91(7):986–1001, 2003.

A. Varga, G. Looye, D. Moormann, and G. Gräbel. Automated generation of lft-based parametric uncertainty descriptions from generic aircraft models.

Mathematical and Computer Modelling of Dynamical Systems, 4(4):249–274, 1998. doi: 10.1080/13873959808837082.

L. Žlajpah. Simulation in robotics. *Mathematics and Computers in Simulation*, 79(4):879–897, 2008. ISSN 0378-4754. doi: 10.1016/j.matcom.2008.02.017.

A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano. Determining Lyapunov exponents from a time series. *Physica D: Nonlinear Phenomena*, 16(3):285–317, 1985. ISSN 0167-2789. doi: 10.1016/0167-2789(85)90011-9.

L. Zhang, B. Huang, and J. Lam. h_∞ model reduction of Markovian jump linear systems. *Systems & Control Letters*, 50(2):103–118, 2003.

D. Zimmer and M. Otter. Real-time models for wheels and tyres in an object-oriented modelling framework. *Vehicle System Dynamics*, 48(2):189–216, 2010. doi: 10.1080/00423110802687596.