Redundancy reduction of IC models : by multirate time-integration and model order reduction

Citation for published version (APA):
Verhoeven, A. (2008). *Redundancy reduction of IC models : by multirate time-integration and model order reduction*. [Phd Thesis 1 (Research TU/e / Graduation TU/e), Mathematics and Computer Science]. Technische Universiteit Eindhoven. https://doi.org/10.6100/IR631462

DOI:
10.6100/IR631462

Document status and date:
Published: 01/01/2008

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 04. Oct. 2023

Redundancy Reduction of IC Models

by Multirate Time-Integration and
Model Order Reduction

Copyright ©2008 by Arie Verhoeven, Eindhoven, The Netherlands.
All rights are reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without prior permission of the author.

CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN

Verhoeven, Arie

Redundancy reduction of IC models : by multirate time-integration and model order reduction / door Arie Verhoeven. -
Eindhoven : Technische Universiteit Eindhoven, 2008.
Proefschrift. - ISBN 978-90-386-1174-7

NUR 919
Subject headings: numerical integration / numerical analysis / multigrid methods / differential equations / integrated circuits / simulation
2000 Mathematics Subject Classification: 65Lxx, 65M55, 34E13, 47N70

Printed by Jafra Drukwerkservice.
Cover design by Verspaget & Bruinink.

Redundancy Reduction of IC Models
by Multirate Time-Integration and
Model Order Reduction

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
Rector Magnificus, prof.dr.ir. C.J. van Duijn, voor een
commissie aangewezen door het College
voor Promoties in het openbaar te verdedigen
op dinsdag 8 januari 2008 om 16.00 uur

door

Arie Verhoeven

geboren te Utrecht

Dit proefschrift is goedgekeurd door de promotor:

prof.dr. R.M.M. Mattheij

Copromotor:

dr. E.J.W. ter Maten

Contents

1 Introduction 1

1.1 Motivation for the project 1
1.2 Formulation of the problem 4
1.3 Guideline to the document 7

2 Mathematical models for electrical circuits 9

2.1 Theory of electrical circuits 9
2.2 Modified nodal analysis (MNA) 11
2.3 Circuit analyses 13

3 Differential-algebraic equations (DAEs) 15

3.1 Introduction 15
3.2 Uniqueness and existence of the solution 16
3.3 Index of differential-algebraic equations 18
3.4 Stability 19

4 Numerical integration of DAEs 21

4.1 Introduction 21
4.2 Linear Multistep Methods 23
4.3 The Nordsieck data representation 26
4.4 The Backward Difference Formula (BDF) 28
4.5 Error analysis of the BDF method 30
4.6 Adaptive stepsize control 32

5 BDF multirate time-integration for DAEs 39

5.1 Introduction 39
5.2 Types of multirate 41
5.3 The BDF Compound-Fast multirate algorithm 44
5.4 Analysis of conditions for partitioning 49
5.5 Stability analysis of multirate methods 53
5.6 Stability analysis of Euler Backward multirate algorithm 55
5.7 Stability analysis of multistep BDF multirate algorithms 62
5.8 Multirate for hierarchical models 74
5.9 Implicit interpolation techniques 76

6 **Adaptive multirate stepsize control and partitioning** **81**
6.1 Analysis of the local discretisation error 81
6.2 Adaptive stepsize and order control 85
6.3 Optimal value for the balance number 87
6.4 Adaptive partitioning control 90
6.5 Local efficiency analysis of multirate methods 91
6.6 Partitioning algorithms 93
6.7 Influence of interpolation errors on partitioning sequence 97
6.8 Dynamical partitioning techniques 99
6.9 Further extensions 101

7 **Model order reduction (MOR)** **103**
7.1 Introduction 103
7.2 Model order reduction of LTI systems 104
7.3 Nonlinear model order reduction 108
7.4 Trajectory Piecewise Linear Model Order Reduction 109
7.5 Reduced DAE models by Galerkin projection 111
7.6 Proper Orthogonal Decomposition 112
7.7 The Missing Point Estimation (MPE) 115
7.8 Interpolation of function snapshots 118

8 **Applications** **123**
8.1 Multirate experiments 123
8.2 Applications of Model Order Reduction 131

List of frequently used symbols **137**

Bibliography **139**

Index **145**

Summary **149**

Samenvatting **151**

Curriculum vitae **153**

Acknowledgements **155**

Chapter 1

Introduction

1.1 Motivation for the project

Electronic devices play an increasingly important role in everyday life. Just think of the computer, mobile phone, digital camera and radio, that are heavily using electronics. If one could open such a device one can easily check that inside the nice package real electronic devices can be found. For the design of these simulation techniques are essential, at the basis of which mathematics is the key ingredient.

Integrated circuits or chips are made of very small silicon slices; silicon is a semiconductor material. They communicate with the environment by means of external electronic pins. Figure 1.1 shows a typical application of a chip produced by NXP Semiconductors. The chips will process external input signals in a predefined way that is determined by its physical and topological properties. Many input signals are analogue, which means that they really occur during a certain time interval. Examples are human speech, sound

Figure 1.1: A possible application of a silicon chip produced by NXP.

Figure 1.2: Some raw material of Silion.

intensity, vibrations and all other physical phenomenons that can happen in nature. Those signals can be measured and processed directly by analogue circuits. The new generation of chips is digital, which means that they only can process digital signals. Digital signals can be obtained from analogue signals by quantification. They only have a finite number of values at a finite number of time-points. Digital circuits are e.g. the backbone of the computer. They can also be used to process analogue signals once they are converted by an A/D converter. Vice versa the produced digital signal can be converted back to an analogue output signal.

Building an integrated circuit like a computer chip is a very complex process. During this process first the silicon components are made and then these components are properly connected by a conducting metal like copper. Of course, one needs a construction plan to design a chip and test it with a computer. From the construction plans, masks with the circuit patterns are made. Under carefully monitored conditions, a pure silicon crystal is grown. Circuit manufacturing requires the use of crystals with an extremely high grade of perfection. The silicon is sawed into thin wafers with a diamond saw. The wafers are then polished in a number of steps until their surface has a perfect mirror-like finish. The silicon wafer is covered with a layer of insulating silicon oxide. A covering film of protective material that is sensitive to light is put on top of the insulating silicon oxide. UV-light is shone through a mask and onto the chip. On the parts of the chip that are hit by light, the protective material breaks apart. The wafer is developed, rinsed and baked. The development process removes the parts of the protective material exposed to light. The wafer is treated with chemicals in a process called ″etching.″ This removes the unprotected insulating material, creating a pattern of non-protected silicon wafer parts surrounded by areas protected by silicon oxide. The wafer is run through a ″doping″ process that alters the electrical properties of the unprotected areas of the wafer. These steps are repeated to build the integrated circuit, layer by layer.

Finally, when all components of the chip are done, metal is added to connect the components to each other in a process called metalization. First the conducting metal is deposited on the chip. On top of the metal a layer of UV-sensitive photo resist is added. Next a mask that describes the desired lay-out of the metal wires connecting the com-

Figure 1.3: A pure silicon crystal.

ponents of the chip is used. UV-light is shone through this mask and hits the photo resist that isn't protected by the mask. In the next step chemicals are used to remove the photo resist hit by UV-light. Another step of etching removes the metal not protected by photo resist. Today advanced circuits may need many different layers of metal to form all the necessary connections. Once the final layer of connecting metal wires has been added, the chips on the silicon wafer are tested to see if they perform as intended. Finally, the chips on the wafer are separated to form individual integrated circuits. Each chip is packed and subject to another series of tests before it can be used.
The *design* of *Integrated Circuits* has become very complex during the last decades. The transistor is the buildingblock of both analogue and digital circuits. The number of transistors on a chip has grown nearly exponentially, while the size of a chip stayed nearly constant. This implies that a single transistor can only occupy a very tiny place of the IC and thus that very detailed effects have to be taken into account. Therefore the used transistor models themselves have become very complex. A very famous observation, also called Moore's Law, says that the number of transistors per chip doubles in each 18 months. Figure 1.4 illustrates that this law has been approximately realised during the last thirty years.

The challenge of a chip manufacturer is to design a chip that will have the desired specifications. This has become a very hard task because of the increasing complexity. Before producing a chip it is necessary to analyse whether a chip design satisfies the specifications. Although this can be done by experiments on prototypes, this becomes impossible for really complex designs. Today the existing computing power allows for computer-aided design. Here mathematical circuit models are used derived from the theory of electrical circuits. In particular for analogue circuits it is still important to simulate all electrical effects. Thus circuit designs can be analysed without making a prototype. In order to produce first-time-right these simulation techniques have become essential. Figure 1.5 shows several circuit analyses that can be used in the design flow before IC fabrication. The results of the several types of simulations tell whether the design satisfies the specifications. It can be used for both optimisation and verification. In particular transient analysis is an important tool because then the full nonlinear dynamics of the circuit can be analysed.

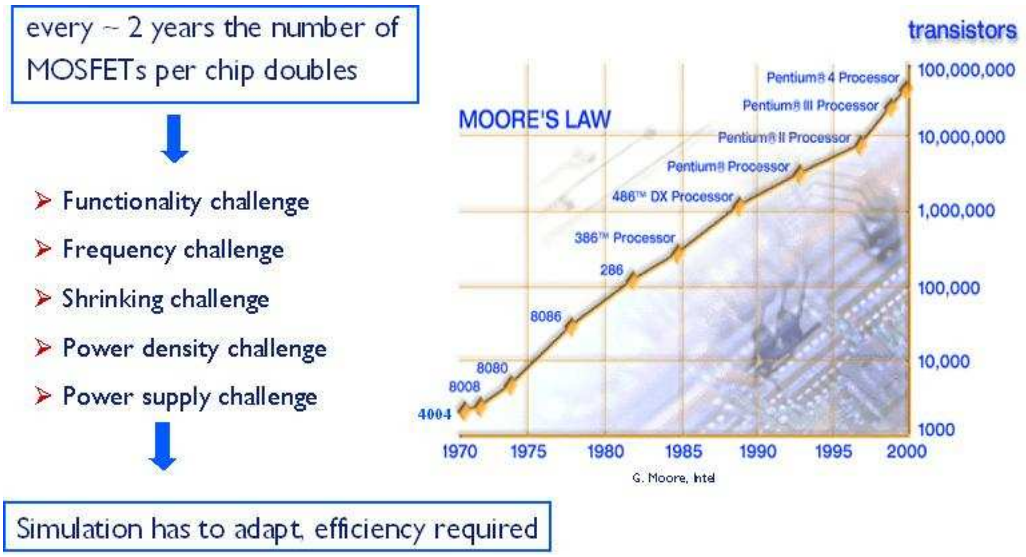
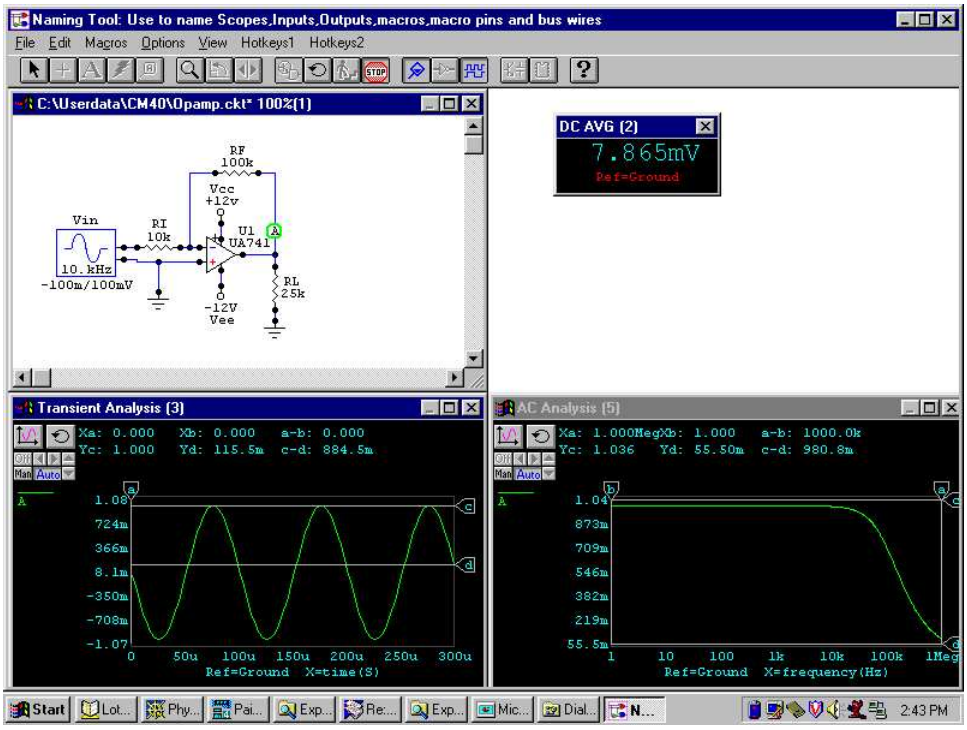


Figure 1.4: The law of Moore.

NXP Semiconductors (founded by Philips) is a chip manufacturer that also needs tools to analyse the designed integrated circuits. Its department Design Methods provides circuit simulation software, among them the in-house analogue circuit simulator Pstar that is also used at Philips. This PhD project ”High performance circuit simulation for analogue and mixed digital-analogue applications” is intended to improve the transient simulation by Pstar.

1.2 Formulation of the problem

Mathematical models play an important role in many applications, such as electronics, mechanics and control. In this thesis we will consider Integrated Circuit models consisting of so called differential-algebraic equations (DAEs). These equations are usually solved by robust implicit numerical integration methods. The time-step is determined using error control, which is based on the most active element. Classical integration methods pay a lot of attention to robust but also efficient linear algebra software, which uses features like symmetry, positive definiteness, sparsity or a specific structure like hierarchy. The resulting method may thus be very robust but not always very efficient. Because no assumption is made about the structure of the DAE, the linear algebra costs may be very high.
In particular for linear time-invariant models this is up for improvement if one can employ their nice structure. Other types for which the classical methods are not very efficient are DAEs with *redundancy*. This redundancy may come from the continuous



```
7.865mV
Ref=Ground
```

```
Xa: 0.000     Xb: 0.000     a-b: 0.000
Yc: 1.000     Yd: 115.5m    c-d: 884.5m
```

```
Xa: 1.000MegXb: 1.000     a-b: 1000.0k
Yc: 1.036     Yd: 55.50m    c-d: 980.8m
```

Figure 1.5: Circuit simulation is essential for the design world to make chips which satisfy their specifications.

DAE formulation (the real model). This means that we effectively have a much smaller solution space. But redundancy can also arise after numerical discretisation of the DAE. In this case not all unknowns need the same time-steps to guarantee at least a prescribed accuracy. *The main topic of this thesis therefore is to design methods that increase the speed of the transient analysis without loss of accuracy by exploitation of this redundancy.*

The redundancy of electrical circuit models can be exploited by using multirate time-integration or by replacing them by simpler models. First, a redundant DAE model can be reduced by model order reduction (MOR). This technique tries to find a model of smaller size that approximates the solution of the original DAE, while keeping the error sufficiently small. We study in particular methods for nonlinear DAE systems, like Proper Orthogonal Decomposition (POD) and Trajectory PieceWise Linear (TPWL). Both techniques offer a good starting point for further research on MOR of non-linear dynamical systems. The TPWL method is a very useful MOR technique to reduce the simulation time for nonlinear DAE systems. Its main advantage is the application of well-developed linear model reduction techniques. The POD method delivers reduced models that are more accurate but unfortunately also much more expensive to simulate. Hence modifications are necessary like Missing Point Estimation. This topic is studied in more detail in this thesis.

Second, the redundancy of the numerical model can be reduced by using multirate schemes. The slowly-varying unknowns are integrated at a coarse time-grid, which

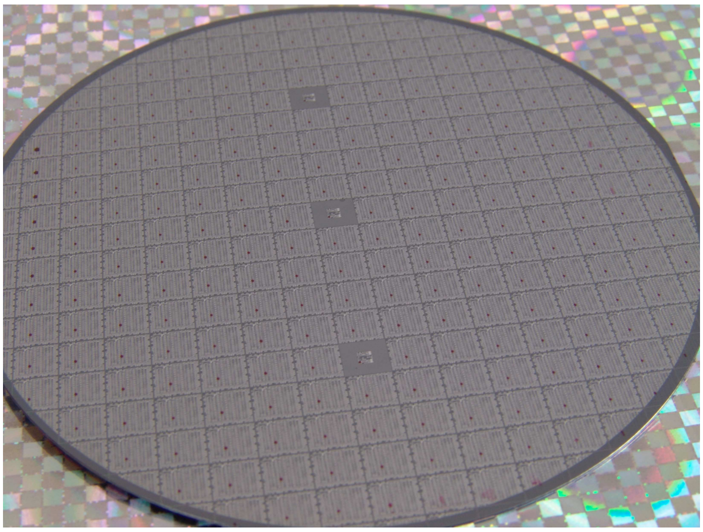


Figure 1.6: A finished wafer containing the unseparated chips.

has much larger time-steps. A nice property of multirate is that it does not use any linear structure, in contrast to MOR, but only a relaxation concept. If the coupling is sufficiently monitored and the partitioning is well chosen, multirate can be very efficient.

Multirate time-integration methods appear to be attractive for initial value problems for DAEs with latency or multirate behaviour. We study the so called Slow-Fast version of the BDF scheme because of stepsize control reasons. The BDF methods are very suitable for the interpolation at the refined time-grid. We also develop the so called Compound-Fast version that is more stable than the Slow-Fast method. Stability is always an important and necessary property of time-integration schemes, that should be conserved for multirate schemes. Therefore it is proven that the Compound-Fast - and Slow-Fast BDF multirate schemes applied to a stable test equation are stable if the subsystems are sufficiently decoupled and the active and slow parts of the system are stable and solvable. For a general partitioning the active part of a stable DAE or ODE is not automatically stable. For DAEs, moreover, even the solvability and DAE-index are not automatically preserved for the active part.

Besides stability also the accuracy is an important topic. For multirate schemes the standard theory for error estimation is no longer valid because of the coupling. We show how the local discretisation error can be estimated and controlled. Among other things we show that the interpolation errors at the interface have to be included.

Since finding a partitioning for a multirate scheme is hard another topic is to study how this can be done automatically. The found partitioning should optimise the speed-up factor of the corresponding multirate scheme. For dynamical partitionings it is allowed to update the partitioning during the transient simulation. A multirate scheme including dynamical partitioning is implemented in Pstar and tested for various circuit models. For circuit models with small high-frequency subcircuits that are coupled to relatively large low-frequency subcircuits the results are very satisfactory.

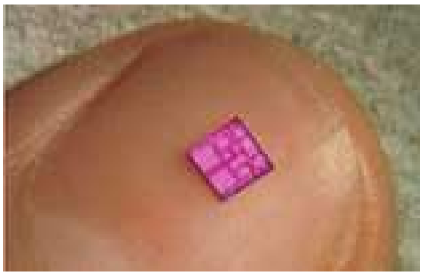


Figure 1.7: A typical integrated circuit shown on a fingernail.

1.3 Guideline to the document

In Chapter 2 the dynamics of electrical circuits are formulated as a system of differential-algebraic equations (DAE). Thus the electrical behaviour of a chip can be described mathematically by the solution of a DAE. We show how the DAE can be derived from the physical laws for the electrical circuits by means of Kirchhoff's Laws and the branch equations. Furthermore several kinds of circuit analysis are enumerated.
Chapter 3 recapitulates some well-known properties of DAEs, like uniqueness and existence of the solution, stability and the DAE-index. Because it is rarely possible to compute the solution of a DAE analytically one has to rely on numerical time integration methods. These methods discretise the time and compute the solution on the time-grid with the help of integration methods. The time-steps have to be chosen in such a way that the error will be small enough. Chapter 4 describes several numerical integration methods for DAEs, such as Linear Multistep Methods and BDF-methods. It is also shown how the local discretisation error can be estimated and controlled in an adaptive way.
Chapters 5 and 6 deal with one of the main topics of this thesis, i.e. multirate time-integration. First an introduction of multirate is given in Chapter 5. Here it is shown that not all partitionings are allowed in order to preserve stability, solvability or the DAE-index. We focus on the BDF Compound-Fast method in particular. It is explained how this method can be efficiently implemented using the Nordsieck data representation. Section 5.5 deals with the stability of the BDF Compound-Fast multirate schemes for both the one-step and multistep case. Chapter 5 ends with multirate for hierarchical circuit models, where the circuit model is modularly organised. Chapter 6 shows how the local error of these multirate schemes can be controlled by use of the stepsizes and the partitioning. First in Section 6.1 a mathematical analysis of the local error is carried out. It turns out that the interpolation errors of the boundary variables play an essential role. Then it is possible to design error and stepsize controllers for the multirate case. Chapter 6 also shows how the partitioning of a multirate method can be determined automatically or even dynamically. A good partitioning will optimise the speed-up factor of the corresponding multirate scheme. From an efficiency analysis it follows that

the optimal partitioning can be solved from a discrete optimisation problem. Because this is too complicated to solve in general, several semi-optimal methods are suggested. Finally it is described how the partitioning can be updated dynamically during the simulation.

Chapter 7 deals with the other main topic of this thesis, i.e. model order reduction. First a general overview and a recapitulation of the MOR theory for linear and nonlinear DAE-systems are given. Because MOR methods of Galerkin type only reduce the dimension and not the simulation time, Sections 7.7 and 7.8 show some possible solutions, like Missing Point Estimation and interpolation.

Finally, Chapter 8 highlights numerical results of multirate time-integration and nonlinear model order reduction for several circuit models. Because the BDF Compound-Fast multirate scheme has been implemented in Pstar, also some large industrial test cases are included.

Chapter 2

Mathematical models for electrical circuits

2.1 Theory of electrical circuits

Before a circuit can be analyzed it is necessary to have a proper mathematical model. This section describes a well-known method, Modified Nodal Analysis (MNA), that is able to model a circuit by a differential-algebraic equation [13, 24, 43]. In general, *electrical circuits* consist of *electrical components* and connecting conductors. Because the electrical resistance of the conductors is negligible, only the junctions of the conductors are important. Each junction (or node) in the circuit has its nodal voltage or potential V. If two nodes in the network have different potentials, there is a voltage difference v between these nodes. A voltage difference will result in a current $\imath$ to the highest potential, that aims to level out the voltages. Conversely, a current implies a voltage difference. By including voltage and/or current sources one can establish a non-trivial voltage and current distribution. This distribution obeys the Kirchhoff current and voltage laws as is explained next.

It is possible to consider a circuit as a graph of two types of elements: *nodes* and *branches*. The branches represent the components, while the nodes represent the junctions of the conductors. Each branch is connected to a pair of nodes. For this branch these nodes are divided into positive and negative nodes. The voltage difference v across a branch is equal to $V^+ - V^-$, where V^+ and V^- are the potentials of the positive and negative node, respectively. The direction of a positive current is always to the positive nodes. If the current is negative, it means a positive current to the negative nodes.

Assume that the *network* consists of n nodes and b branches. Structurally this can be stored in the *topology matrix* $\mathbf{A} \in \mathbb{R}^{n \times b}$, that is defined by

$$\mathbf{A}_{ij} = \begin{cases} 1 & \text{if branch } j \text{ is incident at node } i \text{ and node } i \text{ is positive node of } j, \\ -1 & \text{if branch } j \text{ is incident at node } i \text{ and node } i \text{ is negative node of } j, \\ 0 & \text{if branch } j \text{ is not incident at node } i. \end{cases}$$

Introduce the vector $\mathbf{v}_n \in \mathbb{R}^n$ that contains the nodal voltages, arranged in the same order as the rows of $\mathbf{A}$. Introduce furthermore the vector $\mathbf{i}_b \in \mathbb{R}^b$ and $\mathbf{v}_b \in \mathbb{R}^b$ that contain the branch currents and branch voltage differences in the same order as the columns of $\mathbf{A}$. Note that in general $\mathbf{v}_n = \mathbf{v}_n(t), \mathbf{i}_b = \mathbf{i}_b(t)$ and $\mathbf{v}_b = \mathbf{v}_b(t)$ are time-dependent functions. We will show that these variables are sufficient to model the circuit.
There are two types of equations from physics, that together describe the circuit. First of all, there are the balance laws or the laws of Kirchhoff. These equations have a topological character, because they do not depend on the type of the components, but only on the topology of the circuit.

Kirchhoff's Current Law (KCL):	The algebraic sum of all branch currents leaving a node is zero at all instants of time.
Kirchhoff's Voltage Law (KVL):	The algebraic sum of all voltage differences around any closed loop of a network is zero at all instants of time.

With $\mathbf{A}, \mathbf{v}_n, \mathbf{v}_b$ and $\mathbf{i}_b$ it is easy to formulate KCL and KVL:

$$\begin{aligned} \mathbf{A}\mathbf{i}_b &= \mathbf{0}, && \text{(KCL)}, \\ \mathbf{A}^T\mathbf{v}_n &= \mathbf{v}_b. && \text{(KVL)}. \end{aligned} \tag{2.1}$$

The *constitutive relations* (or *branch equations*) depend on the type of the component. There are many types of components, but here only the basic circuit components have been considered. Let $\mathbf{i}_b$ and $\mathbf{v}_b$ be the vectors that consist of all branch currents and voltage differences across the branches. In general each equation of the components can then be described in an implicit way:

$$f_k(\mathbf{v}_b, \frac{d\mathbf{v}_b}{dt}, \mathbf{i}_b, \frac{d\mathbf{i}_b}{dt}, t) = 0, \quad k = 1, \dots, b \tag{2.2}$$

Note that this is a very general formulation indeed, because branch equations are locally defined. This means that f_k only depends on the quantities $\mathbf{v}_b(k), \mathbf{i}_b(k)$ in branch k. Electrical components can be *current-defined* or *voltage-defined*. Capacitors and current sources are current-defined, inductors and voltage sources are voltage-defined, while resistors can be of both types. Let the vectors $\mathbf{v}_b \in \mathbb{R}^b$ and $\mathbf{i}_b \in \mathbb{R}^b$ be split into two parts

$$\mathbf{v}_b = \begin{pmatrix} \mathbf{v}_{b_1} \\ \mathbf{v}_{b_2} \end{pmatrix}, \quad \mathbf{i}_b = \begin{pmatrix} \mathbf{i}_{b_1} \\ \mathbf{i}_{b_2} \end{pmatrix}, \tag{2.3}$$

where $\mathbf{v}_{b_1}, \mathbf{i}_{b_1} \in \mathbb{R}^{b_1}$ and $\mathbf{v}_{b_2}, \mathbf{i}_{b_2} \in \mathbb{R}^{b_2}$ consists of the branch voltages and branch currents of the current-defined and voltage-defined elements, respectively. Define the functions $\hat{\mathbf{q}} : \mathbb{R}\times\mathbb{R}^b\times\mathbb{R}^{b_2} \to \mathbb{R}^b$ and $\check{\mathbf{q}} : \mathbb{R}\times\mathbb{R}^b\times\mathbb{R}^{b_2} \to \mathbb{R}^b$ such that all current-defined elements satisfy

$$\forall_{k=1,\dots,b_1}\ \mathbf{i}_{b_1}(k) = \frac{d}{dt}\hat{q}_k(t,\mathbf{v}_b,\mathbf{i}_{b_2}) + \hat{j}_k(t,\mathbf{v}_b,\mathbf{i}_{b_2}), \tag{2.4}$$

and all voltage-defined elements satisfy

$$\forall_{k=1,\dots,b_2}\ \mathbf{v}_{b_2}(k) = \frac{d}{dt}\check{q}_k(t,\mathbf{v}_b,\mathbf{i}_{b_2}) + \check{j}_k(t,\mathbf{v}_b,\mathbf{i}_{b_2}). \tag{2.5}$$

Current-defined elements are also called *voltage-controlled components*, while voltage-defined elements are also called *current-controlled components*. Let the matrix $\mathbf{A}$ be partitioned into two parts:

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 & \mathbf{A}_2 \end{pmatrix}.$$

In the new variables the balance laws can be written as

$$\mathbf{A}_1\mathbf{i}_{b_1} + \mathbf{A}_2\mathbf{i}_{b_2} = \mathbf{0}, \quad \text{(KCL)} \tag{2.6}$$

$$\left.\begin{aligned} \mathbf{A}_1^T\mathbf{v}_n &= \mathbf{v}_{b_1}, \\ \mathbf{A}_2^T\mathbf{v}_n &= \mathbf{v}_{b_2}. \end{aligned}\right\} \quad \text{(KVL)} \tag{2.7}$$

2.2 Modified nodal analysis (MNA)

In the previous section equations are derived that describe the circuit. In this section the functions $\mathbf{q}$ and $\mathbf{j}$ will be derived, such that the circuit is modeled by the following system of differential-algebraic equations (DAE)

$$\frac{d}{dt}\mathbf{q}(t,\mathbf{x}(t)) + \mathbf{j}(t,\mathbf{x}(t)) = \mathbf{0}. \tag{2.8}$$

Circuits without voltage-defined elements as in (2.4) are usually modeled by *Nodal Analysis*. However, for the general case we need an extension which is also called *Modified Nodal Analysis (MNA)*. We already saw that the constitutive relations of the two types of components can be formulated as

$$\left.\begin{aligned} \mathbf{i}_{b_1} &= \tfrac{d}{dt}\hat{\mathbf{q}}(t,(\mathbf{v}_{b_1}^T,\mathbf{v}_{b_2}^T)^T,\mathbf{i}_{b_2}) + \hat{\mathbf{j}}(t,(\mathbf{v}_{b_1}^T,\mathbf{v}_{b_2}^T)^T,\mathbf{i}_{b_2}), \\ \mathbf{v}_{b_2} &= \tfrac{d}{dt}\check{\mathbf{q}}(t,(\mathbf{v}_{b_1}^T,\mathbf{v}_{b_2}^T)^T,\mathbf{i}_{b_2}) + \check{\mathbf{j}}(t,(\mathbf{v}_{b_1}^T,\mathbf{v}_{b_2}^T)^T,\mathbf{i}_{b_2}). \end{aligned}\right\} \quad \text{(CR)} \tag{2.9}$$

Left-multiplying the first equation by $\mathbf{A}_1$ and using KCL results in

$$-\mathbf{A}_2\mathbf{i}_{b_2} = \frac{d}{dt}\mathbf{A}_1\hat{\mathbf{q}}(t,(\mathbf{v}_{b_1}^T,\mathbf{v}_{b_2}^T)^T,\mathbf{i}_{b_2}) + \mathbf{A}_1\hat{\mathbf{j}}(t,(\mathbf{v}_{b_1}^T,\mathbf{v}_{b_2}^T)^T,\mathbf{i}_{b_2}).$$

Using KVL, we can express $\mathbf{v}_{b_1}$ and $\mathbf{v}_{b_2}$ in terms of $\mathbf{v}_n$. Because $(\mathbf{v}_{b_1}^T,\mathbf{v}_{b_2}^T)^T = (\mathbf{v}_n^T\mathbf{A}_1,\mathbf{v}_n^T\mathbf{A}_2)^T = \mathbf{A}^T\mathbf{v}_n$, this yields

$$\left.\begin{aligned} -\mathbf{A}_2\mathbf{i}_{b_2} &= \tfrac{d}{dt}\mathbf{A}_1\hat{\mathbf{q}}(t,\mathbf{A}^T\mathbf{v}_n,\mathbf{i}_{b_2}) + \mathbf{A}_1\hat{\mathbf{j}}(t,\mathbf{A}^T\mathbf{v}_n,\mathbf{i}_{b_2}), \\ \mathbf{A}_2^T\mathbf{v}_n &= \tfrac{d}{dt}\check{\mathbf{q}}(t,\mathbf{A}^T\mathbf{v}_n,\mathbf{i}_{b_2}) + \check{\mathbf{j}}(t,\mathbf{A}^T\mathbf{v}_n,\mathbf{i}_{b_2}). \end{aligned}\right\} \tag{2.10}$$

The result is a system of $n + b_2$ equations for the $n + b_2$ elements of $\mathbf{v}_n, \mathbf{i}_{b_2}$. It is well-known that for a connected circuit the rank of its topology matrix $\mathbf{A}$ equals $n - 1$ [24]. This is caused by the property that the column sum of each column of $\mathbf{A}$ is zero. This means that $\mathbf{v}_n$ is not yet uniquely determined by equation (2.10). This is the reason why at least one node has to be *grounded*. Assume, the k-th element of $\mathbf{v}_n$ is grounded at value V^*. In that case, the k-th row of $\mathbf{A}, \mathbf{A}_1, \mathbf{A}_2$, the k-th column of $\mathbf{A}, \mathbf{A}_1$ and the k-th coordinate of $\mathbf{v}_n$ have to be removed. This will lead to a new system of $n + b_2 - 1$ equations for the $n + b_2 - 1$ elements of $\hat{\mathbf{v}}_n, \mathbf{i}_{b_2}$. For most connected circuit models that occur in practice this system will be uniquely solvable, although this is not the case for general DAE models. Let $\mathbf{e}_k \in \mathbb{R}^{n-1}$ be the k-th unit vector, then we get the additional condition $\mathbf{e}_k^T \mathbf{v}_n = V^*$. Thus we can write $\mathbf{v}_n = \mathbf{P}_k^T \hat{\mathbf{v}}_n + \mathbf{e}_k V^*$, where $\hat{\mathbf{v}}_n \in \mathbb{R}^{n-1}$ and $\mathbf{P}_k \in \{0,1\}^{(n-1)\times n}$ is a selection operator with $\mathbf{P}_k \mathbf{e}_k = \mathbf{0}$. Then we get

$$\left.\begin{aligned} -\mathbf{P}_k \mathbf{A}_2 \mathbf{i}_{b_2} &= \tfrac{d}{dt} \mathbf{P}_k \mathbf{A}_1 \hat{\mathbf{q}}(t, \mathbf{A}^T \mathbf{P}_k^T \hat{\mathbf{v}}_n + \mathbf{A}^T \mathbf{e}_k V^*, \mathbf{i}_{b_2}) + \mathbf{P}_k \mathbf{A}_1 \hat{\mathbf{j}}(t, \mathbf{A}^T \mathbf{P}_k^T \hat{\mathbf{v}}_n + \mathbf{A}^T \mathbf{e}_k V^*, \mathbf{i}_{b_2}), \\ \mathbf{A}_2^T \mathbf{P}_k^T \hat{\mathbf{v}}_n + \mathbf{A}_2^T \mathbf{e}_k V^* &= \tfrac{d}{dt} \check{\mathbf{q}}(t, \mathbf{A}^T \mathbf{P}_k^T \hat{\mathbf{v}}_n + \mathbf{A}^T \mathbf{e}_k V^*, \mathbf{i}_{b_2}) + \check{\mathbf{j}}(t, \mathbf{A}^T \mathbf{P}_k^T \hat{\mathbf{v}}_n + \mathbf{A}^T \mathbf{e}_k V^*, \mathbf{i}_{b_2}). \end{aligned}\right\} \tag{2.11}$$

In practice the ground value V^* of a circuit is always equal to zero. Define the state vector $\mathbf{x} = (\mathbf{x}_1^T, \mathbf{x}_2^T)^T = \left(\hat{\mathbf{v}}_n^T, \mathbf{i}_{b_2}^T\right)^T$ of dimension d and the functions $\mathbf{q}, \mathbf{j} : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$, such that

$$\mathbf{q}(t, \mathbf{x}) = \begin{pmatrix} \mathbf{P}_k \mathbf{A}_1 \hat{\mathbf{q}}(t, \mathbf{A}^T \mathbf{P}_k^T \mathbf{x}_1, \mathbf{x}_2) \\ \check{\mathbf{q}}(t, \mathbf{A}^T \mathbf{P}_k^T \mathbf{x}_1, \mathbf{x}_2) \end{pmatrix},$$

$$\mathbf{j}(t, \mathbf{x}) = \begin{pmatrix} \mathbf{P}_k \mathbf{A}_1 \hat{\mathbf{j}}(t, \mathbf{A}^T \mathbf{P}_k^T \mathbf{x}_1, \mathbf{x}_2) + \mathbf{P}_k \mathbf{A}_2 \mathbf{x}_2 \\ \check{\mathbf{j}}(t, \mathbf{A}^T \mathbf{P}_k^T \mathbf{x}_1, \mathbf{x}_2) - \mathbf{A}_2^T \mathbf{P}_k^T \mathbf{x}_1 \end{pmatrix}.$$

Now the grounded circuit is mathematically described by the system of differential-algebraic equations (2.8) indeed.
It is possible to expand $\mathbf{q}$ and $\mathbf{j}$ as sums of local contributions of all components [31].

$$\mathbf{q}(t, \mathbf{x}) = \sum_{\text{elements}} \mathbf{A}_e \mathbf{q}_e(t, \mathbf{B}_e^T \mathbf{x}), \quad \mathbf{j}(t, \mathbf{x}) = \sum_{\text{elements}} \mathbf{A}_e \mathbf{j}_e(t, \mathbf{B}_e^T \mathbf{x}). \tag{2.12}$$

Here $\mathbf{q}_e, \mathbf{j}_e$ are the functions that model the branches themselves and $\mathbf{A}_e, \mathbf{B}_e$ are local extraction and selection mappings. Often $\mathbf{A}_e = \mathbf{B}_e$. For an ordinary capacitor $\mathbf{q}_e$ is simply a scalar. The sizes of $\mathbf{q}_e, \mathbf{j}_e$ are equal to 1 and 2 for current-defined and voltage-defined elements, respectively. Thus, in general the *Jacobian matrices*

$$\mathbf{C}(t, \mathbf{x}) = \frac{\partial \mathbf{q}(t, \mathbf{x})}{\partial \mathbf{x}} = \sum_{\text{elements}} \mathbf{A}_e \mathbf{C}_e(t, \mathbf{B}_e^T \mathbf{x}) \mathbf{B}_e^T, \tag{2.13}$$

$$\mathbf{G}(t, \mathbf{x}) = \frac{\partial \mathbf{j}(t, \mathbf{x})}{\partial \mathbf{x}} = \sum_{\text{elements}} \mathbf{A}_e \mathbf{G}_e(t, \mathbf{B}_e^T \mathbf{x}) \mathbf{B}_e^T \tag{2.14}$$

are sparse matrices, formed by low-rank updates.

2.3 Circuit analyses

After deriving the model of an electrical circuit, that is determined by the functions $\mathbf{q}$ and $\mathbf{j}$, one is able to analyse the circuit. In this section several types of *circuit analysis-methods* are described.

The *Direct current (DC) analysis* computes the time-independent steady-state solution $\mathbf{x}_{DC}$ of the circuit. In a steady-state there are only time-invariant equations. This means that

$$\begin{aligned} \mathbf{q}(t,\mathbf{x}) &= \mathbf{q}_{DC}(\mathbf{x}), \quad & \mathbf{C}(t,\mathbf{x}) &= \mathbf{C}_{DC}(\mathbf{x}), \\ \mathbf{j}(t,\mathbf{x}) &= \mathbf{j}_{DC}(\mathbf{x}), \quad & \mathbf{G}(t,\mathbf{x}) &= \mathbf{G}_{DC}(\mathbf{x}). \end{aligned}$$

Furthermore the *steady-state* solution has the property:

$$\dot{\mathbf{x}}_{DC} = \mathbf{0}.$$

This implies that $\frac{d}{dt}\mathbf{q}_{DC}(\mathbf{x}_{DC}) = \frac{\partial}{\partial \mathbf{x}_{DC}}\mathbf{q}_{DC}(\mathbf{x}_{DC})\frac{\partial \mathbf{x}_{DC}}{\partial t} = \mathbf{0}$. Hence the steady-state equation to be solved is the algebraic equation

$$\mathbf{j}_{DC}(\mathbf{x}_{DC}) = \mathbf{0}. \tag{2.15}$$

This is a nonlinear equation in general, that can be solved e.g. by the Newton method. If the equation is linear, Gaussian elimination is sufficient.

The *Alternating Current (AC) analysis* considers the effect of applying small signal perturbations $\mathbf{e}(t)$, with $\mathbf{e}(0) = \mathbf{0}$, to the equation of the DC-solution

$$\frac{d}{dt}\mathbf{q}_{DC}(\mathbf{x}(t)) + \mathbf{j}_{DC}(\mathbf{x}(t)) - \mathbf{e}(t) = \mathbf{0}, \quad \mathbf{x}(0) = \mathbf{x}_{DC}. \tag{2.16}$$

Note that the function $\mathbf{q}$ does not depend explicitly on t. At $t = 0$, the solution $\mathbf{x}(t)$ is equal to the steady-state solution $\mathbf{x}_0 = \mathbf{x}_{DC}$, while the dynamic behaviour is caused by an independent small sine-wave excitation $\mathbf{e}(t)$, that is added to the circuit as a source function. It is convenient to consider a sine-wave as the imaginary part of the complex harmonic function. The imaginary part of a complex vector $\mathbf{z}$ is defined as $\Im[\mathbf{z}]$. Thus

$$\mathbf{e}(t) = \mathbf{e}^* \sin(\omega_0 t) = \mathbf{e}^* \Im[e^{\mathbf{i}\omega_0 t}],$$

with $\mathbf{e}^* \in \mathbb{R}^n$, $\omega_0 \in \mathbb{R}$ the angular frequency of $\mathbf{e}(t)$ and $\mathbf{i} \in \mathbb{C}$ the complex unity. Because $\mathbf{e}^*$ is small, $\|\mathbf{e}^*\| << 1$, the solution will have small differences with the steady-state solution. Introduce $\mathbf{y}(t) = \mathbf{x}(t) - \mathbf{x}_{DC}$, then

$$\mathbf{q}_{DC}(\mathbf{x}(t)) \doteq \mathbf{q}_{DC}(\mathbf{x}_{DC}) + \frac{\partial \mathbf{q}_{DC}}{\partial \mathbf{x}}(\mathbf{x}_{DC})\mathbf{y}(t),$$

$$\mathbf{j}_{DC}(\mathbf{x}(t)) \doteq \mathbf{j}_{DC}(\mathbf{x}_{DC}) + \frac{\partial \mathbf{j}_{DC}}{\partial \mathbf{x}}(\mathbf{x}_{DC})\mathbf{y}(t).$$

Since $\mathbf{j}_{DC}(\mathbf{x}_{DC}) = \mathbf{0}$, we obtain in first order

$$\begin{aligned} \frac{d}{dt}\mathbf{q}_{DC}(\mathbf{x}(t)) + \mathbf{j}_{DC}(\mathbf{x}(t)) &\doteq \frac{d}{dt}\left(\mathbf{q}_{DC}(\mathbf{x}_{DC}) + \mathbf{C}_{DC}\mathbf{y}(t)\right) + \mathbf{j}_{DC}(\mathbf{x}_{DC}) + \mathbf{G}_{DC}\mathbf{y}(t) \\ &= \mathbf{C}_{DC}\dot{\mathbf{y}}(t) + \mathbf{G}_{DC}\mathbf{y}(t). \end{aligned}$$

If

$$\mathbf{C}_{DC}\dot{\mathbf{y}}_{AC}(t)+\mathbf{G}_{DC}\mathbf{y}_{AC}(t)=\mathbf{e}^*e^{\mathbf{i}\omega_0 t}\quad \mathbf{y}_{AC}(0)=\mathbf{0}, \tag{2.17}$$

the AC-solution of the circuit is given by $\mathbf{x}_{AC}(t)=\mathbf{x}_0+\Im[\mathbf{y}_{AC}(t)]$. Since $\mathbf{y}_{AC}(0)=0$ we can use the Fourier transforms $\mathcal{F}\{\mathbf{y}_{AC}(t)\}(\omega)=\boldsymbol{y}_{AC}$, $\mathcal{F}\{\dot{\mathbf{y}}_{AC}(t)\}(\omega)=\mathbf{i}\omega\boldsymbol{y}_{AC}(\omega)$ and $\mathcal{F}\{\mathbf{e}(t)\}(\omega)=\mathbf{e}^*\delta(\omega-\omega_0)$. Thus in the frequency domain (2.17) is equivalent to

$$(\mathbf{i}\omega\mathbf{C}_{DC}+\mathbf{G}_{DC})\boldsymbol{y}_{AC}(\omega)=\mathbf{e}^*\delta(\omega-\omega_0).$$

We can write $\boldsymbol{y}_{AC}(\omega)=\mathbf{z}_{AC}\delta(\omega-\omega_0)$, where $\mathbf{z}_{AC}\in\mathbb{C}^n$ solves the complex linear system

$$(\mathbf{i}\omega_0\mathbf{C}_{DC}+\mathbf{G}_{DC})\mathbf{z}_{AC}=\mathbf{e}^*.$$

An AC analysis solves this last system.

The *Transient analysis* treats the full real nonlinear circuit. In general, one is interested in the behaviour of the system on the time interval $[0,T]$. Normally the initial state $\mathbf{x}_0$ is known. It could be the steady-state solution to study large signal perturbations, but this is not necessary. For a transient analysis the *initial value problem (IVP)*

$$\left\{\begin{array}{c}\frac{d}{dt}\mathbf{q}(t,\mathbf{x}(t))+\mathbf{j}(t,\mathbf{x}(t))=\mathbf{0},\\ \mathbf{x}(0)=\mathbf{x}_0\end{array}\right. \tag{2.18}$$

has to be solved. If $\mathbf{x}_0=\mathbf{x}_{DC}$ we have a sequence of analyses: first solve (2.15) ("implicit DC") and then integrate (2.18) in time. The functions $\mathbf{q}:[0,T]\times\mathbb{R}^d\to\mathbb{R}^d$ and $\mathbf{j}:[0,T]\times\mathbb{R}^d\to\mathbb{R}^d$ can be derived from physical laws. The solution $\mathbf{x}(t)$ describes the dynamic behaviour of the system for a known initial value, for example the steady-state. Because (2.18) is a differential-algebraic equation, not all initial states are allowed. If $\mathbf{q}$ is independent of t, the steady-state solution at $t=0$ is always a consistent initial solution. The initial value problem can be solved by numerical tools like Linear Multistep Methods or Runge Kutta methods. Due to the algebraic equations and the possibly stiff behaviour, it is necessary to use implicit methods. In chapter four, the numerical tools for the transient analysis are investigated.

In many circuits there exists a periodic solution. This solution is often called the *periodic steady-state*. In this case, instead of an initial value, a periodicity constraint is specified. In the periodic steady-state, the functions $\mathbf{q}(t,\mathbf{x}(t))$ and $\mathbf{j}(t,\mathbf{x}(t))$ are periodic functions with respect to t. Then the *two-point boundary value problem* (BVP)

$$\left\{\begin{array}{c}\frac{d}{dt}\mathbf{q}(t,\mathbf{x}(t))+\mathbf{j}(t,\mathbf{x}(t))=\mathbf{0},\\ \mathbf{x}(0)=\mathbf{x}(T)\end{array}\right. \tag{2.19}$$

has to be solved. Here T is the *period* of the solution. If the functions $\mathbf{q}$ and $\mathbf{j}$ are periodic, the period T can be derived. But if that is not the case it is still possible to have periodic solutions. Sometimes T is known, but T could also be an additional unknown of the system. In that case the *free oscillator problem* has to be solved [25]. Then an additional equation is solved for the timeshift by setting a specific coordinate $\mathbf{x}_k(0)=\mathbf{x}_{k,o}$ to a special value (that should be in the range of $\mathbf{x}_k(t)$). These problems, with periodicity constraints, are more difficult to solve, because there are no initial conditions. If the period T is known, one could use e.g. a shooting method to solve this problem. For unknown period, a nonlinear eigenvalue problem has to be solved, that is more difficult. However, for linear systems it is sufficient to determine the eigenvalues of the system that determine the possible periods.

Chapter 3

Differential-algebraic equations (DAEs)

3.1 Introduction

Many physical applications, like electronics and mechanics, can be very well modeled by dynamical systems consisting of differential equations. If the system also included pure algebraic equations, one speaks about *differential-algebraic equations*. In this chapter we will consider the following DAE with state vector $\mathbf{x} : [0, T] \to \mathbb{R}^d$

$$\begin{cases} \frac{d}{dt}\mathbf{q}(t, \mathbf{x}(t)) + \mathbf{j}(t, \mathbf{x}(t)) = \mathbf{0}, \\ \mathbf{x}(0) = \mathbf{x}_0. \end{cases} \tag{3.1}$$

The functions $\mathbf{q} : [0, T] \times \mathbb{R}^d \to \mathbb{R}^d, \mathbf{j} : [0, T] \times \mathbb{R}^d \to \mathbb{R}^d$ and the Jacobian matrices $\mathbf{C}(t, \mathbf{x}) = \frac{\partial \mathbf{q}}{\partial \mathbf{x}}(t, \mathbf{x})$ and $\mathbf{G}(t, \mathbf{x}) = \frac{\partial \mathbf{j}}{\partial \mathbf{x}}(t, \mathbf{x})$ can be derived from physical laws. If the system describes an electrical circuit, MNA can be used to derive $\mathbf{q}, \mathbf{j}$ and $\mathbf{C}, \mathbf{G}$ as is explained in Chapter 2. The solution $\mathbf{x}(t)$ of (3.1) describes the dynamic behaviour of the system for a known initial value, for example the steady state.
A special case of the DAEs is the *ordinary differential equation (ODE)*. In that case, a transient analysis computes the solution of the initial value problem

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}), \\ \mathbf{x}(0) = \mathbf{x}_0. \end{cases} \tag{3.2}$$

with $\mathbf{f} : [0, T] \times \mathbb{R}^d \to \mathbb{R}^d$.
The DAE is called *semi-explicit* if it can be written as:

$$\begin{cases} \dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{x}, \mathbf{y}), \\ \mathbf{0} = \mathbf{g}(t, \mathbf{x}, \mathbf{y}). \end{cases}$$

The benefit of this type is the strict separation between the differential equations and the algebraic equations. After introducing the new variable $\mathbf{y} = \mathbf{q}(t, \mathbf{x})$, the DAE (3.1) can easily be written into the semi-explicit form

$$\begin{cases} \dot{\mathbf{y}} = -\mathbf{j}(t, \mathbf{x}), \\ \mathbf{0} = \mathbf{y} - \mathbf{q}(t, \mathbf{x}). \end{cases}$$

The price of this transformation is that the number of unknowns is doubled. There are more representations of (3.1). Expanding the derivative of $\mathbf{q}$ and using the Jacobian matrix $\mathbf{C}(t, \mathbf{x}) = \frac{\partial \mathbf{q}(t,\mathbf{x})}{\partial \mathbf{x}}$ results in

$$\mathbf{C}(t, \mathbf{x})\dot{\mathbf{x}} + \frac{\partial \mathbf{q}}{\partial t}(t, \mathbf{x}) + \mathbf{j}(t, \mathbf{x}) = \mathbf{0}. \tag{3.3}$$

From this representation , it follows that if $\mathbf{C}(t, \mathbf{x})$ is invertible for all $\mathbf{x}$, the DAE can be written as an ODE. But in practice, $\mathbf{C}(t, \mathbf{x})$ will almost always be singular, because of the algebraic equations. Then the solution has to satisfy a number of algebraic equations. Because these algebraic equations also apply at $t = 0$, a proper initial solution also has to satisfy the algebraic equations. Such initial solution is called *consistent*. For DAE-index 1 the stationary solution is a consistent solution, but a general initial value is not. For a higher index DAE also constraints on time derivatives of $\mathbf{x}$ at $t = 0$ have to be met.

3.2 Uniqueness and existence of the solution

For ODEs of type (3.2) it is well-known that there exists a unique continuously differentiable solution if $\mathbf{f}(t, \mathbf{x})$ is continuous on $[0, T]$ and *Lipschitz continuous* with respect to $\mathbf{x}$ [32]. For DAEs of type (3.1) we need different conditions for $\mathbf{q}$ and $\mathbf{j}$.
Here we can use *Banach's contraction theorem* that states that each operator $\mathcal{T} : S \rightarrow S$ has a unique fixed point, if S is a closed Banach space and $\|\mathcal{T}\mathbf{x} - \mathcal{T}\mathbf{y}\| \leq \alpha\|\mathbf{x} - \mathbf{y}\|$ for $\alpha < 1$ (see also [37]).
Because $\mathbf{q}$ does not always have an inverse, we use the function

$$\mathbf{g}(t, \mathbf{x}) := \mathbf{q}(t, \mathbf{x}) + \lambda\mathbf{j}(t, \mathbf{x}). \tag{3.4}$$

Here $\lambda > 0$ is an arbitrary positive constant, such that $\mathbf{g}$ is an invertible function with inverse $\mathbf{g}_{\text{inv}}$. Let $L > 0$ and $S > 0$ be the Lipschitz constants of $\mathbf{j}$ and $\mathbf{g}_{\text{inv}}$, respectively. If $\mathbf{q}, \mathbf{j}$ and $\mathbf{g}_{\text{inv}}$ are continuously differentiable with Jacobian matrices $\mathbf{C}(t, \mathbf{x}), \mathbf{G}(t, \mathbf{x})$, the Lipschitz constants L, S can be defined as

$$\begin{aligned} L &:= \max\{\|\mathbf{G}(t, \mathbf{x})\|, (t, \mathbf{x}) \in \mathcal{I} \times \Omega\}, && (3.5) \\ S &:= \max\{\| [\mathbf{C}(t, \mathbf{x}) + \lambda\mathbf{G}(t, \mathbf{x})]^{-1} \|, (t, \mathbf{x}) \in \mathcal{I} \times \Omega\}. && (3.6) \end{aligned}$$

Theorem 3.1 *The DAE (3.1) has a unique solution for* $t \in [0, t_{\text{end}})$ *with* $t_{\text{end}} > 0$ *if the functions* $\mathbf{j}$ *and* $\mathbf{g}_{\text{inv}}$ *are Lipschitz continuous with finite Lipschitz constants* L, S.

Proof: Consider two solutions $\mathbf{x}, \mathbf{y}$ that satisfy

$$\left\{ \begin{array}{lcl} \frac{d}{dt}[\mathbf{q}(t,\mathbf{x})] + \mathbf{j}(t,\mathbf{x}) & = & 0, \quad \mathbf{x}(0) = \mathbf{x}_0, \\ \frac{d}{dt}[\mathbf{q}(t,\mathbf{y})] + \mathbf{j}(t,\mathbf{y}) & = & 0, \quad \mathbf{y}(0) = \mathbf{x}_0, \end{array} \right. \tag{3.7}$$

respectively. Since $\mathbf{q}(0,\mathbf{x}(0)) = \mathbf{q}(0,\mathbf{y}(0)) = \mathbf{q}(0,\mathbf{x}_0) = \mathbf{q}_0$ we can write

$$\left\{ \begin{array}{lcl} \mathbf{q}(T,\mathbf{x}(T)) - \mathbf{q}_0 + \int_0^T \mathbf{j}(\tau,\mathbf{x}(\tau))d\tau & = & \mathbf{0}, \\ \mathbf{q}(T,\mathbf{y}(T)) - \mathbf{q}_0 + \int_0^T \mathbf{j}(\tau,\mathbf{y}(\tau))d\tau & = & \mathbf{0}, \end{array} \right.$$

and

$$\mathbf{q}(T,\mathbf{x}(T)) - \mathbf{q}(T,\mathbf{y}(T)) + \int_0^T [\mathbf{j}(\tau,\mathbf{x}(\tau)) - \mathbf{j}(\tau,\mathbf{y}(\tau))]\, d\tau = \mathbf{0}. \tag{3.8}$$

Using (3.4), this equation (3.8) is equivalent to

$$\mathbf{g}(T,\mathbf{x}(T)) - \mathbf{g}(T,\mathbf{y}(T)) - \lambda(\mathbf{j}(T,\mathbf{x}(T)) - \mathbf{j}(T,\mathbf{y}(T))) + \int_0^T [\mathbf{j}(\tau,\mathbf{x}(\tau)) - \mathbf{j}(\tau,\mathbf{y}(\tau))]\, d\tau = \mathbf{0}.$$

If $\mathbf{g}$ is differentiable, there exists a $\mathbf{c} \in \mathbb{R}^d$ for which

$$\frac{\partial \mathbf{g}}{\partial \mathbf{x}}(T,\mathbf{c})(\mathbf{x}(T) - \mathbf{y}(T)) = \lambda(\mathbf{j}(T,\mathbf{x}(T)) - \mathbf{j}(T,\mathbf{y}(T))) - \int_0^T [\mathbf{j}(\tau,\mathbf{x}(\tau)) - \mathbf{j}(\tau,\mathbf{y}(\tau))]\, d\tau.$$

Finally we get

$$\|\mathbf{x}(T) - \mathbf{y}(T)\| \leq \| \left[\frac{\partial \mathbf{g}}{\partial \mathbf{x}}(T,\mathbf{c})\right]^{-1} \| (\lambda + T) \|\mathbf{j}(t,\mathbf{x}(t)) - \mathbf{j}(t,\mathbf{y}(t))\|_{[0,T]}.$$

Because $\mathbf{j}$ and $\mathbf{g}_{\text{inv}}$ are Lipschitz continuous with finite Lipschitz constants $L > 0$ and $S > 0$, we obtain

$$\|\mathbf{x}(t) - \mathbf{y}(t)\|_{[0,T]} \leq (T + \lambda)LS\|\mathbf{x}(t) - \mathbf{y}(t)\|_{[0,T]}.$$

Then we can always construct a small T for which $(T+\lambda)LS < 1$ and $\mathbf{x}(t) = \mathbf{y}(t)$. Apply the above mentioned Banach's contraction theorem with $S = L^2([0,T],\mathbb{R}^d)$ to prove existence and uniqueness of the solution of (3.1). □

Thus a finite S means that $\mathbf{C}(t,\mathbf{x}) + \lambda\mathbf{G}(t,\mathbf{x})$ is invertible for all $(t,\mathbf{x})$. This is exactly the famous condition for the matrix pencil $(\mathbf{C},\mathbf{G})$ of LTI systems. For finite L and S it means that we get a unique bounded solution for $T = \frac{1}{LS}$. Note that this operator could even be used to find a numerical solution by a fixed-point iteration on $[0,T]$. The previous results are sufficient to prove the existence and uniqueness of the solution. Nevertheless we still need an alternative version of Gronwall's Lemma, that also can prove well-posedness with respect to small perturbations of the initial solution and the system itself.

3.3 Index of differential-algebraic equations

The general form of a DAE is given by

$$\mathbf{f}(t, \mathbf{x}, \dot{\mathbf{x}}) = \mathbf{0}. \tag{3.9}$$

This equation consists of algebraic and differential equations. If $\frac{\partial \mathbf{f}}{\partial \dot{\mathbf{x}}}$ is invertible, the DAE can be transformed into an ODE. If $\mathbf{f}$ represents the dynamics of an electrical circuit this will not be the case in general. But after differentiating the DAE sufficiently often and replacing the algebraic equations by the extra derived differential equations, one can explicitly express $\dot{\mathbf{x}}$ in terms of $\mathbf{x}$.

Definition 3.2 *The* (global) DAE-index ν *of the DAE (3.9) is the necessary amount of differentiations to get an ODE.*

Clearly, ODEs have DAE-index $\nu = 0$. For (3.1), it follows that $\nu = 0$ if $\mathbf{C}(t, \mathbf{x})$ is invertible. But in general, it is hard to determine the global index of a system.

Definition 3.3 *Consider the* matrix pencils*:* $\lambda \boldsymbol{C}(t, \boldsymbol{x}) + \boldsymbol{G}(t, \boldsymbol{x})$ *with* $\lambda \in \mathbb{C}$*. The DAE (3.1) is* solvable *if det*$(\lambda \boldsymbol{C}(t, \boldsymbol{x}) + \boldsymbol{G}(t, \boldsymbol{x}))$ *is only zero for a finite number of values for* λ*. If* $\boldsymbol{G}(t, \boldsymbol{x})$ *is invertible and if* $-\frac{1}{\lambda}$ *is an eigenvalue of the matrix* $\boldsymbol{C}(t, \boldsymbol{x})\boldsymbol{G}(t, \boldsymbol{x})^{-1}$*, the pencils are never invertible.*

The following *Kronecker decomposition* of the matrices $\mathbf{C}, \mathbf{G}$ is very useful to analyse the index of linear time-invariant DAEs [32].

Theorem 3.4 *Consider the invertible matrix pencil* $\lambda \boldsymbol{C} + \boldsymbol{G}$*. Then there exist nonsingular matrices* $\boldsymbol{P}$ *and* $\boldsymbol{Q}$*, such that*

$$\boldsymbol{PCQ} = \begin{pmatrix} \boldsymbol{I}_{d-s} & 0 \\ 0 & \boldsymbol{N} \end{pmatrix}, \quad \boldsymbol{PGQ} = \begin{pmatrix} \boldsymbol{A} & 0 \\ 0 & \boldsymbol{I}_s \end{pmatrix}. \tag{3.10}$$

Here, $\boldsymbol{I}_{d-s}, \boldsymbol{I}_s$ *are identity matrices, while* $\boldsymbol{N} \in \mathbb{R}^{s \times s}$ *is a nilpotent matrix.*

Consider the DAE (3.1) with exact solution $\mathbf{x}(t)$. From Theorem (3.4) it follows that there exist invertible matrices $\mathbf{P}$ and $\mathbf{Q}$ such that $\mathbf{PC}(t, \mathbf{x}(t))\mathbf{Q}$ and $\mathbf{PG}(t, \mathbf{x}(t))\mathbf{Q}$ at time-point t can be written like (3.10).

Definition 3.5 *The local DAE-index* μ *of (3.1) at time-point* t *equals the nilpotency index of* $\boldsymbol{N}$*. The nilpotency index* μ *is defined as:*

$$\mu = \min\{k \in \mathbb{N} : \boldsymbol{N}^k = \boldsymbol{O}\}.$$

It follows that for all λ

$$\begin{aligned}\det(\mathbf{C}+\lambda\mathbf{G}) &= \det(\mathbf{P}^{-1})\det(\begin{pmatrix}\mathbf{I}+\lambda\mathbf{A} & 0\\ 0 & \mathbf{N}+\lambda\mathbf{I}\end{pmatrix})\det(\mathbf{Q}^{-1})\\ &= \frac{1}{\det(\mathbf{PQ})}\det(\mathbf{I}+\lambda\mathbf{A})\det(\mathbf{N}+\lambda\mathbf{I}).\end{aligned}$$

Because $\mathbf{N}$ is a nilpotent matrix all its eigenvalues are equal to zero and it can be transformed into a Jordan upper block triangular matrix with zeros in the diagonal. This implies that $\det(\mathbf{N}+\lambda\mathbf{I}) = \lambda^s$ and

$$\det(\mathbf{C}+\lambda\mathbf{G}) = \frac{\lambda^s}{\det(\mathbf{PQ})}\det(\mathbf{I}+\lambda\mathbf{A}) = O(\lambda^s).$$

Thus s equals the algebraic multiplicity of the eigenvalue zero. Let g be the geometric multiplicity of the corresponding eigenspace: $g = \dim(\mathrm{Null}(\mathbf{C}))$. Then the DAE-index μ is defined as

$$\mu = \begin{cases} 0 & \text{if } s = 0,\\ 1+s-g & \text{if } s > 0.\end{cases}$$

3.4 Stability

Besides the solvability, also the *stability* of the system is important.

Definition 3.6 *Consider the perturbed IVP of (3.1) with initial value* $\hat{x}_0$ *and solution* $\hat{x}(t)$*. The system is stable if:*

$$\forall_{\epsilon>0}\ \exists_{\delta>0}\ \|\hat{x}_0 - x_0\| < \delta\ \Rightarrow\ \forall_t\ \|\hat{x}(t) - x(t)\| < \epsilon. \tag{3.11}$$

Stability ensures that the difference between the exact and the approximated solution remains small, if the initial value is changed only a little. This is a useful property, because the local discretisation errors of an integration method can be considered as perturbations of the initial value for that timepoint. For many physical systems like electrical circuits, the time-dependent behaviour is caused by source functions. This means that these systems can be described by only the DAE

$$\begin{cases} \frac{d}{dt}\mathbf{q}(\mathbf{x}) + \mathbf{j}(\mathbf{x}) + \mathbf{u}(t) = \mathbf{0},\\ \mathbf{x}(0) = \mathbf{x}_0,\end{cases} \tag{3.12}$$

where $\mathbf{u}(t)$ is an input function that only depends on t. It is difficult to check the global stability in general. A possible approach is to check the local stability around the initial steady state. In this case the stability of the linearised systems at all possible times and states is determined. For a *linear time-invariant (LTI) system*, that has constant Jacobian matrices it is well-known that the system is stable, if the Jacobian matrix is a stable matrix. This means that all eigenvalues of the Jacobian matrix have strict negative real parts.

Let $\mathbf{x}_0$ be the steady state of (3.12), with $\mathbf{j}(\mathbf{x}_0) = \mathbf{0}$. Consider the linearised homogeneous system around $\mathbf{x}_0$

$$\mathbf{C}\dot{\mathbf{y}} + \mathbf{G}\mathbf{y} = \mathbf{0}. \tag{3.13}$$

This system is stable if all roots of

$$\det(s\mathbf{C} + \mathbf{G}) = 0$$

have strict negative real part. Furthermore, if (3.13) is stable then also the nonlinear system itself is *locally stable* around $\mathbf{x}_0$.

Chapter 4

Numerical integration of DAEs

4.1 Introduction

Consider the following system of differential-algebraic equations:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{q}(t, \mathbf{x}) + \mathbf{j}(t, \mathbf{x}) = \mathbf{0}. \tag{4.1}$$

It is assumed that the exact solution $\mathbf{x}(t)$ exists on $[0, T]$. This system can describe many dynamical phenomena in nature, economy or other applications. Very often we are interested to know the state $\mathbf{x}(t)$ on a certain time interval $[0, T]$ for a given initial condition. It is rarely possible to compute the solution analytically, in which case one has to rely on *numerical time integration methods*. All these methods discretise the time and compute the solution on the time grid with the help of integration methods. The timesteps have to be chosen in such a way that the error will be small enough. The time interval $[0, T]$ is discretised into a timegrid $\{t_n, n = 0, \ldots, N\}$, with $h_n = t_n - t_{n-1}$ the n-th timestep. The numerical approximation at $t = t_n$ is denoted by $\mathbf{x}_n$. Now, (4.1) is approximated with use of numerical discretisation or integration methods. In general, this results in a large system of N nonlinear equations with N unknowns. Usually, it is possible to compute $\mathbf{x}_n$ for $n \in \{1, \ldots, N\}$ in a recursive manner:

$$\mathbf{g}(t_n, \mathbf{x}_{n-k}, \ldots, \mathbf{x}_n) = \mathbf{0}. \tag{4.2}$$

If $\mathbf{x}_n$ only depends on the previous solution $\mathbf{x}_{n-1}$, the method is a *one step method*, e.g. the *Runge Kutta methods*. For *multistep methods* $k > 1$ such that also $\mathbf{x}_{n-2}, \ldots, \mathbf{x}_{n-2}$ are used, which makes them more complex to analyse [15,21,32,48].

Definition 4.1 *Consider the numerical solutions* $\{x_n, n = 0, \ldots, N\}$ *at the timegrid* $\{t_n, n = 0, \ldots, N\}$.

- *The* global error e_n *is equal to* $x(t_n) - x_n$.

- *The* local discretisation error (LDE) *per unit step is equal to the residual of (4.2), after inserting the exact solution:* $\delta_n = \boldsymbol{g}(t_n, \boldsymbol{x}(t_{n-k}), \ldots, \boldsymbol{x}(t_n))$.
- *Let* $\boldsymbol{x}_n^*$ *be the solution of the numerical scheme (4.2), if all previous solutions are exact.*

$$\boldsymbol{g}(t_n, \boldsymbol{x}(t_{n-k}), \ldots, \boldsymbol{x}(t_{n-1}), \boldsymbol{x}_n^*) = \boldsymbol{0}.$$

Then, the scaled local discretisation error *is equal to* $\boldsymbol{d}_n = \boldsymbol{x}(t_n) - \boldsymbol{x}_n^*$.

A numerical scheme is called *consistent* with order p, if for constant stepsize h, the local discretisation error $\delta_n = O(h^{p+1})$. Here we use the local discretisation error per unit step in contrast to the also existing local discretisation error per step [48]. The scheme is convergent with order p, if also the global errors satisfy $\|\mathbf{e}_n\| = O(h^p)$. Theorem (4.2) gives some important relations between the local and global error.

Theorem 4.2 *Assume that method (4.2) is consistent with order* p. *Then, the scaled local discretisation error satisfies*

$$\left(\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}_n}\right) \boldsymbol{d}_n = \delta_n.$$

The global error satisfies the following equation:

$$\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}_{n-k}} \boldsymbol{e}_{n-k} + \ldots + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}_n} \boldsymbol{e}_n \doteq \delta_n.$$

This is equivalent to

$$\left(\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}_n}\right) \boldsymbol{e}_n \doteq -\left(\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}_{n-k}} \boldsymbol{e}_{n-k} + \ldots + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}_{n-1}} \boldsymbol{e}_{n-1}\right) + \left(\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}_n}\right) \boldsymbol{d}_n.$$

Proof: For the scaled local discretisation error we obtain

$$\begin{aligned} \mathbf{0} &= \mathbf{g}(t_n, \mathbf{x}(t_{n-k}), \ldots, \mathbf{x}(t_{n-1}), \mathbf{x}_n^*) \\ &= \delta_n - \tfrac{\partial \mathbf{g}}{\partial \mathbf{x}_n} \mathbf{d}_n. \end{aligned}$$

Because of the definition of δ_n and (4.2), it follows that

$$\begin{aligned} \mathbf{0} &= \mathbf{g}(t_n, \mathbf{x}_{n-k}, \ldots, \mathbf{x}_n) \\ &\doteq \delta_n - \left(\tfrac{\partial \mathbf{g}}{\partial \mathbf{x}_{n-k}} \mathbf{e}_{n-k} + \ldots + \tfrac{\partial \mathbf{g}}{\partial \mathbf{x}_n} \mathbf{e}_n\right). \end{aligned}$$

□

Note that there exists a close relationship between the scaled local error and the global error. It appears that the global error also depends on the previous global errors. Although theorem (4.2) holds, this asymptotic relation can only be used to estimate the global error at the next timepoint. Because the rather complex behaviour of the global error, the LDE or the scaled LDE are controlled instead. If a method is consistent with order p, it is possible to control the local discretisation error by the stepsizes. Then, it is necessary to use an estimate $\hat{\delta}_n$ of the LDE with $\hat{\delta}_n = \delta_n + O(h^{p+2})$. If a method is convergent, this means that the global errors tend to zero, if $h \to 0$. However, in practice the stepsizes are always positive numbers. In that case, stability is also an important property of a numerical scheme.

Definition 4.3 *A method is called* absolutely stable *for a given stepsize* h *and a given differential equation if the change due to a perturbation of size* ϵ *in one of the mesh values* x_n *is not larger than* ϵ *in all subsequent values* x_m, $m > n$.

In practice, the absolute stability is investigated for the test equation $\dot{y} = \lambda y$ with $\lambda \in \mathbb{C}$. This results in a stability region $S \subset \mathbb{C}$, with

$$h\lambda \in S \Rightarrow \text{Method is stable for linear test equation with eigenvalue } \lambda.$$

Definition 4.4 *A method is called* $A(\alpha)$*-stable with* $0 < \alpha < \frac{\pi}{2}$ *if*

$$S \supseteq S_\alpha = \{z : |\arg(-z)| < \alpha, z \neq 0\}.$$

A method is called absolutely stable (A-stable) or unconditionally stable if it is $A(\frac{\pi}{2})$*-stable.*

4.2 Linear Multistep Methods

Besides one step methods, it is also possible to use *Linear Multistep Methods (LMM)*. In contrast to Runge Kutta methods, they do not use only the last numerical solution. This leads to smoother solutions, but can cause problems with discontinuities. For variable stepsizes, it is more difficult to analyse them, because now the local errors depend also on the previous stepsizes. A general representation of the LMM-method to approximate the solution of (4.1) with fixed stepsizes is:

$$\sum_{m=0}^{k} \left[\rho_m \mathbf{q}(t_{n-m}, \mathbf{x}_{n-m}) + h\sigma_m \mathbf{j}(t_{n-m}, \mathbf{x}_{n-m})\right] = \mathbf{0}. \tag{4.3}$$

A k-step method needs k given initial values $\mathbf{x}_0, \ldots, \mathbf{x}_{k-1}$ to start. If only $\mathbf{x}_0$ is known, the other values can be computed iteratively using methods with increasing amount of steps. Define the vector $\mathbf{r}_n \in \mathbb{R}^n$, which is known at $t = t_n$.

$$\mathbf{r}_n = -\sum_{m=1}^{k} \left[\rho_m \mathbf{q}(t_{n-m}, \mathbf{x}_{n-m}) + h\sigma_m \mathbf{j}(t_{n-m}, \mathbf{x}_{n-m})\right].$$

Then each timestep, the next nonlinear algebraic equation has to be solved:

$$\rho_0 \mathbf{q}(t_n, \mathbf{x}_n) + h\sigma_0 \mathbf{j}(t_n, \mathbf{x}_n) = \mathbf{r}_n.$$

This equation can be solved by e.g. the Newton method. An initial guess for the Newton process could be the previous solution $\mathbf{x}_{n-1}$, but better predictions can be made by means of extrapolation. Implicit schemes have the property that $\sigma_0 \neq 0$, which are necessary if the function $\mathbf{q}$ is not invertible which is the case for circuit models.

An important class of LMM-methods are the *Backward Difference Formulae (BDF)*. Define for $i \in \{0, \dots, k\}$ the *Lagrangian basis polynomial* with

$$l_n(t_{n-i}) := \delta_{i0}. \tag{4.4}$$

Then the function $\mathbf{q}(t, \mathbf{x}(t))$ obeys the next approximation:

$$\mathbf{q}(t, \mathbf{x}(t)) \doteq \sum_{m=0}^{k} l_{n-m}(t)\mathbf{q}(t_{n-m}, \mathbf{x}_{n-m}).$$

Differentiating the DAE in $t = t_n$, we get:

$$h\,\frac{d}{dt}\mathbf{q}(t, \mathbf{x}(t))\bigg|_{t=t_n} + h\mathbf{j}(t_n, \mathbf{x}_n) \doteq \sum_{m=0}^{k} h\left(\frac{d}{dt}l_{n-m}(t)\bigg|_{t=t_n}\right)\mathbf{q}(t_{n-m}, \mathbf{x}_{n-m}) + h\mathbf{j}(t_n, \mathbf{x}_n).$$

Hence we derive the Backward Difference Formula

$$\sum_{m=0}^{k} \rho_m \mathbf{q}(t_{n-m}, \mathbf{x}_{n-m}) + h\mathbf{j}(t_n, \mathbf{x}_n) = \mathbf{0},$$

where ρ_m, σ_m satisfy

$$\begin{aligned} \forall_{m\in\{0,\dots,k\}}\ \rho_m &= h\,\tfrac{d}{dt}l_m(t)\big|_{t=t_n}, \\ \forall_{m\in\{0,\dots,k\}}\ \sigma_m &= \begin{cases} 1 & m = k, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

Define the polynomials:

$$\rho(z) = \textstyle\sum_{m=0}^{k} \rho_m z^m, \quad \sigma(z) = \sum_{m=0}^{k} \sigma_m z^m.$$

Consider the shift-operator q with $(q\ \mathbf{x})_n = \mathbf{x}_{n+1}$. This is often also denoted by $q\mathbf{x}_n = \mathbf{x}_{n+1}$. Now it is possible to describe an LMM-method with use of the shift-operator as follows:

$$\rho(q)\mathbf{x}_{n-k} = h\sigma(q)\mathbf{f}(t_{n-k}, \mathbf{x}_{n-k}). \tag{4.5}$$

Note that for a BDF-method one has the restriction that $\sigma(z) = z^k$ is fixed.

Theorem 4.5 *An LMM-method is consistent with order* p *if and only if the polynomials* $\rho(z)$ *and* $\sigma(z)$ *satisfy next equations:*

$$\begin{gathered} \rho(1) = 0, \\ \rho'(1) - \sigma(1) = 0, \\ \forall_{s\in\{2,\dots,p\}} \textstyle\sum_{m=0}^{k} \left[\rho_m (m-k)^s - s\sigma_m (m-k)^{s-1}\right] = 0. \end{gathered} \tag{4.6}$$

The local discretisation error (LDE) of a consistent LMM-method with order p *and fixed stepsizes can be written as*

$$\delta_n = C_p h^p \boldsymbol{x}^{(p+1)}(t_n) + O(h^{p+1})$$

with $C_p = \sum_{m=0}^{k} \left[\rho_m \frac{(m-k)^{p+1}}{(p+1)!} - \sigma_m \frac{(m-k)^p}{p!}\right]$. *For BDF-methods,* $C_p = -\frac{1}{p+1} = -\frac{1}{k+1}$.

Proof: The local discretisation error (LDE) δ_n is equal to

$$\delta_n = \rho(q)\mathbf{x}(t_{n-k}) - h\sigma(q)\mathbf{f}(t_{n-k}, \mathbf{x}(t_{n-k})).$$

Then, δ_n can be approximated in the next way:

$$\begin{array}{rcl} \delta_n & = & \sum_{m=0}^{k} \rho_m \mathbf{x}(t_{n+m-k}) - h\sum_{m=0}^{k} \sigma_m \mathbf{f}(t_{n+m-k}, \mathbf{x}(t_{n+m-k})) \\ & = & \sum_{m=0}^{k} \rho_m \mathbf{x}(t_n) + h\sum_{m=0}^{k} [\rho_m(m-k) - \sigma_m]\,\dot{\mathbf{x}}(t_n) \\ & + & h^2 \sum_{m=0}^{k} \left[\rho_m \frac{(m-k)^2}{2} - \sigma_m(m-k)\right] \ddot{\mathbf{x}}(t_n) + \dots \\ & + & h^p \sum_{m=0}^{k} \left[\rho_m \frac{(m-k)^p}{p!} - \sigma_m \frac{(m-k)^{p-1}}{(p-1)!}\right] \mathbf{x}^{(p)}(t_n) + O(h^{p+1}). \end{array} \tag{4.7}$$

Because an LMM-method is consistent with order p if $\delta_n = O(h^{p+1})$, indeed $\rho(z)$ and $\sigma(z)$ has to satisfy the equations (4.6). ☐

Definition 4.6 *Consider a polynomial* $P(z)$. *Then,* $P(z)$ *obeys the root condition, if*

$$\begin{cases} \forall_{z\in\mathbb{C}} P(z) = 0 \Rightarrow |z| \leq 1, \\ \forall_{z\in\mathbb{C}} \left(P(z) = 0 \wedge z \textit{ is not simple.}\right) \Rightarrow |z| < 1. \end{cases} \tag{4.8}$$

The LMM-method is called root-stable if $\rho(z)$ obeys the root condition (4.8).

Theorem 4.7 *An LMM-scheme is convergent with order* k *for an ordinary ODE, if it is both consistent with order* k *and root stable.*

Proof: See [32,48]. ☐

Because convergent LMM-methods are root-stable, it always holds for these LMM-methods that $\rho'(1) \neq 0$, because 1 can only be a simple root of $\rho(z)$. Then, it follows from the equation (4.6) that also $\sigma(1) \neq 0$.

To check the absolute stability, the LMM-method is performed for the linear test equation, which results in:

$$\rho(q)y_{n-k} = \bar{h}\sigma(q)y_{n-k}$$

with $\bar{h} = \lambda h$. Now, the characteristic polynomial $\pi(z, \bar{h})$ of the numerical scheme is equal to $\rho(z) - \bar{h}\sigma(z)$. Then

$$\pi(q, \bar{h})y_{n-k} = 0.$$

The method is absolutely stable (A-stable) in $\bar{h}$ if $\pi(z, \bar{h})$ obeys the root condition. Note that root stability is equivalent to absolutely stability in $\bar{h} = 0$. Then it is possible to derive stability regions for the LMM-methods with:

$$S = \{\bar{h} \in \mathbb{C} : \pi(z, \bar{h}) \text{ satisfies the root condition.}\}. \tag{4.9}$$

There exists a Theorem proved by Dahlquist [32], that states that the order of absolutely stable LMM-methods is at most 2, while the order of $A(\alpha)$-stable methods is at most 6. In practice, the stability regions for special nonlinear functions f may be quite different from these stability regions. But after linearising f at $x = x^*$, the eigenvalues of $\frac{\partial f}{\partial x}\big|_{x=x^*}$ can be computed. The timestep h has to be chosen, such that for all eigenvalues $h\lambda \in S$.

4.3 The Nordsieck data representation

For an efficient implementation of a Linear Multistep method, like the BDF scheme, it is important to have a proper data structure. There are several ways to store and process polynomials, such as the Lagrange, scaled divided difference and Nordsieck representation. This topic has been studied very thoroughly e.g. in [49]. In this section we describe the *Nordsieck data representation* [13,59,73]. Let $\mathbf{p}(t) : \mathbb{R} \to \mathbb{R}^d$ be a vector-valued polynomial of degree k. This polynomial can be written as a truncated Taylor series around an arbitrarily chosen time-point τ_1, e.g. $\tau_1 = t_n$, as

$$\mathbf{p}(t) = \sum_{i=0}^{k} \frac{1}{i!} \frac{d^i}{dt^i} \mathbf{p}(\tau_1)(t - \tau_1)^i. \tag{4.10}$$

It is very common to expand $\mathbf{p}(t)$ in the following scaled form (for some $h > 0$)

$$\mathbf{p}(t) = \sum_{i=0}^{k} \left(\frac{h^i \frac{d^i}{dt^i} \mathbf{p}(\tau_1)}{i!} \right) \left(\frac{t - t_1}{h} \right)^i .$$

This polynomial could describe the time-behaviour of $\mathbf{x}(t)$ or $\mathbf{q}(t, \mathbf{x}(t))$ at a certain time-interval $[t_{n-1}, t_n]$, where $\tau_1 = t_n$ and $h = t_n - t_{n-1}$. The *Nordsieck matrix* $\bar{\mathbf{P}}(\tau_1, h) \in \mathbb{R}^{d\times(k+1)}$ of this polynomial, defined as

$$\bar{\mathbf{P}} = \left(\mathbf{p}(t_1) \;\middle|\; h\frac{d}{dt}\mathbf{p}(t_1) \;\middle|\; \frac{h^2}{2}\frac{d^2}{dt^2}\mathbf{p}(t_1) \;\middle|\; \dots \;\middle|\; \frac{h^k}{k!}\frac{d^k}{dt^k}\mathbf{p}(t_1) \right) \tag{4.11}$$

contains all coefficients of this polynomial. Often one does not realise that $\bar{\mathbf{P}}$ is a matrix for multi-dimensional systems of DAEs. This matrix $\bar{\mathbf{P}}$ is called the *transposed Nordsieck vector* if the state dimension d is equal to one. Now, the vector-valued polynomial and its Nordsieck matrix are related by the following equation, where $\bar{\mathbf{p}}_i$ is the i-th column of $\bar{\mathbf{P}}$

$$\mathbf{p}(t) = \sum_{i=0}^{k} \bar{\mathbf{p}}_{i+1} \left(\frac{t - t_1}{h} \right)^i = \bar{\mathbf{P}} \begin{pmatrix} 1 \\ \omega \\ \vdots \\ \omega^k \end{pmatrix}, \quad \omega = \frac{t - \tau_1}{h}. \tag{4.12}$$

Define

$$\mathbf{e}(k, \omega) := \left[1, \omega, \dots, \omega^k\right]^T . \tag{4.13}$$

Clearly, the Nordsieck matrix depends on the time-point τ_1 and the stepsize h. This means that changing τ_1 or h will also change the Nordsieck vector. If the Nordsieck matrix $\bar{\mathbf{P}}(\tau_1, h_1)$ is available, $\mathbf{p}(t)$ can be found from

$$\mathbf{p}(t) = \sum_{i=0}^{k} \bar{\mathbf{p}}_{i+1} \left(\frac{t - t_1}{h_1} \right)^i = \bar{\mathbf{P}} \cdot \mathbf{e}(k, \frac{t - \tau_1}{h_1}). \tag{4.14}$$

Two Nordsieck matrices $\bar{\mathbf{P}}(\tau_1, h_1)$ and $\bar{\mathbf{P}}(\tau_2, h_2)$ are called *equivalent* if they represent the same polynomial $\mathbf{p}(t)$. If one Nordsieck matrix $\bar{\mathbf{P}} := \bar{\mathbf{P}}(\tau_1, h_1)$ is known, all other Nordsieck matrices $\bar{\mathbf{Q}} := \bar{\mathbf{P}}(\tau_2, h_2)$ can also be computed.

Define $\mathbf{A}(k, \omega, \mu) \in \mathbb{R}^{(k+1)\times(k+1)}$ by

$$a_{ij} := \begin{cases} 0 & i < j, \\ \binom{i-1}{j-1}\omega^{i-1}\mu^{i-j} & i \geq j. \end{cases} \tag{4.15}$$

As the following Theorem shows this matrix appears to be needed to describe the relationship between two equivalent Nordsieck matrices.

Theorem 4.8 *Assume that* $\bar{\boldsymbol{P}} := \bar{\boldsymbol{P}}(\tau_1, h_1)$ *and* $\bar{\boldsymbol{Q}} := \bar{\boldsymbol{P}}(\tau_2, h_2)$ *are two equivalent Nordsieck matrices that represent the same polynomial* $\boldsymbol{p}(t)$*. Then the Nordsieck matrices are related by*

$$\bar{\boldsymbol{Q}} = \bar{\boldsymbol{P}} \cdot \boldsymbol{A}(k, \frac{h_2}{h_1}, \frac{\tau_2 - \tau_1}{h_2}), \tag{4.16}$$

where $\boldsymbol{A} \in \mathbb{R}^{(k+1)\times(k+1)}$ *is defined in (4.15).*

This Theorem can be used to transform the Nordsieck matrices if $\bar{\mathbf{P}}$ and $\bar{\mathbf{Q}}$ have the same number of columns k. In practice $\bar{\mathbf{P}}$ and $\bar{\mathbf{Q}}$ may also have a different number of columns, k_1 and k_2 say. If the polynomial $\mathbf{p}(t)$ of degree k_1 is represented by $\bar{\mathbf{P}}$, it is only possible to describe it also with $\bar{\mathbf{P}}$ if $k_2 \geq k_1$. Otherwise, $\mathbf{p}(t)$ can only be approximated by a lower degree polynomial $\mathbf{q}(t)$, that is represented by $\bar{\mathbf{Q}}$.

Define the non-square Vandermonde matrix $\mathbf{V}(k, l) \in \mathbb{R}^{(k+1)\times(l+1)}$ by

$$v_{ij} := \begin{cases} 1 & i = j = 1, \\ (1 - j)^{i-1} & \text{otherwise.} \end{cases} \tag{4.17}$$

Definition 4.9 *For* $1 \leq i \leq k+1$ *and* $1 \leq j \leq l+1$ *the non-square Vandermonde matrix* $\boldsymbol{V}(k, l) \in \mathbb{R}^{(k+1)\times(l+1)}$ *is defined by For* $1 \leq i \leq k+1$ *and* $1 \leq j \leq l+1$ *the matrices* $\boldsymbol{T}^{(1)}(k, l, \omega, \mu), \boldsymbol{T}^{(2)}(k, l, \omega, \mu) \in \mathbb{R}^{(k+1)\times(l+1)}$ *are defined by*

$$\boldsymbol{T}^{(1)} := \boldsymbol{A}(k, \omega, \mu) \cdot \boldsymbol{V}(k, l) \cdot \boldsymbol{V}(l, l)^{-1} \tag{4.18}$$

$$\boldsymbol{T}^{(2)} := \boldsymbol{A}(k, \omega, \mu) \cdot \left[\boldsymbol{V}(k, l-1), \boldsymbol{e}_2^k\right] \cdot \left[\boldsymbol{V}(l, l-1), \boldsymbol{e}_2^l\right]^{-1} \tag{4.19}$$

where $\boldsymbol{e}_2^k, \boldsymbol{e}_2^l$ *are the unit vectors* $[0, 1, 0, \ldots]^T$ *of length* $k+1$ *and* $l+1$*, respectively.*

The transformations $\mathbf{T}^{(1)}, \mathbf{T}^{(2)}$ can be used to describe the relationship between two Nordsieck matrices which are not equivalent. In practice they are used if the integration order changes adaptively.

Theorem 4.10 *Consider the polynomial* $\boldsymbol{p}(t)$ *of degree* k_1 *that is represented by* $\bar{\boldsymbol{P}} \in \mathbb{R}^{d\times(k_1+1)}$ *using time-point* τ_1 *and stepsize* h_1*. Let* $\bar{\boldsymbol{Q}} \in \mathbb{R}^{d\times(k_2+1)}$ *be the Nordsieck matrix of* $\boldsymbol{q}(t)$ *of degree* k_2 *with time-point* τ_2 *and stepsize* h_2*. Then we obtain the relationships*

$$\bar{\boldsymbol{Q}} = \bar{\boldsymbol{P}} \cdot \boldsymbol{T}^{(1)}(k_1, k_2, \frac{h_2}{h_1}, \frac{\tau_2 - \tau_1}{h_2}) \quad \Leftrightarrow \quad \boldsymbol{q}(\tau_2 - jh_2) = \boldsymbol{p}(\tau_2 - jh_2), \quad 0 \leq j \leq k_2,$$

$$\bar{\boldsymbol{Q}} = \bar{\boldsymbol{P}} \cdot \boldsymbol{T}^{(2)}(k_1, k_2, \frac{h_2}{h_1}, \frac{\tau_2 - \tau_1}{h_2}) \quad \Leftrightarrow \quad \begin{cases} \boldsymbol{q}(\tau_2 - jh_2) &= \boldsymbol{p}(\tau_2 - jh_2), \quad 0 \leq j \leq k_2 - 1 \\ \frac{d}{dt}\boldsymbol{q}(\tau_2) &= \frac{d}{dt}\boldsymbol{p}(\tau_2) \end{cases}$$

between the Nordsieck matrices and the corresponding polynomials.

Using the transformation matrix $\mathbf{T}^{(1)}$ is similar to Lagrangian interpolation, while $\mathbf{T}^{(2)}$ guarantees a smooth transition between $\mathbf{p}$ and $\mathbf{q}$ around τ_2.

4.4 The Backward Difference Formula (BDF)

For variable step the coefficients $\rho_0, \dots, \rho_k$ will depend on the chosen stepsizes. Then it becomes useful to view the BDF method as a special Predictor-Corrector method. Then at each timestep the previous data $\mathbf{x}_{n-1}, \dots, \mathbf{x}_{n-k}$ is represented by a polynomial, which is called the *predictor polynomial*. This polynomial is corrected by use of the newly computed $\mathbf{x}_n$ such that we get the *corrector polynomial*. For the next step this corrector polynomial is used as predictor polynomial.

Consider the polynomial $\mathbf{q}_{n-1}(t)$ that interpolates the solutions of $\mathbf{q}(t, \mathbf{x})$ at the previous computed $(k+1)$ time-points $\{t_{n-k-1}, \dots, t_{n-1}\}$. This polynomial is used as predictor polynomial for the new time-interval, so we have

$$\mathbf{p}_n(t) = \mathbf{q}_{n-1}(t) \tag{4.20}$$

At t_n the polynomial $\mathbf{p}_n(t)$ can be used as prediction for the numerical solution, while the corrector polynomial $\mathbf{q}_n(t)$ has a corrected value at $t = t_n$. More precisely, we express the corrector polynomial in terms of the predictor polynomial and the Lagrangian basis polynomial $l_n(t)$, that satisfies (4.4). After finding the correct values $\mathbf{x}_n, \mathbf{q}(t_n, \mathbf{x}_n)$ at $t = t_n$ we can define the *corrector polynomial* as follows

$$\mathbf{q}_n(t) := \mathbf{p}_n(t) + (\mathbf{q}(t_n, \mathbf{x}_n) - \mathbf{p}_n(t_n)) l_n(t).. \tag{4.21}$$

Here $l_n(t)$ is the Lagrangian basis polynomial that has been defined in (4.4). Because of (4.21) we have the property

$$\mathbf{q}_n(t_{n-j}) = \left\{ \begin{array}{ll} \mathbf{q}(t_n, \mathbf{x}_n) & j = 0, \\ \mathbf{p}_n(t_{n-j}) & 1 \le j \le k \end{array} \right. . \tag{4.22}$$

The BDF-method approximates the term $\frac{d}{dt}\mathbf{q}(t, \mathbf{x})$ at $t = t_n$ by $\frac{d}{dt}\mathbf{q}_n(t_n)$.

$$\frac{d}{dt}\mathbf{q}_n(t_n) + \mathbf{j}(t_n, \mathbf{x}_n) = \mathbf{0}. \tag{4.23}$$

Lemma 4.11 *It is well-known that the derivative of each* k*-th degree polynomial* $p(t)$ *can be uniquely expressed as a linear combination of* $p(t_n), \dots, p(t_{n-k})$ *like*

$$h_n \frac{dp}{dt}(t_n) = \rho_{n,0} p(t_n) + \ldots + \rho_{n,k} p(t_{n-k}).$$

Because of the definition of the Lagrangian basis polynomial $l_n(t)$ in (4.4), this Lemma implies

$$h_n l_n'(t_n) = \rho_{n,0}. \tag{4.24}$$

Because both $\mathbf{p}_n(t)$ and $\mathbf{q}_n(t)$ have degree k, Lemma 4.11 also yields

$$\begin{aligned} h_n \tfrac{d}{dt}\mathbf{p}_n(t_n) = \rho_{n,0}\mathbf{p}_n(t_n) + \ldots + \rho_{n,k}\mathbf{p}_n(t_{n-k}), \\ h_n \tfrac{d}{dt}\mathbf{q}_n(t_n) = \rho_{n,0}\mathbf{q}_n(t_n) + \ldots + \rho_{n,k}\mathbf{q}_n(t_{n-k}). \end{aligned}$$

The relation between $\mathbf{p}_n(t)$ and $\mathbf{q}_n(t)$ implies

$$\begin{aligned} h_n \tfrac{d}{dt}\mathbf{q}_n(t_n) &= \rho_{n,0}\mathbf{q}(t_n, \mathbf{x}_n) + \rho_{n,1}\mathbf{p}_n(t_{n-1}) + \ldots + \rho_{n,k}\mathbf{p}_n(t_{n-k}) \\ &= h_n \tfrac{d}{dt}\mathbf{p}_n(t_n) + \rho_{n,0}\left[\mathbf{q}(t_n, \mathbf{x}(t_n)) - \mathbf{p}_n(t_n)\right]. \end{aligned}$$

Using this identity in the equation (4.23) we obtain for $\mathbf{x}_n$ the nonlinear equation

$$h_n \frac{d}{dt}\mathbf{p}_n(t_n) + \rho_{n,0}\left[\mathbf{q}(t_n, \mathbf{x}_n) - \mathbf{p}_n(t_n)\right] + h_n \mathbf{j}(t_n, \mathbf{x}_n) = \mathbf{0}.$$

Define $\alpha_n = \rho_{n,0} = h_n l_n'(t_n)$ and $\mathbf{b}_n = h_n \frac{d}{dt}\mathbf{p}_n(t_n) - \alpha_n \mathbf{p}_n(t_n)$. Then $\mathbf{x}_n$ is the solution of the nonlinear algebraic equation

$$\alpha_n \mathbf{q}(t_n, \mathbf{x}_n) + h_n \mathbf{j}(t_n, \mathbf{x}_n) + \mathbf{b}_n = \mathbf{0}, \tag{4.25}$$

that can e.g. be solved by the Newton method. The vector $\mathbf{b}_n$ represents the contribution of the previous timesteps to the current nonlinear system. In this way the BDF method can be applied on a time-grid using variable stepsizes. Usually the stepsizes are used to control the accuracy of the method.

As shown before polynomials can be stored efficiently by use of Nordsieck matrices. Introduce the dimensionless coefficients $\xi_{n,m}$

$$\xi_{n,m} = \frac{t_n - t_{n-m}}{h_n}, \tag{4.26}$$

then the Lagrangian polynomial $l_n(t)$ can be expanded as

$$\begin{aligned} l_n(t) &= \tfrac{t - t_{n-1}}{t_n - t_{n-1}} \cdot \tfrac{t - t_{n-2}}{t_n - t_{n-2}} \cdots \tfrac{t - t_{n-k}}{t_n - t_{n-k}} \\ &= \tfrac{t - t_n + t_n - t_{n-1}}{t_n - t_{n-1}} \cdot \tfrac{t - t_n + t_n - t_{n-2}}{t_n - t_{n-2}} \cdots \tfrac{t - t_n + t_n - t_{n-k}}{t_n - t_{n-k}} \\ &= \left(\tfrac{1}{\xi_{n,1}} \tfrac{t - t_n}{h_n} + 1\right) \cdot \left(\tfrac{1}{\xi_{n,2}} \tfrac{t - t_n}{h_n} + 1\right) \cdot \ldots \cdot \left(\tfrac{1}{\xi_{n,k}} \tfrac{t - t_n}{h_n} + 1\right) \\ &= 1 + \left(\tfrac{1}{\xi_{n,1}} + \ldots + \tfrac{1}{\xi_{n,k}}\right)\left(\tfrac{t - t_n}{h_n}\right) + \ldots + \left(\tfrac{1}{\xi_{n,1}} \cdots \tfrac{1}{\xi_{n,k}}\right)\left(\tfrac{t - t_n}{h_n}\right)^k. \end{aligned} \tag{4.27}$$

From this expansion it is easy to derive the Nordsieck vector $\bar{l}^n$, such that

$$l_n(t) = \sum_{i=0}^{k} \bar{l}^n_{i+1} \left(\frac{t - t_n}{h_n}\right)^i. \tag{4.28}$$

Define also the Nordsieck matrices $\bar{\mathbf{P}}^n, \bar{\mathbf{Q}}^n$ of the predictor and corrector polynomials $\mathbf{p}_n(t), \mathbf{q}_n(t)$. From what has been said in the previous section it follows that all necessary time derivatives for the BDF methods are sufficiently stored in $\bar{\mathbf{P}}^n, \bar{\mathbf{Q}}^n$. The relations $\mathbf{p}_n(t) = \mathbf{q}_{n-1}(t)$ and $\mathbf{q}_n(t) = \mathbf{p}_n(t) + (\mathbf{q}(t_n, \mathbf{x}_n) - \mathbf{p}_n(t_n))l_n(t)$ are equivalent to

$$\bar{\mathbf{P}}^n = \bar{\mathbf{Q}}^{n-1}\mathbf{A}(k, \frac{h_n}{h_{n-1}}, 1), \tag{4.29}$$

$$\bar{\mathbf{Q}}^n = \bar{\mathbf{P}}^n + (\mathbf{q}(t_n, \mathbf{x}_n) - \bar{\mathbf{p}}^n_1)\left(\bar{\mathbf{l}}^n\right)^T. \tag{4.30}$$

The transformation matrix $\mathbf{A}(k, \frac{h_n}{h_{n-1}}, 1)$ was defined in (4.15). The parameters $\alpha_n, \mathbf{b}_n$ can also be derived from the Nordsieck vectors

$$\left\{ \begin{array}{lcl} \alpha_n & = & \bar{l}_2^n, \\ \beta_n & = & \bar{\mathbf{p}}_2^n - \alpha_n \bar{\mathbf{p}}_1^n. \end{array} \right.$$

If extrapolation is used for the solution $\mathbf{x}$ as an initial guess for the Newton method, the polynomials $\mathbf{p}_n, \mathbf{q}_n$ are not sufficient because $\mathbf{q}(t, \mathbf{x})$ is not invertible. Therefore also the predictor and corrector polynomials $\mathbf{y}_n(t), \mathbf{x}_n(t)$ for $\mathbf{x}(t)$ have to be stored by the Nordsieck matrices $\bar{\mathbf{Y}}^n, \bar{\mathbf{X}}^n$. Like (4.29) these vectors are related by

$$\begin{aligned} \bar{\mathbf{Y}}^n &= \bar{\mathbf{X}}^{n-1} \mathbf{A}(k, \frac{h_n}{h_{n-1}}, 1), && (4.31) \\ \bar{\mathbf{X}}^n &= \bar{\mathbf{Y}}^n + (\mathbf{x}_n - \bar{\mathbf{y}}_1^n) \left(\bar{\mathbf{l}}^n\right)^T. && (4.32) \end{aligned}$$

Now the Newton method can be started using the extrapolated value $\bar{\mathbf{y}}_1^n$.

Assume that the number of time-steps k is variable. Since the degrees of the polynomials $l_n(t), \mathbf{p}_n(t), \mathbf{q}_n(t)$ depend on k, the relation $\mathbf{p}_n(t) = \mathbf{q}_{n-1}(t)$ can not be achieved if k decreases at the new timestep. Then $\mathbf{q}_{n-1}(t)$ can be approximated by a lower degree polynomial $\mathbf{p}_n(t)$, only looking like $\mathbf{q}_{n-1}(t)$ in the neighbourhood of t_n. Usually this is done by means of interpolation, e.g.

$$\mathbf{p}_n(t_n - jh_n) = \mathbf{q}_{n-1}(t_n - jh_n), \quad j = 0, \ldots, k_n \qquad (4.33)$$

or

$$\begin{array}{rcl} \mathbf{p}_n(t_n - jh_n) & = & \mathbf{q}_{n-1}(t_n - jh_n), \quad j = 0, \ldots, k_n - 1, \\ \frac{d}{dt}\mathbf{p}_n(t_n) & = & \frac{d}{dt}\mathbf{q}_{n-1}(t_n). \end{array} \qquad (4.34)$$

The second approach (4.34) has the advantage that the piecewise polynomial solution is differentiable at t_n because $\mathbf{q}_{n-1}(t)$ and $\mathbf{p}_n(t)$ have a smooth transition at t_n. In [73], this subject is investigated in more detail. If the integration order k decreases, $\bar{\mathbf{P}}^n, \bar{\mathbf{Y}}^n$ are approximated by

$$\begin{aligned} \bar{\mathbf{P}}^n &= \bar{\mathbf{Q}}^{n-1} \mathbf{T}^{(i)}(k_{n-1}, k_n, \frac{h_n}{h_{n-1}}, 1), && (4.35) \\ \bar{\mathbf{Y}}^n &= \bar{\mathbf{X}}^{n-1} \mathbf{T}^{(i)}(k_{n-1}, k_n, \frac{h_n}{h_{n-1}}, 1), && (4.36) \end{aligned}$$

where $i = 1$ or $i = 2$. The transformation matrices $\mathbf{T}^{(1)}, \mathbf{T}^{(2)}$ have been defined in Definition 4.9 in section (4.3).

4.5 Error analysis of the BDF method

This section contains some important properties of the local errors for the BDF method. The theorems are given without proofs; they can be found e.g. in [59]. Remind that the local discretisation error per unit step is defined as the residual of the numerical scheme after inserting the exact solution. It appears that the order p of the local discretisation error of the k-step BDF-method with k steps is equal to k.

Theorem 4.12 *The* k*-step BDF-method has a local discretisation error (LDE)* δ_n *of order* $p = k$*, such that*

$$\delta_n \quad = \quad C_{p,n} h_n^{p+1} \boldsymbol{q}^{(p+1)}(t_n, \boldsymbol{x}(t_n)) + O(h_n^{p+2}), \tag{4.37}$$

where the error constant $C_{p,n}$ *satisfies*

$$C_{p,n} = -\frac{\xi_{n,1} \cdots \xi_{n,p}}{(p+1)!}.$$

Because of this Theorem we have the following asymptotic behaviour for δ_n

$$\delta_n \quad = \quad -\tfrac{1}{(p+1)!} \xi_{n,1} \cdots \xi_{n,p} h_n^{p+1} \mathbf{q}^{(p+1)}(t_n, \mathbf{x}(t_n)) + O(h_n^{p+2}). \tag{4.38}$$

Thus all BDF-methods with $k > 0$ are consistent for ODEs. However, for DAEs, it is still possible that the global error does not have the same order if the DAE-index is larger than one. The scaled local error $\mathbf{d}_n$ is defined as the difference between $\mathbf{x}_n^*$ and the exact solution $\mathbf{x}(t_n)$, where $\mathbf{x}_n^*$ is the solution of the scheme if all previous solutions are exact. The relationship between the scaled LDE and the LDE is equal to:

$$\mathbf{J}(t_n, \mathbf{x}(t_n)) \mathbf{d}_n \doteq \delta_n. \tag{4.39}$$

where $\mathbf{J}(t, \mathbf{x}) = \alpha_n \mathbf{C}(t, \mathbf{x}) + h_n \mathbf{G}(t, \mathbf{x})$ is the Jacobian matrix, which is used to solve the implicit equation (4.25).

For an explicit ODE we have $\mathbf{J}(t, \mathbf{x}) = \alpha_n \mathbf{I} + h_n \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(t, \mathbf{x})$, which implies that the scaled LDE has the same order as the LDE. However, this is not true for DAEs in general, if the Jacobian matrix $C(t, \mathbf{x})$ is not invertible.

An advantage of the scaled LDE is that the influence of the higher DAE-index is measured by $\mathbf{d}_n$ but not by δ_n. If for a solvable DAE the local DAE-indices are smaller or equal than μ at all timepoints, we have for the scaled LDE

$$\mathbf{d}_n = O(h_n^{p+1-\mu}).$$

This is a reason to use $\mathbf{d}_n$ instead of δ_n, although δ_n is easier to estimate.

Theorem 4.13 *Let* $\boldsymbol{p}_n(t)$ *be the predictor polynomial and* $\boldsymbol{q}_n(t)$ *the corrector polynomial for the values of* $\boldsymbol{q}(t, \boldsymbol{x})$*. Assume that* $\boldsymbol{q}_n^{(p+1)}(t_n) = \boldsymbol{q}^{(p+1)}(t_n, \boldsymbol{x}(t_n)) + O(h_n)$*. Consider the LDE estimate* $\hat{\delta}_n$ *with*

$$\hat{\delta}_n := \frac{-1}{\xi_{n,p+1}} (\boldsymbol{q}_n(t_n) - \boldsymbol{p}_n(t_n)). \tag{4.40}$$

Then, this estimator has sufficient accuracy, which implies that $\hat{\delta}_n = \delta_n + O(h_n^{p+2})$*.*

Thus (4.40) can be used to estimate the local discretisation error. Note that $\mathbf{q}_n(t_n), \mathbf{p}_n(t_n)$ are the first columns of the corresponding Nordsieck matrices $\bar{\mathbf{Q}}^n, \bar{\mathbf{P}}^n$. It is just the difference between the predicted and corrected value at $t = t_n$. The estimate $\hat{\mathbf{d}}_n$ of the scaled local error can be computed by

$$\hat{\mathbf{d}}_n = [\mathbf{J}(t_n, \mathbf{x}(t_n))]^{-1}\,\hat{\delta}_n. \tag{4.41}$$

Note that $\mathbf{J}(t_n, \mathbf{x}(t_n))$ is already factorised at the end of the Newton method. It also applies for an automated scaling.

4.6 Adaptive stepsize control

As noted above for BDF-methods, the order p is equal to the number of steps k. Then the LDE per unit step δ_n can be estimated by the following scaled difference

$$\hat{\delta}_n = \frac{-1}{\xi_{n,p+1}}(\bar{\mathbf{q}}_{n,1} - \bar{\mathbf{p}}_{n,1}).$$

Here $(\bar{\mathbf{q}}_{n,1} - \bar{\mathbf{p}}_{n,1})$ is the difference between the corrected and predicted values of $\mathbf{q}$ at t_n. In [59] it is shown that if $h_{n-k} = O(h_n)$ for all k the LDE estimate of a BDF-method with order p, satisfies:

$$\begin{aligned} \hat{\delta}_n &= -\tfrac{1}{(p+1)!}\xi_{n,1}\cdots\xi_{n,p}h_n^{p+1}\mathbf{q}^{(p+1)}(t_n, \mathbf{x}(t_n)) + O(h_n^{p+2}) \\ &= -\tfrac{1}{(p+1)!}h_n\prod_{k=1}^{p}\left(\sum_{i=0}^{k-1}h_{n-i}\right)\mathbf{q}^{(p+1)}(t_n, \mathbf{x}(t_n)) + O(h_n^{p+2}). \end{aligned} \tag{4.42}$$

Define $\hat{\varphi}$, such that

$$\hat{\delta}_n = \frac{1}{p!}h_n^2(h_{n-1} + h_n)\cdots(h_{n-p+1} + \cdots + h_n)\hat{\Phi}(t_n, \mathbf{x}(t_n)).$$

This means that

$$\hat{\Phi}(t_n, \mathbf{x}(t_n)) = -\frac{1}{p+1}\mathbf{q}^{(p+1)}(t_n, \mathbf{x}(t_n)) + O(h_n).$$

Introduce for all n

$$\hat{\varphi}_n := \|\hat{\Phi}(t_n, \mathbf{x}(t_n))\|.$$

Then, because all stepsizes are positive, it follows that

$$\hat{r}_n := \|\hat{\delta}_n\| = \hat{\varphi}_n\frac{1}{p!}h_n^2(h_{n-1} + h_n)\cdots(h_{n-p+1} + \cdots + h_n). \tag{4.43}$$

This model describes the behaviour of $\hat{r}_n$ with respect to the stepsizes. It is a rather complex error model, which depends on p previous stepsizes. Note that for one-step methods, $\hat{r}_n$ only depends on the last stepsize h_n. This error estimate $\hat{r}_n$ must satisfy the constraints

$$\forall_n \hat{r}_n \leq \text{TOL}, \tag{4.44}$$

where TOL is the tolerance level. Often model (4.43) is simplified, to make the analysis less complex. Introducing

$$\Omega_n = \frac{1}{p!}\xi_{n,1}\cdots\xi_{n,p}$$

for all n, implies that $\hat{r}_n$ also satisfies the model

$$\hat{r}_n = \hat{\varphi}_n \Omega_n h_n^{p+1}.$$

Now, it is assumed that Ω_n is nearly independent of the stepsize sequence, so $\Omega_n = 1 + O(h_n)$. Note that for fixed stepsizes, $\Omega_n = 1$. Furthermore, define $\hat{\psi}_n$ by $\hat{\psi}_n = \hat{\phi}_n \Omega_n$, then

$$\hat{r}_n = \hat{\psi}_n h_n^{p+1}. \tag{4.45}$$

This means that $\hat{\psi}_n \doteq \hat{\varphi}_n$ also is nearly independent of the stepsizes.
Let $\epsilon = \theta\mathrm{TOL}$ be the wanted accuracy, where $0 < \theta < 1$ is a safety factor. In the most classic DAE-solvers, the *stepsize controllers* use the *elementary control law* (4.46) together with some additional nonlinear control actions. Here

$$h_n = \left(\frac{\epsilon}{\hat{r}_{n-1}}\right)^{\frac{1}{p+1}} h_{n-1}. \tag{4.46}$$

The nonlinearity is caused by the many logical if-else statements. However, the logarithmic version of the elementary control law is linear. In practice, the stepsize controller is a combination of a linear and a non-linear part. In the classical case, the linear part is an ordinary deadbeat-controller, while the non-linear part may consist of saturations, dead-zones, memory, etc. In [52], Söderlind proposes to expand the linear part to a PID-controller with free control parameters and to reduce the non-linear part. Linear controllers result in much smoother *stepsize sequences* than the original non-linear controllers. Therefore the controlled timestep variations are claimed to be less sensitive with respect to parameter variations than in classical time integration procedures. Hence, the results obtained with automatic control will be more robust and better suited for optimisation purposes than before. The *smoothness* of the stepsize and *error sequences* is quantified by means of the number $s(x) = \sqrt{\sum_{m=1}^{N}(x_m - x_{m-1})^2/\|x\|_2}$. Fig. (4.1) shows the structure of the classical stepsize controllers.

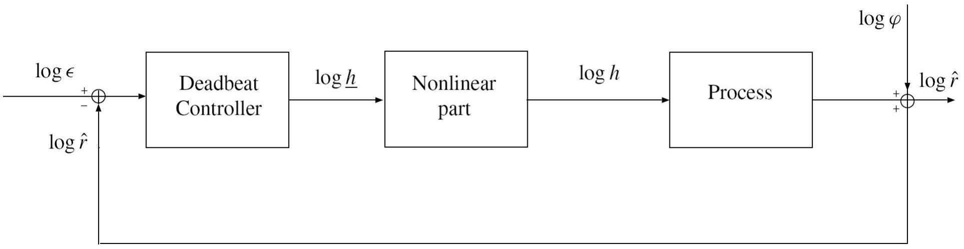


Figure 4.1: Structure of classical approach to stepsize control.

Adaptive stepsize control is used to control the local errors of the numerical solution. For optimisation purposes special stepsize controllers can ensure that the errors and stepsizes also behave smoothly. For onestep methods, the stepsize control process can be viewed as a digital (i.e. discrete) linear control system for the logarithms of the errors and steps. For the multistep BDF-method this control process can be approximated by such a linear control system. This section shows how *digital linear control theory* can be used to design better stepsize controllers. For more information the reader is referred to [59,62,63].

It seems attractive to use control-theoretic techniques for error control. Figure 4.2 shows the block diagram of this feedback control system. The process model $E(q)$ and the controller model $H(q)$ are described below.

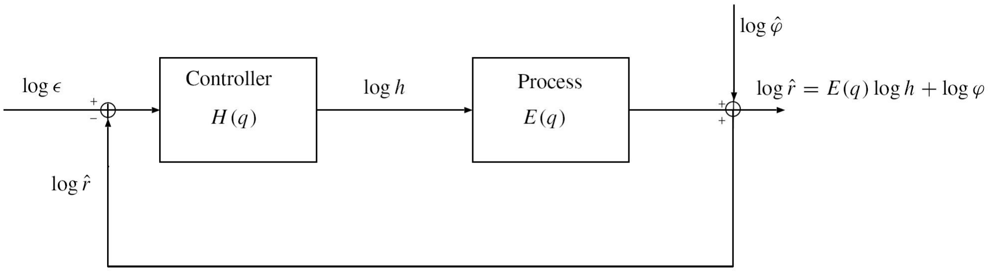


Figure 4.2: Diagram of adaptive stepsize control viewed as a feedback control system.

The logarithmic version of the *onestep error model* (4.45) is

$$\log \hat{r}_n = (p+1)\log h_n + \log \hat{\varphi}_n. \tag{4.47}$$

Writing $\log \hat{r} = \{\log \hat{r}_n\}_{n\in\mathbb{N}}, \log h = \{\log h_n\}_{n\in\mathbb{N}}$ and $\log \hat{\varphi} = \{\log \hat{\varphi}_n\}_{n\in\mathbb{N}}$, this implies that the sequence $\log \hat{r}$ can be viewed as the output of a digital (i.e. discrete) linear control system, where $\log h$ is the input signal and $\log \hat{\varphi}$ is an unknown output disturbance. We can denote all linear models with finite recursions for $\log \hat{r}$ by

$$\log \hat{r} = E(q)\log h + \log \hat{\varphi}, \tag{4.48}$$

where q is the *shift-operator*, with $q(\log h_n) = \log h_{n+1}$, and where $E(q)$ is a rational function of q:

$$E(q) = \frac{L(q)}{K(q)} = \frac{\lambda_0 q^M + \cdots + \lambda_M}{q^M + \kappa_1 q^{M-1} + \cdots + \kappa_M}. \tag{4.49}$$

For the onestep model, we just have that $E(q) = (p+1)$ is constant. However, it is not possible to derive a linear model of this form for the multistep BDF methods. In this case for a p-step method, we have the following nonlinear model for $\log \hat{r}$

$$\log \hat{r}_n = 2\log h_n + \log(h_{n-1}+h_n) + \cdots + \log(h_{n-p+1}+\cdots+h_n) + \log \hat{\varphi}_n - \log p!. \tag{4.50}$$

Note that $\log \hat{r}_n$ also depends on the previous stepsizes, because it is a multistep method. In [33] it is tried to approximate this model by the previous model for onestep methods, but this is not the optimal choice. Another possibility is to use an adaptive process model which is based on parameter identification.

If the stepsizes only have small variations, also linearisation can be used [49]. In [59] it is proved that the linearised model is equal to

$$\log \hat{r}_n = (1+\gamma_p)\log h_n + (\gamma_p - \gamma_1)\log h_{n-1} + \cdots + (\gamma_p - \gamma_{p-1})\log h_{n-p+1} + \log \hat{\varphi}_n, \tag{4.51}$$

where $\gamma_m = \sum_{n=1}^{m} \frac{1}{n}$ for $m \in \mathbb{N}$. This model can also be cast in the form (4.48), where

$$E(q) = \frac{(1+\gamma_p)q^{p-1} + (\gamma_p - \gamma_1)q^{p-2} + \cdots + (\gamma_p - \gamma_{p-1})}{q^{p-1}}. \tag{4.52}$$

The logarithmic version of the controller for the elementary control law in eqn. (4.46) is

$$\log h_n - \log h_{n-1} = \frac{1}{p+1}(\log \epsilon - \log \hat{r}_{n-1}). \tag{4.53}$$

So, also the control action can be viewed as a linear feedback controller for the same linear system. The input $\log h$ is computed based on the previous values of the output $\log \hat{r}$ and the reference $\log \epsilon$. All linear controllers can be denoted by

$$\log h = H(q)(\log \epsilon - \log \hat{r}), \tag{4.54}$$

where $H(q)$ is a rational function of q:

$$H(q) = \frac{B(q)}{A(q)} = \frac{\beta_0 q^{N-1} + \cdots + \beta_{N-1}}{q^N + \alpha_1 q^{N-1} + \cdots + \alpha_N}. \tag{4.55}$$

For the controller of eqn. (4.46) we just have that $H(q) = \frac{1}{p+1}\frac{1}{q-1}$.
Now we consider the *error model* (4.48), which is controlled by the linear controller (4.54). It is assumed that the error model is already available, while the controller still must be designed. This means that K, L are known, while A, B are unknown. Now, the closed loop dynamics are described by the following equations:

$$\begin{cases} \log h = U_r(q)\log \epsilon + U_w(q)\log \hat{\varphi}, \\ \log \hat{r} = Y_r(q)\log \epsilon + Y_w(q)\log \hat{\varphi}. \end{cases} \tag{4.56}$$

The transfer functions satisfy

$$\begin{array}{ll} U_r(q) = \frac{B(q)K(q)}{R(q)}, & U_w(q) = \frac{-B(q)K(q)}{R(q)}, \\ Y_r(q) = \frac{B(q)L(q)}{R(q)}, & Y_w(q) = \frac{A(q)K(q)}{R(q)}, \end{array} \tag{4.57}$$

where $R(q) = A(q)K(q) + B(q)L(q)$. In this section we will derive conditions for A, B such that the closed loop dynamics have some preferred properties.
The output $\log \hat{r}$ depends on the reference signal $\log \epsilon$ and the disturbance $\log \hat{\varphi}$. This means that the control error $\log \epsilon - \log \hat{r}$ deviates from zero in general. However, in (4.56) there is no control error if $Y_w(q)\log \hat{\varphi} = 0$ and $Y_r(1) = 1$. If $\log \hat{\varphi}$ is a polynomial of degree $p_A - 1$ and $Y_w(q)\log \hat{\varphi} = 0$, we call p_A the *order of adaptivity*. It is always required that $p_A \geq 1$ in order to have no control error for a constant disturbance. For higher order adaptivity the controller is capable to follow linear or other polynomial trends of the disturbance $\log \hat{\varphi}$. It can be proved that the controller is adaptive with *adaptivity order* p_A if $(q-1)^{p_A}$ is a divisor of $A(q)$

$$A(q) = (q-1)^{p_A}\hat{A}(q)$$

Because of numerical errors, the disturbance $\log \hat{\varphi}$ can contain alternating noise with frequency near π. The controller acts like a *filter* for the stepsizes if

$$|U_w(e^{i\omega})| = O\left((|\omega| - \pi)^{p_F}\right), \quad |\omega| \to \pi. \tag{4.58}$$

This means that alternations are always filtered, while also frequency components near $|\omega| = \pi$ are removed. Higher filter order results in a smaller transition band between the passed and filtered frequencies. Because

$$\lim_{|\omega|\to\pi} \frac{e^{i\omega}+1}{|\omega|-\pi} = -i,$$

it follows that the equation (4.58) is equivalent to

$$|U_w(e^{i\omega})| = O(|e^{i\omega}+1|^{p_F}), \quad |\omega| \to \pi.$$

Because $\lim_{|\omega|\to\pi} e^{i\omega} = -1$, it follows that

$$|U_w(z)| = O(|z+1|^{p_F}), \quad z \to -1.$$

Thus, if $U_w(z)$ is divisible by $(z+1)^{p_F}$, its *filter order* is equal to p_F. This means that $z^* = -1$ is a zero of $U_w(z)$ with multiplicity p_F. It is also possible to filter the output signal itself. The controller has output filter order p_R with respect to w if

$$|Y_w(e^{i\omega})| = O\left((|\omega|-\pi)^{p_R}\right), \quad |\omega| \to \pi. \tag{4.59}$$

Again, this is satisfied if $Y_w(z)$ is divisible by $(z+1)^{p_R}$. It is not possible to combine an *error filter* with a *stepsize filter*.
The poles of the system are determined by the $N+M$ roots of the characteristic equation

$$A(q)K(q) + B(q)L(q) = 0.$$

If the poles lie inside the complex unity circle, the closed loop system is stable. The absolute values determine the reaction speed of the controllers, while the angles determine the eigenfrequencies. This means that for real positive poles there are no oscillations.
If the controller is adaptive, we know that the error always will be equal to the reference level if the disturbance is a low degree polynomial. However, this will never be the case in practice. Thus it is still possible that the error will be larger than the tolerance level TOL.
Let R, S be polynomials of degree $N+M$, such that

$$\begin{array}{lllll} S(q) & = & A(q)K(q) & = & q^{N+M} + \sigma_1 q^{N+M-1} + \cdots + \sigma_{N+M} \\ R(q) & = & A(q)K(q) + B(q)L(q) & = & q^{N+M} + \rho_1 q^{N+M-1} + \cdots + \rho_{N+M} \end{array}$$

Let θ be the safety factor such that $\epsilon = \theta$ TOL and $R(1) = 1 + \sum_{k=1}^{N+M} \rho_k$. There are no *rejections*, such that $\hat{r}_n \geq$TOL if

- The disturbance $\hat{\varphi}$ satisfies the inequality:

$$\theta^{R(1)} \hat{\varphi}_n (\hat{\varphi}_{n-1})^{\sigma_1} \cdots (\hat{\varphi}_{n-N-M})^{\sigma_{N+M}} \leq 1. \tag{4.60}$$

- The coefficients of $R(q)$ satisfy: $\rho_i \leq 0, \quad i \in \{1, \ldots, N+M\}$, e.g. $R(q) = q^{N+M} - r^{N+M}$.

- The previous $N+M$ stepsizes have been *accepted*.

The first condition for the disturbance $\hat{\varphi}$ also depends on the safety factor θ. Note that a small θ will decrease the number of future rejections indeed. The second condition can not be satisfied if all poles equal to $0 < r < 1$ such that $R(q) = (q-r)^{N+M}$. However, for e.g. $R(q) = q^{N+M} - r^{N+M}$, this property is satisfied.
Note that a small safety factor θ will decrease the number of future rejections because of the first condition. The second condition is not true if all poles are real positive.
In order to get the optimal control parameters, in [20, 51] a systemetic investigation is done for a large range of possible control parameters. Below we propose a theoretical approach which is only based on the closed loop dynamics. Assume that A, B can be factorised like $A(q) = (q-1)^{p_A}(q+1)^{p_R}\tilde{A}(q)$ and $B(q) = (q+1)^{p_F}\tilde{B}(q)$. Then the order of adaptivity is equal to p_A, while the filter orders are p_R and p_F. Because $q = 1$ and $q = -1$ are not stable poles, it is not allowed that $A(1) = B(1) = 0$ or $A(-1) = B(-1) = 0$. Thus it follows that $p_R = 0$ or $p_F = 0$. Let $R(q)$ be the polynomial whose roots are equal to the wanted poles, then the polynomials A, B are determined by

$$(q-1)^{p_A}(q+1)^{p_R}\tilde{A}(q)K(q) + (q+1)^{p_F}\tilde{B}(q)L(q) = R(q). \tag{4.61}$$

The coefficients of A, B are the control parameters, which can be computed from (4.61). Instead of this theoretical approach, in [20] a systemetic investigation is done for a large range of possible control parameters.
If the coefficients of the controller $H(q)$ are derived, this controller can be used to compute the stepsizes. Assume that $p_A \geq 1$, then there exists a polynomial $\bar{A}(q)$ of degree $N-1$, such that $A(q) = (q-1)\bar{A}(q) = (q-1)(q^{N-1} + \bar{\alpha}_1 q^{N-2} + \ldots + \bar{\alpha}_{N-1})$. Now,

$$(q-1)\bar{A}(q)\log h = B(q)(\log \epsilon - \log \hat{r}).$$

This is equivalent to the *control law*

$$h_n = \left(\frac{\epsilon}{\hat{r}_{n-1}}\right)^{\beta_0} \cdots \left(\frac{\epsilon}{\hat{r}_{n-N}}\right)^{\beta_{N-1}} \left(\frac{h_{n-1}}{h_{n-2}}\right)^{-\bar{\alpha}_1} \cdots \left(\frac{h_{n-N+1}}{h_{n-N}}\right)^{-\bar{\alpha}_{N-1}} h_{n-1}. \tag{4.62}$$

This controller can easily be implemented in a numerical DAE solver. It needs the N previous stepsizes $h_{n-1}, \ldots, h_{n-N}$ and the N previous error estimators $\hat{r}_{n-1}, \ldots, \hat{r}_{n-N}$. Controllers with large N have the disadvantage of a larger storage because of the needed time history. After discontinuities, the integration order decreases to one, which implies that the process model is no longer valid. This implies that another stepsize controller has to be used, which can not use the previous stepsizes and errors. If after rejections, other control techniques are used, these results do not satisfy the closed loop dynamics. Since the behaviour of the error estimate is not exactly modelled by the process models, it is not completely sure that controller (4.62) maintains its properties, such as adaptivity and *filter-properties*, for the integration process itself. From the experiments [59] it seems not always attractive to use higher order adaptive controllers. However, filtering appears to be attractive because it reduces the high-frequent noise, which makes the behaviour of the stepsizes and the errors much smoother. To deal with the trade-off between the smoothness and the speed, optimal control could be applied. In this case, a cost function should be defined which depends on the stepsize sequence and the error sequence.
Besides the stepsizes also the integration order can be controlled adaptively. Clearly the

local error also depends on the integration order and this affects the process model. The influence of the order to the error of the used BDF scheme can be modelled by:

$$r_n = \phi_n \cdot [w_n h_n]^{p+1}, \tag{4.63}$$

where w_n is the *bandwidth* (maximal frequency) of the solution $\mathbf{x}(t)$ and p is the order. Also the optimal order p depends on the dynamical behaviour of the solution. Only if the bandwidth satisfies $w_n h_n < 1$, it is attractive to increase the order p. This means that for simulations of low accuracy, with large h_n, low orders are often preferable. Because the bandwidth of $\mathbf{x}(t)$ is unknown it also has to be estimated. This can be done by comparing two error estimates of different order. Note that error estimates for lower order can be derived from the current Nordsieck array [13,59,73].

Chapter 5

BDF multirate time-integration for DAEs

5.1 Introduction

In the previous sections we showed that Integrated Circuits can be modelled mathematically by differential-algebraic equations of type (4.1). In contrast to classical *single-rate time-integration* methods, multirate methods integrate parts with different dynamical behaviour, with different stepsizes or even with different schemes. Besides the coarse time-grid $\{T_n, 0 \leq n \leq N\}$ with stepsizes $H_n = T_n - T_{n-1}$, also a fine time-grid $\{t_{n-1,m}, 1 \leq n \leq N, 0 \leq m \leq q_n\}$ is used with stepsizes $h_{n,m} = t_{n,m} - t_{n,m-1}$. If the two time-grids are *synchronised*, $T_n = t_{n,0} = t_{n-1,q_n}$ holds for all n. Here q_n is

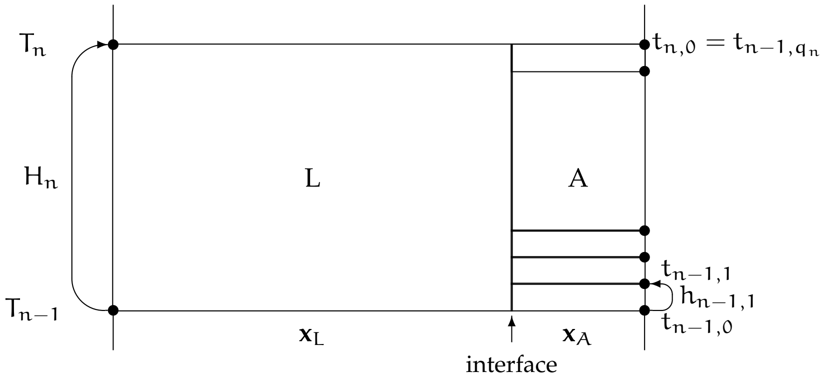


Figure 5.1: Schematic overview of multirate time-integration of latent and active parts at the coarse and fine time-grids.

called the *multirate factor*, which satisfies

$$H_n = \sum_{m=1}^{q_n} h_{n,m}. \tag{5.1}$$

For a multirate method it is necessary to *partition* the variables and equations into an active (A) and a latent (L) part. This can be done by the user or automatically. Let $\mathbf{B}_A \in \mathbb{R}^{d_A \times d}$ and $\mathbf{B}_L \in \mathbb{R}^{d_L \times d}$, with $d_A + d_L = d$, be the *selection operators* with the orthogonality properties: $\mathbf{B}_A\mathbf{B}_A^T = \mathbf{I}_A \in \mathbb{R}^{d_A \times d_A}, \mathbf{B}_L\mathbf{B}_A^T = \mathbf{O}, \mathbf{B}_A\mathbf{B}_L^T = \mathbf{O}, \mathbf{B}_L\mathbf{B}_L^T = \mathbf{I}_L \in \mathbb{R}^{d_L \times d_L}$. Then the variables and functions can be split into *active (A)* and *latent (L)* parts $\mathbf{x}_A \in \mathbb{R}^{d_A}, \mathbf{x}_L \in \mathbb{R}^{d_L}$:

$$\begin{aligned} \mathbf{x} &= \mathbf{B}_A^T\mathbf{x}_A + \mathbf{B}_L^T\mathbf{x}_L, \\ \mathbf{q}(t,\mathbf{x}) &= \mathbf{B}_A^T\mathbf{q}_A(t,\mathbf{B}_A\mathbf{x},\mathbf{B}_L\mathbf{x}) + \mathbf{B}_L^T\mathbf{q}_L(t,\mathbf{B}_A\mathbf{x},\mathbf{B}_L\mathbf{x}), \\ \mathbf{j}(t,\mathbf{x}) &= \mathbf{B}_A^T\mathbf{j}_A(t,\mathbf{B}_A\mathbf{x},\mathbf{B}_L\mathbf{x}) + \mathbf{B}_L^T\mathbf{j}_L(t,\mathbf{B}_A\mathbf{x},\mathbf{B}_L\mathbf{x}). \end{aligned} \tag{5.2}$$

Then equation (3.1) is equivalent to the *partitioned system*

$$\begin{aligned} \tfrac{d}{dt}\left[\mathbf{q}_A(t,\mathbf{x}_A,\mathbf{x}_L)\right] + \mathbf{j}_A(t,\mathbf{x}_A,\mathbf{x}_L) = \mathbf{0}, \\ \tfrac{d}{dt}\left[\mathbf{q}_L(t,\mathbf{x}_A,\mathbf{x}_L)\right] + \mathbf{j}_L(t,\mathbf{x}_A,\mathbf{x}_L) = \mathbf{0}. \end{aligned} \tag{5.3}$$

Of course it is also possible to extend this partitioning into a partitioning of k sub-systems, where the k sub-systems have a decreasing dynamical activity.

$$\begin{aligned} \tfrac{d}{dt}\left[\mathbf{q}_1(t,\mathbf{x}_1,\ldots\mathbf{x}_k)\right] + \mathbf{j}_1(t,\mathbf{x}_1,\ldots,\mathbf{x}_k) = \mathbf{0}, \\ \vdots \\ \tfrac{d}{dt}\left[\mathbf{q}_k(t,\mathbf{x}_1,\ldots\mathbf{x}_k)\right] + \mathbf{j}_k(t,\mathbf{x}_1,\ldots\mathbf{x}_k) = \mathbf{0}. \end{aligned} \tag{5.4}$$

In the sequel we need the selection operators $\mathbf{B}_i \in \mathbb{R}^{d_i \times d}$ for $i = 1,\ldots,k$ with the properties

$$\mathbf{B}_i\mathbf{B}_j^T = \begin{cases} \mathbf{I}_i \in \mathbb{R}^{n_i \times n_i} & \text{if } i = j \\ \mathbf{O} & \text{if } i \neq j \end{cases} \tag{5.5}$$

Then the variables and functions can be split in active (A) and latent (L) parts:

$$\begin{aligned} \mathbf{x} &= \mathbf{B}_1^T\mathbf{x}_1 + \ldots + \mathbf{B}_k^T\mathbf{x}_k, \\ \mathbf{q}(t,\mathbf{x}) &= \mathbf{B}_1^T\mathbf{q}_1(t,\mathbf{B}_1\mathbf{x},\ldots,\mathbf{B}_k\mathbf{x}) + \ldots + \mathbf{B}_k^T\mathbf{q}_k(t,\mathbf{B}_1\mathbf{x},\ldots,\mathbf{B}_k\mathbf{x}), \\ \mathbf{j}(t,\mathbf{x}) &= \mathbf{B}_1^T\mathbf{j}_1(t,\mathbf{B}_1\mathbf{x},\ldots,\mathbf{B}_k\mathbf{x}) + \ldots + \mathbf{B}_k^T\mathbf{j}_k(t,\mathbf{B}_1\mathbf{x},\ldots,\mathbf{B}_k\mathbf{x}). \end{aligned} \tag{5.6}$$

If the circuit consists of a number of nearly decoupled subcircuits, it is possible to apply relaxation techniques. The following theoretic results have been derived from [37,44,58, 72], where this topic is investigated more profoundly. It is possible to apply relaxation to the steady state equations, both in the linear or nonlinear case. However it is also possible to apply relaxation directly to the DAE (4.1) itself.

If $\mathbf{A}\mathbf{x} = \mathbf{b}$ is a linear system, that has to be solved and $\mathbf{A}$ is split such that $\mathbf{A} = \mathbf{B} - \mathbf{C}$ with $\mathbf{B}$ nonsingular, then a *linear relaxation* method is given by the following iterations

$$\mathbf{x}^{k+1} = \mathbf{B}^{-1}\mathbf{C}\mathbf{x}^k + \mathbf{B}^{-1}\mathbf{b}.$$

The idea behind this is that the linear system $\mathbf{Bx} = \mathbf{d}$ is easy to solve. Note that relaxation here is equivalent to applying the modified Newton method to $\mathbf{Ax} - \mathbf{b} = \mathbf{0}$, where the Jacobian matrix is approximated by $\mathbf{B}$. For the Jacobi methods, $\mathbf{B}$ is a diagonal matrix, while for Gauss-Seidel, $\mathbf{B}$ is a lower-triangular matrix. Note that for the Jacobi version, parallelism can be used. It is well-known that if the spectral radius $\rho(\mathbf{B}^{-1}\mathbf{C})$ is smaller than one, the solution $\mathbf{x}^k$ converges to the exact solution. Furthermore, if $\mathbf{A}$ is an M-matrix it follows that $\rho(\mathbf{B}_{GS}^{-1}\mathbf{C}_{GS}) \leq \rho(\mathbf{B}_J^{-1}\mathbf{C}_J)$ [58]. Here $\rho(\mathbf{B}_J^{-1}\mathbf{C}_J)$ is independent of permutations in contrast to $\rho(\mathbf{B}_{GS}^{-1}\mathbf{C}_{GS})$. It can be proved that this is always the case, if $\mathbf{A}$ is diagonally dominant.

The above mentioned idea can be generalised to a nonlinear system. Let $\mathbf{F}(\mathbf{x}) = \mathbf{b}$ be a nonlinear equation, such that the Jacobian matrix $J(\mathbf{x}) = \frac{\partial \mathbf{F}}{\partial \mathbf{x}}$ satisfies

$$\mathbf{J}(\mathbf{x}) = \mathbf{B}(\mathbf{x}) - \mathbf{C}(\mathbf{x}),$$

and $\mathbf{B}(\mathbf{x})$ is nonsingular. This method converges locally if $\mathbf{B}^{-1}\mathbf{C}$ is a contraction.

It is also possible to generalise the relaxation to the continuous DAE (4.1) itself resulting in *Waveform Relaxation (WR)* [72]. The *Gauss-Seidel version* corresponds closely with multirate methods. Note that the order of the equations is important, because of the coupling between the variables. An alternative version of waveform relaxation is the WR *Jacobi method*. Then all subcircuits can be integrated independently, because the order is not important. This implies that for this version parallelism is possible. In general, the Jacobi version will converge slower than the Gauss-Seidel method. The WR-method can also be performed on a sequence of windows $[T_i, T_{i+1}]$. In this case, it is possible to get a better initial guess $\mathbf{x}^0(t)$, which can improve the convergence speed.

5.2 Types of multirate

Multirate time-integration has the common property that one needs to integrate a partitioned system like (5.4) by using different timesteps for each subsystem. A multirate method integrates the slowest part with a (large) *macrostep* H, while the faster subsystems are integrated with smaller *microsteps*. We assume that no additional macrosteps are done before all faster subsystems have also been integrated. There are two very natural types of multirate methods: Two very natural types of multirate methods are *Slowest First* and *Fastest First*.

Slowest First The subsystem with the slowest dynamical behaviour (or with the largest time constant) is first integrated with one step. Then the subsystems with increasingly faster behaviour are integrated until synchronisation at the end of time interval $[T_{n-1}, T_n]$.

Fastest First Here the subsystem with the fastest dynamical behaviour (or with the smallest time constant) is first integrated with one step. Then the subsystems with decreasingly faster behaviour are integrated.

Sometimes it is attractive to integrate a slow system together with all faster sub-systems with one microstep, instead of extrapolating them. Although this compound step is more expensive, it is also more stable.
Sub-systems can subsequentially be solved, using the results of each other. Multirate methods with this property are called of Gauss-Seidel type. There are also multirate methods of Jacobi type, that solve the sub-systems independently. Usually Gauss-Seidel types is prefered because they are more efficient. However, Jacobi types are more suitable for parallelism. It is possible to integrate the subsystems more than once (relaxation). For some types of problems it can be proved that the multirate method will converge to the exact solution. Because convergence is difficult to guarantee and often very slow, it is better to truncate the relaxation. Very often, each part is integrated just once, while the accuracy is controlled by stepsize control.
To keep it simple, we will work with a time-independent type of the partitioned system (5.3) , and denote the active variable $\mathbf{y} = \mathbf{x}_A$ and the slow variable $\mathbf{z} = \mathbf{x}_L$.

$$\frac{d}{dt}\left[\mathbf{q}_A(\mathbf{y},\mathbf{z})\right] + \mathbf{j}_A(\mathbf{y},\mathbf{z}) = \mathbf{0}, \tag{5.7}$$

$$\frac{d}{dt}\left[\mathbf{q}_L(\mathbf{y},\mathbf{z})\right] + \mathbf{j}_L(\mathbf{y},\mathbf{z}) = \mathbf{0}. \tag{5.8}$$

In this section some available multirate methods will be discussed. The multirate methods are independent of the integration method, but are presented for the Euler Backward scheme. Furthermore, the coarse and fine time-grids are assumed to be synchronised, which means that $t_{n-1,q} = T_n = t_{n,0}$. Multirate schemes have been investigated for long, see e.g. [7,16,17,18,46,50,53]. In this section we will summarise some common approaches.
Semi-implicit multirate methods only integrate the equations (5.7) and (5.8) separately, while the other parts are estimated by means of extrapolation or interpolation. The variable **z** should be a latent variable, that can be integrated with a large step-size H. This implies that the interpolation of **z** is expected to be very accurate. A a consequence **z** will almost be independent of the prediction errors of the active variables.
The *Fast-Slow (FS)* method first integrates the fast part (5.7), while $\mathbf{z}_{n-1,m}$ is estimated by means of extrapolation. Next the slow part (5.8) is integrated with one large step-size H, while **y** is already known. Often simple *zeroth order extrapolation* is used with $\hat{\mathbf{z}}_{n-1,m} = \mathbf{z}_{n-1}$. This FS-method is not very useful as step-size control is not possible. Indeed, if the large step-size H has to be reduced, then numerical solutions of **y** at other time-points are required. This implies that all numerical approximations of **y** at time-points larger than the new coarse time-point have to be rejected.
The *Slow-Fast (SF)* method (Algorithm 1) first integrates (5.8) with one large step-size H, while **y** is found through extrapolation. Then equation (5.7) is integrated with a smaller step-size h, while **z** at the intermediate time-points is found by means of interpolation. Often, zeroth order extrapolation and *first order interpolation* are used with $\hat{\mathbf{y}}_n = \mathbf{y}_{n-1}$ and $\hat{\mathbf{z}}_{n-1,m} = \mathbf{z}_{n-1} + \frac{m}{q}(\mathbf{z}_n - \mathbf{z}_{n-1})$ for $m = 0, \dots, q-1$.

ALGORITHM 1 *The Slow-Fast (SF) method*

Extrapolate $\hat{\boldsymbol{y}}_n$ *and solve for* z_n*:*

$$\begin{aligned} \hat{\boldsymbol{y}}_n - \boldsymbol{y}_{n-1} &= \boldsymbol{0} \\ \boldsymbol{q}_L(\hat{\boldsymbol{y}}_n, z_n) - \boldsymbol{q}_L(\boldsymbol{y}_{n-1}, z_{n-1}) + H\boldsymbol{j}_L(\hat{\boldsymbol{y}}_n, \boldsymbol{y}_n) &= \boldsymbol{0} \end{aligned}$$

Interpolate $\hat{z}_{n-1,m}$ *and solve for* $\boldsymbol{y}_{n-1,m+1}$ $\quad (m = 0, \dots, q-1)$*:*

$$\begin{aligned} \hat{z}_{n-1,m} - z_{n-1} - \frac{m}{q}(z_n - z_{n-1}) &= \boldsymbol{0} \\ \boldsymbol{q}_A(\boldsymbol{y}_{n-1,m+1}, \hat{z}_{n-1,m+1}) - \boldsymbol{q}_A(\boldsymbol{y}_{n-1,m}, \hat{z}_{n-1,m}) + h\boldsymbol{j}_A(\boldsymbol{y}_{n-1,m+1}, \hat{z}_{n-1,m+1}) &= \boldsymbol{0} \end{aligned}$$

Because these semi-implicit multirate methods use extrapolation, they have unstable behaviour. To improve the stability the latent part can be integrated using an implicit compound step. The *Compound-Fast (CF)* method (Algorithm 2) first integrates (5.7) and (5.8) together with one large step-size H, which results in $\mathbf{y}_n$ and $\mathbf{z}_n$. Then only equation (5.7) is integrated with a small step-size h, while $\mathbf{z}$ is found by interpolation.

ALGORITHM 2 *The Compound-Fast (CF) method*

Solve for z_n *and* $\boldsymbol{y}_n$*:*

$$\begin{aligned} \boldsymbol{q}_A(\boldsymbol{y}_n, z_n) - \boldsymbol{q}_A(\boldsymbol{y}_{n-1}, z_{n-1}) + H\boldsymbol{j}_A(\boldsymbol{y}_n, z_n) &= \boldsymbol{0} \\ \boldsymbol{q}_L(\boldsymbol{y}_n, z_n) - \boldsymbol{q}_L(\boldsymbol{y}_{n-1}, z_{n-1}) + H\boldsymbol{j}_L(\boldsymbol{y}_n, z_n) &= \boldsymbol{0} \end{aligned}$$

Interpolate $\hat{z}_{n-1,m}$ *and solve for* $\boldsymbol{y}_{n-1,m+1}$ $\quad (m = 0, \dots, q-1)$*:*

$$\begin{aligned} \hat{z}_{n-1,m} - z_{n-1} - \frac{m}{q}(z_n - z_{n-1}) &= \boldsymbol{0} \\ \boldsymbol{q}_A(\boldsymbol{y}_{n-1,m+1}, \hat{z}_{n-1,m+1}) - \boldsymbol{q}_A(\boldsymbol{y}_{n-1,m}, \hat{z}_{n-1,m}) + h\boldsymbol{j}_A(\boldsymbol{y}_{n-1,m+1}, \hat{z}_{n-1,m+1}) &= \boldsymbol{0} \end{aligned}$$

Note that $\mathbf{y}_n$ is computed twice by the CF method. Another possibility is the *Mixed Compound-Fast (MCF)* method, which computes $\mathbf{y}_{n-1,1}$ and $\mathbf{z}_n$ simultaneously. The first active solution $\mathbf{y}_{n-1,1}$ is already computed at the compound step. Note that $\mathbf{y}_{n-1,1}$ is equal to the solution of the integration of the fastest part for $m = 0$. In fact, the CF method and the MCF methods are special cases of the Generalised Compound-Fast method. The CF method has the benefit that it is more stable and easier to implement, while the MCF method results in better scaled nonlinear equations which are easier to solve by the Newton method. This approach is also used by the MROW method [6]. Algorithm 3 shows a generalised version [65] of the Mixed Compound-Fast method, where $\alpha \in \mathbb{R}$. This family of methods contains the Slow-Fast ($\alpha \to \infty$), the Compound-Fast method itself ($\alpha = 1$) and the Mixed Compound-Fast method ($\alpha = \frac{1}{q}$).

ALGORITHM 3 *The Generalised Compound-Fast (GCF) method*

Interpolate $\hat{z}_{n-1,\alpha q}$*, extrapolate* $\hat{\boldsymbol{y}}_n$ *and solve for* $\boldsymbol{y}_{n-1,\alpha q}$ *and* z_n*:*

$$\begin{aligned}
\hat{z}_{n-1,\alpha q} - z_{n-1} - \alpha(z_n - z_{n-1}) &= \boldsymbol{0}\\
\hat{\boldsymbol{y}}_n - \boldsymbol{y}_{n-1} - \frac{1}{\alpha}(\boldsymbol{y}_{n-1,\alpha q} - \boldsymbol{y}_{n-1}) &= \boldsymbol{0}\\
\boldsymbol{q}_A(\boldsymbol{y}_{n-1,\alpha q}, \hat{z}_{n-1,\alpha q}) - \boldsymbol{q}_A(\boldsymbol{y}_n, z_n) + \alpha H \boldsymbol{j}_A(\boldsymbol{y}_{n-1,\alpha q}, \hat{z}_{n-1,\alpha q}) &= \boldsymbol{0}\\
\boldsymbol{q}_L(\hat{\boldsymbol{y}}_n, z_n) - \boldsymbol{q}_L(\boldsymbol{y}_{n-1}, z_{n-1}) + H \boldsymbol{j}_L(\hat{\boldsymbol{y}}_n, z_n) &= \boldsymbol{0}
\end{aligned}$$

Interpolate $\hat{z}_{n-1,m}$ *and solve for* $\boldsymbol{y}_{n-1,m+1}$ $(m = 0, \ldots, q-1)$*:*

$$\begin{aligned}
\hat{z}_{n-1,m} - z_{n-1} - \frac{m}{q}(z_n - z_{n-1}) &= \boldsymbol{0}\\
\boldsymbol{q}_A(\boldsymbol{y}_{n-1,m+1}, \hat{z}_{n-1,m+1}) - \boldsymbol{q}_A(\boldsymbol{y}_{n-1,m}, \hat{z}_{n-1,m}) + h \boldsymbol{j}_A(\boldsymbol{y}_{n-1,m+1}, \hat{z}_{n-1,m+1}) &= \boldsymbol{0}
\end{aligned}$$

Although the integration methods used for the sub-circuits can be A-stable, this is not the case for the multirate version [50]. Indeed, for multirate methods the results also depend on the extrapolated or interpolated results of the other part. Thus the stability will always strongly depend on the used partitioning and on the coupling. In particular, the extrapolation may cause unstable behaviour. Therefore it is expected that methods with an implicit compound step are more stable, because they do not employ explicit extrapolation. Besides the previous methods, in [50] also *Implicit multirate method* fully implicit multirate methods exist. Here a large system of algebraic equations is solved for all subsystems. This means that no interpolation or extrapolation is necessary. Compared to the other multirate methods, it needs more computational time but also has better stability properties. The solution of the nonlinear equations requires dedicated solution techniques. Alternatively the solution of a fully implicit multirate method can be approximated by a convergent Waveform Relaxation method. Table 5.1 summarises all mentioned types of multirate time-integration.

In the sections below we will focus on the Compound-Fast multirate methods. Multirate is also applied to the time-integration of ODEs and spatial discretised PDEs [16, 19, 29, 30, 46, 53]. Since Integrated Circuits are modelled by differential-algebraic (DAE) models, we will concentrate on multirate methods applied to differential-algebraic systems.

5.3 The BDF Compound-Fast multirate algorithm

This section describes how the *BDF Compound-Fast (BDF-CF) multirate algorithm* can be efficiently implemented by means of the Nordsieck data representation. Although also other implicit methods can be used, like Runge Kutta methods, we use the Backward Difference Formulae (BDF method) because they use less function evaluations and are

Table 5.1: Several types of multirate time-integration.

Name:	Brief description:
Waveform-Relaxation (WR)	All subsystems are independently solved and updated until convergence,
Jacobi	In each iteration the subsystems are independently solved allowing parallelism.
Gauss-Seidel	In each iteration the subsystems are subsequentially solved in a specific order.
Fully Implicit	Multirate type for strongly coupled subsystems without interpolation or extrapolation. It could be efficiently implemented by using WR.
Semi-implicit	Multirate type for weakly coupled subsystems with interpolation or extrapolation.
Fastest First	All subsystems are subsequentially solved and updated once in Fastest First order.
Slowest First	All subsystems are subsequentially solved and updated once in Slowest First order.
Fast-Slow (FS)	The Fast and Slow subsystems are subsequentially solved and updated once in Fastest First order.
Slow-Fast (SF)	The Fast and Slow subsystems are subsequentially solved and updated once in Slowest First order.
Compound-Fast (CF)	A more stable modification of the SF method where the complete system is solved at the coarse time-grid.
Generalised Compound-Fast (GCF)	A generalisation of the CF method where during the compound step the latent part $\mathbf{z}_n$ is solved simultaneously with the active part $\mathbf{y}_{n-1,\alpha q}$ at a different time-point $t_{n-1,\alpha q}$.
Mixed Compound-Fast (MCF)	Special case of the GCF method with $\alpha = \frac{1}{q}$, saving one refinement step.

very well suited for interpolation. For linear multistep methods the solution can always be represented by a piecewise polynomial, which can be used to interpolate the latent interface variables without accuracy loss. The BDF algorithm has been described in more detail in the section 4.1. It stores the previous numerical values in Nordsieck matrices.

Before we describe the proposed multirate algorithm in detail for both the *compound step* and the *refinement phase*, we have to define the required polynomials for the BDF time-integration algorithm. Firstly, the BDF integration of order K on the coarse time-grid needs the Lagrangian basis polynomial $l^n(t)$ on $\{T_{n-K}, \ldots, T_n\}$, with

$$l^n(T_{n-j}) = \begin{cases} 1 & \text{if } j = 0, \\ 0 & \text{if } j \neq 0. \end{cases}$$

Let $\bar{\mathbf{l}}^n \in \mathbb{R}^{(K+1)}$ be the corresponding Nordsieck vector (Chapter 4) of $l^n(t)$ which can be expressed as

$$l^n(t) = \sum_{i=0}^{K} \bar{l}^n_{i+1} \left(\frac{t - T_n}{H_n}\right)^i,$$

where $\bar{l}^n_{i+1}$ is the $i+1$-st element of the Nordsieck vector

$$\bar{\mathbf{l}}^n = \left(l^n(T_n), H_n \frac{d}{dt} l^n(T_n), \dots, \frac{H_n^k}{k!} \frac{d^k}{dt^k} l^n(T_n)\right)^T.$$

Because of (4.27) and (4.26)

$$\bar{\mathbf{l}}^n = \left(1, \frac{1}{\xi_{n,1}} + \dots + \frac{1}{\xi_{n,k}}, \dots, \frac{1}{\xi_{n,1}} \cdots \frac{1}{\xi_{n,k}},\right)^T.$$

The Nordsieck matrices $\bar{\mathbf{Y}}^n, \bar{\mathbf{X}}^n, \bar{\mathbf{P}}^n, \bar{\mathbf{Q}}^n \in \mathbb{R}^{d \times (K+1)}$ are needed, where $\bar{\mathbf{Y}}^n, \bar{\mathbf{X}}^n$ represent the local predictor and corrector polynomials for $\mathbf{x}(t)$ and $\bar{\mathbf{P}}^n, \bar{\mathbf{Q}}^n$ are the corresponding ones for $\mathbf{q}(t, \mathbf{x}(t))$, respectively. For instance, the predictor polynomial $\mathbf{y}^n(t)$ for $\mathbf{x}(t)$ on $[T_{n-1}, T_n]$ satisfies

$$\mathbf{y}^n(t) = \sum_{i=0}^{K} \bar{\mathbf{y}}^n_{i+1} \left(\frac{t - T_n}{H_n}\right)^i,$$

where $\bar{\mathbf{y}}^n_{i+1}$ is the $(i+1)$-st column of the Nordsieck matrix

$$\bar{\mathbf{Y}}^n = \left(\mathbf{y}^n(T_n) | H_n \frac{d}{dt} \mathbf{y}^n(T_n) | \dots | \frac{H_n^k}{k!} \frac{d^k}{dt^k} \mathbf{y}^n(T_n)\right).$$

The multirate k-th order BDF method also re-integrates the active part independently on a fine time-grid. There it needs a different Lagrange basis polynomial $l^{n-1,m}(t)$ on $\{t_{n-1,m-k}, \dots, t_{n-1,m}\}$, with

$$l^{n-1,m}(t_{n-1,m-j}) = \begin{cases} 1 & \text{if } j = 0, \\ 0 & \text{if } j \neq 0. \end{cases}$$

Here the fine-grid Nordsieck matrices $\bar{\mathbf{Y}}_A^{n-1,m}, \bar{\mathbf{X}}_A^{n-1,m}, \bar{\mathbf{P}}_A^{n-1,m}, \bar{\mathbf{Q}}_A^{n-1,m} \in \mathbb{R}^{d_A \times (k+1)}$ are needed, of which $\bar{\mathbf{Y}}_A^{n-1,m}, \bar{\mathbf{X}}_A^{n-1,m}$ represent the local fine-grid predictor and corrector polynomials at the fine time-grid for $\mathbf{x}_A(t)$ and $\bar{\mathbf{P}}_A^{n-1,m}, \bar{\mathbf{Q}}_A^{n-1,m}$ are the corresponding ones for $\mathbf{q}_A(t, \mathbf{x}_A(t), \hat{\mathbf{x}}_L)$, respectively. The fine-grid predictor polynomial $\mathbf{y}_A^{n-1,m}(t)$ for $\mathbf{x}_A(t)$ on $[t_{n-1,m-1}, t_{n-1,m}]$ satisfies

$$\mathbf{y}_A^{n-1,m}(t) = \sum_{i=0}^{k} \bar{\mathbf{y}}_{A,i+1}^{n-1,m} \left(\frac{t - t_{n-1,m}}{h_{n-1,m}}\right)^i,$$

where

$$\bar{\mathbf{Y}}_A^{n-1,m} = \left(\mathbf{y}_A^{n-1,m}(t_{n-1,m}), \dots, \frac{h_{n-1,m}^k}{k!} \frac{d^k}{dt^k} \mathbf{y}_A^{n-1,m}(t_{n-1,m})\right).$$

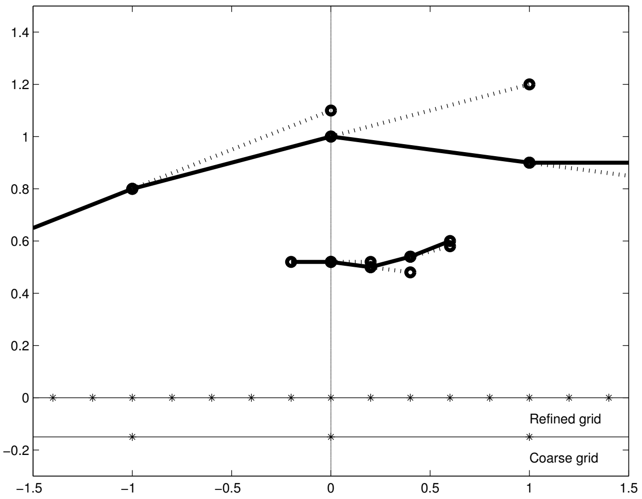


Figure 5.2: Typical form of the predictor (dashed) and corrector (solid) polynomials at the coarse (graph at the top) and at the fine time-grid (graph in the middle) with $q = 5$.

Figure 5.2 shows the typical form of the predictor and corrector polynomials on the coarse and fine grids. The polynomials are of degree one, which implies the use of linear extrapolation for the prediction. Clearly, the solution becomes smoother for higher degree polynomials.
In fact during the compound step a normal BDF step is done for the complete DAE. This means that the following algebraic system is solved

$$\rho_0^n \mathbf{q}(T_n, \mathbf{x}_n) + H_n \mathbf{j}(T_n, \mathbf{x}_n) + \mathbf{b}_n = \mathbf{0}, \tag{5.9}$$

where ρ_0^n is an order-dependent parameter and $\mathbf{b}_n$ is a vector that represents the collected terms of the numerical integration. We can easily express ρ_0^n and $\mathbf{b}_n$ in terms of $\bar{\mathbf{l}}^n$ and $\bar{\mathbf{P}}^n$ by $\rho_0^n = \bar{l}_2^n$ and $\mathbf{b}_n = \bar{\mathbf{p}}_2^n - \rho_0^n \bar{\mathbf{p}}_1^n$ (Chapter 4). Here $\bar{\mathbf{p}}_1^n$ and $\bar{\mathbf{p}}_2^n$ are just the first two columns of $\bar{\mathbf{P}}^n$, the Nordsieck matrix of the predictor polynomial $\mathbf{p}^n(t)$, which is set to be equal to the corrector polynomial $\mathbf{q}^{n-1}(t)$ of the previous compound step if the integration order is the same as in the previous timestep.

$$\mathbf{p}^n(t) = \mathbf{q}^{n-1}(t), \quad \mathbf{y}^n(t) = \mathbf{x}^{n-1}(t). \tag{5.10}$$

In section 4.3 it is shown that for the corresponding Nordsieck matrices this implies that

$$\bar{\mathbf{P}}^n = \bar{\mathbf{Q}}^{n-1} \mathbf{T}_n, \quad \bar{\mathbf{Y}}^n = \bar{\mathbf{X}}^{n-1} \mathbf{T}_n, \tag{5.11}$$

where $\mathbf{T}_n = \mathbf{T}^{(*)}(K_n, K_{n-1}, \frac{H_n}{H_{n-1}}, 1)$ has been defined before in Definition 4.9 (p. 27). Usually the initial guess for the solution of (5.9) is computed by extrapolation, so $\hat{\mathbf{x}}_n = \bar{\mathbf{y}}_1^n$. Because the compound step will be much larger than the time-constant of the active part, we use a modified Newton scheme which relaxes the active part of the residual by using a small positive weighting factor in front of the active part. For a compound step, the active part of $\mathbf{x}_n$ still must be improved by the refinement phase. After the refinement we use the updated $\mathbf{x}_n$ and $\mathbf{q}_n$ to correct the complete predictor polynomials $\mathbf{p}^n(t)$ and $\mathbf{y}^n(t)$

$$\begin{aligned} \mathbf{q}^n(t) &:= \mathbf{p}^n(t) + (\mathbf{q}(T_n, \mathbf{x}_n) - \mathbf{p}^n(T_n))l^n(t), && (5.12) \\ \mathbf{x}^n(t) &:= \mathbf{y}^n(t) + (\mathbf{x}_n - \mathbf{y}^n(T_n))l^n(t). && (5.13) \end{aligned}$$

For the Nordsieck matrices this means

$$\begin{aligned} \bar{\mathbf{Q}}^n &= \bar{\mathbf{P}}^n + (\mathbf{q}_n - \hat{\mathbf{q}}_n)\left(\bar{\mathbf{l}}^n\right)^T, && (5.14) \\ \bar{\mathbf{X}}^n &= \bar{\mathbf{Y}}^n + (\mathbf{x}_n - \hat{\mathbf{x}}_n)\left(\bar{\mathbf{l}}^n\right)^T. && (5.15) \end{aligned}$$

Here $(\mathbf{q}_n - \hat{\mathbf{q}}_n)\left(\bar{\mathbf{l}}^n\right)^T$ and $(\mathbf{x}_n - \hat{\mathbf{x}}_n)\left(\bar{\mathbf{l}}^n\right)^T$ are rank 1 - matrices.
In fact, the refinement solves a new initial value problem for a perturbed DAE but of smaller size. for each time-point $t_{n-1,m}$ it solves the nonlinear equation

$$\bar{\rho}_0^{n-1,m}\mathbf{q}_A(t_{n-1,m}, \mathbf{x}_A^{n-1,m}, \hat{\mathbf{x}}_L^{n-1,m}) + h_{n-1,m}\mathbf{j}_A(t_{n-1,m}, \mathbf{x}_A^{n-1,m}, \hat{\mathbf{x}}_L^{n-1,m}) + \mathbf{b}_A^{n-1,m} = \mathbf{0}, \tag{5.16}$$

where $\hat{\mathbf{x}}_L^{n-1,m}$ is the interpolated latent part. We are able to compute $\hat{\mathbf{x}}_L^{n-1,m}$ from interpolation-based functions that are constructed between the compound step and the refinement phase. Another possibility is to compute it from the Nordsieck matrix $\bar{\mathbf{X}}^n$, provided that its latent part is updated during the compound step. Because of (4.14) we have

$$\hat{\mathbf{x}}_L^{n-1,m} = \mathbf{B}_L\bar{\mathbf{X}}^n\mathbf{e}(K, \frac{t_{n-1,m} - T_n}{H_n}). \tag{5.17}$$

Furthermore $\bar{\rho}_0^{n-1,m}$ and $\mathbf{b}_A^{n-1,m}$ can be computed in a similar way as for the compound step: $\bar{\rho}_0^{n-1,m} = \bar{l}_2^{n-1,m}, \mathbf{b}_A^{n-1,m} = \bar{\mathbf{p}}_{A,2}^{n-1,m} - \bar{\rho}_0^{n-1,m}\bar{\mathbf{p}}_{A,1}^{n-1,m}$. The predictor Nordsieck matrices $\bar{\mathbf{P}}_A^{n-1,m}, \bar{\mathbf{Y}}_A^{n-1,m}$ are computed similarly by

$$\bar{\mathbf{P}}_A^{n-1,m} = \bar{\mathbf{Q}}_A^{n-1,m-1}\mathbf{T}_{n-1,m}, \quad \bar{\mathbf{Y}}_A^{n-1,m} = \bar{\mathbf{X}}_A^{n-1,m-1}\mathbf{T}_{n-1,m}, \tag{5.18}$$

where $\mathbf{T}_{n-1,m} = \mathbf{T}^{(*)}(k_{n-1,m}, k_{n-1,m-1}, \frac{h_{n-1,m}}{h_{n-1,m-1}}, 1)$ for $m > 0$. Note that the transformation matrix $\mathbf{T}^{(*)}$ has been defined before in Definition 4.9 (p. 27). For $m = 0$ we need the matrices $\bar{\mathbf{Q}}_A^{n-2,q-1}, \bar{\mathbf{X}}_A^{n-2,q-1}$ which are only available for a static partitioning. Otherwise we need to transform a part of the coarse Nordsieck matrices $\bar{\mathbf{Q}}^{n-1}, \bar{\mathbf{X}}^{n-1}$. In contrast to the compound phase it is not very difficult to solve the nonlinear equation (5.16) because we have an accurate initial guess $\hat{\mathbf{y}}_A^{n-1,m} = \mathbf{B}_A\bar{\mathbf{y}}_{A,1}^{n-1,m}$ in order to start the Newton method. The derived solution is used to correct the predictor polynomials. For the Nordsieck representation we get:

$$\begin{aligned} \bar{\mathbf{Q}}_A^{n-1,m} &= \bar{\mathbf{P}}_A^{n-1,m} + \left(\mathbf{q}_A(t_{n-1,m}, \mathbf{x}_A^{n-1,m}, \hat{\mathbf{x}}_L^{n-1,m}) - \hat{\mathbf{q}}_A\right)\left(\bar{\mathbf{l}}^{n-1,m}\right)^T, && (5.19) \\ \bar{\mathbf{X}}_A^{n-1,m} &= \bar{\mathbf{Y}}_A^{n-1,m} + \left(\mathbf{x}_A^{n-1,m} - \hat{\mathbf{x}}_A^{n-1,m}\right)\left(\bar{\mathbf{l}}^{n-1,m}\right)^T. && (5.20) \end{aligned}$$

Finally, if $t_{n-1,m} > T_n$ the fine-grid corrector polynomials are evaluated at T_n:

$$\mathbf{B}_A\mathbf{x}_n = \bar{\mathbf{X}}_A^{n-1,m}\mathbf{e}(k, \frac{T_n - t_{n-1,m}}{h_{n-1,m}}), \quad \mathbf{B}_A\mathbf{q}_n = \bar{\mathbf{Q}}_A^{n-1,m}\mathbf{e}(k, \frac{T_n - t_{n-1,m}}{h_{n-1,m}}), \tag{5.21}$$

where again we used (4.14).

5.4 Analysis of conditions for partitioning

For a proper implementation of the previous multirate schemes we assume that the solvability is preserved for the active part. Furthermore it is also required that the active part of a stable DAE is also stable and has a DAE-index that is less than or equal to that of the original DAE.
Consider the linear time-invariant system operator $\Sigma : L^2(\mathbb{R}^p) \to L^2(\mathbb{R}^d)$, that satisfies $\Sigma \mathbf{x} = \mathbf{C}\dot{\mathbf{x}} + \mathbf{G}\mathbf{x}$. Linear circuit models can also be modeled in this form as

$$\Sigma \mathbf{x} = \mathbf{u}(t). \tag{5.22}$$

Here $\mathbf{u}(t) \in \mathbb{R}^p$ is a time-dependent input source function. Chapter 3 shows that the system (5.22) is solvable if and only if the spectrum $\sigma(\Sigma)$ of Σ is a finite set, and it is stable if and only if all $\lambda \in \sigma(\Sigma)$ have negative real parts. Here $\sigma(\Sigma)$ can be related to the eigenvalues of the matrix pencil $\lambda\mathbf{C} + \mathbf{G}$, as $\sigma(\Sigma) = \{\lambda \in \mathbb{C} : \det(\lambda\mathbf{C} + \mathbf{G}) = 0\}$. For a general partitioning these properties are not preserved for the active part of a DAE.

Example 5.1 Consider the linear 2-dimensional problem $\Sigma : \mathbf{C}\dot{\mathbf{x}} + \mathbf{G}\mathbf{x} = \mathbf{u}$ where

$$\mathbf{C} = \mathbf{G} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

This DAE is solvable since $\det(\lambda\mathbf{C} + \mathbf{G}) = -(\lambda + 1)^2$, which equals zero only for $\lambda = -1$, so $\sigma(\Sigma) = \{-1\}$ is a finite set. If we take the partitioning with

$$\mathbf{B}_A = \begin{pmatrix} 1 & 0 \end{pmatrix}, \quad \mathbf{B}_L = \begin{pmatrix} 0 & 1 \end{pmatrix},$$

we get for the active part the unsolvable problem

$$0\dot{\mathbf{y}} + 0\mathbf{y} = u_1.$$

Note that the active part of an explicit ODE is always solvable, because then $\mathbf{C} = \mathbf{I}$ is an invertible matrix. However, the stability is not automatically preserved for both the ODE and the DAE.

Example 5.2 If we take

$$\mathbf{C} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \mathbf{G} = \begin{pmatrix} -1 & -2 \\ 2 & 2 \end{pmatrix},$$

we have a stable ODE with eigenvalues $-\frac{1}{2} \pm \frac{1}{2}\sqrt{7}i$, but for the active part we get the unstable differential equation

$$\dot{\mathbf{y}} = \mathbf{y} + u_1.$$

This can happen because $\mathbf{G}$ is not a positive definite matrix. Besides the solvability and stability also the DAE-index of the active part is not always preserved. This will become clear from the following examples.

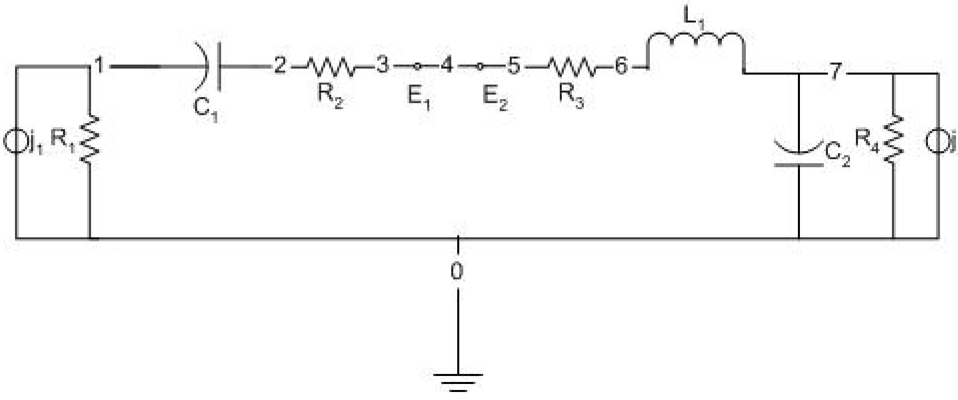


Figure 5.3: Circuit diagram of linear test example for which the nodes 3,4,5 are directly connected by shorts.

Example 5.3 First we look at a circuit model depicted in Fig. 5.3 of which the nodal voltages V_k and currents i_1^E, i_2^E, i^L satisfy the following system of DAEs

$$\begin{array}{rrcl}
\text{(KCL for)}V_1: & \frac{V_1}{R_1} + C_1\frac{d}{dt}(V_1 - V_2) + j_1 & = & 0,\\
V_2: & C_1\frac{d}{dt}(V_2 - V_1) + \frac{1}{R_2}(V_2 - V_3) & = & 0,\\
V_3: & \frac{1}{R_2}(V_3 - V_2) - i_1^E & = & 0,\\
V_4: & i_1^E - i_2^E & = & 0,\\
V_5: & i_2^E + \frac{1}{R_3}(V_5 - V_6) & = & 0,\\
V_6: & \frac{1}{R_3}(V_6 - V_5) - i^L & = & 0,\\
V_7: & i^L + C_2\frac{dV_7}{dt} + \frac{V_7}{R_4} + j_2 & = & 0,\\
\text{(KVL for)}i_1^E: & V_4 - V_3 & = & 0,\\
i_2^E: & V_5 - V_4 & = & 0,\\
i^L: & L\frac{di^L}{dt} - (V_7 - V_6) & = & 0.
\end{array} \tag{5.23}$$

Here we use the current sources $j_1 = \sin(\omega_s t)$ A, $j_2 = \sin(\omega_f t)$ A with $\omega_s = 2000\frac{\pi}{8}$, $\omega_f = 10\omega_s$ and the parameter values $C_1 = C_2 = \frac{1}{\omega_s^2 L}$ F, $R_1 = 10\ \Omega$, $R_2 = R_3 = R_4 = 1\ \Omega$ and $L = 0.1$ H. The left part of the model appears to be slowly varying because of the low frequency of j_1 and the choice of the electrical parameters. Note that the circuit model includes additional shorts that do not affect the solution but make the currents between both parts explicitly available. We consider the partitionings: $(\mathbf{x}_L, \mathbf{x}_A) = \big((V_1, V_2, V_3, i_1^E), (V_4, \ldots, V_7, i_2^E, i^L)\big)$ and $(\mathbf{x}_L, \mathbf{x}_A) = \big((V_1, V_2, V_3), (V_4, \ldots, V_7, i_1^E, i_2^E, i^L)\big)$. Both partitionings correspond to current interpolation and voltage interpolation, respectively. Figures 5.4 and 5.5 show the numerical solution of the active part at the time interval $[0, 10^{-3}]$s for both types of interpolation. It turns out that current interpolation leads to non-smooth behaviour, but the results of voltage interpolation are very good.

Example 5.4 We also look at the following modified circuit model, corresponding to

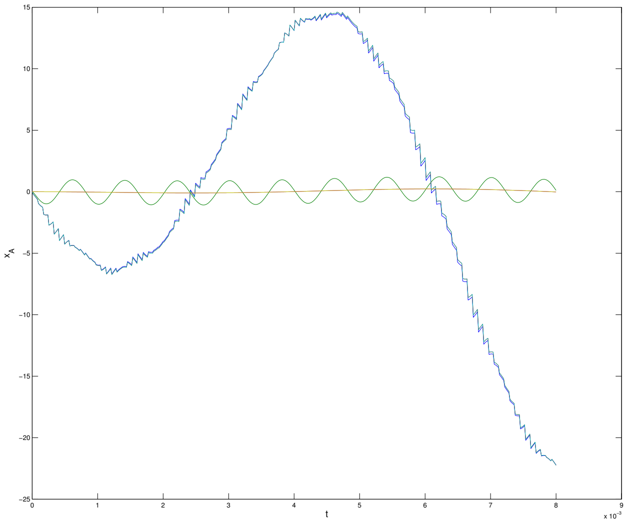


Figure 5.4: Numerical solution of the active part of (5.23) for current interpolation.

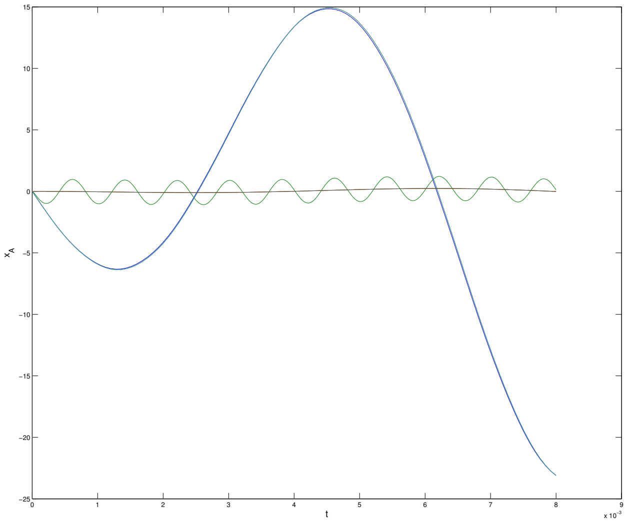


Figure 5.5: Numerical solution of the active part of (5.23) for voltage interpolation.

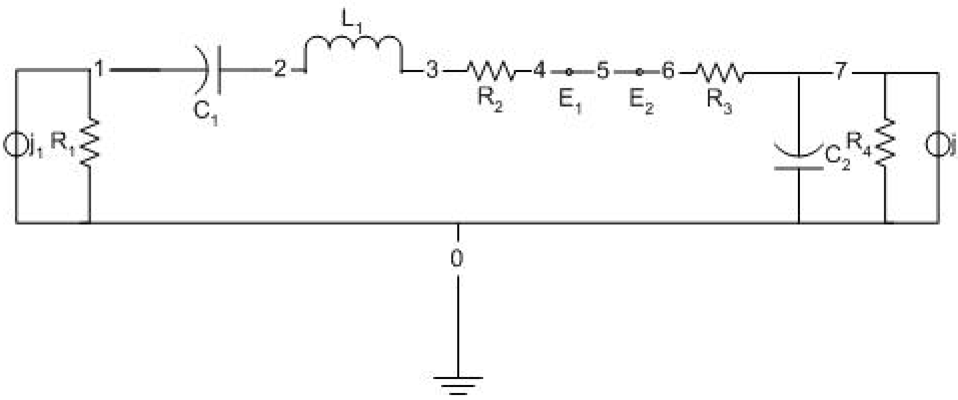


Figure 5.6: Circuit diagram of linear test example, for which the nodes 4,5,6 are directly connected by shorts.

the circuit diagram in Fig. 5.6 and having the same parameters and sources as the first example.

$$\begin{array}{rrcl}
(\text{KCL for })V_1: & \frac{V_1}{R_1}+C_1\frac{d}{dt}(V_1-V_2)+j_1 & = & 0,\\
V_2: & C_1\frac{d}{dt}(V_2-V_1)-i^L & = & 0,\\
V_3: & i^L+\frac{1}{R_2}(V_3-V_4) & = & 0,\\
V_4: & \frac{1}{R_2}(V_4-V_3)-i^E_1 & = & 0,\\
V_5: & i^E_1-i^E_2 & = & 0,\\
V_6: & i^E_2+\frac{1}{R_3}(V_6-V_7) & = & 0,\\
V_7: & \frac{1}{R_3}(V_7-V_6)+C_2\frac{dV_7}{dt}+\frac{V_7}{R_4}+j_2 & = & 0,\\
(\text{KVL for })i^E_1: & V_5-V_4 & = & 0,\\
i^E_2: & V_6-V_5 & = & 0,\\
i^L: & L\frac{di^L}{dt}-(V_3-V_2) & = & 0.
\end{array} \tag{5.24}$$

Again we consider current interpolation and voltage interpolation: $(\mathbf{x}_L,\mathbf{x}_A)=\big((V_1,\ldots,V_4,i^L,i^E_1),(V_5,V_6,V_7,i^E_2)\big)$ and $(\mathbf{x}_L,\mathbf{x}_A)=\big((V_1,\ldots,V_4,i^L),(V_5,V_6,V_7,i^E_1,i^E_2)\big)$. This time it turns out from the numerical results that the accuracy is good for current interpolation, while for voltage interpolation the computed active currents have a large error.

Thus we get a low accuracy for current interpolation applied to the first example (5.23) and for voltage interpolation applied to the second example (5.24). The reason for this behaviour is that for the first model the active circuit contains the inductor. This causes that the differential index of the DAE for the remaining active part is larger than one. In this case current interpolation gives problems such that the active voltages are not approximated correctly. However, with voltage interpolation everything is computed accurately. For the second model the active circuit does not contain the inductor. This ensures here that the index is one. Thus current interpolation gives no problems so that the currents and voltages are correctly approximated. However, with voltage interpolation the active currents are not well computed because they depend now on the

derivative of the interpolated slow voltages.
From these experiments it follows that the differential index of the active part of the DAE is very important for semi-implicit multirate. If the index is larger than one, the active solution will depend on the higher order derivatives of the input. This implies that we get problems with linear interpolation because then the active solution will become discontinuous. For circuit models it is well-known that they are composed of subcircuits in a hierarchical way. If these subcircuits have index one we can exploit this property to get a good partitioning, i.e. make partitionings with interfaces along boundaries of subcircuits [31]. Otherwise, it is required to check whether the DAE-index of the fast and the slow parts of the model are smaller or equal than one. For a Compound-Fast method it is sufficient to check only the DAE-index of the fast part. In [25, 57] it has been shown how the DAE-index of an electrical circuit model can be checked in an automatical way. For example, a circuit has index one if and only if it contains no inductor/current-source cut-sets nor controlled capacitor/voltage-source loops with at least one voltage-source.

5.5 Stability analysis of multirate methods

Multirate methods have worse stability properties than ordinary integration methods. Therefore this section is devoted to analyse the stability of those multirate methods in particular for the Slow-Fast and for the Compound-Fast methods.
Consider the partitioned nonlinear DAE in (5.7),(5.8) with property $\mathbf{j}_A(\mathbf{0},\mathbf{0}) = \mathbf{j}_L(\mathbf{0},\mathbf{0}) = \mathbf{0}$. Again we denote the fast variable $\mathbf{y} = \mathbf{x}_A$ and the slow variable $\mathbf{z} = \mathbf{x}_L$. We assume that the origin is a stable stationary solution, which implies that for all initial conditions $\mathbf{y}(t) \to \mathbf{0}$ and $\mathbf{z}(t) \to \mathbf{0}$ if $t \to \infty$. The stability of multirate schemes will only be analyzed for DAEs with these properties. Furthermore the analysis is done for equidistant time-grids.

Definition 5.5 *Let $\boldsymbol{y}_n$ and z_n be the numerical approximations of the multirate scheme at the time-point $T_n = nH$ on the coarse equidistant time-grid for a solvable DAE. The scheme is called (conditionally) stable if for all initial conditions $\boldsymbol{y}_n \to \boldsymbol{0}$ and $z_n \to \boldsymbol{0}$ if $n \to \infty$. The multirate scheme is* A-stable *(or* unconditionally stable*) if it is stable for all solvable DAEs with $\boldsymbol{y}(t), \boldsymbol{z}(t) \to \boldsymbol{0}$ for $t \to \infty$ and for all $H, q > 0$.*

This criterion would require a stability analysis of a nonlinear multi-dimensional recurrence relation, which is very complex. In [45] it has been shown that for stability of all semi-implicit and implicit Euler Backward multirate methods it is necessary that the system (5.7),(5.8) is *monotonically stable* in the max-norm. Another possible approach is to consider the *Prothero-Robinson equation* [6, 40] as additional test equation, which is used to analyze the stability on a given trajectory $\mathbf{y} = \tilde{\mathbf{y}}(t), \mathbf{z} = \tilde{\mathbf{z}}(t)$

$$\begin{aligned} \tfrac{d}{dt}\left[\mathbf{q}_A(\mathbf{y}-\tilde{\mathbf{y}},\mathbf{z}-\tilde{\mathbf{z}})\right] + \mathbf{j}_A(\mathbf{y}-\tilde{\mathbf{y}},\mathbf{z}-\tilde{\mathbf{z}}) = \mathbf{j}_A(\mathbf{0},\mathbf{0}) = \mathbf{0}, \\ \tfrac{d}{dt}\left[\mathbf{q}_L(\mathbf{y}-\tilde{\mathbf{y}},\mathbf{z}-\tilde{\mathbf{z}})\right] + \mathbf{j}_L(\mathbf{y}-\tilde{\mathbf{y}},\mathbf{z}-\tilde{\mathbf{z}}) = \mathbf{j}_L(\mathbf{0},\mathbf{0}) = \mathbf{0}. \end{aligned} \tag{5.25}$$

For the rest it is only possible to prove local stability for the linearised system around the origin. Then we get the following multi-dimensional linear time-invariant DAE (5.26) or ODE (5.27)

$$\underbrace{\begin{pmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} \\ \mathbf{C}_{21} & \mathbf{C}_{22} \end{pmatrix}}_{\mathbf{C}} \begin{pmatrix} \dot{\mathbf{y}} \\ \dot{\mathbf{z}} \end{pmatrix} + \underbrace{\begin{pmatrix} \mathbf{G}_{11} & \mathbf{G}_{12} \\ \mathbf{G}_{21} & \mathbf{G}_{22} \end{pmatrix}}_{\mathbf{G}} \begin{pmatrix} \mathbf{y} \\ \mathbf{z} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \tag{5.26}$$

$$\begin{pmatrix} \dot{\mathbf{y}} \\ \dot{\mathbf{z}} \end{pmatrix} = \underbrace{\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix}}_{\mathbf{A}} \begin{pmatrix} \mathbf{y} \\ \mathbf{z} \end{pmatrix}. \tag{5.27}$$

In [50] it is shown that Euler Backward multirate methods are stable for (5.27) if the matrix

$$\mathcal{A} = \begin{pmatrix} \mu(\mathbf{A}_{11}) & \|\mathbf{A}_{12}\| \\ \|\mathbf{A}_{21}\| & \mu(\mathbf{A}_{22}) \end{pmatrix} \tag{5.28}$$

is stable, where μ is a logarithmic norm, that is

$$\mu(\mathbf{A}) = \lim_{h\downarrow 0^+} \frac{\|\mathbf{I} + h\mathbf{A}\| - 1}{h} = \lim_{h\downarrow 0^+} \frac{\log(\|e^{h\mathbf{A}}\|)}{h}.$$

The matrix $\mathcal{A}$ in (5.28) is stable when $\sigma(\mathcal{A}) \subset \{z \in \mathbb{C} : |z| < 1\}$, which is equivalent to the requirements $\mu(\mathbf{A}_{11}) < 0, \mu(\mathbf{A}_{22}) < 0$ and $\|\mathbf{A}_{12}\|\|\mathbf{A}_{21}\| < \mu(\mathbf{A}_{11})\mu(\mathbf{A}_{22})$. In qualitative terms this means that each subsystem is stable and the couplings between the subsystems are weak.

For ordinary integration methods stability can be studied by looking at the scalar *test equation* $\dot{x} = \lambda x$ with $\lambda \in \mathbb{C}$ [32]. For multirate methods for DAEs with two time-steps h and H, the following *two-dimensional test equation* could be studied, where y and z are the active and latent variable respectively

$$\underbrace{\begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}}_{\mathbf{C}} \begin{pmatrix} \dot{y} \\ \dot{z} \end{pmatrix} + \underbrace{\begin{pmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{pmatrix}}_{\mathbf{G}} \begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{5.29}$$

For ordinary differential equations the following (real) linear test equation is usually studied [28, 50]

$$\begin{pmatrix} \dot{y} \\ \dot{z} \end{pmatrix} = \underbrace{\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}}_{\mathbf{A}} \begin{pmatrix} y \\ z \end{pmatrix} \tag{5.30}$$

We will concentrate on the stability of multirate schemes for (5.30). Let y_n and z_n be the numerical approximations at the time-point $T_n = nH$ on the coarse time-grid. For Euler Backward multirate schemes the numerical solutions y_n and z_n satisfy the following two-dimensional recurrence relation

$$\begin{pmatrix} z_n \\ y_n \end{pmatrix} = \underbrace{\begin{pmatrix} \sigma & \rho \\ \tau & \nu \end{pmatrix}}_{\mathbf{M}} \begin{pmatrix} z_{n-1} \\ y_{n-1} \end{pmatrix}. \tag{5.31}$$

Note that y_n and z_n in (5.31) are swapped compared to y, z in (5.30). The multirate method is stable if y_n and z_n tend to zero for $n \to \infty$, which is the case if $\rho(\mathbf{M}) < 1$. For $q = 1$, the stability behaviour of the multirate methods is independent of the used coordinate system or partitioning. However, for $q > 1$ this is only the case if the linear system is decoupled. Otherwise the stability does not only depend on the eigenvalues but also on the eigenvectors of the matrix $\mathbf{A}$.

The dynamics of higher order multistep methods need an adaption of (5.31). Assume that the compound step uses a BDF method of order K, while the refinement is done with a BDF method of order k. We introduce the following vectors

$$\mathbf{z}_n := \begin{pmatrix} z_n \\ \vdots \\ z_{n-K+1} \end{pmatrix} \in \mathbb{R}^K, \quad \mathbf{y}_n := \begin{pmatrix} y_n \\ \vdots \\ y_{n-k+1} \end{pmatrix} \in \mathbb{R}^k. \tag{5.32}$$

Then the dynamics of a multirate linear multistep method obey the following multi-dimensional *recurrence relation*

$$\begin{pmatrix} \mathbf{z}_n \\ \mathbf{y}_n \end{pmatrix} = \underbrace{\begin{pmatrix} \mathbf{S} & \mathbf{R} \\ \mathbf{T} & \mathbf{N} \end{pmatrix}}_{\mathbf{M}} \begin{pmatrix} \mathbf{z}_{n-1} \\ \mathbf{y}_{n-1} \end{pmatrix}, \tag{5.33}$$

where $\mathbf{M} \in \mathbb{R}^{(K+k)\times(K+k)}$ is a *companion matrix*. The multistep multirate method is stable if $\mathbf{y}_n$ and $\mathbf{z}_n$ tend to zero for $n \to \infty$. Again this is the case if $\rho(\mathbf{M}) < 1$.

Thus in both cases the stability of multirate schemes can be determined from $\rho(\mathbf{M})$ where $\mathbf{M} \in \mathbb{R}^{(K+k)\times(K+k)}$. The schemes applied to (5.30) are A-stable if $\rho(\mathbf{M}) < 1$ for all $H, q > 0$ and stable matrices $\mathbf{A}$ [50]. Because of simplicity, we start with the stability of the first order Euler Backward multirate method. Finally we also consider BDF multirate methods of higher order.

5.6 Stability analysis of Euler Backward multirate algorithm

We will now consider more particularly the stability of the Slow-Fast and Compound-Fast versions of the Euler Backward multirate algorithm. First we will show that the dynamics can really be described by the recurrence relation in (5.31). Both zeroth and first order interpolation of the latent part will be included. We will formulate a theorem that gives us stability conditions for the matrix $\mathbf{A}$. Since these conditions are rather complex to interpret, we formulate two other theorems which are based on an asymptotic analysis for $H \to 0$ or $q \to \infty$.

Lemma 5.6 *Consider the Slow-Fast and the Compound-Fast versions of the Euler Backward multirate scheme. Then $\{z_n\}$ and $\{y_n\}$ are solutions of the following recurrence relation*

$$\begin{pmatrix} z_n \\ y_n \end{pmatrix} = \underbrace{\begin{pmatrix} \sigma & \rho \\ \tau & \nu \end{pmatrix}}_{M} \begin{pmatrix} z_{n-1} \\ y_{n-1} \end{pmatrix}, \tag{5.34}$$

where, for the Slow-Fast version,

$$\rho = \frac{a_{21}H}{1-a_{22}H}, \quad \sigma = \frac{1}{1-a_{22}H} \tag{5.35}$$

and, for the Compound-Fast version,

$$\rho = \frac{a_{21}H}{1-(a_{11}+a_{22})H+(a_{11}a_{22}-a_{12}a_{21})H^2}, \quad \sigma = \frac{1-a_{11}H}{1-(a_{11}+a_{22})H+(a_{11}a_{22}-a_{12}a_{21})H^2}. \tag{5.36}$$

For both versions, if zeroth order interpolation is used we have

$$\nu = \gamma^q + \textstyle\sum_{l=0}^{q-1} \gamma^l \delta\rho, \quad \tau = \textstyle\sum_{l=0}^{q-1} \gamma^l \delta\sigma \tag{5.37}$$

and for first order interpolation we have

$$\nu = \gamma^q + \textstyle\sum_{l=0}^{q-1} \gamma^l \rho\delta(1-\frac{l}{q}), \quad \tau = \textstyle\sum_{l=0}^{q-1} \gamma^l \delta(\frac{l}{q}(1-\sigma)+\sigma), \tag{5.38}$$

where

$$\gamma = \frac{1}{1-a_{11}h}, \quad \delta = \frac{a_{12}h}{1-a_{11}h}. \tag{5.39}$$

Proof: In both the Slow-Fast and the Compound-Fast methods the latent variable is integrated first. Using zeroth order extrapolation of y_{n-1} for the Slow-Fast method we obtain the relation

$$\frac{z_n - z_{n-1}}{H} = a_{21}y_{n-1} + a_{22}z_n. \tag{5.40}$$

From (5.40) it indeed follows that

$$z_n = \rho y_{n-1} + \sigma z_{n-1}, \tag{5.41}$$

where ρ, σ are given in (5.35). For the Compound-Fast method, we get a recurrence relation for $\{y_n\}$ and $\{z_n\}$:

$$\begin{cases} \frac{y_n - y_{n-1}}{H} = a_{11}y_n + a_{12}z_n, \\ \frac{z_n - z_{n-1}}{H} = a_{21}y_n + a_{22}z_n. \end{cases} \tag{5.42}$$

The solution z_n satisfies again (5.41) with different values for ρ and σ in (5.36).
For both methods $z_{n-1,j}$ is estimated for $j \in \{1, \ldots, q-1\}$ employing z_{n-1} and z_n as follows:

Zeroth order interpolation: $\hat{z}_{n-1,j} = z_n$,
First order interpolation: $\hat{z}_{n-1,j} = z_{n-1} + \frac{j}{q}(z_n - z_{n-1}) = (1-\frac{j}{q})z_{n-1} + \frac{j}{q}z_n$.

Finally, the active part is integrated on the time interval $[T_{n-1}, T_n]$ with q steps h:

$$\frac{y_{n-1,j} - y_{n-1,j-1}}{h} = a_{11}y_{n-1,j} + a_{12}\hat{z}_{n-1,j}. \tag{5.43}$$

The recurrence relation (5.43) is equivalent to

$$y_{n-1,j} = \frac{\frac{1}{h}}{\frac{1}{h} - a_{11}} y_{n-1,j-1} + \frac{a_{12}}{\frac{1}{h} - a_{11}} \hat{z}_{n-1,j} = \gamma y_{n-1,j-1} + \delta \hat{z}_{n-1,j},$$

where $\gamma = \frac{1}{1-a_{11}h}$ and $\delta = \frac{a_{12}h}{1-a_{11}h}$. If zeroth order interpolation is used we have for $j \in \{1, \dots, q\}$

$$\begin{array}{rcl} y_{n-1,j} & = & \gamma y_{n-1,j-1} + \delta z_n \\ & = & \gamma^j y_{n-1,0} + \sum_{k=0}^{j-1} \gamma^{j-1-k} \delta z_n. \end{array} \tag{5.44}$$

If first order interpolation is used we have for $j \in \{1, \dots, q\}$

$$\begin{array}{rcl} y_{n-1,j} & = & \gamma y_{n-1,j-1} + \delta(1 - \frac{j}{q}) z_{n-1} + \delta \frac{j}{q} z_n \\ & = & \gamma^j y_{n-1,0} + \sum_{k=0}^{j-1} \gamma^{j-1-k} \left(\delta(1 - \frac{k+1}{q}) z_{n-1} + \delta \frac{k+1}{q} z_n \right). \end{array} \tag{5.45}$$

Inserting (5.41) into (5.44) for $j = q$ results in

$$\begin{array}{rcl} y_n = y_{n-1,q} & = & \gamma^q y_{n-1,0} + \sum_{k=0}^{q-1} \gamma^{q-1-k} \delta(\rho y_{n-1,0} + \sigma z_{n-1}) \\ & = & \nu y_{n-1,0} + \tau z_{n-1} = \nu y_{n-1} + \tau z_{n-1}, \end{array} \tag{5.46}$$

where ν, τ are given in (5.37). Similarly, inserting (5.41) into (5.45) for $j = q$ results in

$$\begin{array}{rcl} y_n = y_{n-1,q} & = & \gamma^q y_{n-1,0} + \left(\sum_{k=0}^{q-1} \gamma^{q-1-k} \delta(1 - \frac{k+1}{q}) \right) z_{n-1} + \\ & & \left(\sum_{k=0}^{q-1} \gamma^{q-1-k} \delta \frac{k+1}{q} \right) (\rho y_{n-1,0} + \sigma z_{n-1}) \\ & = & \nu y_{n-1,0} + \tau z_{n-1} = \nu y_{n-1} + \tau z_{n-1}, \end{array} \tag{5.47}$$

where ν, τ are given in (5.38). From (5.41), (5.46) and (5.47) it indeed follows that $\{y_n\}, \{z_n\}$ satisfy the recurrence relation in (5.34). ☐

Before we formulate the stability conditions for (5.34) we need the following Lemma.

Lemma 5.7 *Let* $\phi(\lambda) = \det(\boldsymbol{M} - \lambda \boldsymbol{I}) = \lambda^2 - tr(\boldsymbol{M})\lambda + \det(\boldsymbol{M})$ *be the characteristic polynomial of* $\boldsymbol{M}$*, where* $\boldsymbol{M} \in \mathbb{R}^{2\times 2}$*. One can easily show that*

$$\rho(\boldsymbol{M}) < 1 \Leftrightarrow \left\{ \begin{array}{c} \phi(-1) = 1 + tr(\boldsymbol{M}) + \det(\boldsymbol{M}) > 0, \\ \phi(0) = \det(\boldsymbol{M}) < 1, \\ \phi(1) = 1 - tr(\boldsymbol{M}) + \det(\boldsymbol{M}) > 0. \end{array} \right. \tag{5.48}$$

Proof: We are looking for conditions for the coefficients of $\phi(\lambda)$ such that

$$\phi(\lambda) = 0 \Rightarrow |\lambda| < 1. \tag{5.49}$$

In [17, 21] this Lemma is proved by use of the Routh-Hurwitz criterion. We will give here an alternative proof without this criterion.

Assume that there exists a root r with $|r| \geq 1$ such that $\phi(r) = 0$. If $r \in \mathbb{R}$ there exists another root $s \in \mathbb{R}$ such that $\phi(\lambda) = (\lambda - r)(\lambda - s)$. Because of the conditions we have the properties $(1 - r)(1 - s) > 0$, $rs < 1$ and $(1 + r)(1 + s) > 0$. It is not allowed that $r = 1$ nor $r = -1$. If $r > 1$ the first condition implies that also $s > 1$, which is impossible if $rs < 1$. If $r < -1$ the third condition implies that also $s < -1$, which is also impossible if $rs < 1$. Thus it follows that if $r, s \in \mathbb{R}$ it indeed follows that $r, s \in (-1, 1)$. If $r \notin \mathbb{R}$

we can write $\phi(\lambda) = (\lambda - r)(\lambda - \bar{r})$, where $\bar{r}$ is the complex conjugate of $r \in \mathbb{C}$. Then $\phi(0) = r\bar{r} = |r|^2 < 1$. Thus we indeed proved that the three stability conditions (5.48) indeed imply that $|r| < 1$. □

Consider the recurrence relation in (5.34) which describes the dynamical behaviour of the Slow-Fast and Compound-Fast versions of the Euler Backward multirate scheme for the stable test equation (5.30). Lemma 5.7 enables us to derive necessary and sufficient stablity conditions for these multirate methods.

Theorem 5.8 *Both the SF and CF versions of the Euler Backward multirate scheme using zeroth order interpolation are stable for all* H, q *if*

$$\begin{array}{rcl} (1+\sigma)(1+\gamma^q) + \rho\delta\sum_{l=0}^{q-1}\gamma^l & > & 0, \\ 1 - \sigma\gamma^q & > & 0, \\ (1-\sigma)(1-\gamma^q) - \rho\delta\sum_{l=0}^{q-1}\gamma^l & > & 0, \end{array} \tag{5.50}$$

and the schemes using first order interpolation are stable for all H, q *if*

$$\begin{array}{rcl} (1+\sigma)(1+\gamma^q) - \rho\delta\sum_{l=0}^{q-1}\gamma^l(\frac{2l}{q} - 1) & > & 0, \\ \frac{\rho\delta}{q}\sum_{l=0}^{q-1}\gamma^l l - \sigma\gamma^q + 1 & > & 0, \\ (1-\sigma)(1-\gamma^q) - \rho\delta\sum_{l=0}^{q-1}\gamma^l & > & 0. \end{array} \tag{5.51}$$

Proof: The methods are stable if $\rho(\mathbf{M}) < 1$ for all $H, q > 0$ and stable matrices $\mathbf{A}$. Since $\mathbf{M} \in \mathbb{R}^{2\times 2}$ Lemma 5.7 gives us

$$\rho(\mathbf{M}) < 1 \Leftrightarrow \left\{ \begin{array}{c} \phi(-1) = 1 + \mathrm{tr}(\mathbf{M}) + \det(\mathbf{M}) > 0, \\ \phi(0) = \det(\mathbf{M}) < 1, \\ \phi(1) = 1 - \mathrm{tr}(\mathbf{M}) + \det(\mathbf{M}) > 0. \end{array} \right. \tag{5.52}$$

Using the properties $\mathrm{tr}(\mathbf{M}) = \sigma + \nu$ and $\det(\mathbf{M}) = \sigma\nu - \rho\tau$, we get the following stability conditions for the elements of $\mathbf{M}$

$$\begin{array}{rcl} 1 + \sigma + \nu + \sigma\nu - \rho\tau & > & 0, \\ 1 - \sigma\nu + \rho\tau & > & 0, \\ 1 - \sigma - \nu + \sigma\nu - \rho\tau & > & 0. \end{array} \tag{5.53}$$

After substituting the expressions for ν and τ in (5.37) and (5.38) we obtain the three stability conditions in (5.50) and (5.51) respectively. □

Note that we can write the stability conditions in (5.53) in the following form

$$\left\{ \begin{array}{rcl} (1+\sigma)(1+\nu) & > & \rho\tau, \\ \sigma\nu - 1 & < & \rho\tau, \\ (1-\sigma)(1-\nu) & > & \rho\tau. \end{array} \right.$$

Since the stability conditions (5.50) and (5.51) are rather complex, we will derive more compact stability conditions by means of an asymptotic analysis. First we will prove that the multirate schemes studied are always conditionally stable. Second we also will give sufficient conditions for $q \to \infty$ such that the methods are stable for all H.

Theorem 5.9 *Both the Slow-Fast and Compound-Fast versions of the Euler Backward multirate schemes using zeroth or first order interpolation applied to the stable test equation (5.30) are always conditionally stable.*

Proof: The multirate methods are conditionally stable if the stability conditions in (5.50) or (5.51) only become valid for $H \to 0$. Therefore we will derive asymptotic approximations of these conditions. It easily follows that $\sigma = 1 + a_{22}H + O(H^2)$, $\gamma = 1 + \frac{a_{11}}{q}H + O(H^2)$, $\gamma^q = 1 + a_{11}H + O(H^2)$ and $\rho\delta = \frac{a_{12}a_{21}}{q}H^2 + O(H^3)$. Using these approximations, we can derive that $(1+\sigma)(1+\gamma^q) = 4 + O(H)$, $1 - \sigma\gamma^q = -(a_{11} + a_{22})H + O(H^2)$, $(1-\sigma)(1-\gamma^q) = a_{11}a_{22}H^2 + O(H^3)$, $\rho\delta\sum_{l=0}^{q-1}\gamma^l(\frac{2l}{q} - 1) = O(H^2)$ and $\rho\delta\sum_{l=0}^{q-1}\gamma^l = a_{12}a_{21}H^2 + O(H^3)$. In this way we obtain from (5.50)

$$\begin{array}{lcl} (1+\sigma)(1+\gamma^q) + \rho\delta\sum_{l=0}^{q-1}\gamma^l & = & 4 + O(H), \\ 1 - \sigma\gamma^q & = & -(a_{11} + a_{22})H + O(H^2), \\ (1-\sigma)(1-\gamma^q) - \rho\delta\sum_{l=0}^{q-1}\gamma^l & = & (a_{11}a_{22} - a_{12}a_{21})H^2 + O(H^3). \end{array} \tag{5.54}$$

and from (5.51)

$$\begin{array}{lcl} (1+\sigma)(1+\gamma^q) - \rho\delta\sum_{l=0}^{q-1}\gamma^l(\frac{2l}{q} - 1) & = & 4 + O(H), \\ \frac{\rho\delta}{q}\sum_{l=0}^{q-1}\gamma^l l - \sigma\gamma^q + 1 & = & -(a_{11} + a_{22})H + O(H^2), \\ (1-\sigma)(1-\gamma^q) - \rho\delta\sum_{l=0}^{q-1}\gamma^l & = & (a_{11}a_{22} - a_{12}a_{21})H^2 + O(H^3). \end{array} \tag{5.55}$$

After inserting these asymptotic expressions into (5.50) and (5.51), we obtain the following asymptotic stability conditions for (5.34), which coincide with the ones for (5.30)

$$\begin{array}{rcccl} \mathrm{tr}(\mathbf{A}) & = & a_{11} + a_{22} & < & 0, \\ \det(\mathbf{A}) & = & a_{11}a_{22} - a_{12}a_{21} & > & 0. \end{array} \tag{5.56}$$

Thus indeed both the Slow-Fast and Compound-Fast multirate methods using either zeroth or first order interpolation are stable for $H \to 0$ (conditionally stable) when $\mathbf{A}$ is a stable matrix. □

Now we will prove a theorem which gives sufficient stability conditions such that both methods are conditionally stable for $q \to \infty$. In the proof we need the following Lemma, which is given below without proof.

Lemma 5.10 *Consider the following rational function* $P : \mathbb{R}^+ \to \mathbb{R}$ *with*

$$\forall_{H>0} P(H) = \frac{A - BH}{A - CH - DH^2}$$

and $A, B, C, D \in \mathbb{R}$. *If* $A > 0, C < 0, D < 0, |B| < |C|$, *this rational function* P *satisfies*

$$\forall_{H>0} |P(H)| < 1.$$

In Theorem 5.11 we will show that if the subsystems are sufficiently decoupled and the multirate factor $q \to \infty$, both the active and the slow parts of the system are stable and solvable for the Slow-Fast version, while only the active part of the system is stable and solvable for the Compound-Fast version. The first condition is very natural, because strongly coupled subsystems will show the same temporally dynamical activity, which makes multirate not possible. In subsection 5.4 we showed that the active and latent parts are not always stable and solvable for a general partitioning.

Theorem 5.11 *Consider the Euler Backward Slow-Fast and Compound-Fast multirate schemes using zeroth or first order interpolation applied to the stability test equation (5.30). If*

$$\begin{cases} a_{11} < 0, \\ a_{22} < 0, \\ |a_{12}a_{21}| < |a_{11}a_{22}|, \end{cases} \tag{5.57}$$

the Slow-Fast version is unconditionally stable for $q \to \infty$. *If*

$$\begin{cases} a_{11} < 0, \\ -a_{11}a_{22} - 2a_{11}^2 < a_{12}a_{21} < a_{11}a_{22}, \end{cases} \tag{5.58}$$

the Compound-Fast version is unconditionally stable for $q \to \infty$.

Note that the Compound-Fast method is more stable than the Slow-Fast method, because it does not need that $a_{22} < 0$ and $-a_{11}a_{22} - 2a_{11}^2 < a_{12}a_{21} < a_{11}a_{22}$ is a weaker condition than $|a_{12}a_{21}| < |a_{11}a_{22}|$. **Proof:** First we prove that $a_{11} < 0$ is necessary for both methods. If the multirate factor $q \to \infty$, it is necessary that $|\gamma| < 1$ in order to have $\gamma^q \to 0$. This means that the Euler Backward method is stable for the active part, which is the case if $a_{11} < 0$.
Taking the limit $q \to \infty$, it can be shown that $\rho\delta\sum_{l=0}^{q-1}\gamma^l \to \rho\delta\sum_{l=0}^{\infty}\gamma^l = \frac{\rho\delta}{1-\gamma}$ and $\rho\delta\sum_{l=0}^{q-1}\gamma^l\frac{l}{q} \to 0$. Thus it follows that the stability conditions in (5.50) have the same asymptotic behaviour for $q \to \infty$

$$\begin{array}{ccl} (1+\sigma)(1+\gamma^q) + \rho\delta\sum_{l=0}^{q-1}\gamma^l & \to & 1+\sigma+\rho\delta\frac{1}{1-\gamma}, \\ 1-\sigma\gamma^q & \to & 1, \\ (1-\sigma)(1-\gamma^q) - \rho\delta\sum_{l=0}^{q-1}\gamma^l & \to & 1-\sigma-\rho\delta\frac{1}{1-\gamma}. \end{array} \tag{5.59}$$

Similarly for (5.51) when $q \to \infty$

$$\begin{array}{ccl} (1+\sigma)(1+\gamma^q) - \rho\delta\sum_{l=0}^{q-1}\gamma^l(\frac{2l}{q}-1) & \to & 1+\sigma+\rho\delta\frac{1}{1-\gamma}, \\ \frac{\rho\delta}{q}\sum_{l=0}^{q-1}\gamma^l l - \sigma\gamma^q + 1 & \to & 1, \\ (1-\sigma)(1-\gamma^q) - \rho\delta\sum_{l=0}^{q-1}\gamma^l & \to & 1-\sigma-\rho\delta\frac{1}{1-\gamma}. \end{array} \tag{5.60}$$

This means that for $q \to \infty$ we have the following unconditional stability conditions

$$\begin{cases} 1+\sigma+\rho\delta\frac{1}{1-\gamma} > 0, \\ 1-\sigma-\rho\delta\frac{1}{1-\gamma} > 0. \end{cases} \Leftrightarrow \quad |\frac{\rho\delta}{1-\gamma}+\sigma| < 1. \tag{5.61}$$

Because of the definition of γ, δ in (5.39) it follows that $\frac{\delta}{1-\gamma} = -\frac{a_{12}}{a_{11}}$, yielding

$$|-\frac{a_{12}}{a_{11}}\rho+\sigma| < 1. \tag{5.62}$$

Using (5.35) for the SF method, condition (5.62) is equivalent to

$$\forall_{H>0} |P_{SF}(H)| = |-\frac{a_{12}}{a_{11}}\rho + \sigma| = \frac{|1 - \frac{a_{12}a_{21}}{a_{11}}H|}{|1 - a_{22}H|} < 1.$$

Using Lemma 5.10 this gives the stability conditions

$$\begin{cases} a_{22} < 0, \\ |\frac{a_{12}a_{21}}{a_{11}}| < |a_{22}|. \end{cases}$$

The second condition is equivalent to $|a_{12}a_{21}| < |a_{11}a_{22}|$ indeed. Thus we have shown that the Euler Backward Slow-Fast multirate method using zeroth or first order interpolation is indeed unconditionally stable for $q \to \infty$ if the conditions (5.57) hold.
Using (5.36) for the CF method, condition (5.62) is equivalent to

$$\forall_{H>0} |P_{CF}(H)| = |-\frac{a_{12}}{a_{11}}\rho + \sigma| = \frac{|1 - (\frac{a_{12}a_{21}}{a_{11}} + a_{11})H|}{|1 - (a_{11} + a_{22})H + (a_{11}a_{22} - a_{12}a_{21})H^2|} < 1.$$

Again Lemma 5.10 gives us the following sufficient stability conditions

$$\begin{cases} a_{11} + a_{22} < 0, \\ a_{11}a_{22} - a_{12}a_{21} > 0, \\ |\frac{a_{12}a_{21}}{a_{11}} + a_{11}| < |a_{11} + a_{22}|. \end{cases}$$

The first two conditions are automatically satisfied for a stable test equation. Because $a_{11} + a_{22} < 0$, the third condition is equivalent to

$$a_{11} + a_{22} < \frac{a_{12}a_{21}}{a_{11}} + a_{11} < -a_{11} - a_{22}. \tag{5.63}$$

From the left inequality in (5.63) we can derive

$$\frac{1}{a_{11}}(a_{11}a_{22} - a_{12}a_{21}) < 0. \tag{5.64}$$

The other inequality in (5.63) gives

$$a_{12}a_{21} > -a_{11}a_{22} - 2a_{11}^2. \tag{5.65}$$

Using $a_{11} < 0$ and combining the inequalities (5.64) and (5.65) gives us

$$-a_{11}a_{22} - 2a_{11}^2 < a_{12}a_{21} < a_{11}a_{22}. \tag{5.66}$$

Thus we have shown that the Euler Backward Compound-Fast multirate method using zeroth or first order interpolation is indeed unconditionally stable for $q \to \infty$ if the conditions (5.58) hold. □

We have derived simplified sufficient stability conditions for the matrix **A** of the test equation (5.30) such that both Euler Backward multirate schemes are stable. For the asymptotic analysis for $H \to 0$ or $q \to \infty$ it does not matter whether zeroth or first order interpolation is used.

5.7 Stability analysis of multistep BDF multirate algorithms

Since BDF methods of higher order are multistep methods, the previous analysis needs to be adapted to apply in this case. This section therefore is devoted in particular to the stability of the BDF multistep scheme for the Slow-Fast and Compound-Fast multirate versions. First we will show that the dynamics really can be described by the recurrence relation in (5.33). We consider only one type of interpolation of the latent part and do the same stability analysis as in the previous section.
First we have the following definitions.

Definition 5.12 *Define the function* $\boldsymbol{b} : \mathbb{N} \rightarrow \mathbb{R}^K$

$$\boldsymbol{b}_j = \boldsymbol{V}^{-T}\boldsymbol{e}(K, \frac{j}{q} - 1), \tag{5.67}$$

where $\boldsymbol{e} : \mathbb{N} \times \mathbb{R} \rightarrow \mathbb{R}^s$ *and* $\boldsymbol{V} \in \mathbb{R}^{K \times K}$ *are given by*

$$\boldsymbol{e}(s, \omega) := \left[1, \omega, \ldots, \omega^{s-1}\right]^T \tag{5.68}$$

and

$$v_{ij} = \begin{cases} 1 & i = j = 1, \\ (1-i)^{j-1} & \textit{otherwise.} \end{cases} \tag{5.69}$$

Consider the solutions z_n, y_n of the Slow-Fast and the Compound-Fast versions of the BDF multirate scheme, both with integration orders (K, k). Let $\mathbf{z}_n, \mathbf{y}_n$ be defined as

$$\mathbf{z}_n := \begin{pmatrix} z_n \\ \vdots \\ z_{n-K+1} \end{pmatrix} \in \mathbb{R}^K, \quad \mathbf{y}_n := \begin{pmatrix} y_n \\ \vdots \\ y_{n-k+1} \end{pmatrix} \in \mathbb{R}^k. \tag{5.70}$$

Assume that, for the Slow-Fast version, $\mathbf{R} \in \mathbb{R}^{K \times k}$, $\mathbf{S} \in \mathbb{R}^{K \times K}$ are given by

$$\begin{aligned} \mathbf{R} &:= \tilde{\rho} \begin{pmatrix} \tilde{\sigma} & 0 & \ldots & 0 \\ 0 & 0 & & \\ \vdots & & \ddots & \\ 0 & & & 0 \end{pmatrix}, \quad \mathbf{S} := \begin{pmatrix} -\tilde{\sigma}\frac{\rho_1}{\rho_0} & \ldots & & -\tilde{\sigma}\frac{\rho_K}{\rho_0} \\ 1 & & & 0 \\ & \ddots & & \vdots \\ & & 1 & 0 \end{pmatrix}, \\ &\tilde{\rho} = \tfrac{a_{21}H}{\rho_0}, \quad \tilde{\sigma} = \tfrac{\rho_0}{\rho_0 - a_{22}H} \end{aligned} \tag{5.71}$$

and, for the Compound-Fast version,

$$\begin{aligned} \mathbf{R} &:= \tilde{\rho} \begin{pmatrix} -\tilde{\sigma}\frac{\rho_1}{\rho_0} & \ldots & -\tilde{\sigma}\frac{\rho_K}{\rho_0} \\ 0 & \ldots & 0 \\ \vdots & & \vdots \\ 0 & \ldots & 0 \end{pmatrix}, \quad \mathbf{S} := \begin{pmatrix} -\tilde{\sigma}\frac{\rho_1}{\rho_0} & \ldots & & -\tilde{\sigma}\frac{\rho_K}{\rho_0} \\ 1 & & & 0 \\ & \ddots & & \vdots \\ & & 1 & 0 \end{pmatrix}, \\ \tilde{\rho} &= \tfrac{a_{21}H}{\rho_0 - a_{11}H}, \\ \tilde{\sigma} &= \tfrac{\rho_0(\rho_0 - a_{11}H)}{\rho_0^2 - \rho_0(a_{11}+a_{22})H + (a_{11}a_{22} - a_{12}a_{21})H^2} \end{aligned} \tag{5.72}$$

and that $\mathbf{N} \in \mathbb{R}^{k\times k}, \mathbf{T} \in \mathbb{R}^{k\times K}$ are given by

$$\mathbf{N} \;:=\; \mathbf{G}^q + \textstyle\sum_{l=0}^{q-1} \mathbf{G}^l \mathbf{d}\mathbf{b}_{q-l}^T \mathbf{R}, \quad \mathbf{T} \;:=\; \textstyle\sum_{l=0}^{q-1} \mathbf{G}^l \mathbf{d}\mathbf{b}_{q-l}^T \mathbf{S}, \tag{5.73}$$

where $\mathbf{G} \in \mathbb{R}^{k\times k}$ and $\mathbf{d} \in \mathbb{R}^k$ are defined by

$$\mathbf{G} \;:=\; \begin{pmatrix} -\tilde{\gamma}\frac{\bar{\rho}_1}{\bar{\rho}_0} & \cdots & & -\tilde{\gamma}\frac{\bar{\rho}_k}{\bar{\rho}_0} \\ 1 & & & 0 \\ & \ddots & & \vdots \\ & & 1 & 0 \end{pmatrix}, \quad \mathbf{d} \;:=\; \tilde{\delta}\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad \begin{aligned} \tilde{\gamma} &= \tfrac{\bar{\rho}_0}{\bar{\rho}_0 - a_{11}h}, \\ \tilde{\delta} &= \tfrac{a_{12}h}{\bar{\rho}_0 - a_{11}h}. \end{aligned} \tag{5.74}$$

Lemma 5.13 *The sequences $\{\boldsymbol{z}_n\}$ and $\{\boldsymbol{y}_n\}$ are solutions of the following recurrence relation*

$$\begin{pmatrix} z_n \\ y_n \end{pmatrix} = \underbrace{\begin{pmatrix} S & R \\ T & N \end{pmatrix}}_{M} \begin{pmatrix} z_{n-1} \\ y_{n-1} \end{pmatrix}. \tag{5.75}$$

Proof: In both the Slow-Fast and the Compound-Fast method the latent variable is integrated first. Using zeroth order extrapolation of y_{n-1} for the Slow-Fast method we obtain the system

$$\frac{\rho_0 z_n + \ldots + \rho_K z_{n-K}}{H} = a_{21} y_{n-1} + a_{22} z_n. \tag{5.76}$$

Since $K > 1$ we see that z_n also depends on previous values of $\{z_n\}$. From (5.76), it follows that $\mathbf{z}_n$ satisfies the following recurrence relation

$$\mathbf{z}_n = \mathbf{R}\mathbf{y}_{n-1} + \mathbf{S}\mathbf{z}_{n-1}, \tag{5.77}$$

where $\mathbf{R}, \mathbf{S}$ are given in (5.71). For the Compound-Fast method, we get a recurrence relation for $\{y_n\}$ and $\{z_n\}$:

$$\begin{cases} \frac{\rho_0 y_n + \ldots + \rho_K y_{n-K}}{H} = a_{11} y_n + a_{12} z_n, \\ \frac{\rho_0 z_n + \ldots + \rho_K z_{n-K}}{H} = a_{21} y_n + a_{22} z_n. \end{cases} \tag{5.78}$$

Since $\{z_n\}$ satisfies

$$\begin{aligned} z_n \;&=\; \tfrac{\rho_0(\rho_0 - Ha_{21})}{(\rho_0 - Ha_{22})(\rho_0 - Ha_{11}) - a_{12}a_{21}H^2} \left(-\tfrac{\rho_1}{\rho_0} z_{n-1} - \ldots - \tfrac{\rho_K}{\rho_0} z_{n-K}\right) \\ &+\; \tfrac{\rho_0 H a_{21}}{(\rho_0 - Ha_{22})(\rho_0 - Ha_{11}) - a_{12}a_{21}H^2} \left(-\tfrac{\rho_1}{\rho_0} y_{n-1} - \ldots - \tfrac{\rho_K}{\rho_0} y_{n-K}\right), \end{aligned}$$

the solution z_n satisfies again (5.77) with different values for $\mathbf{R}, \mathbf{S}$ in (5.72).
The active part is integrated on the time interval $[T_{n-1}, T_n]$ with q steps h:

$$\frac{\bar{\rho}_0 y_{n-1,j} + \ldots + \bar{\rho}_k y_{n-1,j-k}}{h} = a_{11} y_{n-1,j} + a_{12} \hat{z}_{n-1,j}. \tag{5.79}$$

The recurrence relation (5.79) is equivalent to

$$y_{n-1,j} = \frac{1}{\bar{\rho}_0 - a_{11}h}(-\bar{\rho}_1 y_{n-1,j-1} - \ldots - \bar{\rho}_k y_{n-1,j-k} + a_{12} h \hat{z}_{n-1,j}). \tag{5.80}$$

For both methods $z_{n-1,j}$ is estimated for $j \in \{1, \dots, q-1\}$ employing $z_{n-k}, \dots, z_n$. For the remainder of this proof we first need the following Lemma 5.14.

Lemma 5.14 *The interpolated value* $\hat{z}_{n-1,j}$ *can be retrieved from the coarse latent vector* $\mathbf{z}_n$ *by* $\hat{z}_{n-1,j} = \mathbf{b}_j^T \mathbf{z}_n$*, where* $\mathbf{b}_j$ *is given in (5.67).*

Proof: We can describe the numerical solution for z at the coarse grid by a truncated Taylor expansion around T_n

$$z^n(t) = \sum_{i=0}^{K} \bar{z}_{i+1}^n \left(\frac{t - T_n}{H}\right)^i .$$

The vector $\bar{\mathbf{z}}^n \in \mathbb{R}^K$ is the Nordsieck vector of length K:

$$\bar{\mathbf{z}}^n := \left(z^n(T_n), H\frac{d}{dt}z^n(T_n), \dots, \frac{H^{K-1}}{(K-1)!}\frac{d^{K-1}}{dt^{K-1}}z^n(T_n)\right)^T .$$

Then we have

$$\hat{z}_{n-1,j} = z^n(t_{n-1,j}) = \sum_{i=0}^{K} \bar{z}_{i+1}^n \left(\frac{t_{n-1,j} - T_n}{H}\right)^i = \mathbf{e}(K, \frac{t_{n-1,j} - T_n}{H})^T \cdot \bar{\mathbf{z}}^n.$$

In section 4.3 it is shown that the Nordsieck vector $\bar{\mathbf{z}}^n \in \mathbb{R}^K$ and the vector $\mathbf{z}_n \in \mathbb{R}^K$ are related by

$$\mathbf{V}\bar{\mathbf{z}}^n = \mathbf{z}_n, \tag{5.81}$$

where $\mathbf{V}$ is the Vandermonde matrix defined in (5.69). Thus

$$\hat{z}_{n-1,j} = \mathbf{e}(K, \frac{t_{n-1,j} - T_n}{H})^T \mathbf{V}^{-1}\mathbf{z}_n = \mathbf{e}(K, \frac{j}{q} - 1)^T \mathbf{V}^{-1}\mathbf{z}_n,$$

since $T_n = t_{n-1,q}$ and $H = qh$. Thus $\hat{z}_{n-1,j} = \mathbf{b}_j^T \mathbf{z}_n$, where $\mathbf{b}_j$ is given in (5.67). □

Now we continue with the second part of the proof of Lemma 5.13. For the refinement phase we introduce the following vector $\in \mathbb{R}^k$

$$\mathbf{y}_{n-1,j} := \begin{pmatrix} y_{n-1,j} \\ \vdots \\ y_{n-1,j-k+1} \end{pmatrix}. \tag{5.82}$$

In vector notation this can be written as

$$\mathbf{y}_{n-1,j} = \mathbf{G}\mathbf{y}_{n-1,j-1} + \mathbf{d}\hat{z}_{n-1,j}, \tag{5.83}$$

where $\mathbf{G}, \mathbf{d}$ are given in (5.74). Hence for $j \in \{1, \dots, q\}$ we have

$$\begin{aligned} \mathbf{y}_{n-1,j} &= \mathbf{G}\mathbf{y}_{n-1,j-1} + \mathbf{d}\mathbf{b}_j^T \mathbf{z}_n \\ &= \mathbf{G}^j \mathbf{y}_{n-1,0} + \textstyle\sum_{k=0}^{j-1} \mathbf{G}^{j-1-k}\mathbf{d}\mathbf{b}_{k+1}^T \mathbf{z}_n. \end{aligned} \tag{5.84}$$

Since the coarse and fine time-grids are synchronised, we can insert (5.77) into (5.84) for $j = q$, resulting in

$$\begin{aligned} \mathbf{y}_n = \mathbf{y}_{n-1,q} &= \mathbf{G}^q \mathbf{y}_{n-1,0} + \textstyle\sum_{k=0}^{q-1} \mathbf{G}^{q-1-k} \mathbf{d}\mathbf{b}_{k+1}^T (\mathbf{R}\mathbf{y}_{n-1} + \mathbf{S}\mathbf{z}_{n-1}) \\ &= \mathbf{N}\mathbf{y}_{n-1} + \mathbf{T}\mathbf{z}_{n-1}, \end{aligned} \tag{5.85}$$

where $\mathbf{N}, \mathbf{T}$ are given in (5.73). From (5.77) it indeed follows that $\{\mathbf{y}_n\}, \{\mathbf{z}_n\}$ satisfy the recurrence relation in (5.75). This completes the proof of Lemma 5.13. □

Since the matrix $\mathbf{M} \in \mathbb{R}^{(K+k)\times(K+k)}$ in (5.75) is a higher dimensional matrix if $\max\{K, k\} > 1$, the stability conditions in (5.52) do not hold. One possible approach is to derive more accurate stability conditions using the Routh-Hurwitz criterion. This becomes very tedious for higher order and therefore we analyze the following two-dimensional recurrence relation for $\{\tilde{z}_n\}$ and $\{\tilde{y}_n\}$ instead

$$\begin{pmatrix} \tilde{z}_n \\ \tilde{y}_n \end{pmatrix} = \hat{\mathbf{M}} \begin{pmatrix} \tilde{z}_{n-1} \\ \tilde{y}_{n-1} \end{pmatrix}. \tag{5.86}$$

Here $\hat{\mathbf{M}} \in \mathbb{R}^{2\times 2}$ is properly chosen such that $\rho(\mathbf{M}) < \rho(\hat{\mathbf{M}})$. We introduce the following matrices

$$\mathbf{P} = \frac{1}{\tilde{\rho}} \mathbf{S}^{-1}\mathbf{R}, \quad \mathbf{X} = \frac{1}{\tilde{\delta}} \sum_{l=0}^{q-1} \mathbf{G}^l \mathbf{d}\mathbf{b}_{q-l}^T, \quad \mathbf{Y} = \mathbf{G}^q. \tag{5.87}$$

Since $\mathbf{S}, \mathbf{Y}$ are diagonalisable, there exist $\mathbf{V}, \bar{\mathbf{V}}, \Lambda, \bar{\Lambda}$ such that $\mathbf{S} = \mathbf{V}\Lambda\mathbf{V}^{-1}, \mathbf{Y} = \bar{\mathbf{V}}\bar{\Lambda}\bar{\mathbf{V}}^{-1}$; $\Lambda, \bar{\Lambda}$ diagonal. Furthermore we define a number $L > 0$ with

$$L := \max\{\mathrm{cond}(\mathbf{V}), \mathrm{cond}(\bar{\mathbf{V}})\}. \tag{5.88}$$

Next we define the two-dimensional matrix $\hat{\mathbf{M}}$ by

$$\hat{\mathbf{M}} := \begin{bmatrix} \rho(\mathbf{S}) & L|\tilde{\rho}|\rho(\mathbf{S})\|\mathbf{P}\| \\ L|\tilde{\delta}|\rho(\mathbf{S})\|\mathbf{X}\| & \rho(\mathbf{Y}) + L^2|\tilde{\rho}\tilde{\delta}|\|\mathbf{P}\|\|\mathbf{X}\|\rho(\mathbf{S}) \end{bmatrix}. \tag{5.89}$$

We will show that this two-dimensional matrix can be used to get simpler stability conditions for (5.75).

Lemma 5.15 *Consider the matrices* $\boldsymbol{M} \in \mathbb{R}^{(K+k)\times(K+k)}$ *in (5.75) and* $\hat{\boldsymbol{M}} \in \mathbb{R}^{2\times 2}$ *in (5.89). Then for the spectral radii of* $\mathbf{M}$ *and* $\hat{\mathbf{M}}$ *we have the relation*

$$\rho(\boldsymbol{M}) \leq \rho(\hat{\boldsymbol{M}}). \tag{5.90}$$

Proof: Let $\mathbf{P}, \mathbf{X}, \mathbf{Y}$ be the matrices as defined in (5.87). The following relations between the block matrices of $\mathbf{M}$ exist

$$\mathbf{R} = \tilde{\rho}\mathbf{S}\mathbf{P}, \quad \mathbf{N} = \mathbf{Y} + \tilde{\delta}\mathbf{X}\mathbf{R}, \quad \mathbf{T} = \tilde{\delta}\mathbf{X}\mathbf{S}.$$

Thus the companion matrix $\mathbf{M}$ can be nicely factorised, using a factorisation that is popular when dealing with indefinite matrices

$$\mathbf{M} = \begin{bmatrix} \mathbf{S} & \mathbf{R} \\ \mathbf{T} & \mathbf{N} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \\ \tilde{\delta}\mathbf{X} & \mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{S} & \\ & \mathbf{Y} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{I} & \tilde{\rho}\mathbf{P} \\ & \mathbf{I} \end{bmatrix}.$$

After performing the transformation

$$\begin{aligned}\tilde{\mathbf{M}} &= \begin{bmatrix} \mathbf{V}^{-1} & \\ & \bar{\mathbf{V}}^{-1} \end{bmatrix} \mathbf{M} \begin{bmatrix} \mathbf{V} & \\ & \bar{\mathbf{V}} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{I} & \\ \tilde{\delta}\bar{\mathbf{V}}^{-1}\mathbf{X}\mathbf{V} & \mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} \Lambda & \\ & \bar{\Lambda} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{I} & \tilde{\rho}\mathbf{V}^{-1}\mathbf{P}\bar{\mathbf{V}} \\ & \mathbf{I} \end{bmatrix} \\ &= \begin{bmatrix} \Lambda & \tilde{\rho}\Lambda\mathbf{V}^{-1}\mathbf{P}\bar{\mathbf{V}} \\ \tilde{\delta}\bar{\mathbf{V}}^{-1}\mathbf{X}\mathbf{V}\Lambda & \bar{\Lambda} + \tilde{\rho}\tilde{\delta}\mathbf{V}^{-1}\bar{\mathbf{V}}^{-1}\mathbf{X}\mathbf{V}\Lambda\mathbf{P}\bar{\mathbf{V}} \end{bmatrix},\end{aligned} \tag{5.91}$$

it can be seen that $\rho(\mathbf{M}) = \rho(\tilde{\mathbf{M}})$ and $\rho(\Lambda) = \rho(\mathbf{S}), \rho(\bar{\Lambda}) = \rho(\mathbf{Y})$. We are now able to obtain a beautiful key estimate which allows to compare the norm of the n-th power of the $(K+k) \times (K+k)$-matrix $\tilde{\mathbf{M}}$ to the norm of the n-th power of the 2×2 matrix $\hat{\mathbf{M}}$. The matrix $\hat{\mathbf{M}}$ contains the essential data in condensed form.

$$\|\tilde{\mathbf{M}}^n\| \leq \|\hat{\mathbf{M}}^n\|. \tag{5.92}$$

This key estimate implies that for all n $\|\tilde{\mathbf{M}}^n\|^{\frac{1}{n}} \leq \|\hat{\mathbf{M}}^n\|^{\frac{1}{n}}$. Using the properties

$$\lim_{n\to\infty} \|\tilde{\mathbf{M}}^n\|^{\frac{1}{n}} = \rho(\tilde{\mathbf{M}}), \quad \lim_{n\to\infty} \|\hat{\mathbf{M}}^n\|^{\frac{1}{n}} = \rho(\hat{\mathbf{M}}) \tag{5.93}$$

yields $\rho(\tilde{\mathbf{M}}) \leq \rho(\hat{\mathbf{M}})$. Because $\mathbf{M}, \tilde{\mathbf{M}}$ are similar we get the required identity. □

Now we are able to prove the following theorem, which gives necessary and sufficient stability conditions for the studied Slow-Fast and Compound-Fast versions of the BDF multirate schemes applied to the stable test equation (5.30). In contrast to Theorem 5.8 the found results are also valid if the integration orders K and k for the coarse and fine time-grids are larger than one.

Theorem 5.16 *Consider the recurrence relation in (5.75) which describes the dynamical behaviour of the Slow-Fast and Compound-Fast versions of the BDF multirate schemes for the stable test equation (5.30). Then the schemes are stable for all* H, q *if*

$$\begin{aligned} (1+\rho(\boldsymbol{S}))(1+\rho(\boldsymbol{G}^q)) &> -L^2|\tilde{\rho}\tilde{\delta}|\|\boldsymbol{P}\|\|\boldsymbol{X}\|, \\ \rho(\boldsymbol{S})\rho(\boldsymbol{G}^q) &< 1 + L^2|\tilde{\rho}\tilde{\delta}|\|\boldsymbol{P}\|\|\boldsymbol{X}\|, \\ (1-\rho(\boldsymbol{S}))(1-\rho(\boldsymbol{G}^q)) &> L^2|\tilde{\rho}\tilde{\delta}|\|\boldsymbol{P}\|\|\boldsymbol{X}\|. \end{aligned} \tag{5.94}$$

Proof: The methods are stable if $\rho(\mathbf{M}) < 1$ for all $H, q > 0$ and stable matrices $\mathbf{A}$. In Lemma 5.15 it is shown that $\rho(\hat{\mathbf{M}}) < 1 \Rightarrow \rho(\mathbf{M}) < 1$, where $\hat{\mathbf{M}} \in \mathbb{R}^{2\times 2}$ is given in (5.89). Because $\hat{\mathbf{M}}$ is a real two-dimensional matrix, the stability conditions in Lemma 5.7 can be used. It simply follows that

$$\begin{aligned} \rho(\hat{\mathbf{M}}) < 1 \quad &\Leftrightarrow \quad \left\{ \begin{aligned} 1 + \operatorname{tr}(\hat{\mathbf{M}}) + \det(\hat{\mathbf{M}}) &> 0, \\ \det(\hat{\mathbf{M}}) &< 1, \\ 1 - \operatorname{tr}(\hat{\mathbf{M}}) + \det(\hat{\mathbf{M}}) &> 0. \end{aligned} \right. \\ &\Leftrightarrow \quad \left\{ \begin{aligned} 1 + \rho(\mathbf{S}) + \rho(\mathbf{Y}) + L^2|\tilde{\rho}\tilde{\delta}|\|\mathbf{P}\|\|\mathbf{X}\| + \rho(\mathbf{S})\rho(\mathbf{Y}) &> 0, \\ \rho(\mathbf{S})\rho(\mathbf{Y}) &< 1, \\ 1 - \rho(\mathbf{S}) - \rho(\mathbf{Y}) - L^2|\tilde{\rho}\tilde{\delta}|\|\mathbf{P}\|\|\mathbf{X}\| + \rho(\mathbf{S})\rho(\mathbf{Y}) &> 0. \end{aligned} \right. \\ &\Leftrightarrow \quad \left\{ \begin{aligned} (1+\rho(\mathbf{S}))(1+\rho(\mathbf{Y})) &> -L^2|\tilde{\rho}\tilde{\delta}|\|\mathbf{P}\|\|\mathbf{X}\|, \\ \rho(\mathbf{S})\rho(\mathbf{Y}) &< 1 + L^2|\tilde{\rho}\tilde{\delta}|\|\mathbf{P}\|\|\mathbf{X}\|, \\ (1-\rho(\mathbf{S}))(1-\rho(\mathbf{Y})) &> L^2|\tilde{\rho}\tilde{\delta}|\|\mathbf{P}\|\|\mathbf{X}\|. \end{aligned} \right. \end{aligned} \tag{5.95}$$

After substituting the expressions in (5.87) for $\mathbf{P}, \mathbf{X}, \mathbf{Y}$ we obtain the sufficient stability conditions for (5.94). The remainder of the theorem follows immediately. □

Using (5.87) the stability conditions in (5.94) can be expanded as

$$\begin{array}{rcl} (1+\rho(\mathbf{S}))(1+\rho(\mathbf{G}^q)) & > & -L^2|\tilde{\rho}\tilde{\delta}|\|\mathbf{S}^{-1}\mathbf{R}\|\|\sum_{l=0}^{q-1}\mathbf{G}^l\mathbf{db}_{q-l}^T\|, \\ \rho(\mathbf{S})\rho(\mathbf{G}^q) & < & 1+L^2|\tilde{\rho}\tilde{\delta}|\|\mathbf{S}^{-1}\mathbf{R}\|\|\sum_{l=0}^{q-1}\mathbf{G}^l\mathbf{db}_{q-l}^T\|, \\ (1-\rho(\mathbf{S}))(1-\rho(\mathbf{G}^q)) & > & L^2|\tilde{\rho}\tilde{\delta}|\|\mathbf{S}^{-1}\mathbf{R}\|\|\sum_{l=0}^{q-1}\mathbf{G}^l\mathbf{db}_{q-l}^T\|. \end{array} \tag{5.96}$$

Note that the stability conditions for the multistep case are very similar to the conditions for the onestep case in (5.51). The first inequality in (5.94) is always satisfied. Thus we have the following sufficient stability conditions

$$\left\{ \begin{array}{rcl} \rho(\mathbf{S})\rho(\mathbf{G}^q) & < & 1, \\ |\tilde{\rho}\tilde{\delta}|\|\mathbf{P}\|\|\mathbf{X}\| & < & \frac{1}{L^2}(1-\rho(\mathbf{S}))(1-\rho(\mathbf{G}^q)). \end{array} \right. \tag{5.97}$$

The following Lemma enables us to express the conditions for $\rho(\mathbf{S}), \rho(\mathbf{G}^q)$ in terms of $\tilde{\sigma}, \tilde{\gamma}$.

Lemma 5.17 *For both companion matrices* $\boldsymbol{S}, \boldsymbol{G}$ *of order* $p \in \{1, \ldots, 6\}$ *there exist* $\mu_p, \nu_p \in [0,1]$ *with* $\mu_p + \nu_p = 1$*, such that for* $\tilde{\sigma}, \tilde{\gamma} \in [0,1]$

$$\rho(\boldsymbol{S}) \leq \mu_p + \nu_p\tilde{\sigma}, \quad \rho(\boldsymbol{G}) \leq \mu_p + \nu_p\tilde{\gamma}. \tag{5.98}$$

Proof: Both matrices $\mathbf{G}$ (5.74) and $\mathbf{S}$ (5.71),(5.72) are companion matrices of the following structure:

$$\mathbf{S} = \begin{pmatrix} -\tilde{\sigma}\frac{\rho_1}{\rho_0} & \cdots & & -\tilde{\sigma}\frac{\rho_K}{\rho_0} \\ 1 & & & 0 \\ & \ddots & & \vdots \\ & & 1 & 0 \end{pmatrix}, \quad \mathbf{G} = \begin{pmatrix} -\tilde{\gamma}\frac{\bar{\rho}_1}{\bar{\rho}_0} & \cdots & & -\tilde{\gamma}\frac{\bar{\rho}_k}{\bar{\rho}_0} \\ 1 & & & 0 \\ & \ddots & & \vdots \\ & & 1 & 0 \end{pmatrix}.$$

Since $\rho_i = \bar{\rho}_i$ if $K = k$, we can write $\mathbf{S} = \mathbf{C}(\tilde{\sigma}, K)$ and $\mathbf{G} = \mathbf{C}(\tilde{\gamma}, k)$, where $\{\mathbf{C}(x,p) : x \in [0,1], p \in \{1, \ldots, 6\}\}$ is a general family of companion matrices. Because $\mathbf{S} = \mathbf{G}$ if $K = k$ and $\tilde{\sigma} = \tilde{\gamma}$, it is sufficient to prove

$$\rho(\mathbf{C}(x,p)) \leq \mu_p + \nu_p x.$$

Figure 5.7 shows the relationship between $\rho(\mathbf{C}(x,p))$ and $\mu_p + \nu_p x$ for $p \in \{1, \ldots, 6\}$ for the following values of μ_p, ν_p

p	1	2	3	4	5	6
ρ_0	1	3/2	11/6	25/12	137/60	49/20
μ_p	0	0.1	0.23	0.41	0.65	0.88
ν_p	1	0.9	0.77	0.59	0.35	0.12

It is clear that for all $x \in [0,1]$ $\rho(\mathbf{C}(x,p)) \leq \mu_p + \nu_p x$. Because $\mathbf{S} = \mathbf{C}(\tilde{\sigma}, K)$ and $\mathbf{G} = \mathbf{C}(\tilde{\gamma}, k)$ this also proves (5.98).

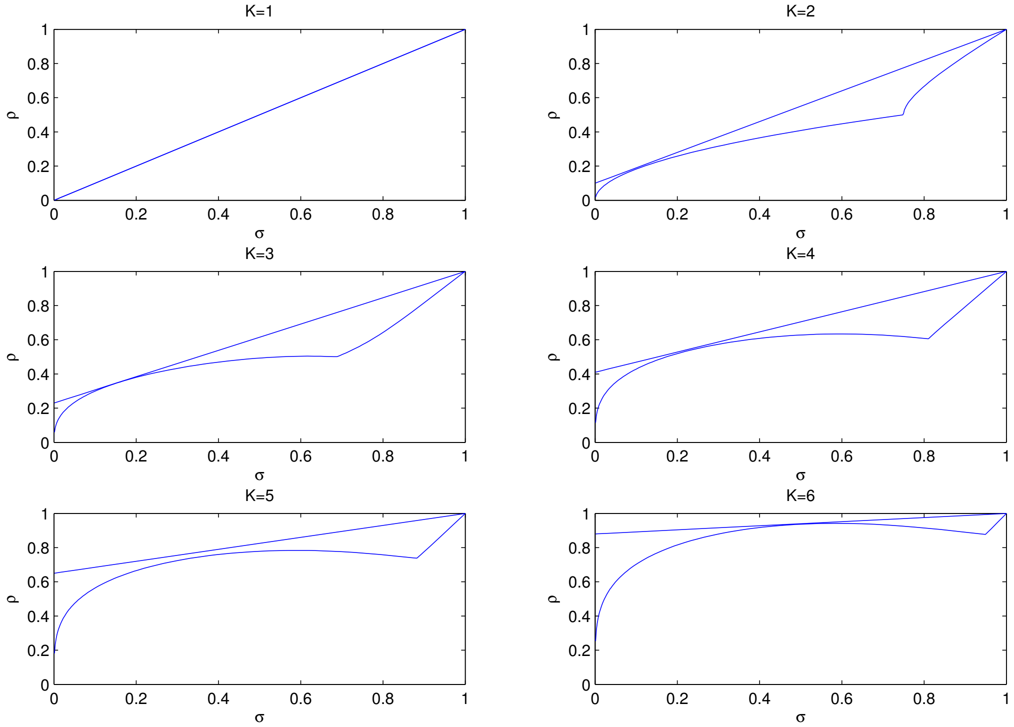


Figure 5.7: The relationship between $\rho(\mathbf{C}(x, p))$ and $\mu_p + \nu_p x$.

□

Note that these bounds are true only if $\tilde{\sigma}, \tilde{\gamma} \geq 0$ and less than 1. Using these bounds $p = K$ for $\rho(\mathbf{S})$ and $p = k$ for $\rho(\mathbf{G}^q)$, the first stability condition in (5.97) is always satisfied if

$$(\mu_K + \nu_K \tilde{\sigma})(\mu_k + \nu_k \tilde{\gamma}) < 1.$$

Because $\mu_K + \nu_K = \mu_k + \nu_k = 1$, we obtain therefore the following sufficient stability conditions

$$\left\{ \begin{array}{c} (1 + \nu_K(\tilde{\sigma} - 1))(1 + \nu_k(\tilde{\gamma} - 1))^q < 1, \\ |\tilde{\rho}\tilde{\delta}| \|\mathbf{P}\| \|\mathbf{X}\| < \frac{1}{L^2} \nu_K (1 - \tilde{\sigma})(1 - (1 + \nu_k(\tilde{\gamma} - 1))^q), \\ \tilde{\sigma} \geq 0, \\ \tilde{\gamma} \geq 0. \end{array} \right. \tag{5.99}$$

In the previous section we derived more compact stability conditions from (5.51) by means of an asymptotic analysis. This idea will be generalised for the BDF multirate methods of higher order. Since also the stability conditions (5.94) are very complex, we

will derive more compact stability conditions based on an asymptotic analysis. Firstly, we will prove that the studied multirate schemes are always conditionally stable. Secondly, we also will give sufficient conditions for $q \to \infty$ such that the methods are stable for all H.
In this section we will investigate the conditions for conditional stability which can be retrieved by an asymptotic analysis for $H \to 0$.

Theorem 5.18 (Sufficient stability conditions for $H \to 0$) *If*

$$\left\{\begin{array}{rcl} \mathbf{A} & \textit{is} & \textit{stable}, \\ a_{11} & < & 0, \\ a_{22} & < & 0, \\ |a_{21}a_{12}| & < & C|a_{11}a_{22}|, \end{array}\right. \tag{5.100}$$

where $C := \frac{q\nu_K\nu_k}{L^2\|\boldsymbol{P}\|\|\boldsymbol{X}\|}$*, the Slow-Fast and Compound-Fast versions of the BDF multirate schemes applied to the stable test equation (5.30) are conditionally stable for* $H \to 0$*.*

Proof: Since $\mathbf{A}$ is a stable matrix, we have

$$\begin{array}{rcccl} \mathrm{tr}(\mathbf{A}) & = & a_{11} + a_{22} & < & 0, \\ \det(\mathbf{A}) & = & a_{11}a_{22} - a_{12}a_{21} & > & 0. \end{array} \tag{5.101}$$

For $H \to 0$ we have the following asymptotic expansions in H

$$\begin{array}{rclrcl} \tilde{\sigma} & \doteq & 1 + \frac{a_{22}}{\rho_0}H, & 1 + \nu_K(\tilde{\sigma} - 1) & \doteq & 1 + \nu_K\frac{a_{22}}{\rho_0}H, \\ \tilde{\gamma} & \doteq & 1 + \frac{a_{11}}{q\bar{\rho}_0}H, & (1 + \nu_k(\tilde{\gamma} - 1))^q & \doteq & 1 + \nu_k\frac{a_{11}}{\bar{\rho}_0}H, \\ \tilde{\rho}\tilde{\delta} & \doteq & \frac{a_{21}a_{12}}{\rho_0\bar{\rho}_0}Hh, & (1 - (1 + \nu_k(\tilde{\gamma} - 1))^q) & \doteq & -\nu_k\frac{a_{11}}{\bar{\rho}_0}H. \end{array}$$

For $H \to 0$ $\tilde{\sigma}, \tilde{\gamma}$ are positive numbers. Since $\|\mathbf{P}\| = \|\mathbf{P}_0\|$ and $\|\mathbf{X}\| \doteq \|\sum_{l=0}^{q-1}\mathbf{G}_0^l\mathbf{e}_1\mathbf{b}_{q-l}^T\|$, where $\mathbf{e}_1 = [1, 0, \ldots, 1] \in \mathbb{R}^k$, we get the following asymptotic stability conditions, instead of (5.99),

$$\left\{\begin{array}{rcl} 1 + \nu_K\frac{a_{22}}{\rho_0}H + \nu_k\frac{a_{11}}{\bar{\rho}_0}H & < & 1, \\ \frac{|a_{21}a_{12}|}{\rho_0\bar{\rho}_0}Hh\|\mathbf{P}\|\|\mathbf{X}\| & < & \frac{1}{L^2}\nu_K\frac{a_{22}}{\rho_0}H\nu_k\frac{a_{11}}{\bar{\rho}_0}H, \end{array}\right. \tag{5.102}$$

yielding first order conditions

$$\left\{\begin{array}{rcl} \nu_K\frac{a_{22}}{\rho_0}H + \nu_k\frac{a_{11}}{\bar{\rho}_0}H & < & 0, \\ |a_{21}a_{12}|\frac{1}{q}\|\mathbf{P}\|\|\mathbf{X}\| & < & \frac{1}{L^2}\nu_K\nu_k a_{11}a_{22}. \end{array}\right. \tag{5.103}$$

If $K = k$, such that $\frac{\nu_K}{\rho_0} = \frac{\nu_k}{\bar{\rho}_0}$, the first stability condition in (5.103) is always satisfied for a stable $\mathbf{A}$. This first condition is also satisfied for a stable $\mathbf{A}$ if $a_{11} < 0$ and $k \leq K$, such that $\frac{\nu_k}{\bar{\rho}_0} \geq \frac{\nu_K}{\rho_0}$, or if $a_{22} < 0$ and $k \geq K$, such that $\frac{\nu_k}{\bar{\rho}_0} \leq \frac{\nu_K}{\rho_0}$.
The second stability condition in (5.103) is satisfied if

$$|a_{21}a_{12}| < Ca_{11}a_{22}, \quad C = \frac{q\nu_K\nu_k}{L^2\|\mathbf{P}\|\|\mathbf{X}\|}. \tag{5.104}$$

Because of $a_{11}, a_{22} \leq 0$ in (5.100) it follows that $a_{11}a_{22} \geq 0$. In this case the condition (5.104) is indeed equivalent to

$$|a_{21}a_{12}| < C|a_{11}a_{22}|,$$

where C is unchanged. ☐

In this part we investigate the stability for $q \to \infty$ and $H > 0$. It appears that the stability conditions in (5.97) can be simplified by using the limit values $\mathbf{X}, \mathbf{R}$. We use the notation $\mathbf{e}_i$ for the i-th unit basis vector and $\mathbf{e} = [1, \ldots, 1]$ for the vector consisting of ones.

Lemma 5.19 *For* $q \to \infty$ *we have*

$$\boldsymbol{X} \to (\boldsymbol{I} - \boldsymbol{G})^{-1}\boldsymbol{e}_1\boldsymbol{e}_1^T = \frac{1}{\tilde{\gamma} - 1}\boldsymbol{e}\boldsymbol{e}_1^T,$$

and

$$\boldsymbol{R} = \tilde{\rho}\boldsymbol{S}\boldsymbol{P},$$

where $\boldsymbol{P}$ *is* $\boldsymbol{e}_K\boldsymbol{w}^T$*, with* $\boldsymbol{w}^T = \frac{\rho_0}{\rho_K}\boldsymbol{e}_1^T$ *for the Slow-Fast method and* $\boldsymbol{w}^T = [\frac{\rho_1}{\rho_K}, \ldots, \frac{\rho_K}{\rho_K}]$ *for the Compound-Fast method.*

Proof: For $q \to \infty$ we have that

$$\mathbf{X} = \frac{1}{\tilde{\delta}}\sum_{l=0}^{q-1}\mathbf{G}^l\mathbf{d}\mathbf{b}_{q-l}^T \to \sum_{l=0}^{\infty}\mathbf{G}^l\mathbf{e}_1\mathbf{e}_1^T = (\mathbf{I} - \mathbf{G})^{-1}\mathbf{e}_1\mathbf{e}_1^T.$$

It can be derived that

$$(\mathbf{I} - \mathbf{G})^{-1}\mathbf{e}_1\mathbf{e}_1^T = \frac{1}{1 + \tilde{\gamma}(\frac{\bar{\rho}_1}{\bar{\rho}_0} + \ldots + \frac{\bar{\rho}_K}{\bar{\rho}_0})} = \frac{1}{1 - \tilde{\gamma}},$$

because of the consistency condition $\bar{\rho}_0 + \ldots + \bar{\rho}_K = 0$. The other property follows directly from the definition of $\mathbf{P}$ in (5.87). ☐

Before we state the stability theorem 5.21 we need the following Lemma.

Lemma 5.20 *For both companion matrices* $\boldsymbol{S}, \boldsymbol{G}$ *of order* $p \in \{1, \ldots, 6\}$ *we have*

$$0 < \tilde{\sigma} < 1 \Rightarrow \rho(\boldsymbol{S}) < 1, \quad 0 < \tilde{\gamma} < 1 \Rightarrow \rho(\boldsymbol{G}) < 1, \quad 0 < \tilde{\gamma}^q < 1 \Rightarrow \rho(\boldsymbol{G}^q) < 1.$$

Proof: The matrix $\mathbf{S}$ has the characteristic equation

$$\lambda^p + \frac{\tilde{\sigma}}{\rho_0}(\rho_1\lambda^{p-1} + \ldots + \rho_p) = 0,$$

which is equivalent to

$$\frac{\rho_0}{\tilde{\sigma}}\lambda^p + \rho_1\lambda^{p-1} + \ldots + \rho_p = 0.$$

The BDF-p method for the test equation $\dot{y} = \lambda y$ gives us the characteristic polynomial

$$(\rho_0 - h\lambda)\lambda^p + \rho_1\lambda^{p-1} + \ldots + \rho_p = 0.$$

It is well-known that for $h\lambda \in \mathbb{R}^-$ the numerical solution will be stable up to order $p = 6$. It follows that $\rho(\mathbf{S}) < 1$ if

$$\frac{\rho_0}{\tilde{\sigma}} > \rho_0,$$

being equivalent to

$$0 < \tilde{\sigma} < 1 \Rightarrow \rho(\mathbf{S}) < 1.$$

Since $\mathbf{G} = \mathbf{S}$ if $K = k$ and $\tilde{\sigma} = \tilde{\gamma}$, it immediately follows that

$$0 < \tilde{\gamma} < 1 \Rightarrow \rho(\mathbf{G}) < 1.$$

Because $0 < \tilde{\gamma} < 1 \Leftrightarrow 0 < \tilde{\gamma}^q < 1$ and $\rho(\mathbf{G}) < 1 \Leftrightarrow \rho(\mathbf{G}^q) < 1$, we also have

$$0 < \tilde{\gamma}^q < 1 \Rightarrow \rho(\mathbf{G}^q) < 1.$$

□

Now we are able to derive sufficient stability conditions such that the studied BDF multirate algorithms are stable for $q \to \infty$. It turns out that if both subsystems are sufficiently decoupled and the active and slow parts of the system are stable and solvable, both the SF and CF versions are stable.

Theorem 5.21 (Sufficient stability conditions for $q \to \infty$) *If*

$$\left\{\begin{array}{rcl} \boldsymbol{A} & \textit{is} & \textit{stable}, \\ a_{11} & < & 0, \\ a_{22} & < & 0, \\ |a_{21}a_{12}| & < & D|a_{11}a_{22}|, \end{array}\right. \tag{5.105}$$

where $D := \frac{\nu_K}{L^2\|\boldsymbol{P}\|}\left(1 - \frac{a_{22}}{\rho_0}H_{\max}\right)^{-1}$*, then the Slow-Fast BDF multirate schemes applied to the stable test equation (5.30) are unconditionally stable for* $q \to \infty$*. If (5.105) hold, where* $D := \frac{\nu_K}{L^2\|\boldsymbol{P}\|}\left(1 - \frac{(a_{11}+a_{22})}{\rho_0}H_{\max} + \frac{(a_{11}a_{22}-a_{12}a_{21})}{\rho_0^2}H_{\max}^2\right)^{-1}$*, then the Compound-Fast BDF multirate schemes applied to the stable test equation (5.30) are unconditionally stable for* $q \to \infty$*.*

Proof: The stability conditions in (5.97) are only satisfied for $q \to \infty$ if $\rho(\mathbf{G}) < 1$, such that $\rho(\mathbf{G}^q) \to 0$. Then we get the stability conditions

$$\left\{\begin{array}{rcl} \rho(\mathbf{G}) & < & 1, \\ \rho(\mathbf{S}) + L^2|\tilde{\rho}\tilde{\delta}|\,\|\mathbf{P}\|\,\|\mathbf{X}\| & < & 1. \end{array}\right.$$

Using the Lemma 5.17 and 5.20 it is possible to derive the sufficient stability conditions for $\tilde{\gamma} = \frac{\tilde{\rho}_0}{\tilde{\rho}_0 - a_{11}h}$

$$\left\{\begin{array}{rcl} \tilde{\gamma} & \in & [0,1], \\ \mu_K + \nu_K\tilde{\sigma} + L^2|\tilde{\rho}\tilde{\delta}|\,\|\mathbf{P}\|\,\|\mathbf{X}\| & < & 1, \\ \tilde{\sigma} & > & 0. \end{array}\right.$$

The first condition is fulfilled indeed if $a_{11} < 0$. Because of Lemma 5.19 we know that $\|\mathbf{X}\| \to \frac{1}{|\tilde{\gamma}-1|} \|\mathbf{e}\mathbf{e}_1^T\| = \frac{1}{|\tilde{\gamma}-1|}$, where we use $\|\mathbf{e}\mathbf{e}_1^T\| = 1$. Then the second stability condition is satisfied if

$$\mu_K + \nu_K \tilde{\sigma} + L^2 \|\mathbf{P}\| \, \frac{|\tilde{\rho}\tilde{\delta}|}{|\tilde{\gamma} - 1|} < 1.$$

Since $\frac{\tilde{\delta}}{\tilde{\gamma}-1} = -\frac{a_{12}}{a_{11}}$, we get

$$\mu_K + \nu_K \tilde{\sigma} + L^2 \|\mathbf{P}\| \, |\tilde{\rho}| \frac{|a_{12}|}{|a_{11}|} < 1. \tag{5.106}$$

For the Slow-Fast method this implies

$$\mu_K + \nu_K \frac{\rho_0}{\rho_0 - a_{22}H} + L^2 \|\mathbf{P}\| \, |a_{21}| \frac{1}{\rho_0} H \frac{|a_{12}|}{|a_{11}|} < 1,$$

or

$$L^2 \|\mathbf{P}\| \, |a_{21}| \frac{1}{\rho_0} H \frac{|a_{12}|}{|a_{11}|} < 1 - \mu_K - \nu_K \frac{\rho_0}{\rho_0 - a_{22}H}.$$

Using $\mu_K = 1 - \nu_K$, we derive

$$L^2 \|\mathbf{P}\| \, |a_{21}| \frac{1}{\rho_0} H \frac{|a_{12}|}{|a_{11}|} < \nu_K - \nu_K \frac{\rho_0}{\rho_0 - a_{22}H}$$

or

$$L^2 \|\mathbf{P}\| \, |a_{21}| H \frac{|a_{12}|}{|a_{11}|} < -\frac{\nu_K a_{22} H}{1 - \frac{a_{22}}{\rho_0} H}.$$

Thus we obtain the following sufficient stability condition, where we insert $D = \frac{\nu_K}{L^2 \|\mathbf{P}\|} \big(1 - \frac{a_{22}}{\rho_0} H_{\max}\big)^{-1}$

$$|a_{21}||a_{12}| < -D|a_{11}|a_{22}. \tag{5.107}$$

Since $a_{22} < 0$ we derive the sufficient condition

$$|a_{21}a_{12}| < D|a_{22}||a_{11}|.$$

Because $a_{22} < 0$ we see that $\tilde{\sigma} > 0$. Thus the Slow-Fast multirate method is indeed stable for $q \to \infty$ if the stability conditions (5.105) are satisfied.
For the Compound-Fast method we obtain for (5.106)

$$\mu_K + \nu_K \frac{\rho_0(\rho_0 - a_{11}H)}{\rho_0^2 - \rho_0(a_{11} + a_{22})H + (a_{11}a_{22} - a_{12}a_{21})H^2} + L^2 \|\mathbf{P}\| \, |\frac{a_{21}H}{\rho_0 - a_{11}H}| \frac{|a_{12}|}{|a_{11}|} < 1$$

or

$$L^2 \|\mathbf{P}\| \, |\frac{a_{21}H}{\rho_0 - a_{11}H}| \frac{|a_{12}|}{|a_{11}|} < 1 - \mu_K - \nu_K \frac{\rho_0(\rho_0 - a_{11}H)}{\rho_0^2 - \rho_0(a_{11} + a_{22})H + (a_{11}a_{22} - a_{12}a_{21})H^2}.$$

Because $a_{11} < 0$ we can use the estimate $|\frac{a_{21}H}{\rho_0 - a_{11}H}| \leq |\frac{a_{21}}{\rho_0}|H$. Using also $\mu_K = 1 - \nu_K$, it is sufficient to prove that

$$\begin{aligned} L^2 \|\mathbf{P}\| \, |a_{21}| \tfrac{1}{\rho_0} H \tfrac{|a_{12}|}{|a_{11}|} &< \nu_K - \nu_K \tfrac{\rho_0(\rho_0 - a_{11}H)}{\rho_0^2 - \rho_0(a_{11}+a_{22})H + (a_{11}a_{22} - a_{12}a_{21})H^2} \\ &= -\tfrac{\nu_K(\rho_0 a_{22} H - \det(\mathbf{A})H^2)}{\rho_0^2 - \rho_0(a_{11}+a_{22})H + (a_{11}a_{22} - a_{12}a_{21})H^2}. \end{aligned}$$

Multiplication by $|a_{11}|\frac{\rho_0}{H}$ gives us

$$L^2\|\mathbf{P}\|\;|a_{21}a_{12}| < -\frac{\nu_K|a_{11}|a_{22}(\rho_0^2 - \rho_0\frac{\det(\mathbf{A})}{a_{22}}H)}{\rho_0^2 - \rho_0(a_{11}+a_{22})H + (a_{11}a_{22} - a_{12}a_{21})H^2}.$$

Since $\mathrm{tr}(\mathbf{A}) = a_{11} + a_{22} < 0$ and $\det(\mathbf{A}) = a_{11}a_{22} - a_{12}a_{21} > 0$, we obtain the sufficient stability condition, where we insert $D = \frac{\nu_K}{L^2\|\mathbf{P}\|}\big(1 - \frac{(a_{11}+a_{22})}{\rho_0}H_{\max} + \frac{(a_{11}a_{22}-a_{12}a_{21})}{\rho_0^2}H_{\max}^2\big)^{-1}$.

$$|a_{21}a_{12}| < -D|a_{11}|a_{22}(1 - \frac{H}{\rho_0 a_{22}}\det(\mathbf{A})).$$

If $a_{22} < 0$ and $\det(\mathbf{A}) > 0$ it is immediately clear that

$$-a_{22}(1 - \frac{\det(\mathbf{A})}{\rho_0 a_{22}}H) = |a_{22}|(1 - \frac{\det(\mathbf{A})}{\rho_0 a_{22}}H) > |a_{22}|. \tag{5.108}$$

Thus the BDF Compound-Fast multirate method is stable for $q \to \infty$ indeed if the stability conditions in (5.105) hold because then $D|a_{11}a_{22}| < -D|a_{11}|a_{22}(1 - \frac{\det(\mathbf{A})}{\rho_0 a_{22}}H)$. Since $a_{22} < 0$ it immediately follows that $\tilde{\sigma} > 0$ is always satisfied. Thus also the Compound-Fast multirate method is stable for $q \to \infty$ if the stability conditions in (5.105) are satisfied. □

Because of the restriction $\rho(\mathbf{G}) < 1$ it appears possible to reduce the stability conditions in Theorem 5.21.

Lemma 5.22 *For $q \to \infty$ the matrix $\boldsymbol{M}$ in (5.75) is stable if*

$$\rho(\boldsymbol{S}(\boldsymbol{I} + \tilde{\rho}\tilde{\delta}\boldsymbol{P}\boldsymbol{X})) < 1. \tag{5.109}$$

Proof: For $q \to \infty$ we have

$$\mathbf{M} \to \begin{pmatrix} \mathbf{S} & \mathbf{R} \\ \tilde{\delta}\mathbf{X}\mathbf{S} & \tilde{\delta}\mathbf{X}\mathbf{R} \end{pmatrix}.$$

For each eigenpair $(\lambda, \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix})$ we see

$$\begin{aligned} \mathbf{S}\mathbf{x} + \mathbf{R}\mathbf{y} &= \lambda\mathbf{x} \\ \tilde{\delta}\mathbf{X}\mathbf{S}\mathbf{x} + \tilde{\delta}\mathbf{X}\mathbf{R}\mathbf{y} &= \lambda\mathbf{y}. \end{aligned}$$

This implies

$$\lambda\tilde{\delta}\mathbf{X}\mathbf{x} = \tilde{\delta}\mathbf{X}\mathbf{S}\mathbf{x} + \tilde{\delta}\mathbf{X}(\lambda\mathbf{x} - \mathbf{S}\mathbf{x}) = \lambda\mathbf{y}.$$

Thus for each eigenvector with $\lambda \neq 0$ we get $\mathbf{y} = \tilde{\delta}\mathbf{X}\mathbf{x}$. Hence we can reduce the eigenvalue problem to

$$\mathbf{S}\mathbf{x} + \tilde{\delta}\mathbf{R}\mathbf{X}\mathbf{x} = (\mathbf{S} + \tilde{\delta}\mathbf{R}\mathbf{X})\mathbf{x} = \lambda\mathbf{x}.$$

Clearly the method is stable for $q \to \infty$ if $\rho(\mathbf{S} + \tilde{\delta}\mathbf{R}\mathbf{X}) < 1$. Since $\mathbf{R} = \tilde{\rho}\mathbf{S}\mathbf{P}$, this is equivalent to

$$\rho(\mathbf{S}(\mathbf{I} + \tilde{\rho}\tilde{\delta}\mathbf{P}\mathbf{X})) < 1.$$

□

5.8 Multirate for hierarchical models

Integrated Circuits can be modeled by a *hierarchical system* of differential-algebraic equations [31]:

$$\frac{d}{dt}[\mathbf{q}(t,\mathbf{x})] + \mathbf{j}(t,\mathbf{x}) = \sum_{i=1}^{N} \mathbf{B}_{(i)}^{T} \left[\frac{d}{dt}[\mathbf{q}^{(i)}(t,\mathbf{x}^{(i)})] + \mathbf{j}^{(i)}(t,\mathbf{x}^{(i)})\right] = \mathbf{0}. \tag{5.110}$$

Clearly this circuit model consists of N coupled subcircuit models. Each local state vector $\mathbf{x}^{(i)}$ (voltages,currents) consists of a terminal ($\hat{\mathbf{x}}^{(i)}$) and an internal ($\check{\mathbf{x}}^{(i)}$) part:

$$\mathbf{x}^{(i)} = \mathbf{B}_{(i)}\mathbf{x} = \begin{bmatrix} \hat{\mathbf{x}}^{(i)} \\ \check{\mathbf{x}}^{(i)} \end{bmatrix}.$$

The matrices $\mathbf{B}_{(i)} \in \{0,1\}^{d_i \times d}$, defined by (5.111), are used to select the proper parts $\hat{\mathbf{x}}^{(i)} = \begin{bmatrix} \hat{\mathbf{B}}_{(i)} & \end{bmatrix}\mathbf{x}$ and $\check{\mathbf{x}}^{(i)} = \begin{bmatrix} & \check{\mathbf{B}}_{(i)} \end{bmatrix}\mathbf{x}$ of $\mathbf{x}^{(i)}$ from $\mathbf{x}$. This even allows for a hierarchical structure.

$$\mathbf{B}_{(i)} = \begin{bmatrix} \hat{\mathbf{B}}_{(i)} & \\ & \check{\mathbf{B}}_{(i)} \end{bmatrix}. \tag{5.111}$$

Similarly, also the functions $\mathbf{q}^{(i)}$ (charges,fluxes) and $\mathbf{j}^{(i)}$ (currents,voltages) have a similar structure:

$$\mathbf{q}^{(i)} = \mathbf{B}_{(i)}\mathbf{q} = \begin{bmatrix} \hat{\mathbf{q}}^{(i)} \\ \check{\mathbf{q}}^{(i)} \end{bmatrix}, \quad \mathbf{j}^{(i)} = \mathbf{B}_{(i)}\mathbf{j} = \begin{bmatrix} \hat{\mathbf{j}}^{(i)} \\ \check{\mathbf{j}}^{(i)} \end{bmatrix}.$$

We can rewrite (5.110) in a part consisting of the collected equations for the terminal unknowns (5.112) and a part consisting of the remaining equations for the internal unknowns (5.113):

$$\sum_{i=1}^{N} \hat{\mathbf{B}}_{(i)}^{T} \left[\frac{d}{dt}[\hat{\mathbf{q}}^{(i)}(t,\mathbf{x}^{(i)})] + \hat{\mathbf{j}}^{(i)}(t,\mathbf{x}^{(i)})\right] = \mathbf{0}, \tag{5.112}$$

$$\frac{d}{dt}[\check{\mathbf{q}}^{(i)}(t,\mathbf{x}^{(i)})] + \check{\mathbf{j}}^{(i)}(t,\mathbf{x}^{(i)}) = \mathbf{0}, \quad i = 1,\ldots,N. \tag{5.113}$$

Each subcircuit model can again be further decomposed in this manner.

For single-rate time integration all equations are discretised simultaneously by the same time step. If the time constants per subcircuits are quite different, it is attractive to perform multirate time integration. Then the fast subcircuits can be integrated independently on a fine time-grid. Especially when the fast subcircuits are small in size, the additional costs for synchronisation and partitioning can be overcome and the overall multirate procedure becomes much more efficient than the single-rate time integration. An attractive multirate method is the Compound-Fast version [65, 68], which first integrates the whole system at the new coarse time gridpoint and after that re-integrates only the active part at the fine time-grid. We will denote the coarse and fine time gridpoints by $\{T_n, 0 \le n \le N\}$ and $\{t_{n-1,m}, 1 \le n \le N, 0 \le m \le q_n\}$ with macro-steps $H_n := T_n - T_{n-1}$, and micro-steps $h_{n,m} := t_{n,m} - t_{n,m-1}$ and multirate factors q_n. For a partitioning in a latent (slow) and an active (fast) part, $\mathbf{x}^{(L)} \in \mathbb{R}^{d_L}$ and $\mathbf{x}^{(A)} \in \mathbb{R}^{d_A}$, we typically get:

$$\begin{aligned}
\hat{\mathbf{B}}_{(L)}^{T}\left[\frac{d}{dt}[\hat{\mathbf{q}}^{(L)}(t,\mathbf{x}^{(L)})]+\hat{\mathbf{j}}^{(L)}(t,\mathbf{x}^{(L)})\right]+\hat{\mathbf{B}}_{(A)}^{T}\left[\frac{d}{dt}[\hat{\mathbf{q}}^{(A)}(t,\mathbf{x}^{(A)})]+\hat{\mathbf{j}}^{(A)}(t,\mathbf{x}^{(A)})\right] &= \mathbf{0},\\
\frac{d}{dt}[\check{\mathbf{q}}^{(L)}(t,\mathbf{x}^{(L)})]+\check{\mathbf{j}}^{(L)}(t,\mathbf{x}^{(L)}) &= \mathbf{0},\\
\frac{d}{dt}[\check{\mathbf{q}}^{(A)}(t,\mathbf{x}^{(A)})]+\check{\mathbf{j}}^{(A)}(t,\mathbf{x}^{(A)}) &= \mathbf{0}.
\end{aligned}$$

In contrast to flat models we have the property that $d_L + d_A \geq d$ because there can be shared terminal variables.
A first approach is to integrate only the internal part of the active subcircuit at the fine time-grid. Then we get the following active circuit model for $\check{\mathbf{x}}^{(A)}$

$$\frac{d}{dt}[\check{\mathbf{q}}^{(A)}(t,\mathbf{x}^{(A)})]+\check{\mathbf{j}}^{(A)}(t,\mathbf{x}^{(A)}) = \mathbf{0}, \quad \mathbf{x}^{(A)} = \begin{bmatrix} \hat{\mathbf{x}}^{(A)} \\ \check{\mathbf{x}}^{(A)} \end{bmatrix}. \tag{5.114}$$

In practise $\hat{\mathbf{x}}^{(A)}$ will also behave latently, and in this case it is preferable to use voltage interpolation of the terminal voltages $\hat{\mathbf{x}}^{(A)}$. From the hierarchical linear solver in Pstar [13] we know that (5.114) is solvable for $\check{\mathbf{x}}^{(A})$. However, stability is now not automatically preserved from the original model. Furthermore the DAE-index can be larger than one, which typically leads to sawtooth-like shapes of $\check{\mathbf{x}}^{(A})$.
A second approach is to integrate the complete active subcircuit (i.e. using $\mathbf{q}^{(A)}$ rather than $\check{\mathbf{q}}^{(A)}$, etc) at the fine time-grid. Then we get the following active circuit model for $\mathbf{x}^{(A)}$

$$\frac{d}{dt}[\mathbf{q}^{(A)}(t,\mathbf{x}^{(A)})]+\mathbf{j}^{(A)}(t,\mathbf{x}^{(A)}) = -\hat{\mathbf{B}}_{(A)}\hat{\mathbf{B}}_{(L)}^{T}\left[\frac{d}{dt}[\hat{\mathbf{q}}^{(L)}(t,\mathbf{x}^{(L)})]+\hat{\mathbf{j}}^{(L)}(t,\mathbf{x}^{(L)})\right]. \tag{5.115}$$

This leads to a more stable situation including the conservation of Kirchhoff's Current Law at the terminals and preservation of the DAE-index. In this case it is preferable to interpolate the terminal currents

$$\mathbf{i}_{L\to A} = -\hat{\mathbf{B}}_{(A)}\hat{\mathbf{B}}_{(L)}^{T}\left[\frac{d}{dt}[\hat{\mathbf{q}}^{(L)}(t,\mathbf{x}^{(L)})]+\hat{\mathbf{j}}^{(L)}(t,\mathbf{x}^{(L)})\right]. \tag{5.116}$$

This can be done by adding $\mathbf{i}_{L\to A}$ as unknown or by calculating it explicitly. Direct interpolation of the terminal slow voltages $\mathbf{x}^{(L)}$ is not attractive because $\mathbf{i}_{L\to A}$ could also depend on slow internals $\check{\mathbf{x}}^{(L)}$. In Pstar for each subcircuit the corresponding terminal current $\frac{d}{dt}[\hat{\mathbf{q}}^{(i)}(t,\mathbf{x}^{(i)})]+\hat{\mathbf{j}}^{(i)}(t,\mathbf{x}^{(i)})$ is stored. Then the vector $\mathbf{i}_{L\to A}$ can be constructed for each multirate-partitioning. When model (5.115) is integrated, the equations are discretised and solved by an iterative Newton method. Each Newton iteration a linear algebraic system of size d_F has to be solved. Note that then the right-hand-sides consist not only of a sampled $\mathbf{i}_{L\to A}$ but also on the previous solutions of the active state (time history).
Interpolation of the currents $\mathbf{i}_{L\to A}$ causes solvability problems for the active part if the active subcircuits are not grounded (so one may have to ground the most latent coupled

terminal unknown). Stability and the differential index are only preserved if all subcircuits are stable DAEs of index one. In general this property can not be assumed for a circuit simulator.
It is well-known that the ungrounded circuit model (5.110) does not have a unique solution. Usually one nodal voltage therefore is set to zero while the corresponding equation is skipped. If the active part is not directly coupled to the ground, but only via slowly time-varying terminal unknowns, we could of course add these terminal unknowns to the active part, but then we would like to find another solution.
We know that each circuit and subcircuit is of the type

$$\frac{d}{dt}[\mathbf{q}^{(i)}(t, \mathbf{x}^{(i)})] + \mathbf{j}^{(i)}(t, \mathbf{x}^{(i)}) = \mathbf{0}.$$

Here $\mathbf{x}^{(i)} = \begin{bmatrix} \hat{\mathbf{x}}^{(i)} \\ \check{\mathbf{x}}^{(i)} \end{bmatrix}$ consists of terminal unknowns $\hat{\mathbf{x}}^{(i)}$ and internal unknowns $\check{\mathbf{x}}^{(i)}$. For the root circuit the only terminal unknown is the voltage of the ground-node, which is given. All other unknowns are considered to be internal at the circuit level. For a proper circuit this automatically means that these internals are uniquely solvable. If we consider a subcircuit with the same structure the state vector again consists of a terminal and an internal part. If the terminal unknowns are given then the internal unknowns can always be computed. The major problem of interpolation of the currents is that also the terminal part of $\mathbf{x}^{(A)}$ is treated as an unknown. Although we can assume that $\check{\mathbf{x}}^{(A)}$ can be found for given $\hat{\mathbf{x}}^{(A)}$, it is not clear whether we can find $\mathbf{x}^{(A)}$ for a given $\mathbf{i}_T$. It appears necessary to select one element of $\hat{\mathbf{x}}^{(A)}$ which has to be grounded to the corresponding slow unknowns. This means that for an ungrounded fast subcircuit, one terminal unknown has to be grounded to the coarse grid solution. Of course this is only possible if that unknown varies slowly in time. Thus if $\hat{\mathbf{x}}^{(A)}$ contains at least one latent unknown (voltage), e.g. with a constant value, this unknown can be approximated by interpolation. In electrical terms this means that such an unknown is connected to a grounded voltage source.
There are also subcircuits which do not have a unique solution if one terminal variable is grounded. Typically these subcircuits are *reducible*, which means that they consist of two unconnected parts. If the parts are really unconnected, this is easy to check from the topology matrix. For normal circuit models its rank is equal to $d - 1$. If the rank is equal to $d - k$, it can be proved that the subcircuit consists of k unconnected parts. Unfortunately, also irreducible models can be reducible in a numerical sense, e.g. if two parts are connected by an electrical element with a huge impedance. It is even possible that the reducibility varies with time if a connecting transistor behaves like a switch.

5.9 Implicit interpolation techniques

In [67] it is shown that the Compound-Fast algorithm can have solvability problems with dynamical partitioning. The major problem is that the active subcircuit does not automatically preserve all dynamical properties of the complete model, like solvability,

stability and irreducibility. Also the differential-index can be higher than one if the complete model has DAE-index one. Irreducibility means that all variables of the model are connected. Note that in a numerical sense it can be even worse.
An alternative could be a modified BDF multirate algorithm with *implicit interpolation*. Such a method may solve this problem and may also reduce the average number of Newton iterations per compound step. For this we will develop a proper description of implicit current interpolation which is necessary for this method.
We saw that both voltage and current interpolation can still have problems with solvability, stability or the differential index. These problems can be avoided by an alternative approach, where $\hat{\mathbf{x}}^{(A)}$ or $\mathbf{i}_{L\to A}$ are approximated by an implicit interpolation formula. Because of the datastructure organised by the hierarchically connected subcircuits it seems more attractive to consider the second case with current interpolation. Because of a blockwise treatment of subcircuits the contribution to the current equations at the parent subcircuit level are just terminal currents. In (5.116) we already introduced the terminal current $\mathbf{i}_{L\to A}$ from latent-to-active. We also introduce the terminal current $\mathbf{i}_{A\to L}$ from active-to-latent. Then one can also write the hierarchical circuit model of (5.115) like

$$\left\{\begin{array}{ll} \frac{d}{dt}[\mathbf{q}^{(L)}(t,\mathbf{x}^{(L)})] + \mathbf{j}^{(L)}(t,\mathbf{x}^{(L)}) = \mathbf{i}_{A\to L}, & i \\ \mathbf{i}_{A\to L} = -\hat{\mathbf{B}}_{(L)}\hat{\mathbf{B}}^T_{(A)}\left[\frac{d}{dt}[\hat{\mathbf{q}}^{(A)}(t,\mathbf{x}^{(A)})] + \hat{\mathbf{j}}^{(A)}(t,\mathbf{x}^{(A)})\right], & ii \\ \mathbf{i}_{L\to A} = -\hat{\mathbf{B}}_{(A)}\hat{\mathbf{B}}^T_{(L)}\left[\frac{d}{dt}[\hat{\mathbf{q}}^{(L)}(t,\mathbf{x}^{(L)})] + \hat{\mathbf{j}}^{(L)}(t,\mathbf{x}^{(L)})\right], & iii \\ \frac{d}{dt}[\mathbf{q}^{(A)}(t,\mathbf{x}^{(A)})] + \mathbf{j}^{(A)}(t,\mathbf{x}^{(A)}) = \mathbf{i}_{L\to A}. & iv \end{array}\right. \tag{5.117}$$

Here $\mathbf{i}_{A\to L}$ and $\mathbf{i}_{L\to A}$ are the terminal currents that couple both subcircuits. If the vector $\mathbf{i}_{L\to A}$ is given it is possible to perform the refinement for $\mathbf{x}^{(A)}$. For the Slow-Fast multirate method $\mathbf{i}_{L\to A}$ is approximated at the coarse time-grid, based on $\mathbf{x}^{(L)}$. However, it is also possible to approximate $\mathbf{i}_{L\to A}$ by a different approach.
Note that $\mathbf{i}_{L\to A}$ and $\mathbf{i}_{A\to L}$ are related by the Kirchhoff's Current Law

$$\hat{\mathbf{B}}^T_{(L)}\mathbf{i}_{A\to L} + \hat{\mathbf{B}}^T_{(A)}\mathbf{i}_{L\to A} = \mathbf{0}.$$

Let us discretise (5.117i) by Euler Backward with step $H_n = t_{n-1,m} - t_{n-1,0}$, where $t_{n-1,0} = T_{n-1}$

$$\mathbf{q}^{(L)}(t_{n-1,m},\mathbf{x}^{(L)}_{n-1,m}) - \mathbf{q}^{(L)}(t_{n-1,0},\mathbf{x}^{(L)}_{n-1,0}) + H_m\mathbf{j}^{(L)}(t_{n-1,m},\mathbf{x}^{(L)}_{n-1,m}) = \mathbf{i}_{A\to L}(t_{n-1,m}). \tag{5.118}$$

In this section we assume that just one Newton step is needed to correct the prediction $\hat{\mathbf{x}}^{(L)}_{n-1,m}$, which is acceptable if $\mathbf{x}^{(L)}$ behaves latently. Thus

$$\mathbf{J}^{(L)}_{n-1,m}\big(\mathbf{x}^{(L)}_{n-1,m} - \hat{\mathbf{x}}^{(L)}_{n-1,m}\big) = \mathbf{i}_{A\to L}(t_{n-1,m}) - \mathbf{f}^{(L)}_{n-1,m},$$

where $\mathbf{J}^{(L)}_{n-1,m} = \mathbf{C}^{(L)}(t_{n-1,m},\hat{\mathbf{x}}^{(L)}_{n-1,m}) - \mathbf{C}^{(L)}(t_{n-1,0},\mathbf{x}^{(L)}_{n-1,0}) + H_m\mathbf{G}^{(L)}(t_{n-1,m},\hat{\mathbf{x}}^{(L)}_{n-1,m})$ and $\mathbf{f}_{n-1,m} = \mathbf{q}^{(L)}(t_{n-1,m},\hat{\mathbf{x}}^{(L)}_{n-1,m}) - \mathbf{q}^{(L)}(t_{n-1,0},\mathbf{x}^{(L)}_{n-1,0}) + H_m\mathbf{j}^{(L)}(t_{n-1,m},\hat{\mathbf{x}}^{(L)}_{n-1,m})$. Hence

$$\mathbf{x}^{(L)}_{n-1,m} = \hat{\mathbf{x}}^{(L)}_{n-1,m} + \mathbf{J}^{-1}_{n-1,m}\big(\mathbf{i}_{A\to L}(t_{n-1,m}) - \mathbf{f}_{n-1,m}\big).$$

We do not want to compute $\mathbf{J}_{n-1,m}^{-1}\mathbf{f}_{n-1,m}$ for all m, so we use linear interpolation of $\mathbf{J}_{n-1,0}^{-1}\mathbf{f}_{n-1,0} = \mathbf{J}_{n-1,0}^{-1}\mathbf{f}(\mathbf{x}_{n-1,0}^{(L)})$ and $\mathbf{J}_{n,0}^{-1}\mathbf{f}_{n,0} = \mathbf{J}_{n,0}^{-1}\mathbf{f}(\mathbf{x}_{n,0}^{(L)})$. Thus we obtain

$$\mathbf{J}_{n-1,m}^{-1}\mathbf{f}_{n-1,m} := \lambda_m \mathbf{J}_{n-1,0}^{-1}\mathbf{f}_{n-1,0} + \mu_m \mathbf{J}_{n,0}^{-1}\mathbf{f}_{n,0}.$$

Here $\lambda_m = 1 - \frac{m}{q}$ and $\mu_m = 1 - \lambda_m = \frac{m}{q}$. Since for $m = 0$ we have that $\mathbf{x}_{n-1,m}^{(L)} = \hat{\mathbf{x}}_{n-1,0}^{(L)}$ solves (5.118), we have that $\mathbf{f}_{n-1,0} = \mathbf{0}$. After interpolating the operator $\mathbf{J}^{-1}$ applied to $\mathbf{i}_{A\to L}(t_{n-1,m})$, we obtain the following approximation of $\mathbf{x}_{n-1,m}^{(L)}$:

$$\begin{array}{rcl} \mathbf{x}_{n-1,m}^{(L)} & \approx & \hat{\mathbf{x}}_{n-1,m}^{(L)} + (\lambda_m \mathbf{J}_{n-1,0}^{-1} + \mu_m \mathbf{J}_{n,0}^{-1})\mathbf{i}_{A\to L}(t_{n-1,m}) - \mu_m \mathbf{J}_{n,0}^{-1}\mathbf{f}_{n,0} \\ & \approx & \mathbf{a}(t_{n-1,m}) + \mathbf{A}(t_{n-1,m})\mathbf{i}_{A\to L}(t_{n-1,m}), \end{array} \tag{5.119}$$

where $\mathbf{a}(t_{n-1,m}) = \hat{\mathbf{x}}_{n-1,m}^{(L)} - \mu_m \mathbf{J}_{n,0}^{-1}\mathbf{f}_{n,0}$ and $\mathbf{A}(t_{n-1,m}) = \lambda_m \mathbf{J}_{n-1,0}^{-1} + \mu_m \mathbf{J}_{n,0}^{-1}$. Instead of solving the complete system (5.117), we can solve the following reduced system for $\mathbf{i}_{A\to L}, \mathbf{i}_{L\to A}, \mathbf{x}^{(A)}$

$$\left\{ \begin{array}{c} \mathbf{i}_{A\to L} = -\hat{\mathbf{B}}_{(L)}\hat{\mathbf{B}}_{(A)}^{T}\left[\frac{d}{dt}[\hat{\mathbf{q}}^{(A)}(t,\mathbf{x}^{(A)})] + \hat{\mathbf{j}}^{(A)}(t,\mathbf{x}^{(A)})\right], \\ \mathbf{i}_{L\to A} = -\hat{\mathbf{B}}_{(A)}\hat{\mathbf{B}}_{(L)}^{T}\left[\frac{d}{dt}[\hat{\mathbf{q}}^{(L)}(t,\mathbf{a}(t) + \mathbf{A}(t)\mathbf{i}_{A\to L})] + \hat{\mathbf{j}}^{(L)}(t,\mathbf{a}(t) + \mathbf{A}(t)\mathbf{i}_{A\to L})\right], \\ \frac{d}{dt}[\mathbf{q}^{(A)}(t,\mathbf{x}^{(A)})] + \mathbf{j}^{(A)}(t,\mathbf{x}^{(A)}) = \mathbf{i}_{L\to A}. \end{array} \right. \tag{5.120}$$

We can then compute $\mathbf{x}^{(L)}$ by evaluating formula (5.119). Thus we get a multirate method of Fastest First type instead of Slowest First type. In contrast to the Compound-Fast multirate method we do not need a compound step now to predict $\mathbf{i}_{L\to A}$. However the stability properties of this new method are not yet clear. Section 5.5 only considers the stability analysis for the Slow-Fast and Compound-Fast versions of the BDF multirate algorithm.

Similarly, $\mathbf{i}_{L\to A}$ satisfies

$$\begin{array}{rcl} \mathbf{i}_{L\to A}(t_{n-1,m}) & = & -\hat{\mathbf{B}}_{(A)}\hat{\mathbf{B}}_{(L)}^{T}\left[\frac{d}{dt}[\hat{\mathbf{q}}^{(L)}(t_{n-1,m},\mathbf{x}_{n-1,m}^{(L)})] + \hat{\mathbf{j}}^{(L)}(t_{n-1,m},\mathbf{x}_{n-1,m}^{(L)})\right] \\ & = & -\hat{\mathbf{P}}\left[\frac{d}{dt}[\mathbf{q}^{(L)}(t_{n-1,m},\mathbf{x}_{n-1,m}^{(L)})] + \mathbf{j}^{(L)}(t_{n-1,m},\mathbf{x}_{n-1,m}^{(L)})\right], \end{array}$$

where $\hat{\mathbf{P}} = \hat{\mathbf{B}}_{(A)}\hat{\mathbf{B}}_{(L)}^{T}\begin{bmatrix}\hat{\mathbf{B}}_{(L)} & \mathbf{O}\end{bmatrix}$. In a similar way as for $\mathbf{x}^{(L)}$ in (5.119) we can derive the following feedback law for $\mathbf{i}_{L\to A}$:

$$\mathbf{i}_{L\to A}(t) \approx \mathbf{b}(t) + \mathbf{B}(t)\mathbf{x}^{(L)}(t). \tag{5.121}$$

If we insert the formula for $\mathbf{x}^{(L)}(t)$ in (5.119) we can derive $\mathbf{c}(t)$ and $\mathbf{C}$ such that

$$\mathbf{i}_{L\to A}(t) \approx \mathbf{c}(t) + \mathbf{C}\mathbf{i}_{A\to L}(t).$$

Now we can reduce the system (5.117) even further to the following system for $\mathbf{i}_{A\to L}, \mathbf{x}^{(A)}$:

$$\left\{ \begin{array}{l} \mathbf{i}_{A\to L} = -\hat{\mathbf{B}}_{(L)}\hat{\mathbf{B}}_{(A)}^{T}\left[\frac{d}{dt}[\hat{\mathbf{q}}^{(A)}(t,\mathbf{x}^{(A)})] + \hat{\mathbf{j}}^{(A)}(t,\mathbf{x}^{(A)})\right], \\ \frac{d}{dt}[\mathbf{q}^{(A)}(t,\mathbf{x}^{(A)})] + \mathbf{j}^{(A)}(t,\mathbf{x}^{(A)}) = \mathbf{c}(t) + \mathbf{C}\mathbf{i}_{A\to L}. \end{array} \right. \tag{5.122}$$

Note that we can eliminate $\mathbf{i}_{A\to L}$, which results in the following system for $\mathbf{x}^{(A)}$

$$\frac{d}{dt}[\mathbf{q}^{(A)}(t,\mathbf{x}^{(A)})] + \mathbf{j}^{(A)}(t,\mathbf{x}^{(A)}) = \mathbf{c}(t) - \mathbf{C}\hat{\mathbf{B}}_{(L)}\hat{\mathbf{B}}_{(A)}^{T}\left[\frac{d}{dt}[\hat{\mathbf{q}}^{(A)}(t,\mathbf{x}^{(A)})] + \hat{\mathbf{j}}^{(A)}(t,\mathbf{x}^{(A)})\right]. \tag{5.123}$$

From its structure it can be seen that only the terminal active equations that are directly coupled to the latent part are modified. In fact they are multiplied by a linear transformation. This linear transformation is such that the dynamical behaviour of the original system has been preserved. Again, the vector-valued function $\mathbf{c}(t)$ is again an interpolation-based current source.

In a similar way as for $\mathbf{i}_{L\to A}$ we can derive the formula

$$\mathbf{i}_{A\to L}(t) \approx \mathbf{f}(t) + \mathbf{F}(t)\mathbf{x}^{(A)}(t). \tag{5.124}$$

Combining all three formulae (5.119), (5.121) and (5.124) enables us to express $\mathbf{i}_{L\to A}$ directly in terms of $\mathbf{x}^{(A)}$:

$$\mathbf{i}_{L\to A}(t) = \mathbf{g}(t) + \mathbf{G}\mathbf{x}^{(A)}(t).$$

Then we get the following system for $\mathbf{x}^{(A)}$:

$$\left\{ \; \frac{d}{dt}[\mathbf{q}^{(A)}(t,\mathbf{x}^{(A)})] + \mathbf{j}^{(A)}(t,\mathbf{x}^{(A)}) = \mathbf{g}(t) + \mathbf{G}`\mathbf{x}^{(A)}. \right. \tag{5.125}$$

Thus there are three interpolation variants from which the first one is the most accurate and the last one the easiest one to implement. All variants assume that the large slowest part is independently solvable, which is reasonable. Variant I also needs to evaluate all terminal equations for the slow models and solves all terminal currents, which leads to a second order system and can be expensive. But it can also be applied for fast terminal currents $\mathbf{i}_{L\to A}, \mathbf{i}_{A\to L}$. Variant II only needs to evaluate active elements but it still needs $\mathbf{i}_{A\to L}$ as additional unknown. Therefore it still can be applied for active $\mathbf{i}_{A\to L}$. Furthermore it is useful if the active subcircuit is irreducible. Variant III really reduces to a system for only the active part. It is only allowed if all terminal currents $\mathbf{i}_{L\to A}, \mathbf{i}_{A\to L}$ behave slowly.

For the third variant it is clear that $\mathbf{i}_{L\to A}$ is replaced by a combination of current sources and resistors. In fact this is model reduction of the large latent part. This improves the dynamical properties of the active circuit model. Hopefully now the solvability, stability and index should be preserved. Another important advantage is that it makes the compound step unnecessary. We only had to invert the Jacobian matrix of the slow part once. This means that the number of Newton iterations will be reduced.

From the previous variants the second variant is in particular very attractive. It allows a reducible active part with fast terminal currents. Furthermore, even problems with solvability and stability could be solved. Nevertheless it is less expensive than the first variant which has to evaluate all submodels connected to the terminal part.

For an implementation of interpolation variant II we have to do the following steps. First we compute $\mathbf{c}(t)$ by an integration of the slow part by a large step, where $\mathbf{i}_{A\to L} = \mathbf{0}$. The columns of the matrix $\mathbf{C}$ can be computed simultaneously based on $\mathbf{i}_{A\to L} = \mathbf{e}_i$ by re-use of the same LU factorisation of $\mathbf{J}$. Next, the refinement is performed for the modified active system. Each refinement step the errors of $\mathbf{x}^{(A)}$ and $\mathbf{i}_{A\to L}$ are monitored. Note that the error of $\mathbf{i}_{A\to L}$ determines the extrapolation error of the slow part. If the estimated error becomes too large, because of sudden wake-ups, we immediately can stop. At the end of the refinement, the slow variables $\mathbf{x}^{(L)}, \mathbf{i}_{L\to A}$ are updated using the implicit formulae. Note that this formula can also be used if the refinement was stopped earlier. Clearly sudden wake-ups are very well detected now. We are also able now to estimate the error of the latent part. If it is acceptable small, we can repartition and go to the next step. Otherwise, we could do another Newton iteration or reduce the compound step.

Chapter 6

Adaptive multirate stepsize control and partitioning

6.1 Analysis of the local discretisation error

The accuracy of a multirate method can be controlled by the stepsizes of the compound step and the refinement phase. In this section we will show how H_n and $h_{n-1,m}$ can be controlled such that the local error is smaller than a given tolerance level. We will also analyse how the local discretisation errors at the coarse and fine grid asymptotically behave. Finally an error model is constructed which is used to develop stepsize controllers for the coarse and fine time-grids. We generalise some techniques in [49, 51] to our multirate case [68]. The local discretisation error $\mathbf{d}^n$ is defined as the residual of the scheme at the coarse time-grid after inserting the exact solution. It still has the familiar behaviour $\mathbf{d}^n = O(H_n^{K+1})$. In practice the local discretisation errors are not known and should be estimated up to a sufficient accuracy. The local discretisation error $\mathbf{d}^n$ can be estimated by $\hat{\mathbf{d}}^n$ using the Nordsieck representation of the predictor and corrector polynomial of $\mathbf{q}$ (at $t = [T_{n-1}, T_n]$):

$$\hat{\mathbf{d}}^n := \frac{-H_n}{T_n - T_{n-K-1}} \left[\bar{\mathbf{q}}_1^n - \bar{\mathbf{p}}_1^n\right]. \tag{6.1}$$

Now

$$\hat{r}_C^n := \|\mathbf{B}_L \hat{\mathbf{d}}^n\| + \tau \|\mathbf{B}_A \hat{\mathbf{d}}^n\| \tag{6.2}$$

is the used weighted error norm for the coarse grid, which must satisfy $\hat{r}_C^n < \mathrm{TOL}_C$. Here $\tau \geq 0$ is a small non-negative *relaxation number* which must improve the convergence of the compound step. The optimal value is not known yet, but in our experiments even $\tau = 0$ worked satisfactorily.

For multirate methods we also need to consider the local discretisation error $\mathbf{d}^{n,m}$ at the fine time-grid. At the fine time-grid the DAE has been perturbed by the interpolated latent variables at the interface between latent and active areas. The local discretisation error at the fine time-grid equals there

$$\mathbf{d}^{n,m} = \bar{\rho}_0^{n-1,m}\mathbf{q}_A(t_{n-1,m}, \mathbf{x}_A(.), \mathbf{x}_L(.)) + h_{n-1,m}\mathbf{j}_A(t_{n-1,m}, \mathbf{x}_A(.), \mathbf{x}_L(.)) + \mathbf{b}^{n-1,m},$$

where $\mathbf{b}^{n-1,m}$ contains the time history from the multistep method

$$\mathbf{b}^{n-1,m} = \bar{\rho}_1^{n-1,m}\mathbf{q}_A(t_{n-1,m-1}, \mathbf{x}_A(.), \mathbf{x}_L(.)) + \ldots + \bar{\rho}_k^{n-1,m}\mathbf{q}_A(t_{n-1,m-k}, \mathbf{x}_A(.), \mathbf{x}_L(.)).$$

We also consider the perturbed local discretisation error $\tilde{\mathbf{d}}^{n,m}$, which is the residual of the refinement scheme with fixed interpolated slow part after inserting the exact active solution. Clearly, $\tilde{\mathbf{d}}^{n,m}$ satisfies

$$\tilde{\mathbf{d}}^{n,m} = \bar{\rho}_0^{n-1,m}\mathbf{q}_A(t_{n-1,m}, \mathbf{x}_A(.), \hat{\mathbf{x}}_L^{n-1,m}) + h_{n-1,m}\mathbf{j}_A(t_{n-1,m}, \mathbf{x}_A(.), \hat{\mathbf{x}}_L^{n-1,m}) + \tilde{\mathbf{b}}^{n-1,m},$$

where

$$\begin{aligned} \tilde{\mathbf{b}}^{n-1,m} &= \bar{\rho}_1^{n-1,m}\mathbf{q}_A(t_{n-1,m-1}, \mathbf{x}_A(.), \hat{\mathbf{x}}_L^{n-1,m-1}) + \ldots \\ &+ \bar{\rho}_k^{n-1,m}\mathbf{q}_A(t_{n-1,m-k}, \mathbf{x}_A(.), \hat{\mathbf{x}}_L^{n-1,m-k}). \end{aligned}$$

The perturbed local discretisation error $\mathbf{B}_A\tilde{\mathbf{d}}^{n-1,m}$ behaves as $O(h_{n-1,m}^{k+1})$ and can be estimated in a similar way as $\mathbf{d}^n$ from the perturbed Nordsieck matrices of the refinement phase.

$$\mathbf{B}_A\hat{\tilde{\mathbf{d}}}^{n,m} := \frac{-h_{n-1,m}}{t_{n-1,m} - t_{n-1,m-k-1}}\mathbf{B}_A\left[\bar{\mathbf{q}}_1^{n-1,m} - \bar{\mathbf{p}}_1^{n-1,m}\right]. \tag{6.3}$$

The norm of $\|\mathbf{B}_A\hat{\tilde{\mathbf{d}}}^{n,m}\|$ will be denoted by

$$\hat{r}_A^{n-1,m} := \|\mathbf{B}_A\hat{\tilde{\mathbf{d}}}^{n,m}\|. \tag{6.4}$$

For single-rate methods the accuracy can completely be controlled by controlling the local discretisation errors. This is no longer the case for multirate methods, where also the *interpolation errors* play an important role. Let $\hat{\mathbf{x}}_L(t)$ be the interpolation-based function which is exact at the coarse or fine time-grid. Then the interpolation errors $\mathbf{r}^n$ and $\mathbf{r}^{n-1,m}$ are defined as the maximum errors of $\hat{\mathbf{x}}_L(t)$ between the time-points of the coarse and fine time-grids on the interval $[T_{n-1},\ T_n]$ and $[t_{n-1,m-1},\ t_{n-1,m}]$, respectively. If the interpolation order is equal to the integration order, they again have the familiar behaviour $\mathbf{r}^n = O(H_n^{K+1}), \mathbf{r}^{n-1,m} = O(h_{n-1,m}^{k+1})$.

Lemma 6.1 *For the interpolation errors* $\boldsymbol{r}^n, \boldsymbol{r}^{n-1,m}$ *we have*

$$\|\boldsymbol{r}^n\| \leq \|\hat{\boldsymbol{r}}^n\|, \quad \|\boldsymbol{r}^{n-1,m}\| \leq \|\hat{\boldsymbol{r}}^{n-1,m}\|, \tag{6.5}$$

where

$$\hat{\boldsymbol{r}}^n := \frac{1}{4}\frac{-H_n}{T_n - T_{n-K-1}}(\bar{\boldsymbol{x}}_1^n - \bar{\boldsymbol{y}}_1^n), \quad \hat{\boldsymbol{r}}^{n-1,m} := \frac{1}{4}\frac{-h_{n-1,m}}{t_{n-1,m} - t_{n-1,m-k-1}}(\bar{\boldsymbol{x}}_1^{n-1,m} - \bar{\boldsymbol{y}}_1^{n-1,m}). \tag{6.6}$$

Proof: Let $\mathbf{x}(t)$ be the exact solution and $\hat{\mathbf{x}}$ a polynomial of degree K that interpolates $\mathbf{x}$ at the previous $K+1$ time-points $\{T_{n-K}, \dots, T_n\}$. At the coarse time-grid we have the following asymptotic behaviour

$$\begin{aligned} \mathbf{x}(t) - \hat{\mathbf{x}}(t) &= (t - T_{n-K}) \cdots (t - T_{n-1})(t - T_n) \frac{\mathbf{x}^{(K+1)}(\tau)}{(K+1)!}, \quad \tau \in (T_{n-K}, T_n) \\ &= (t - T_{n-K}) \cdots (t - T_{n-1})(t - T_n) \frac{\mathbf{x}^{(K+1)}(T_n)}{(K+1)!} + O(H_n^{K+2}). \end{aligned} \tag{6.7}$$

We easily derive the upper bound $\max_{t \in [T_{n-1}, T_n]} \|\mathbf{x}(t) - \hat{\mathbf{x}}(t)\|$ for all $t \in [T_{n-1}, T_n]$, which satisfies up to $O(H_n^{K+2})$

$$\begin{aligned} \max_{t \in [T_{n-1}, T_n]} \|\mathbf{x}(t) - \hat{\mathbf{x}}(t)\| &\leq \textstyle\prod_{j=2}^{K} (T_n - T_{n-j}) \max_{t \in [T_{n-1}, T_n]} |(t - T_{n-1})(t - T_n)| \, \|\frac{\mathbf{x}^{(K+1)}(T_n)}{(K+1)!}\| \\ &= \tfrac{1}{4} \textstyle\prod_{j=2}^{K} (T_n - T_{n-j})(T_n - T_{n-1})^2 \|\frac{\mathbf{x}^{(K+1)}(T_n)}{(K+1)!}\| \\ &= \|\tfrac{1}{4} H_n^2 (H_{n-1} + H_n) \cdots (H_{n-K+1} + \cdots + H_n) \frac{\mathbf{x}^{(K+1)}(T_n)}{(K+1)!}\|. \end{aligned} \tag{6.8}$$

Consider $\bar{\mathbf{y}}(t)$ which interpolates $\hat{\mathbf{x}}$ (and $\mathbf{x}$) at the previous K+1 time-points $\{T_{n-K-1}, \dots, T_{n-1}\}$. Thus we have

$$\hat{\mathbf{x}}(t) - \hat{\mathbf{y}}(t) = (t - T_{n-K-1}) \cdots (t - T_{n-1}) \frac{\mathbf{x}^{(K+1)}(T_n)}{(K+1)!} + O(H_n^{K+2}).$$

In particular we obtain

$$\begin{aligned} \bar{\mathbf{x}}_1^n - \bar{\mathbf{y}}_1^n &= \hat{\mathbf{x}}(T_n) - \hat{\mathbf{y}}(T_n) = (T_n - T_{n-K-1}) \cdots (T_n - T_{n-1}) \frac{\mathbf{x}^{(K+1)}(T_n)}{(K+1)!} + O(H_n^{K+2}) \\ &= (H_{n-K} + \cdots + H_n) \cdots (H_{n-1} + H_n) H_n \frac{\mathbf{x}^{(K+1)}(T_n)}{(K+1)!} + O(H_n^{K+2}). \end{aligned}$$

Because

$$\|\frac{1}{4} H_n^2 (H_{n-1} + H_n) \cdots (H_{n-K+1} + \cdots + H_n) \frac{\mathbf{x}^{(K+1)}(T_n)}{(K+1)!}\| = \frac{H_n}{4(T_n - T_{n-K-1})} \|\bar{\mathbf{x}}_1^n - \bar{\mathbf{y}}_1^n\| + O(H_n^{K+2}),$$

it follows that

$$\max_{t \in [T_{n-1}, T_n]} \|\mathbf{x}(t) - \hat{\mathbf{x}}(t)\| \leq \frac{H_n}{4(T_n - T_{n-K-1})} \|\bar{\mathbf{x}}_1^n - \bar{\mathbf{y}}_1^n\|. \tag{6.9}$$

Now it follows indeed that $\|\mathbf{r}^n\| \leq \|\hat{\mathbf{r}}^n\|$ if $\hat{\mathbf{r}}^n = \frac{1}{4} \frac{-H_n}{T_n - T_{n-K-1}} (\bar{\mathbf{x}}_1^n - \bar{\mathbf{y}}_1^n)$. At the fine time-grid we can prove in a similar way that also $\|\mathbf{r}^{n-1,m}\| \leq \|\hat{\mathbf{r}}^{n-1,m}\|$, where $\hat{\mathbf{r}}^{n-1,m}$ is given in (6.6). □

Before we state the following theorem, we introduce the notion of the *coupling matrix*.

Definition 6.2 *The coupling matrix* $\boldsymbol{K}_{n-1,m} \in \mathbb{R}^{d_A \times d_L}$ *is defined by*

$$\boldsymbol{K}_{n-1,m} := \boldsymbol{B}_A \frac{d}{dt} \left[\boldsymbol{C}(t_{n-1,m}, \boldsymbol{x}(t_{n-1,m})) \right] \boldsymbol{B}_L^T + \boldsymbol{B}_A \boldsymbol{G}(t_{n-1,m}, \boldsymbol{x}(t_{n-1,m})) \boldsymbol{B}_L^T, \tag{6.10}$$

where $\boldsymbol{C}(t, \boldsymbol{x}) = \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{x}}(t, \boldsymbol{x})$ *and* $\boldsymbol{G}(t, \boldsymbol{x}) = \frac{\partial \boldsymbol{j}}{\partial \boldsymbol{x}}(t, \boldsymbol{x})$.

This coupling matrix will turn out to be essential for the error analysis.

Theorem 6.3 *The active part of* $\boldsymbol{d}^{n-1,m}$ *satisfies*

$$\|\boldsymbol{B}_A \boldsymbol{d}^{n-1,m}\| \leq \|\boldsymbol{B}_A \tilde{\boldsymbol{d}}^{n-1,m}\| + h_{n-1,m} \|\boldsymbol{K}_{n-1,m}\| \|\boldsymbol{B}_L \boldsymbol{r}^{n-1,m}\|. \tag{6.11}$$

Proof: From now on we use the abbreviations $t = t_{n-1,m}$, $h = h_{n-1,m}$, $\bar{\rho}_0 = \bar{\rho}_0^{n-1,m}$, $\tilde{\mathbf{b}} = \tilde{\mathbf{b}}_{n-1,m}$, $\mathbf{b} = \mathbf{b}_{n-1,m}$. For the error difference $\Delta \mathbf{d}^{n-1,m} = \mathbf{d}^{n-1,m} - \tilde{\mathbf{d}}^{n-1,m}$ we have

$$\begin{array}{rcl} \|\Delta \mathbf{d}^{n-1,m}\| & = & \|\bar{\rho}_0 (\mathbf{q}_A(t, \mathbf{x}_A(t), \mathbf{x}_L(t)) - \mathbf{q}_A(t, \mathbf{x}_A(t), \hat{\mathbf{x}}_L^{n-1,m})) \\ & + & h(\mathbf{j}_A(t, \mathbf{x}_A(t), \mathbf{x}_L(t)) - \mathbf{j}_A(t, \mathbf{x}_A(t), \hat{\mathbf{x}}_L^{n-1,m})) + \mathbf{b} - \tilde{\mathbf{b}}\| \\ & \leq & \|\bar{\rho}_0 \frac{\partial \mathbf{q}_A}{\partial \mathbf{x}_L} + h \frac{\partial \mathbf{j}_A}{\partial \mathbf{x}_L} + \frac{\partial \mathbf{b}_A}{\partial \mathbf{x}_L}\| \ \max_{0 \leq j \leq K} \|\hat{\mathbf{x}}_L^{n-1,m-j} - \mathbf{x}_L(t_{n-1,m-j})\|. \end{array} \tag{6.12}$$

Here we assumed that

$$\|\tilde{\mathbf{b}} - \mathbf{b}\| \leq \|\frac{\partial \mathbf{b}}{\partial \mathbf{x}_L}\| \max_{0 \leq j \leq K} \|\hat{\mathbf{x}}_L^{n-1,m-j} - \mathbf{x}_L(t_{n-1,m-j}))\|. \tag{6.13}$$

Note that $\mathbf{K}_{n-1,m}$ satisfies

$$\mathbf{K}_{n-1,m} = \frac{\bar{\rho}_0}{h} \frac{\partial \mathbf{q}_A}{\partial \mathbf{x}_L}(t, \mathbf{x}_A(t), \mathbf{x}_L(t)) + \frac{\partial \mathbf{j}_A}{\partial \mathbf{x}_L}(t, \mathbf{x}_A(t), \mathbf{x}_L(t)) + \frac{1}{h} \frac{\partial \mathbf{b}_A}{\partial \mathbf{x}_L}(t, \mathbf{x}_A(t), \mathbf{x}_L(t)) + O(H_n). \tag{6.14}$$

Because of Lemma 6.1 we get

$$\|\Delta \mathbf{d}^{n-1,m}\| \leq h \|\mathbf{K}_{n-1,m}\| \|\mathbf{B}_L \mathbf{r}^{n-1,m}\| + \text{h.o.t.}. \tag{6.15}$$

Since $\mathbf{B}_A \mathbf{d}^{n-1,m} = \mathbf{B}_A \tilde{\mathbf{d}}^{n-1,m} + \Delta \mathbf{d}^{n-1,m}$ it immediately follows from (6.15) that (6.11) is fulfilled if higher order terms are neglected. □

Theorem 6.3 states that the interpolation error because of the interface equals $\|\mathbf{K}_{n-1,m}\| \|\mathbf{B}_L \mathbf{r}^{n-1,m}\|$. An estimated upper bound at the coarse time-grid is given by

$$\hat{r}_I^n := \|\hat{\mathbf{K}}_n\| \|\mathbf{B}_L \hat{\mathbf{r}}^n\|. \tag{6.16}$$

Here the coupling matrix $\mathbf{K}$ can be discretised at the coarse time-grid as follows

$$\mathbf{K}_{n-1,m} \doteq \frac{1}{H_n} \mathbf{B}_A \left[\mathbf{C}(T_n, \mathbf{x}^n) - \mathbf{C}(T_{n-1}, \mathbf{x}^{n-1})\right] \mathbf{B}_L^T + \mathbf{B}_A \mathbf{G}(T_n, \mathbf{x}^n) \mathbf{B}_L^T =: \hat{\mathbf{K}}_n. \tag{6.17}$$

Now $\|\mathbf{B}_A \mathbf{d}^{n-1,m}\|$ can be bounded by

$$\hat{r}_A^{n-1,m} := \hat{\hat{r}}_A^{n-1,m} + h_{n-1,m} \hat{r}_I^n, \tag{6.18}$$

where $\hat{\hat{r}}_A^{n-1,m}$ is an upperbound of $\|\mathbf{B}_A \tilde{\mathbf{d}}^{n-1,m}\|$.
It is also possible to work with other error definitions than the local discretisation error, like the local scaled error or the interpolation error. For all types we have $\hat{r}_C^n = O(H_n^{K+1})$ and $\hat{r}_A^{n-1,m} = O(h_{n-1,m}^{k+1})$ can be estimated and controlled by the already existing error

Table 6.1: Multirate error estimates for several types of error estimation.

$\hat{r}$	LDE	scaled LDE
$\hat{r}_C^n$	$\lVert\mathbf{B}_L\hat{\mathbf{d}}^n\rVert + \tau\lVert\mathbf{B}_A\hat{\mathbf{d}}^n\rVert$	$\lVert\mathbf{B}_L\hat{\mathbf{e}}^n\rVert + \tau\lVert\mathbf{B}_A\hat{\mathbf{e}}^n\rVert$
$\hat{r}_I^n$	$\lVert\hat{\mathbf{K}}_n\rVert\lVert\mathbf{B}_L\hat{\mathbf{r}}^n\rVert$	$\lVert[\mathbf{B}_A\mathbf{J}_n\mathbf{B}_A^T]^{-1}\hat{\mathbf{K}}_n\rVert\lVert\mathbf{B}_L\hat{\mathbf{r}}^n\rVert$
$\hat{\tilde{r}}_A^{n-1,m}$	$\lVert\mathbf{B}_A\hat{\tilde{\mathbf{d}}}^{n-1,m}\rVert$	$\lVert\mathbf{B}_A\hat{\tilde{\mathbf{e}}}^{n-1,m}\rVert$

control mechanism of the normal transient simulation. However, for the interpolation error $\hat{r}_I^n$ we have to analyse the equation (6.11) more profoundly. Recall that the local discretisation error satisfies

$$\lVert\mathbf{B}_A\mathbf{d}^{n-1,m}\rVert \leq \lVert\mathbf{B}_A\tilde{\mathbf{d}}^{n-1,m}\rVert + h_{n-1,m}\lVert\mathbf{K}_{n-1,m}\rVert\lVert\mathbf{B}_L\mathbf{r}^{n-1,m}\rVert.$$

Introducing

$$\mathbf{B}_A\tilde{\mathbf{e}}^{n-1,m} := [\mathbf{B}_A\mathbf{J}_{n-1,m}\mathbf{B}_A^T]^{-1}\mathbf{B}_A\tilde{\mathbf{d}}^{n-1,m} \tag{6.19}$$

it is easy to prove that

$$\lVert\mathbf{B}_A\mathbf{e}^{n-1,m}\rVert \leq \lVert\mathbf{B}_A\tilde{\mathbf{e}}^{n-1,m}\rVert + h_{n-1,m}\lVert[\mathbf{B}_A\mathbf{J}_{n-1,m}\mathbf{B}_A^T]^{-1}\mathbf{K}_{n-1,m}\rVert\lVert\mathbf{B}_L\mathbf{r}^{n-1,m}\rVert. \tag{6.20}$$

Because of the hierarchical structure of circuit models the original circuit is always solvable, which implies that $\mathbf{J}_n$ is invertible. Here we assume that $\mathbf{B}_A\mathbf{J}_{n-1,m}\mathbf{B}_A^T$ is an invertible matrix that can be approximated by $\mathbf{B}_A\mathbf{J}_n\mathbf{B}_A^T$, which is only the case if the active part is a solvable system. If $\mathbf{J}$ is symmetric positive definite this is always the case of course. Conditions for this property have been given in section 5.4. Table 6.1 shows the error estimates for the investigated error types.

6.2 Adaptive stepsize and order control

Adaptive stepsize control of H_n and $h_{n,m}$ can be used to keep $\hat{r}_I^n = O(H_n^{K+1})$ (6.16) and $\hat{\tilde{r}}_A^{n-1,m} = O(h_{n-1,m}^{k+1})$ (6.4) close to $\theta\mathrm{TOL}_I$ and $\theta\mathrm{T\tilde{O}L}_A$ respectively, where $0 < \theta < 1$ is a safety factor. The *models of these error estimates* can be found in (6.16) and (6.4), respectively.

The local discretisation error of a Compound-Fast multirate method for a fixed partitioning can be controlled by independent control of the compound step and the refinement phase. The purpose of the control is that both the slow and fast errors r_L^n and $r_A^{n-1,m}$ respectively are always smaller than a given tolerance level TOL. For the latent part we just have

$$r_L^n = \lVert\mathbf{B}_L\mathbf{d}^n\rVert = O(H_n^{K+1}),$$

where $\mathbf{d}^n$ is the vector of local discretisation errors per element at T_n and $\lVert.\rVert = \lVert.\rVert_\infty$. In [68] it has been shown that the active local error at $t_{n-1,m}$ can be bounded by

$$r_A^{n-1,m} = \lVert\mathbf{B}_A\mathbf{d}^{n-1,m}\rVert \leq \lVert\mathbf{B}_A\tilde{\mathbf{d}}^{n-1,m}\rVert + h_{n-1,m}\lVert\mathbf{B}_A\mathbf{K}_{n-1,m}\mathbf{B}_L^T\rVert\lVert\mathbf{B}_L\mathbf{r}^{n-1,m}\rVert. \tag{6.21}$$

Here $\mathbf{B}_A\tilde{\mathbf{d}}^{n-1,m}$ is the usual local discretisation error of the active part (i.e. under the assumption of no interpolation errors). The following term comprises the error due to interpolation. The vector $\mathbf{B}_L\mathbf{r}^{n-1,m}$ is the interpolation error of the slow part and $\mathbf{B}_A\mathbf{K}_{n-1,m}\mathbf{B}_L^T$ the coupling matrix, where

$$\mathbf{K}_{n-1,m} = \frac{d}{dt}\left[\mathbf{C}(t_{n-1,m}, \mathbf{x}(t_{n-1,m}))\right] + \mathbf{G}(t_{n-1,m}, \mathbf{x}(t_{n-1,m})),$$

which involves the solution also at the latent part. Thus the interpolation error at the interface, $h_{n-1,m}\|\mathbf{B}_A\mathbf{K}_{n-1,m}\mathbf{B}_L^T\|\|\mathbf{B}_L\mathbf{r}^{n-1,m}\|$, cannot be controlled by $h_{n-1,m}$ completely. Therefore we approximate it at the coarse time-grid as follows

$$h_{n-1,m}\|\mathbf{B}_A\mathbf{K}_{n-1,m}\mathbf{B}_L^T\|\|\mathbf{B}_L\mathbf{r}^{n-1,m}\| \approx h_{\max}\|\mathbf{B}_A\mathbf{K}_n\mathbf{B}_L^T\|\|\mathbf{B}_L\mathbf{r}^n\|,$$

which only depends on H_n. If

$$\begin{aligned} r_I^n := h_{\max}\|\mathbf{B}_A\mathbf{K}_n\mathbf{B}_L^T\|\|\mathbf{B}_L\mathbf{r}^n\| &\leq wTOL, &(6.22)\\ \tilde{r}_A^{n-1,m} := \|\mathbf{B}_A\tilde{\mathbf{d}}^{n-1,m}\| &\leq (1-w)TOL, &(6.23)\end{aligned}$$

it immediately follows from (6.21) that $r_A^{n-1,m} \leq$TOL. Section 6.3 will show how $w \in (0,1)$ can be optimized [68] with respect to the default value 0.5. Then we get the following conditions for H, h for the Compound-Fast method

$$\begin{aligned} r_C^n := \max\{r_L^n, \frac{1}{w}r_I^n\} &\leq TOL, &(6.24)\\ r_R^{n-1,m} := \frac{1}{1-w}\tilde{r}_A^{n-1,m} &\leq TOL. &(6.25)\end{aligned}$$

For hierarchical circuit models with current interpolation it is not necessary to compute the coupling matrix $\mathbf{K}_n$ because the terminal currents for each subcircuit can be explicitly stored in Nordsieck arrays. Then we can write

$$r_I^n = h_{\max}\|\tilde{\mathbf{B}}_{(A)}\tilde{\mathbf{B}}_{(L)}^T\|\|\mathbf{B}_L\mathbf{r}^n\|. \tag{6.26}$$

In Pstar we have $\hat{\mathbf{d}}^n$ and $\hat{\mathbf{r}}^n$ are estimated by the interpolation errors at the coarse time-grid of the voltages and terminal currents, respectively. Hence estimates (indicated by the hats) are available.

For the Slow-Fast method, without compound step, we get a different model for r_L^n because of the extrapolation errors. Then we need the extended model:

$$r_L^n \approx \|\mathbf{B}_L\tilde{\mathbf{d}}^n\| + H_{\max}\|\mathbf{B}_L\mathbf{K}_{n-1,m}\mathbf{B}_A^T\|\|\mathbf{B}_A\mathbf{r}^{n-1,m}\|.$$

Here $\mathbf{B}_L\tilde{\mathbf{d}}^n$ is the local discretisation error of the latent part under the assumption that there are no extrapolation errors. Furthermore $H_{\max}\|\mathbf{B}_L\mathbf{K}_{n-1,m}\mathbf{B}_A^T\|\|\mathbf{B}_A\mathbf{r}^{n-1,m}\|$ only depends on $h_{n-1,m}$ and can be controlled in the previous refinement phase. In a similar way, if

$$\begin{aligned} r_I^{n-1,m} := H_{\max}\|\mathbf{B}_L\mathbf{K}_{n-1,m}\mathbf{B}_A^T\|\|\mathbf{B}_A\mathbf{r}^{n-1,m}\| &\leq wTOL, &(6.27)\\ \tilde{r}_L^n := \|\mathbf{B}_L\tilde{\mathbf{d}}^n\| &\leq (1-w)TOL, &(6.28)\end{aligned}$$

we get $r_L^n \leq$TOL. Now the term $H_{\max}\|\mathbf{B}_L \mathbf{K}_{n-1,m} \mathbf{B}_A^T\|\|\|\mathbf{B}_A \mathbf{r}^{n-1,m}\|\|\|$ can be controlled by $h^{n-1,m}$ completely during the previous refinement. For this method we get the following conditions for $H_n, h_{n-1,m}$

$$r_C^n := \max\{\frac{1}{1-w}\tilde{r}_L^n, \frac{1}{w}r_I^n\} \quad \leq \quad \text{TOL}, \tag{6.29}$$

$$r_R^{n-1,m} := \max\{\frac{1}{w}r_I^{n-1,m}, \frac{1}{1-w}\tilde{r}_A^{n-1,m}\} \quad \leq \quad \text{TOL}. \tag{6.30}$$

Clearly, for both the Compound-Fast and Slow-Fast multirate methods, it is possible to control the local errors by independent control of $r_C^n = O(H_n^{K+1})$ and $r_R^{n-1,m} = O(h_{n-1,m}^{k+1})$ as shown in the previous section. The stepsizes $H_{n+1}, h_{n,1}$ could e.g. be computed by elementary multirate stepsize controllers like

$$H_{n+1} \quad = \quad \left(\frac{\theta \text{TOL}}{\hat{r}_C^n}\right)^{\frac{1}{K+1}} H_n, \tag{6.31}$$

$$h_{n-1,m+1} \quad = \quad \left(\frac{\theta \text{TOL}}{\hat{r}_R^{n-1,m}}\right)^{\frac{1}{k+1}} h_{n-1,m}. \tag{6.32}$$

Here, $\theta \in (0,1)$ is a safety factor which reduces the number of rejected timesteps. The steps are used to track both $\hat{r}_C^n$ and $\hat{r}_R^{n-1,m}$ close to θTOL.
Because the last fine step $h_{n-1,q}$ is clipped for synchronisation reasons, the error estimate $\hat{r}_A^{n-1,q}$ can become very small. Therefore at the synchronisation time-point T_n $h_{n,1}$ is chosen as the unclipped $h_{n-1,q}$ which depends on $\hat{r}^{n-1,q-1}$.

6.3 Optimal value for the balance number

It appears that the *balance number* w in (6.22) can be chosen in an optimal way. Theorem 6.5 shows how the *optimal balance number* $w_{\text{opt}} \in (0,1)$. To prove that we need the following Lemma.

Lemma 6.4 *Let* $f \in \mathcal{C}^\infty((0,1), \mathbb{R}^+)$ *such that*

$$\forall x \in (0,1) \ \ f(x) = A\ (1-x)^{-\frac{1}{r}} + B\ x^{-\frac{1}{s}},$$

where $A, B \in \mathbb{R}^+$ *and* $r, s \in \mathbb{N}$. *Then*

$$\forall x \in (0,1) \ \ f(x) \geq f(x_*),$$

where $x_* \in (0,1)$ *is the unique solution of*

$$A\ s\ x_*^{1+\frac{1}{s}} = B\ r\ (1-x_*)^{1+\frac{1}{r}}.$$

If $r = s$ *one explicitly has*

$$x_* = \frac{1}{1+\left(\frac{A}{B}\right)^{\frac{r}{1+r}}} \in (0,1),$$

such that

$$\forall x \in (0,1) \;\; f(x) \geq A \left(1 + \left(\frac{A}{B}\right)^{\frac{-r}{1+r}}\right)^{\frac{1}{r}} + B \left(1 + \left(\frac{A}{B}\right)^{\frac{r}{1+r}}\right)^{\frac{1}{r}}. \tag{6.33}$$

Proof: Because $f \in \mathcal{C}^{\infty}((0,1), \mathbb{R}^{+})$ and is unbounded for $x \in \{0,1\}$, f has at least one minimum $x_* \in (0,1)$ that satisfies $f'(x_*) = 0$. It can be proved that this minimum is unique. Because

$$f'(x) = \frac{A\ s\ x^{1+\frac{1}{s}} - B\ r\ (1-x)^{1+\frac{1}{r}}}{r\ s\ x^{1+\frac{1}{s}}(1-x)^{1+\frac{1}{r}}},$$

we get the following equation for x_*:

$$A\ s\ x_*^{1+\frac{1}{s}} = B\ r(1-x_*)^{1+\frac{1}{r}}.$$

If $r = s$ it is easily derived that

$$\frac{x_*}{1-x_*} = \left(\frac{B}{A}\right)^{\frac{r}{r+1}}, \text{ or } x_* = \frac{\left(\frac{B}{A}\right)^{\frac{r}{r+1}}}{1+\left(\frac{B}{A}\right)^{\frac{r}{r+1}}} = \frac{1}{1+\left(\frac{A}{B}\right)^{\frac{r}{r+1}}}.$$

Then indeed (6.33) follows immediately. □

In each multirate step we need the computational workload

$$W_{\text{mult}} = W_R \frac{T}{h} + W_C \frac{T}{H}, \tag{6.34}$$

where W_C, W_R are the computational work per timestep for the compound phase and the refinement phase respectively. This definition will also be used in the efficiency analysis in section 6.4. Because of (6.23) and (6.22) we have for $w < w_{\max}$

$$h = \left(\frac{(1-w)\theta \mathrm{TOL}_A}{\hat{\tilde{\phi}}_A}\right)^{\frac{1}{k+1}} = D_R(1-w)^{\frac{-1}{k+1}}, \quad H = \left(\frac{w\ \theta \mathrm{TOL}_A}{h_{\max}\hat{\phi}_I}\right)^{\frac{1}{K+1}} = D_C w^{\frac{-1}{K+1}}, \tag{6.35}$$

where

$$D_R = \left(\frac{\theta \mathrm{TOL}_A}{\hat{\tilde{\phi}}_A}\right)^{\frac{1}{k+1}}, \quad D_C = \left(\frac{\theta \mathrm{TOL}_A}{h_{\max}\hat{\phi}_I}\right)^{\frac{1}{K+1}}. \tag{6.36}$$

Theorem 6.5 *Let w_{max} be defined by*

$$w_{max} := \frac{TOL_C}{TOL_A} \frac{h_{max}\hat{r}_I^n}{\hat{r}_C^n} \tag{6.37}$$

and assume that $w_* \in (0,1)$ solves

$$G(K+1)w_*^{1+\frac{1}{K+1}} = (k+1)(1-w_*)^{1+\frac{1}{k+1}}, \tag{6.38}$$

where $G := \frac{W_R}{W_C}\frac{D_C}{D_R}$ *and* D_R, D_C *are given in (6.36). Then the optimal value for* $w \in (0,1)$ *equals*

$$w_{opt} = \min\{w_{max}, w_*\}. \tag{6.39}$$

Proof: There exists a number w_{max} such that for $w \leq w_{\mathrm{max}}$ the first constraint (6.22) becomes dominant. This means that w always must be smaller than w_{max} because otherwise the tolerance level $\tilde{\mathrm{TOL}}_A = (1-w)\mathrm{TOL}_A$ for the refinement becomes too small. This value w_{max} can be determined by the property

$$\frac{\mathrm{TOL}_C}{\hat{r}_C^n} = \frac{w_{\mathrm{max}}\mathrm{TOL}_A}{h_{\mathrm{max}}\hat{r}_I^n},$$

which indeed implies (6.37).
If $w \leq w_{\mathrm{max}}$ the computational workload $W_{\mathrm{mult}} = W_{\mathrm{mult}}(w)$ in (6.34) satisfies

$$W_{\mathrm{mult}}(w) = \frac{W_R}{D_R}T(1-w)^{\frac{-1}{k+1}} + \frac{W_C}{D_C}Tw^{\frac{-1}{K+1}}, \tag{6.40}$$

where D_R, D_C are given in (6.36). Clearly, W_{mult} is unbounded for $w = 0$ or $w = 1$, which implies that $w \in (0,1)$. We apply Lemma 6.4 to (6.40) in order to find the optimal balance number w_*. Thus it follows for W_{mult}, where $A = \frac{W_R}{D_R}, B = \frac{W_C}{D_C}$ and $r = k+1, s = K+1$ that in $(0, w_{\mathrm{max}})$ the optimal value w_* indeed solves (6.38) with $G = \frac{W_R}{W_C}\frac{D_C}{D_R}$. ☐

If $K = k$ we can compute the analytical solution of (6.38), which is equal to

$$w_* = \frac{1}{1 + G^{\frac{K+1}{K+2}}} \in (0,1), \tag{6.41}$$

with $G = \frac{W_R}{W_C}\frac{D_C}{D_R}$. From this expression we can conclude that the optimal value w_* becomes very small if $G \gg 1$. This is the case if the multirate factor $q \gg 1$ and $\hat{\tilde{r}}_A \gg h_{\mathrm{max}}\hat{r}_I$ which is satisfied if the active part behaves much faster and is nearly decoupled. Note that an upper bound for H disturbs this optimality because then the steps do not satisfy (6.35). If H has to be limited it is always better to decrease w such that the limited H exactly fits for our case. Then the refinement can be done with larger steps.
It is important that $w \leq w_{\mathrm{max}}$ because for $w > w_{\mathrm{max}}$ the compound step H is determined by determined by $\frac{\mathrm{TOL}_C}{\hat{r}_C^n}$ and stays constant, while the small refinement steps h are determined by $\tilde{\mathrm{TOL}}_A = (1-w)\mathrm{TOL}_A$ and become smaller. For $K = k$ and $w \leq w_{\mathrm{max}}$ the optimized workload equals

$$\begin{aligned} W_{\mathrm{mult}}(w_*) &= \tfrac{W_C}{D_C}T\left(w_*^{\frac{-1}{k+1}} + G(1-w_*)^{\frac{-1}{k+1}}\right) \\ &= \tfrac{W_C}{D_C}T\left(\left(\frac{1}{1+G^{\frac{k+1}{k+2}}}\right)^{\frac{-1}{k+1}} + G\left(1 - \frac{1}{1+G^{\frac{k+1}{k+2}}}\right)^{\frac{-1}{k+1}}\right). \end{aligned} \tag{6.42}$$

Then we get the gain with respect to e.g. $w = 0.5$

$$
\begin{aligned}
\frac{W_{\text{mult}}(w_*)}{W_{\text{mult}}(0.5)} &= \frac{\left(\frac{1}{1+G^{\frac{k+1}{k+2}}}\right)^{\frac{-1}{k+1}} + G\left(1 - \frac{1}{1+G^{\frac{k+1}{k+2}}}\right)^{\frac{-1}{k+1}}}{2^{\frac{1}{k+1}}(1+G)} \\
&= \frac{\frac{1}{G}\left(\frac{1}{1+G^{\frac{k+1}{k+2}}}\right)^{\frac{-1}{k+1}} + \left(1 - \frac{1}{1+G^{\frac{k+1}{k+2}}}\right)^{\frac{-1}{k+1}}}{2^{\frac{1}{k+1}}(\frac{1}{G}+1)}.
\end{aligned}
$$

This implies that for $G \to \infty$ we have

$$
\frac{W_{\text{mult}}(w_*)}{W_{\text{mult}}(0.5)} \to 2^{\frac{-1}{k+1}}.
$$

Thus for very large G and $k = 1$ the optimized workload needs about 70% of the workload for $w = 0.5$.

6.4 Adaptive partitioning control

In the previous sections we described how the local error can be controlled by independent control of $H_n, h_{n-1,m}$ for a fixed partitioning. But for e.g. digital circuits (with dynamically changing activities) it is impractical to apply multirate with a static partitioning. Then dynamical partitioning techniques [19, 46, 60, 67] are needed which are able to follow the moving active part. This means that the partitionings should be updated during the multirate time integration. Because *repartitionings* are not cheap, one will not allow to change the partition during the refinement. Thus repartitionings only can occur just after the compound step or just after the refinement phase. By keeping the old partitioning for an acceptable speed-up factor for some time, the number of repartitionings is reduced. There exist the following three alternatives.

A The partitioning is modified just after the compound step such that the compound step can be accepted.

B The partitioning is controlled simultaneously with the stepsizes just after the refinement phase. Here the available estimates of the errors per element are used.

C First a single-rate step is done and then its discretisation error vector is used to repartition.

Methods B and C are better suited for the stepsize control and the relaxation of the Newton process, because the active part of the coarse error vector in method A is not accurate. Nevertheless, method A is better able to detect sudden wake-ups of latent variables. Note that method A is applied with the restriction that only latent elements can be transfered to the refinement part, while method B is always applied at the end of a multirate-step.

Dynamical partitioning has some consequences for the existing multirate algorithm. Firstly, there is the storage problem. Now, theoretically, each node may have its own time-points, theoretically. Because the lengths of these time-grids will differ for each unit, it is impossible to store the solutions and time-grids in a normal array. Furthermore, for multistep methods there is the initialisation problem for the waked up fast nodes that were slow during the previous compound step. Restarting with onestep methods, like Euler Backward, can reduce the gained efficiency. We use the previous coarse-grid polynomial also as a predictor polynomial for the new refinement phase using upgraded results. For real-world applications it might be necessary to take care of the already existing structure, e.g. a hierarchical structure for circuit models. Instead of element-wise partitioning the submodels are treated like connected blocks that can be active or latent as a block.

6.5 Local efficiency analysis of multirate methods

In this subsection we will analyze the *local efficiency* of a multirate method during one multirate-step $[T_n, T_{n+1}]$. Although we introduced the Compound-Fast BDF algorithm, this analysis is valid for a much larger family of multirate methods. Let W_C, W_R be the *computational work per multirate step* for the compound phase and the refinement phase and define the *workload ratio* by $E = \frac{W_R}{W_C}$. Let W_S be the *computational work per single-rate step* for the standard single-rate version, that satisfies $\frac{W_S}{W_C} = F \approx 1$. If H and h are the compound step and refinement step and $q = \frac{H}{h}$ the multirate factor, then a multirate method on $[T_n, T_{n+1}]$ will need the *computational workload*

$$W_{\text{mult}} = W_R q + W_C = W_C(Eq + 1), \tag{6.43}$$

while a single-rate method with step h_s would need $W_{\text{sing}} = W_S \frac{H}{h_s}$. Thus we have the following speed-up factor for the multirate method

$$S = \frac{W_{\text{sing}}}{W_{\text{mult}}} = \frac{W_S \frac{H}{h_s}}{W_C(Eq + 1)} = F \frac{h}{h_s} \frac{1}{\frac{1}{q} + E} \approx \frac{h}{h_s} \frac{1}{\frac{1}{q} + E}. \tag{6.44}$$

Here q is the multirate factor, that is large if the dynamics of the refined part are more active than the other slow part. The ratio E is determined by the partitioning and describes the relative costs of a refinement step that depends on the size d_A of the refinement part. Assuming fixed h, h_s, we observe that $S \to \frac{F}{E} \frac{h}{h_s}$ for $q \to \infty$ and $S \to Fq \frac{h}{h_s}$ for $E \to 0$. Clearly, we get a large speed-up factor if q is large and E is small. Only if $S > 1$ it could be attractive to use for instance the multirate version of a certain integration scheme.
The workload ratio E typically depends on the (variable) relative size d_A/d of the refinement. We use the approximate model

$$\hat{E} = E_0 + (1 - E_0) \left(\frac{d_A}{d} \right)^{\alpha}, \tag{6.45}$$

where E_0 is a constant part corresponding to the overhead costs of a multirate method. In practice we will assume that $E_0 \approx 0.1$. The variable part depends on $\alpha \in (1, 2)$. By

default we use $\alpha = 2$. Note that it is also possible to model E by a parameterised rational function of d_A and d, where the parameters can be identified by using experimental data. For hierarchical partitioning it can be necessary to use a weighted formula that takes care of the individual sizes of the subcircuits.
The multirate factor q can be *estimated* by the ratio

$$\hat{q}_{n+1} = \frac{H_{n+1}}{h_{n,1}}. \tag{6.46}$$

Here $H_{n+1}, h_{n,1}$ are the proposed stepsizes for the compound step and refinement step, respectively. Section 6.1 shows how these stepsizes are computed, e.g. by the elementary stepsize controllers (6.31),(6.32). Because we used $h_{n,1} = h_{n-1,q}$ without clipping, we get

$$\hat{q}_{n+1} = \frac{H_{n+1}}{h_{n-1,q}} = \left(\frac{\theta\mathrm{TOL}}{\hat{r}_C^n}\right)^{\frac{1}{K+1}} \left(\frac{\hat{r}_R^{n-1,q-1}}{\theta\mathrm{TOL}}\right)^{\frac{1}{k+1}} \frac{H_n}{h_{n-1,q-1}}.. \tag{6.47}$$

For the Compound-Fast method the error estimates $\hat{r}_C$, and $\hat{r}_R$ are defined as

$$\begin{aligned} \hat{r}_C^n &= \max\{\hat{r}_L^n, \frac{1}{w}\hat{r}_I^n\} = \max\{\|\mathbf{B}_L\hat{\mathbf{d}}^n\|, \frac{1}{w}h_{\max}\|\mathbf{B}_A\hat{\mathbf{K}}_n\mathbf{B}_L^T\|\|\mathbf{B}_L\hat{\mathbf{r}}^n\|\}, & (6.48) \\ \hat{r}_R^{n-1,m} &= \frac{1}{1-w}\hat{\hat{r}}_A^{n-1,m} = \frac{1}{1-w}\|\mathbf{B}_A\hat{\hat{\mathbf{d}}}^{n-1,m}\|. & (6.49) \end{aligned}$$

Clearly they strongly depend on the partitioning. The previous steps $H_n, h_{n-1,q-1}$ are considered to be independent of $\mathbf{B}_A, \mathbf{B}_L$. This is the case for adaptive partition control, where the partitionings at the previous steps are given and only the partitioning for the new multirate step is optimised. The procedure (6.47)-(6.49) directly computes the multirate factor $\hat{q}$ for the current partitioning (and with estimates of H and h as well). Because the partitioning is determined by $\mathbf{B}_A, \mathbf{B}_L$, this formula for $\hat{q}_{n+1}$ enables us to express the expected speed-up factor $\hat{S}_{n+1} = \frac{1}{\hat{E}_{n+1} + \frac{1}{\hat{q}_{n+1}}}$ in terms of the partitioning.
For $K = k$ we can use the following formula for $\hat{q}$

$$\hat{q}_{n+1} = \left(\frac{\hat{r}_R^{n-1,q-1}}{\hat{r}_C^n}\right)^{\frac{1}{K+1}} \frac{H_n}{h_{n-1,q-1}}. \tag{6.50}$$

In section 4.6 it is shown how digital linear control theory can be used to develop higher order stepsize controllers that deliver smoother stepsize sequences. Exploiting this here implies that the multirate factor also depends on previous values of the error estimates. This also improves the smoothness of the partitioning sequence. A drawback of higher order stepsize controllers is that they are difficult to apply combined with variable order. In fact they should only be used if the order remains constant during a long time interval. But even then the stepsize controller depends on the current used order.
In (6.47) we showed how we can express the multirate factor in terms of the partitioning. The major problem is that the error vectors $\hat{\mathbf{d}}^n, \hat{\mathbf{r}}^n$ and the coupling matrix $\hat{\mathbf{K}}_n$ are only accurate for the current latent part and that the error vector $\hat{\hat{\mathbf{d}}}^{n-1,m}$ only exists for the current active part. Indeed, we assumed that the corresponding state elements have converged in the Newton method, which is not true for the active elements (i.e. without

relaxed error control).

To solve this problem we need to construct approximations that are valid for all elements. Using the Nordsieck polynomial technique (Section 4.3), we know that error vectors are estimated by the difference between the corrected and predicted solution. The error corresponds to the performed step, so it depends on the current stepsize H or h and order K or k. If we rescale and resize the predictor polynomial we can use its prediction to estimate the local error for other steps or order. Here we can use the transformation matrices that are defined in [68].

We can easily obtain the scaled and resized corrector polynomial. This polynomial is necessary for slow elements that become active. Now we should transform both the current predictor and corrector polynomials to get a proper error estimate and a good initial Nordsieck array. Then we can use the rescaled and transformed vectors to get an error estimate for the case that different step and order would have been used. Note that the method works well for different steps, but for higher order the error estimate will in fact be equal to the current estimate. We can use the same trick to rescale the fine error vector $\hat{\mathbf{d}}^{n-1,m}$ to the coarse grid. This even goes better, because $K \leq k$. Also note that we only should rescale and resize the elements that we are investigating. *Rescaling and resizing* the previous error vectors allows us to use the proposed formula for $\hat{q}$. With respect to the coupling matrix we should evaluate the Jacobian matrices in the state that is corrected by the refinement. We will assume that the coupling matrix behaves slowly and is properly approximated at the coarse time-grid. Note that for LTI systems the coupling matrix is even constant.

6.6 Partitioning algorithms

Although the multirate step h can be smaller than the single-rate step h_s such that the accuracy is maintained, we assume that

$$F\frac{h}{h_s} \approx 1 \text{ is independent of the partitioning.} \tag{6.51}$$

Thus the partitioning is optimal if $\hat{S} = \frac{1}{\frac{1}{q}+E}$ achieves its maximal value. Let the index sets of the active and latent parts be $\mathrm{ind}_A, \mathrm{ind}_L$ of lengths d_A, d_L such that $d = d_A + d_L$. Then both q and E are functions of $\mathrm{ind}_A, \mathrm{ind}_L$. Thus the *optimal partitioning* at $[T_n, T_{n+1}]$ satisfies:

$$\max_{\mathrm{ind}_A,\mathrm{ind}_L} \hat{S}_{n+1} = \max_{\mathrm{ind}_A,\mathrm{ind}_L} \frac{1}{\frac{1}{\hat{q}_{n+1}} + \hat{E}_{n+1}}. \tag{6.52}$$

Here formulae (6.45) and (6.47) are used to compute the estimates $\hat{E}_{n+1}$ and $\hat{q}_{n+1}$, respectively. This optimisation problem is solved after each multirate-step. The exact solution of optimisation problem (6.52) is a discrete 0-1 optimisation problem with a complexity growing very fast because there are 2^d possible partitionings.

The previous section showed how the partitioning can be adaptively controlled by optimizing $\hat{S}$ after each multirate-step. This approach works well for a very small d, but already for $d = 15$ parts there are $2^{15} = 32768$ possible partitionings. Considering all these combinations would be too expensive. In the following sections we propose some algorithms that find a reasonable approximation of the optimal partitioning in an acceptable time.

Instead of optimizing the speed-up factor $\hat{S}$ in the most general way it is also possible to optimize $\hat{S}$ under the restriction that d_A is fixed. This is simpler because it is equivalent to optimizing $\hat{q}$ because now the workload ratio $\hat{E}$ (eq. 6.45) is constant. Since for the Compound-Fast version also the micro step $h_{n-1,q}$ is independent of the partitioning, it is even sufficient to maximise the macro step H_{n+1}. Remember that H_{n+1} in (6.31) depends on the new partitioning, because of

$$\hat{r}_C^n = \max\{\|\mathbf{B}_L\hat{\mathbf{d}}^n\|, \frac{1}{w}h_{\max}\|\mathbf{B}_A\hat{\mathbf{K}}_n\mathbf{B}_L^T\|\|\mathbf{B}_L\hat{\mathbf{r}}^n\|\}.$$

We assume that the vectors $\hat{\mathbf{d}}^n, \hat{\mathbf{r}}^n$ and matrix $\hat{\mathbf{K}}_n$ are valid for all elements by use of proper rescaling and resizing of the fine Nordsieck arrays.

We define a sequence of those *semi-optimal partitionings* for fixed $d_A = x$, where $x \in \{1, \dots, d\}$. Then the real optimal partitioning for optimal x can be selected from all semi-optimal partitionings. We also use this sequence in order to find the semi-optimal partitionings recursively for increasing the value of x. If we compute the semi-optimal partitioning for all x we are able to compute the exact solution as will be described next. Note that under the restriction $d_A = x$ there are still $\binom{d}{x}$ possible partitionings. Indeed

$$\sum_{x=0}^{d}\binom{d}{x} = 2^d.$$

The semi-optimal partitioning for $d_A = x$ can be found from the semi-optimal partitioning for $d_A = x - 1$ by considering all possible *transitions*. If L fast elements from a set with $d_A = x - 1$ are moved to the slow part, it follows that $L + 1$ slow elements have to be moved to the fast part in order to get a new set with $d_A = x$. Note that L is the number of interchanges, that is bounded by $L_{\max} = \min\{x, d - x\}$. The total number of transitions with l interchanges sums up to

$$A_L = \sum_{l=0}^{L}\binom{x}{l}\binom{d-x}{l+1}.$$

Then the new partitioning can be found by maximizing the difference $\Delta\hat{S}$ for all A_L transitions. If we take $L = L_{\max}$ it can be proven indeed that

$$\sum_{l=0}^{L_{\max}}\binom{x}{l}\binom{d-x}{l+1} = \binom{d}{x+1}.$$

This means that then all possible partitionings for $d_A = x + 1$ are considered. Thus taking $L = \min\{x, d - x\}$ leads to the optimal partitioning for each x. But computing the *partitioning sequence* in this manner still considers all 2^d possible partitionings, which is too expensive.

Therefore we make the following assumption:

Assumption 6.6 *A semi-optimal partitioning with* $d_A = x$ *can be found from a semi-optimal partitioning with* $d_A = x - 1$ *by at most* $L_* \in \mathbb{N}$ *interchanges.*

The case $L_* = 0$ is a very special case because then the partitioning sequence is assumed to be nicely ordered. If an element is refined for $d_A = x$ it remains in the refinement for all higher values of x. Furthermore, it allows cheap partitioning algorithms because $A_0 = d - x$. If interchanges sometimes occur one could take $L_* = 1$ that already investigates $A_1 = d - x + x\binom{d-x}{2}$ possible transitions.
The proposed assumption allows us to compute the partitioning sequence much faster. In this manner it is possible to compute the semi-optimal partitioning for all $x \in \{0, \dots, d\}$. Next we could find the global optimum by comparing all these found partitionings. Clearly, this approach is still expensive for large values of d. We could compute the semi-optimal partitioning only for $0 \leq x \leq x_{\max} \ll d$ or by using local search. This algorithm will only give the optimal partitioning if the assumption was correct. Otherwise the accuracy can be improved by constructing two partitioning sequences for both increasing and decreasing x. For each x the partitioning with maximal $\hat{S}$ is selected.
With local search one starts with an initial value $d_A = x_0$ and computes the semi-optimal partitionings for $d_A = x_0 \pm 1$. This initial value x_0 could e.g. be equal to $0.1d$. Then x_1 is equal to the size of the considered partitioning with maximal $\hat{S}$. This procedure is repeated iteratively until a local maximum x_{opt} has been found. Although this algorithm does not always find the global maximum, it is much cheaper because it does not need to consider all possible values for x.
Let us consider the four partitioning algorithms below. All four algorithms are only based on the local error vector for all elements, and neglect the interpolation errors. Note that rescaling and resizing are required to get accurate values for all elements of $\hat{\mathbf{d}}$. By (6.31) it is also possible to compute a similar vector $\hat{\mathbf{H}}$. From the previous section we know that the semi-optimal partitioning sequence is nicely ordered if the interpolation errors are completely neglected. Thus the elements can be sorted corresponding to the corresponding local error. As shown before in (6.47) we can express the new multirate factor $\hat{q}$ in terms of $\hat{\mathbf{d}}$. Then the optimal partitioning can be found by optimizing $\hat{S}(x)$.

I Algorithm I really searches this maximum of $\hat{S}(x)$ by local search. Each iteration it selects the most active latent element and the most latent active element. Then all four possible transitions of these two found elements are compared with respect to their corresponding estimated speed-up factors and the optimal transition is performed. Iteratively the transition with maximum estimated speed-up factor is performed until convergence. This algorithm needs an *initial partitioning* to start with. It could be the previous partitioning, a partitioning computed by another algorithm or it is given by the user. Note that the allowed interchange of elements are in fact not necessary for an ordered partitioning.

II Algorithm II flags unknowns as active when the estimated local error satisfies

$$\hat{d}_i > \epsilon_{\text{rel}} \|\hat{\mathbf{d}}\|_{\max}, \tag{6.53}$$

where $\epsilon_{\text{rel}} < 1$. An alternative is to compute the needed stepsizes per unknown from the local error vector $\hat{\mathbf{d}}$ and store them in a vector $\hat{\mathbf{H}}$. Then we get an equivalent condition

$$\hat{H}_i < \mu_{\text{rel}} \min\{\hat{H}_i\}, \tag{6.54}$$

where $\mu_{\text{rel}} > 1$.

III Algorithm III considers the absolute criterion

$$\hat{d}_i > \epsilon_{\text{abs}}, \tag{6.55}$$

where ϵ_{abs} <TOL or

$$\hat{H}_i < \mu_{\text{abs}}, \tag{6.56}$$

where μ_{abs} <TOL. The tolerance level TOL is given by the user such that the local error satisfies $\|\hat{\mathbf{d}}\|_{\max} \leq$TOL or $\min\{\hat{H}_i\} \geq$TOL.

IV Algorithm IV detects the largest gap between the elements of the vectors $\hat{\mathbf{d}}$ or $\hat{\mathbf{H}}$. Then this gap is used to separate the system in a fast and a slow part. So x is taken at the location of the largest gap between the individual error elements or timesteps. But finding this largest gap is not less difficult than finding the optimum of $\hat{S}(x)$.

For algorithms II and III the values of ϵ_{abs} or ϵ_{rel} still can be chosen. The optimal values depend on the properties of the vector $\hat{\mathbf{d}}$ or $\hat{\mathbf{H}}$. Neglecting the interpolation errors we have approximately

$$\hat{q}_{n+1} \approx \mu_{\text{rel}}. \tag{6.57}$$

This nice relation does not exist for the tolerance level ϵ_{rel} in general. Only if $K = k$ we have

$$\hat{q}_{n+1} \approx \frac{H_n}{h_{n-1,q-1}} \epsilon_{\text{rel}}^{\frac{-1}{K+1}}.$$

These properties motivate to use relative tolerance levels (Algorithm II) instead of absolute tolerance levels (Algorithm III). Also in practice Algorithm II works better than Algorithm III. In particular (6.54) is very useful because of the mentioned relationship with the new multirate factor. For algorithms II to IV it is necessary to add an upper bound for the size of the refinement, because it was not included in [67]. Let $S_{\min} > 1$ be the minimum allowed speed-up and $\hat{q}$ the estimated multirate-factor for the chosen partitioning. Then we get the following upper bound for the workload ratio $\hat{E}$ (eq. 6.45):

$$\hat{E} \leq \frac{1}{S_{\min}} - \frac{1}{\hat{q}}, \tag{6.58}$$

that by (6.57) and (6.54) for Algorithm II yields

$$\hat{E} \leq \frac{1}{S_{\min}} - \frac{1}{\mu_{\text{rel}}}.$$

The Algorithms I-IV still neglect the interpolation errors. Hence they need some add-ons that improve the robustness. One could add a safety region of distance γ around the active part. This means that all nodes which distance to the active part is at most γ are also refined. This makes it possible to predict sudden wake-ups and reduces the number of repartitionings. Furthermore one could always refine the nodes that are connected to sources if they can suddenly become active.

6.7 Influence of interpolation errors on partitioning sequence

The algorithms presented in the previous section are based on Assumption 6.6 with $L_* = 0$. In this case the partitioning sequence is assumed to be nicely ordered without exchanges. Hence, if the partitioning for $d_A = x-1$ is given and we are going to look for a partitioning with $d_A = x$, we only have to consider transitions from latent to active. If the partitioning for $d_A = x$ is given and we are going to look for a partitioning with $d_A = x - 1$ (or smaller) we only have to consider transitions from active to latent. If we neglect the interpolation errors, this assumption is valid because then the multirate factor only depends on

$$\hat{r}_C^n = \|\mathbf{B}_L\hat{\mathbf{d}}^n\|.$$

Indeed, then the elements can be sorted based on $\hat{\mathbf{d}}^n$, such that the semi-optimal partitioning for $d_A = x$ simply consists of the first x sorted elements.

Nevertheless, including the interpolation errors is preferable because it can lead to better partitionings. Although then the assumption that $L_* = 0$ is not longer true, we can still use it in the partitioning. This implies that for a given semi-optimal partitioning with $d_A = x - 1$ the error $\hat{r}_C^n$ only can be reduced by making the dominant element active. If the discretisation part $\|\mathbf{B}_L\hat{\mathbf{d}}^n\|$ was larger, we simply refine the element(s) corresponding to the largest error. If the interpolation part $\frac{1}{w}h_{\max}\|\mathbf{B}_A\hat{\mathbf{K}}_n\mathbf{B}_L^T\|\|\mathbf{B}_L\hat{\mathbf{r}}^n\|$ was larger, we have to consider the latent element(s) corresponding to the column of $\mathbf{B}_A\hat{\mathbf{K}}_n\mathbf{B}_L^T$ with maximal norm and the latent element corresponding to the largest element with the largest contribution to $\mathbf{B}_L\hat{\mathbf{r}}^n$. Because one transition was sufficient to reach the semi-optimal partitioning for $d_A = x$ we take one with the best option. This assumption $L_* = 0$ is not fulfilled by the optimal partitionings in general. But using it implies that the partitioning sequence will be much smoother than the optimal partitioning sequence. From a computational point of view this is nice, because it saves repartitioning costs.

If Assumption 6.6 is valid for $L_* = 1$ an optimal partitioning under the restriction that $d_A = x$ can be found by at most one element exchange from the previous partitioning with $d_A = x-1$. Then we have to consider for all active elements the effect of replacing them by two latent variables. Note that for a fixed active element we can simply take the two most strongly coupled latent elements. Looking at only the most active element is not sufficient because it is possible that only this element is strongly coupled. Nevertheless the costs are not high because of the low number of variables connected to the interface. Instead of an one-dimensional array we get a two-dimensional array. Also if $L_* = 1$ the partitioning costs are still of polynomial complexity.

Example 6.7 In this example we will compare the previous presented partitioning algorithms for the following example

$$\hat{\mathbf{d}}^n = \hat{\mathbf{r}}^n = \begin{bmatrix} 0.1 \\ 0.0001 \\ -0.1 \\ 0.02 \end{bmatrix}, \hat{\mathbf{d}}^{n-1,q-1} = \begin{bmatrix} 0.01 \\ 0.0001 \\ -0.01 \\ 0.02 \end{bmatrix}, \hat{K}_n = \begin{bmatrix} 1 & 1 & & \\ -1 & 2 & 0.1 & \\ & 20 & 1 & 0.001 \\ 0.001 & & & 1 \end{bmatrix},$$

$$\begin{aligned}&K = k = 1,\\ &H_n = 0.01, h_{n-1,q-1} = 0.001, h_{\max} = 0.01,\\ &w = \tfrac{1}{2}, \theta\mathrm{TOL} = 10^{-3}.\end{aligned}$$

Thus we already got proper error estimates for all elements by use of resizing and rescaling. We will compare the partitionings obtained by the several presented partitioning algorithms.

From $\hat{\mathbf{d}}^{n-1,q-1}$ we can e.g. compute that

$$\hat{h}_{n,1} = \left(\frac{\theta\mathrm{TOL}}{0.01}\right)^{\frac{1}{2}} h_{n-1,q-1} = 3.2 \cdot 10^{-4}.$$

The new compound step depends on the partitioning. Because of the assumptions, we can sort it based on $\hat{\mathbf{d}}^{n}$.

$$\hat{H}_{n+1} \left(\frac{\theta\mathrm{TOL}}{\|\mathbf{B}_L \mathbf{d}^n\|}\right)^{\frac{1}{2}} H_n = 3.2 \cdot 10^{-4}.$$

ind_A	$\hat{H}_{n+1}$	$\hat{q}_{n+1}$
$\{1\}$	10^{-3}	3.1
$\{1,3\}$	$2.2 \cdot 10^{-3}$	6.9
$\{1,3,4\}$	0.032	100
$\{1,2,3,4\}$	$H_{\max}$	$3125 H_{\max}$

Note that the last partitioning is in fact equivalent to single-rate. Thus the largest gap for $\hat{H}$ is between $\{1,3\}$ and $\{1,3,4\}$.

We saw that the relative tolerance μ_{rel} is in fact the maximum allowed speed-up factor. Thus for $\mu_{\mathrm{rel}} = 10$ we refine all elements that have relative stepsizes smaller than 10

$$\hat{H}_i < 10 \min\{\hat{H}_i\},$$

Hence, Algorithm II, using the relative tolerance level, indicates that the active set must be $\{1,3\}$.

For Algorithm I using local optimisation without interpolation errors we add the corresponding workload ratio to the table (column 4). We take the simple model $\hat{E} = 0.05 + 0.95\frac{d_A}{d}$. Furthermore we add the estimates of the speed-up factor as well (column 5).

ind_A	$\hat{H}_{n+1}$	$\hat{q}_{n+1}$	$\hat{E}_{n+1}$	$\hat{S}_{n+1}$
$\{1\}$	10^{-3}	3.1	0.29	1.63
$\{1,3\}$	$2.2 \cdot 10^{-3}$	6.9	0.53	1.48
$\{1,3,4\}$	0.032	100	0.76	1.3
$\{1,2,3,4\}$	$H_{\max}$	$3125 H_{\max}$	1	1

This criterion indicates that $\{1\}$ is the best option now. For larger active parts, the costs per refinement step become too large.

The previous results only apply if we neglect the interpolation errors. If we include the interpolation errors there are in general much more possible partitionings. The values of $\hat{S}$ are now different because H is now also used to control the interpolation errors.

ind_A	$\|\mathbf{B}_L\mathbf{d}^n\|$	$\|\mathbf{B}_A\mathbf{K}_n\mathbf{B}_L^T\|$	$\hat{r}_C^n$	$\hat{H}_{n+1}$	$\hat{q}_{n+1}$	$\hat{E}_{n+1}$	$\hat{S}_{n+1}$
$\{1\}$	0.1	1	0.1	10^{-3}	3.1	0.29	1.63
$\{2\}$	0.1	1.1	0.1	10^{-3}	3.1	0.29	1.63
$\{3\}$	0.1	20.001	0.4	$5\cdot10^{-4}$	1.56	0.29	1.07
$\{4\}$	0.1	0.001	0.1	10^{-3}	3.1	0.29	1.63
$\{1,2\}$	0.1	0.1	0.1	10^{-3}	3.1	0.53	1.17
$\{1,3\}$	0.02	20.001	0.4	$5\cdot10^{-4}$	1.56	0.53	0.85
$\{1,4\}$	0.1	1	0.1	10^{-3}	3.1	0.53	1.17
$\{2,3\}$	0.1	1	0.1	10^{-3}	3.1	0.53	1.17
$\{2,4\}$	0.1	1	0.1	10^{-3}	3.1	0.53	1.17
$\{3,4\}$	0.1	0	0.1	10^{-3}	3.1	0.53	1.17
$\{1,2,3\}$	0.02	0.001	0.02	$2.2\cdot10^{-3}$	6.9	0.76	1.1
$\{1,2,4\}$	0.1	0.1	0.1	10^{-3}	3.1	0.76	0.93
$\{1,3,4\}$	0.0001	20	0.4	$5\cdot10^{-4}$	1.56	0.76	0.73
$\{2,3,4\}$	0.1	1	0.1	10^{-3}	3.1	0.76	0.93
$\{1,2,3,4\}$	0	0	0	$H_{\max}$	$q_{\min}$	1	1

Including the interpolation error estimates indicates that the optimal active set is equal to $\{1\}$, $\{2\}$ or $\{4\}$ (because then $\hat{S}$ is maximal). Thus we really get different results because of the additional interpolation errors. Including the interpolation errors will lead to better partitionings, but is also more expensive. Therefore we advise to neglect them in the default case.

6.8 Dynamical partitioning techniques

In the previous section we showed how (6.52) can be used to optimise the partitioning after each multirate-step. However, it is also possible to determine $\hat{S}_{n+1}$ after a single-rate step. If a multirate partitioning can be found for which $\hat{S}_{n+1} > 1$ it is possible to switch from single-rate to multirate. On the other hand, if no acceptable partitioning can be found after a multirate-step it is possible to switch from multirate to single-rate. Therefore we need a *decision strategy about switching between single-rate and multirate*. Note that the speed-up factor estimate should take care of the integration order of the single-rate method, which can be larger than one, while the order for the compound steps is always equal to one.

If some multirate-steps have been done, it can happen that at a certain moment it is no longer possible to find a partitioning for which $\hat{S}$ is acceptable large. Then it can be attractive to switch from multirate to single-rate mode. Because we know that the optimal order is determined by the active part, it is natural to use the integration order k of the refinement also for the new single-rate steps. Furthermore we compute the new single-rate step by

$$h_{n+1} = h_{n,1} = \left(\frac{\theta \mathrm{TOL}}{\hat{r}_R^{n-1,q}}\right)^{\frac{1}{k+1}} h_{n-1,q}, \tag{6.59}$$

that is only based on the error of the previous active part.
If $K < k$ it is necessary to rescale the Nordsieck array for the slow part to the uniform single-rate format. Typically the higher order derivatives will be approximated by zero, which is a reasonable assumption for latent variables. For the single-rate BDF scheme this means that previous latent variables are approximated by interpolation.
After some single-rate-steps it can be an interesting option to reconsider whether multirate can be an efficient alternative. Therefore, it is necessary to estimate $\hat{S}_{n+1}$ for a possible new multirate step. If the multirate factor is estimated after a single-rate step, model (6.45) is still valid but we can not use formula (6.47). Indeed then both H_{n+1} and $h_{n,1}$ are predicted based on the same previous error $\hat{r}^n$ and step h_n. Although we could also use the same order for the compound step, this is not always optimal because of the much larger steps. We even assume that $K = 1$ is always fixed. Despite of $K < k$ and $\hat{r}^n = O(h_n^{k+1})$ we will use it also to compute the new compound step. Thus we typically get

$$\hat{q}_{n+1} = \frac{H_{n+1}}{h_{n,1}} = \left(\frac{\theta \mathrm{TOL}}{\hat{r}_C^n}\right)^{\frac{1}{K+1}} \left(\frac{\hat{r}_R^n}{\theta \mathrm{TOL}}\right)^{\frac{1}{k+1}} . \tag{6.60}$$

Here we use for the BDF Compound-Fast multirate method the same formula for r_C^n as in (6.24) but use the single-rate version of $r_R^{n-1,m}$ in (6.25), resulting in

$$\begin{aligned} \hat{r}_C^n &= \max\{\|\mathbf{B}_L \mathbf{d}^n\|, h_{\max}\|\mathbf{B}_A \mathbf{K}_n \mathbf{B}_L^T\|\|\mathbf{B}_L \mathbf{r}^n\|\}, & (6.61) \\ \hat{r}_R^n &= \frac{1}{1-w}\|\mathbf{B}_A \mathbf{d}^n\|. & (6.62) \end{aligned}$$

Partitioning control is only allowed after the refinement if for all elements there are accepted error estimates. Between the compound step and the refinement phase, we do not really control the partitioning. Nevertheless, we still allow small adjustments of the refinement such that we can accept the already performed compound step. Because a compound step is an expensive task, such a flexibility in the multirate algorithm is very important for circuit models. In particular for circuit models where elements can suddenly wake up. Then we need a *rejection strategy* to choose between rejecting the performed compound step or moving some parts to the refinement.
Consider a partitioning with d_A active and d_L latent elements. Assume that for a performed compound step x latent elements were rejected. If we reject the complete compound step, we have to pay and additionally cost $W_C(d)$. But if we move these x elements to the active part, we obtain additional cost $qW_R(d_A + x) - qW_R(d_A)$ during the refinement phase. The second choice is more efficient if

$$R = q\frac{W_R(d_A + x) - W_R(d_A)}{W_C(d)} < 1.$$

Assume that $W_R(d_A) = d_A^\alpha$ amd $W_C(d) = d^\alpha$, then

$$R(x) = q\frac{(d_A + x)^\alpha - d_A^\alpha}{d^\alpha}. \tag{6.63}$$

Hence, for general α, when

$$x < d_A\left[\left(1 + \frac{1}{q}(\frac{d}{d_A})^\alpha\right)^{\frac{1}{\alpha}} - 1\right], \tag{6.64}$$

it is cheaper to accept the current compound step and to move these x elements to the active set. For $\alpha = 1$ we get the simple expression

$$R = q\frac{x}{d},$$

and

$$x < \frac{d}{q}. \tag{6.65}$$

Clearly, if the multirate factor q is rather small it follows that also compound steps with a relatively large number of rejected elements still should be accepted. Here the multirate factor q can be estimated by the number of refinement steps in the previous multirate-step.

6.9 Further extensions

In (6.45) we proposed a rather simple model for the workload ratio $E = \frac{W_R}{W_C}$, where W_C, W_R are the computational work per timestep for the compound phase and the refinement phase. It is based on the assumption that it only depends on the relative size of the active part like

$$E = E_0 + (1 - E_0)\left(\frac{d_A}{d}\right)^{\alpha}.$$

In fact this model is based on the assumption that the used integration scheme needs about the same number of Newton iterations for both the active and the latent parts. Thus $r \approx R$, where r, R are the average number of Newton iterations per refinement or compound step. But if this assumption is not fulfilled, we could better use a different model for E, e.g.

$$E = E_0 + (1 - E_0)q\left(\frac{d_A}{d}\right)^{\alpha}. \tag{6.66}$$

This model assumes that $\frac{R}{r}$ scales like Cq where q is the multirate factor. This model predicts a relatively large number of compound steps for large multirate factors, which is reasonable for strongly coupled systems. However, using model (6.66) instead of (6.45) in the partitioning algorithms appears to be more complicated. The problem is that now for a fixed $\hat{E}$ ($d_A = x$) optimizing $\hat{S}$ is no longer equivalent to maximizing $\hat{q}$. For the new model there exists for each x a finite optimal value for $\hat{q} = \hat{q}(x)$.

Another extension deals with the fact that all elements do not need the same evaluation costs. For circuit models it is well-known that e.g. elements of (transistor) device models need a very large evaluation time per element (that increased with a factor ≤ 10 over the last 10 years due to the more accurate modeling that is needed with the ongoing decrease of device sizes and the increase of frequencies at which circuits have to operate). Then it can be useful to consider also the relative number of expensive elements in the active part. Let $e \leq d$ be the total of expensive elements and e_A the number of fine expensive elements. Then we typically get

$$E = E_0 + E_1\left(\frac{d_A}{d}\right)^{\alpha} + E_2\left(\frac{e_A}{e}\right)^{\beta}.$$

Again we have the relationship $E_0 + E_1 + E_2 = 1$, while α, β depend on the application. Sometimes it can happen that it is more attractive to split the system in more than two subsystems with different levels of activity, e.g. to reduce also the linear algebra. Unfortunately the previous presented algorithms do not work here directly because the model (6.44) is not valid now. Furthermore they are based on an one-dimensional optimisation of $\hat{S}(x)$, where $\hat{S}(x)$ is the optimal speed-up with $d_A = x$. Now we need a multi-dimensional optimisation problem of $\hat{S}(x_1, \ldots, x_{N-1})$, where $\hat{S}(x)$ is the optimal speed-up factor with $d_{(1)} = x_1, \ldots, d_{(N-1)} = x_{N-1}$. Considering all possible values for $x_1, \ldots, x_{N-1}$ is no longer feasible, but we still can use local search.

An alternative is to apply multirate in a recursive way [53, 54]. Thus also in the refinement the active part can be solved by a multirate method. A benefit is that we can reuse a larger part of the presented partitioning algorithms. For each parent *partition level* we can use a previous presented partitioning algorithm to split the system in a slow and active part. The only difference is that now the active part again can be splitted in moderate and very active parts, etc. Note that model (6.44) for $\hat{S}$ is still valid but we get a different model for $\hat{q}$ or $\hat{E}$.

Chapter 7

Model order reduction (MOR)

7.1 Introduction

In the two previous Chapters we showed how the redundancy of a numerical model can be reduced by using multirate schemes. However, it is also possible to reduce a redundant continuous DAE model directly by model order reduction (MOR). Mathematical *model order reduction (MOR)* aims at replacing the original circuit model by a system of much smaller dimension, which can then be solved by suitable DAE solvers within acceptable time. The idea is to preserve the behaviour at the output terminals or nodes. This is usually tested by studying the response to signals with increasing frequencies. To do this we assume that the original model has the following differential-algebraic structure

$$\left\{ \begin{array}{c} \frac{\mathrm{d}}{\mathrm{d}t}\left[\mathbf{q}(t,\mathbf{x})\right] + \mathbf{j}(t,\mathbf{x}) + \mathbf{B}\mathbf{u} = \mathbf{0}, \quad \mathbf{x}(0) = \mathbf{x}_0, \\ \mathbf{y} - \mathbf{h}(\mathbf{x}) = \mathbf{0}. \end{array} \right. \tag{7.1}$$

For circuit models the current and voltage sources are considered as input signals, of which the contributions can be modeled by $\mathbf{Bu}$. In chapter 2 these contributions were added to the function $\mathbf{j}(t,\mathbf{x})$.

At present, only for linear time-invariant systems MOR techniques are developed well enough and understood properly to be employed [1]. Hence we either linearise the system or decouple it into nonlinear and linear subcircuits (interconnect macromodeling or parasitic subcircuits [14]) and apply MOR to the linear part. The nonlinear MOR techniques are less developed and less understood than the linear ones. In this chapter we present the application of two most promising nonlinear reduction methods on an academic diode chain model. These are the Trajectory PieceWise Linear approach (TPWL) [41,42] and the Proper Orthogonal Decomposition (POD) [2], the last one supported by the newly developed so called Missing Point Estimation (MPE) technique [3].

Model Order Reduction of nonlinear circuits is a hot topic nowadays. One likes to find a reduced model that describes the relationship between the input and output based on the original model. But one of the main drawbacks is the high cost of the model evaluations. This chapter therefore proposes a completely new method that generates much more efficient methods.

7.2 Model order reduction of LTI systems

Consider the Linear time-invariant (LTI) system

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} - \mathbf{C}\mathbf{x} = \mathbf{0} \end{cases}, \tag{7.2}$$

where $\mathbf{x} \in \mathbb{R}^d$ is the state vector, $\mathbf{u} \in \mathbb{R}^p$ and $\mathbf{y} \in \mathbb{R}^q$ are the input and output functions, while $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are constant system matrices. The *controllability function* L_c and *observability function* L_0 are defined by

$$\begin{aligned} L_c(\mathbf{x}_0) &:= \min\{\frac{1}{2}\int_{-\infty}^{0} \|\mathbf{u}(t)\|^2 dt : \mathbf{u} \in L_2(-\infty, 0), \mathbf{x}(-\infty) = 0, \mathbf{x}(0) = \mathbf{x}_0\}, & (7.3) \\ L_o(\mathbf{x}_0) &:= \frac{1}{2}\int_{0}^{\infty} \|\mathbf{y}(t)\|^2 dt, \mathbf{x}(0) = \mathbf{x}_0, \forall_{\tau \in [0,\infty)} \mathbf{u}(\tau) = \mathbf{0}. & (7.4) \end{aligned}$$

These functions describe the minimum amount of control energy necessary to reach state $\mathbf{x}_0$ at $t = 0$ and the total output energy generated by $\mathbf{x}_0$ for $t \geq 0$. We have

$$L_c(\mathbf{x}_0) = \frac{1}{2}\mathbf{x}_0^T \mathbf{W}^{-1} \mathbf{x}_0, \quad L_o(\mathbf{x}_0) = \frac{1}{2}\mathbf{x}_0^T \mathbf{M} \mathbf{x}_0, \tag{7.5}$$

where $\mathbf{W} = \mathbf{W}^T$ and $\mathbf{M} = \mathbf{M}^T$ are the controllability and observability *Gramians*, that are defined by

$$\mathbf{W} := \int_0^\infty e^{\mathbf{A}t} \mathbf{B}\mathbf{B}^T e^{(\mathbf{A}^T t)} dt, \quad \mathbf{M} := \int_0^\infty e^{(\mathbf{A}^T t)} \mathbf{C}^T \mathbf{C} e^{\mathbf{A}t} dt. \tag{7.6}$$

For a proof see e.g. [1]. If the matrix $\mathbf{A}$ is stable, such that the system (7.2) is asymptotically stable, the Gramians $\mathbf{W}, \mathbf{M}$ can be solved from the following *Lyapunov equations*.

$$\begin{aligned} \mathbf{A}\mathbf{W} + \mathbf{W}\mathbf{A}^T &= -\mathbf{B}\mathbf{B}^T, & (7.7) \\ \mathbf{A}^T\mathbf{M} + \mathbf{M}\mathbf{A} &= -\mathbf{C}^T\mathbf{C}. & (7.8) \end{aligned}$$

Consider the linear time-invariant transformation $\mathbf{x} = \mathbf{T}\mathbf{z}$, such that the transformed system obeys

$$\begin{cases} \dot{\mathbf{z}} = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}\mathbf{z} + \mathbf{T}^{-1}\mathbf{B}\mathbf{u}, \\ \mathbf{y} = \mathbf{C}\mathbf{T}\mathbf{z}. \end{cases} \tag{7.9}$$

This transformation $\mathbf{T}$ is called a *balancing transformation* if the Gramians of the transformed system are simultaneously equal to the diagonal matrix Σ: $\mathbf{W} = \mathbf{T}\Sigma\mathbf{T}^T, \mathbf{M} = \mathbf{T}^{-T}\Sigma\mathbf{T}^{-1}$. Then we get the following set of equations for $\mathbf{T}$ and Σ

$$\begin{aligned} \mathbf{T}^{-1}\mathbf{A}\mathbf{T}\Sigma + \Sigma\mathbf{T}^T\mathbf{A}^T\mathbf{T}^{-T} &= -\mathbf{T}^{-1}\mathbf{B}\mathbf{B}^T\mathbf{T}^{-T}, & (7.10) \\ \mathbf{T}^T\mathbf{A}^T\mathbf{T}^{-T}\Sigma + \Sigma\mathbf{T}^{-1}\mathbf{A}\mathbf{T} &= -\mathbf{T}^T\mathbf{C}^T\mathbf{C}\mathbf{T}, & (7.11) \end{aligned}$$

that is equivalent to

$$\begin{aligned}\mathbf{A}\underbrace{\mathbf{T}\Sigma\mathbf{T}^{\mathsf{T}}}_{\mathbf{W}}+\underbrace{\mathbf{T}\Sigma\mathbf{T}^{\mathsf{T}}}_{\mathbf{W}}\mathbf{A}^{\mathsf{T}} &= -\mathbf{B}\mathbf{B}^{\mathsf{T}}, &(7.12)\\ \mathbf{A}^{\mathsf{T}}\underbrace{\mathbf{T}^{-\mathsf{T}}\Sigma\mathbf{T}^{-1}}_{\mathbf{M}}+\underbrace{\mathbf{T}^{-\mathsf{T}}\Sigma\mathbf{T}^{-1}}_{\mathbf{M}}\mathbf{A} &= -\mathbf{C}^{\mathsf{T}}\mathbf{C}. &(7.13)\end{aligned}$$

In practice one performs the following steps to get a reduced model of (7.2):

- Solve the Lyapunov equations for the symmetric positive definite Gramians $\mathbf{W},\mathbf{M}$:

  $$\begin{aligned}\mathbf{A}\mathbf{W}+\mathbf{W}\mathbf{A}^{\mathsf{T}} &= -\mathbf{B}\mathbf{B}^{\mathsf{T}}, &(7.14)\\ \mathbf{A}^{\mathsf{T}}\mathbf{M}+\mathbf{M}\mathbf{A} &= -\mathbf{C}^{\mathsf{T}}\mathbf{C}. &(7.15)\end{aligned}$$

  This is an expensive operation, that usually requires $O(d^3)$ operations and for sparse approximations requires $O(d^2 \log d)$ operations [8].
- Compute the Choleski factorisations

  $$\mathbf{W}=\mathbf{L}_w\mathbf{L}_w^{\mathsf{T}}, \quad \mathbf{M}=\mathbf{L}_m\mathbf{L}_m^{\mathsf{T}},$$

  where $\mathbf{L}_w, \mathbf{L}_m$ are lower triangular matrices.
- Compute the singular value decomposition

  $$\mathbf{L}_w^{\mathsf{T}}\mathbf{L}_m=\mathbf{U}\Sigma\mathbf{V}^{\mathsf{T}}.$$
- Compute the balanced matrix

  $$\mathbf{T}=\mathbf{L}_w\mathbf{U}\Sigma^{-\frac{1}{2}}.$$
- Define the truncated matrix $\mathbf{T}_r=[\mathbf{t}_1 \ \cdots \ \mathbf{t}_r]$, where r is chosen based on size of the singular values.

The approach described above is the basis for Truncated Balanced Realisation (TBR) methods. The Choleski factorisations are done to get rid of numerical instabilities, cf [1]. The *transfer function* of system (7.2) is a rational matrix-valued function defined by

$$\mathbf{H}(s)=\mathbf{C}(s\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}. \tag{7.16}$$

It describes the relationship between the input and output signals $\mathbf{u}(s), \mathbf{y}(s)$ in the Laplace domain by

$$\mathbf{y}(s)=\mathbf{H}(s)\mathbf{u}(s). \tag{7.17}$$

Krylov subspace methods create a Krylov subspace in order to estimate the moments of this transfer function. We know that most Krylov subspace methods match the moments of an expansion of the transfer function around a certain frequency s_0. Thus the transfer function is locally very well estimated by Krylov methods. TBR computes a basis for which the global error can be bounded. However, this bound is not sharp because there is a slight difference between the real global error and the error using the Gramians. There exists a general theory that deals with this problem: interpolation of the transfer function. The method "Poor Man's TBR" approximates the TBR solution by using rational Krylov, that employ a multipoint expansion of the transfer function $\mathbf{H}(s)$ [38]. Consider the controllability and observability Gramians in (7.6). Using Parseval's theorem, it can be shown that $\mathbf{W}, \mathbf{M}$ are also equal to

$$\mathbf{W} = \int_{-\infty}^{\infty} (i\omega\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\mathbf{B}^T(i\omega\mathbf{I} - \mathbf{A})^{-*} d\omega, \tag{7.18}$$

$$\mathbf{M} = \int_{-\infty}^{\infty} (i\omega\mathbf{I} - \mathbf{A})^{-*}\mathbf{C}^T\mathbf{C}(i\omega\mathbf{I} - \mathbf{A})^{-1} d\omega. \tag{7.19}$$

Poor Man's TBR computes these integrals numerically at the interesting frequency interval $f = \frac{\omega}{2\pi} \in [F, F]$.

Now consider the DAE LTI system

$$\left\{ \begin{array}{rcl} \mathbf{E}\dot{\mathbf{x}} & = & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} & = & \mathbf{C}\mathbf{x} \end{array} \right. \tag{7.20}$$

where $\mathbf{x} \in \mathbb{R}^d$ is the state vector, $\mathbf{u} \in \mathbb{R}^p$ and $\mathbf{y} \in \mathbb{R}^q$ are the input and output functions, while $\mathbf{E}, \mathbf{A}, \mathbf{B}, \mathbf{C}$ are the system matrices. We note that $\mathbf{E} = \mathbf{I}$ in the previous case but now it can even be a singular matrix. Therefore the classical approach is not applicable here. In [55] it is shown how TBR can be extended to DAE systems. Using the Kronecker decomposition it is possible to rewrite the LTI system (7.20)

$$\left\{ \begin{array}{l} \dot{\mathbf{z}}_1 = \tilde{\mathbf{A}}_1\mathbf{z}_1 + \tilde{\mathbf{B}}_1\mathbf{u}, \\ \mathbf{y}_1 - \tilde{\mathbf{C}}_1\mathbf{z}_1 = \mathbf{0}, \end{array} \right. \tag{7.21}$$

$$\left\{ \begin{array}{l} \mathbf{N}\dot{\mathbf{z}}_2 = \tilde{\mathbf{A}}_2\mathbf{z}_2 + \tilde{\mathbf{B}}_2\mathbf{u}, \\ \mathbf{y}_2 - \tilde{\mathbf{C}}_2\mathbf{z}_2 = \mathbf{0}, \end{array} \right. \tag{7.22}$$

$$\mathbf{y} = \mathbf{y}_1 + \mathbf{y}_2, \tag{7.23}$$

where $\mathbf{N}$ is a nilpotent matrix. Thus the output $\mathbf{y}$ is written as the sum of the outputs of two independent systems that are both driven by the same input $\mathbf{u}$. The first subsystem (7.21) is called *proper* and the second subsystem (7.22) *improper*. We will use this decomposition for TBR. Because the first system is an ODE, we can there apply TBR to balance the input-output map $\mathbf{u} \to \mathbf{y}_1$. For the second part we have to apply a new type of balancing, where the nilpotency of $\mathbf{N}$ is used such that also the input-output map $\mathbf{u} \to \mathbf{y}_2$ is balanced. Then also the input-output map $\mathbf{u} \to \mathbf{y}$ is automatically balanced and can be approximated by truncating both systems. It is not always necessary to perform the Kronecker decomposition, that can be numerically unstable. In [55] the Schur decomposition is used that is numerically stable. Then an additional Sylvester equation has to be solved that decouples the proper and improper parts. The output of an LTI DAE system (7.23) consists of the outputs of two independent system. The first part is just the output of a normal LTI ODE system of the following type

$$\left\{ \begin{array}{l} \dot{\mathbf{z}}_1 = \tilde{\mathbf{A}}_1\mathbf{z}_1 + \tilde{\mathbf{B}}_1\mathbf{u}, \\ \mathbf{y}_1 - \tilde{\mathbf{C}}_1\mathbf{z}_1 = \mathbf{0}. \end{array} \right. \tag{7.24}$$

This system can be reduced by the MOR techniques for ODE systems as has been described in the beginning of this section, using $\mathbf{A} = \tilde{\mathbf{A}}_1, \mathbf{B} = \tilde{\mathbf{B}}_1, \mathbf{C} = \tilde{\mathbf{C}}_1$. The second part is the output of a system of the following type

$$\begin{cases} \mathbf{N}\dot{\mathbf{z}}_2 = \mathbf{z}_2 + \tilde{\mathbf{B}}_2\mathbf{u}, \\ \mathbf{y}_2 - \tilde{\mathbf{C}}_2\mathbf{z}_2 = \mathbf{0}. \end{cases}, \tag{7.25}$$

where $\mathbf{N}$ is a nilpotent matrix. This means that the solution can be given as

$$\mathbf{z}_2 = \left(\mathbf{N}\frac{d}{dt} - \mathbf{I}\right)^{-1} \tilde{\mathbf{B}}_2\mathbf{u} = \left(\mathbf{N}\frac{d}{dt} - \mathbf{I}\right)^{-1} \tilde{\mathbf{B}}_2\mathbf{u}.$$

By using a Neumann series this results in

$$\mathbf{z}_2 = \left(\mathbf{N}\frac{d}{dt} - \mathbf{I}\right)^{-1} \tilde{\mathbf{B}}_2\mathbf{u} = -\left(\mathbf{I} + \mathbf{N}\frac{d}{dt} + \dots\right) \tilde{\mathbf{B}}_2\mathbf{u}.$$

If $\mathbf{N}$ has nilpotency index k, we have

$$\mathbf{z}_2 = -\left(\hat{\mathbf{B}}_2\mathbf{u} + \mathbf{N}\hat{\mathbf{B}}_2\mathbf{u}^{(1)} + \dots + \mathbf{N}^{k-1}\hat{\mathbf{B}}_2\mathbf{u}^{(k-1)}\right).$$

Thus $\mathbf{z}_2$ depends statically on the input $\mathbf{u}$ and its higher order derivatives in contrast to the normal ODE for $\mathbf{y}_2$ in (7.24). The initial condition has to satisfy

$$\mathbf{z}_2(0) = -\left(\hat{\mathbf{B}}_2\mathbf{u}(0) + \mathbf{N}\hat{\mathbf{B}}_2\mathbf{u}^{(1)}(0) + \dots + \mathbf{N}^{k-1}\hat{\mathbf{B}}_2\mathbf{u}^{(k-1)}(0)\right).$$

For the output $\mathbf{y}_2$ we get then

$$\mathbf{y}_2 = -\tilde{\mathbf{C}}_2\hat{\mathbf{B}}_2\left(\mathbf{u} + \mathbf{N}\mathbf{u}^{(1)} + \dots + \mathbf{N}^{k-1}\mathbf{u}^{(k-1)}\right).$$

We can rewrite this as

$$\mathbf{y}_2 = \hat{\mathbf{C}}_{2,0}\mathbf{u} + \hat{\mathbf{C}}_{2,1}\mathbf{u}^{(1)} + \dots + \hat{\mathbf{C}}_{2,k-1}\mathbf{u}^{(1)}\mathbf{u}^{(k-1)}. \tag{7.26}$$

It can be proven that in this case improper Gramians occur, that also have to be balanced in addition to the proper Gramians. Briefly one has to do the following steps to deal with $\mathbf{z}_2$ and $\mathbf{y}_2$:

- Solve the Lyapunov equations of the improper type for the additional symmetric Gramians $\hat{\mathbf{W}}, \hat{\mathbf{M}}$, using $\mathbf{A} = \tilde{\mathbf{A}}_2, \mathbf{B} = \tilde{\mathbf{B}}_2, \mathbf{C} = \tilde{\mathbf{C}}_2$

  $$\hat{\mathbf{W}} - \mathbf{N}\hat{\mathbf{W}}\mathbf{N}^T = \mathbf{B}\mathbf{B}^T, \tag{7.27}$$

  $$\hat{\mathbf{M}} - \mathbf{N}^T\hat{\mathbf{M}}\mathbf{N} = \mathbf{C}^T\mathbf{C}. \tag{7.28}$$

  These equations are completely different from the normal continuous Lyapunov equations for a proper dynamical system because of the type of equation (7.26) for $\mathbf{y}_2$.

- Compute the Choleski factorisations

  $$\hat{\mathbf{W}} = \hat{\mathbf{L}}_w \hat{\mathbf{L}}_w^{\mathrm{T}}, \quad \hat{\mathbf{M}} = \hat{\mathbf{L}}_m \hat{\mathbf{L}}_m^{\mathrm{T}},$$

  where $\mathbf{L}_w, \mathbf{L}_m$ are lower triangular matrices.

- Compute also the singular value decomposition (balancing for the improper part)

  $$\hat{\mathbf{L}}_w^{\mathrm{T}} \hat{\mathbf{L}}_m = \hat{\mathbf{U}} \hat{\Sigma} \hat{\mathbf{V}}^{\mathrm{T}}.$$

- Compute the balanced matrix

  $$\mathbf{T} = \mathbf{L}_w \mathbf{U} \Sigma^{-\frac{1}{2}}.$$

  This method is called the Generalised Square Root (GSR) method [55].

- Define the truncated matrix $\mathbf{T}_r = [\mathbf{t}_1 \ \cdots \ \mathbf{t}_r]$ based on the singular values.

Thus we see that TBR applied to the improper subsystem works roughly the same. The only difference is that we have to solve the discrete Lyapunov equations instead of the previous continuous ones.

7.3 Nonlinear model order reduction

For nonlinear systems as (7.1) it is no longer possible to apply the MOR techniques described in the previous section. Then we try to exploit the (piecewise) linear structure as well as possible. The reduced model can be constructed for a benchmark simulation, such that it is accurate if the solution is in the neighbourhood of the benchmark solution. The nonlinear model can also be approximated by a piecewise linear model around a certain benchmark solution. Finally the Gramians can be estimated based on empirical data. For nonlinear systems there also exists a generalised theory, but unfortunately, this theory is not directly applicable yet.
A promising approach expresses the Gramians in terms of the correlations of the states and outputs. For LTI systems it can be proven that these empirical Gramians are completely equivalent to the normal Gramians. However, for nonlinear systems this is no longer true. Then one may collect a number of snapshots of the states and outputs to construct "empirical" Gramians. The model is constructed by TBR based on these empirical Gramians. At the end Galerkin projection is used to reduce the nonlinear model. On the one hand this Empirical Balanced Truncation is a very powerful method because it really approximates the relationship between the input and output and neglects all other phenomenons. On the other hand, the major drawback is that the reduced model still needs evaluations of the original model and is not sparse.
A third very promising approach is based on parameter extraction. One constructs a family of models [12] with preferable properties, like sparse, diagonal, decoupled,

cheap to evaluate, etc. It is important that this family still contains a number of unknown parameters that can be optimised. This optimisation is done based on a given input and output. Note that this approach also works for a set of values for the input and state (one-sided reduction). In each case, at the end we get an optimal model in the considered class that describes the wanted relationship very well. Its robustness is tested on an independent test set. This technique appears to be very similar to the theory of neural networks. The neural network is the parameterised model that describes the relationship between the input, output and parameters. In neural network theory, the parameters are called neurons, while the set of values of the input and output for which the model is constructed is called the training set. The drawback of this method is that it is expensive because it needs an optimisation with respect to the parameters. For advanced neural network models this is far from trivial [10, 34, 35]. But in many cases, the accuracy will be very good while the model also can have nice properties that makes it easy to use in practice.
Finally, it is possible to combine model order reduction with multirate time-integration, as is described in Section 5.9. Then the latent part of a multirate partition is not only solved at a coarse time-grid but also replaced by a reduced model.

7.4 Trajectory Piecewise Linear Model Order Reduction

The idea behind the *Trajectory Piecewise Linear (TPWL)* method is to linearise (7.1) several times along a given trajectory $\tilde{\mathbf{x}}(t)$ (corresponding to some typical input $\tilde{\mathbf{u}}(t)$). Define $\mathbf{y}(t) = \mathbf{x}(t) - \tilde{\mathbf{x}}(t)$ and $\mathbf{v}(t) = \mathbf{u}(t) - \tilde{\mathbf{u}}(t)$. Linearising the nonlinear equation (7.1) gives us

$$\frac{d}{dt}\mathbf{q}(t,\tilde{\mathbf{x}}) + \mathbf{j}(t,\tilde{\mathbf{x}}) + \mathbf{B}\tilde{\mathbf{u}} + \frac{d}{dt}\left[\mathbf{C}(t,\tilde{\mathbf{x}})\mathbf{y}\right] + \mathbf{G}(t,\tilde{\mathbf{x}})\mathbf{y} + \mathbf{B}\mathbf{v} = \mathbf{0}.$$

Because the trajectory $\tilde{\mathbf{x}}(t)$ satisfies

$$\frac{d}{dt}\mathbf{q}(t,\tilde{\mathbf{x}}) + \mathbf{j}(t,\tilde{\mathbf{x}}) + \mathbf{B}\tilde{\mathbf{u}} = \mathbf{0},$$

we obtain the following time-varying linear system for $\mathbf{y}(t)$

$$\frac{d}{dt}\left[\mathbf{C}(t,\tilde{\mathbf{x}}(t))\mathbf{y}(t)\right] + \mathbf{G}(t,\tilde{\mathbf{x}}(t))\mathbf{y}(t) + \mathbf{B}\mathbf{v}(t) = \mathbf{0}. \tag{7.29}$$

The main idea of TPWL is to approximate the time-varying Jacobian matrices $\mathbf{C}(t,\tilde{\mathbf{x}}(t)), \mathbf{G}(t,\tilde{\mathbf{x}}(t))$ by a weighted combination of piecewise constant matrices. Then a (finite) sequence of linearised local systems is used to create a globally reduced subspace. The final TPWL model is constructed as a weighted sum of all locally linearised reduced systems. The disadvantage of standard linearisation methods is that they only deliver good results in the neighbourhood of the chosen *linearisation tuple (LT)* $(t_i, \mathbf{x}(t_i))$. To overcome this several linearised models are created in TPWL. The procedure for selection of LTs can be described by the following steps:

1. Set an absolute accuracy factor $\varepsilon > 0$, set $i = 1$.

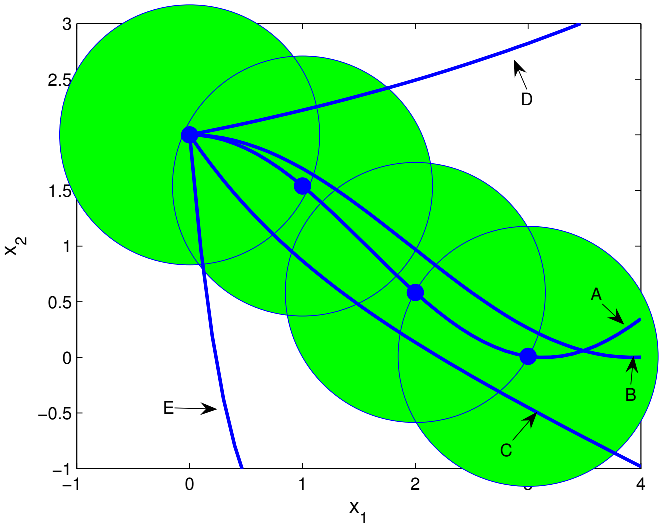


Figure 7.1: The Linearisation Tuples of this TPWL model are derived from the trajectory A. Because solutions B and C are in the neighbourhood of the surrounding balls, they can be efficiently simulated using a TPWL model. But this is not the case for the solutions D and E.

2. Linearise the system around the i-th LT $(t_i, \tilde{\mathbf{x}}_i)$. This implies:
$$\mathbf{C}_i\dot{\mathbf{y}} + \mathbf{G}_i\mathbf{y} + \mathbf{B}_i\mathbf{v}(t) = \mathbf{0}, \tag{7.30}$$
with $\mathbf{C}_i = \frac{\partial}{\partial \mathbf{x}}\mathbf{q}(t,\tilde{\mathbf{x}})\big|_{t_i,\tilde{\mathbf{x}}_i}$ and $\mathbf{G}_i = \frac{\partial}{\partial \mathbf{x}}\mathbf{j}(t,\tilde{\mathbf{x}})\big|_{t_i,\tilde{\mathbf{x}}_i}$, where $\tilde{\mathbf{x}}_i$ stays for $\tilde{\mathbf{x}}(t_i)$. Save $\mathbf{C}_i$, $\mathbf{G}_i$ and $\mathbf{B}_i$.

3. Reduce the linearised system to dimension $r \ll d$ by an appropriate linear MOR method, like "Poor Man's TBR" [38] or by Krylov-subspace methods [36]. This implies
$$\mathbf{C}_i^r\dot{\mathbf{z}} + \mathbf{G}_i^r\mathbf{z} + \mathbf{B}_i^r\mathbf{u}(t) = \mathbf{0}, \tag{7.31}$$
where $\mathbf{C}_i^r = \mathbf{V}_i^T\mathbf{C}_i\mathbf{V}$, $\mathbf{G}_i^r = \mathbf{V}_i^T\mathbf{G}_i\mathbf{V}_i$, $\mathbf{B}_i^r = \mathbf{V}_i^T\mathbf{B}$ with $\mathbf{V}_i \in \mathbb{R}^{d\times r_i}$, $\mathbf{z} \in \mathbb{R}^{r_i}$ and $\mathbf{y} \approx \mathbf{V}_i\mathbf{z}$. Save the local projection matrix $\mathbf{V}_i$.

4. Integrate both the reduced system (7.31) and the original system (7.1) choosing the same time-steps t_k. When $\frac{||\mathbf{V}_i\mathbf{z}(t_k)||}{||\tilde{\mathbf{x}}(t_k)||} > \varepsilon$ choose $(t_k, \tilde{\mathbf{x}}(t_k))$ as $(i+1)$-th LT . Set $i = i + 1$. Go to step 2.

Steps 2 to 4 are repeated until the end of the given trajectory. In this way, a finite number of locally reduced subspaces with bases $\mathbf{V}_1, ..., \mathbf{V}_s$ are created corresponding to the LTs $\{(t_1, \mathbf{x}(t_1)), \ldots, (t_s, \mathbf{x}(t_s))\}$. All locally reduced subspaces are merged into a globally reduced subspace and each locally linearised system (7.30) is now projected onto this global subspace. The procedure can be described by the following steps:

1. Define $\tilde{\mathbf{V}} = [\mathbf{V}_1, \dots, \mathbf{V}_s] \in \mathbb{R}^{d \times (r_1 + \dots + r_s)}$.
2. Calculate the SVD of $\tilde{\mathbf{V}}$: $\tilde{\mathbf{V}} = \mathbf{U}\Sigma\mathbf{W}^T$ with $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_d] \in \mathbb{R}^{d \times d}$,$\Sigma \in \mathbb{R}^{d \times \bar{r}s}$ and $\mathbf{W} \in \mathbb{R}^{\bar{r}s \times \bar{r}s}$, where $\bar{r} = (r_1 + \dots + r_s)/s$.
3. Define the new global projection matrix $\mathbf{V} \in \mathbb{R}^{d \times r}$ as $[\mathbf{u}_1, \dots, \mathbf{u}_r]$.
4. Project each local linearised system (7.30) onto $\mathbf{V}$.

All locally reduced linearised reduced systems are combined in a weighted sum to build the global TPWL model

$$\sum_{i=1}^{s} w_i(\mathbf{z}) \left[\mathbf{V}^T\mathbf{C}_i\mathbf{V}\dot{\mathbf{z}} + \mathbf{V}^T\mathbf{G}_i\mathbf{V}\mathbf{z} + w_i\mathbf{V}^T\mathbf{B}_i\mathbf{v}(t)\right] = \mathbf{0}. \tag{7.32}$$

Because of the construction of the global projection matrix $\mathbf{V}$ it is approximately true that $\mathcal{R}(\mathbf{V}_i) \subset \mathcal{R}(\mathbf{V})$ for $i = 1, \dots, s$. A weight w_i determines the influence of the i-th local system to the global system. The weights can be chosen by making them distance depending, which means that w_i is chosen large if the solution $\mathbf{z}$ of (7.31) is close to the i-th LT, else the weight should be small. For more details on how to choose weights, see [69].

7.5 Reduced DAE models by Galerkin projection

For each t let the state $\mathbf{x}(t) \in \mathbb{R}^d$ belong to a separable Hilbert space $\mathcal{X}$, equiped with the Euclidian inner product space. Then for all t the state $\mathbf{x}$ can be expanded in an orthonormal basis $\mathbf{V} = \begin{pmatrix}\mathbf{v}_1 & \dots & \mathbf{v}_d\end{pmatrix}$

$$\mathbf{x}(t) = \sum_{i \in \mathbb{I}} z_i(t)\mathbf{v}_i. \tag{7.33}$$

The orthonormal basis is derived from various criteria based on the approximation quality of the original state $\mathbf{x}$ by its truncated expansion $\mathbf{x}_r$ as defined in (7.34)

$$\mathbf{x}(t) \approx \mathbf{x}_r(t) = \sum_{i=1}^{r} z_i(t)\mathbf{v}_i. \tag{7.34}$$

The order r of the truncated expansion is lower than the order d of the original expansion. Different reduction methods yield different basis.
The reduced order model is the model that describes the dynamics of the basis coefficients or the reduced state $\mathbf{z} = \{a_1, \dots, a_r\}$. In many methods the reduced order model is derived by replacing the original state $\mathbf{x}$ by its truncated expansion $\mathbf{x}_r$ and projecting the original equations onto the truncated basis

$$\mathbf{V}_r = \begin{pmatrix}\mathbf{v}_1 & \dots & \mathbf{v}_r\end{pmatrix}.$$

This projection scheme is known as the *Galerkin projection* scheme. The resulting reduced order model defines the evolutions of the reduced state

$$\mathbf{z}(t) = \{z_1(t), \dots, z_r(t)\}.$$

Suppose the original model is a DAE model defined in (7.1). Substituting the original state $\mathbf{x}$ by its truncated state $\mathbf{x}_r$ as defined in (7.34) yields

$$\frac{d}{dt}\mathbf{q}(t, \mathbf{V}_r\mathbf{z}) + \mathbf{j}(t, \mathbf{V}_r\mathbf{z}) = \mathbf{0}. \tag{7.35}$$

Galerkin projection of (7.35) onto the truncated basis $\mathbf{V}_r = \begin{pmatrix}\mathbf{v}_1 & \cdots & \mathbf{v}_r\end{pmatrix}$ results in the reduced DAE model:

$$\frac{d}{dt}\mathbf{V}_r^T\mathbf{q}(t, \mathbf{V}_r\mathbf{z}) + \mathbf{V}_r^T\mathbf{j}(t, \mathbf{V}_r\mathbf{z}) = \mathbf{0}, \quad \mathbf{z} \in \mathbb{R}^r. \tag{7.36}$$

The original nonlinear d-dimensional DAE model is reduced to a nonlinear r-dimensional DAE reduced order model by means of the Galerkin projection. Unfortunately, the resulting reduced order model (7.36) for $\mathbf{z} \in \mathbb{R}^r$ is not always solvable for any arbitrary truncation degree r.

7.6 Proper Orthogonal Decomposition

The *proper orthogonal decomposition (POD)*, also known as the *Principal Component Analysis (PCA)* and the *Karhunen-Loéve expansion*, is a special Galerkin projection method. The POD basis $\mathbf{V}_r = \begin{pmatrix}\mathbf{v}_1 & \dots & \mathbf{v}_r\end{pmatrix}$ is an orthonormal basis and derived from the collected state evolutions (snapshots)

$$\mathbf{X} = \begin{pmatrix}\mathbf{x}(t_1) & \dots & \mathbf{x}(t_N)\end{pmatrix}.$$

The POD method is particularly popular for systems governed by nonlinear partial differential equations describing computational fluid dynamics. Analytical solutions do not exist for such systems and the collected data may serve as the only adequate description of the system dynamics. The POD basis is found by minimising the time-averaged approximation error given in (7.37)

$$\operatorname{av}\left(\| \mathbf{x}(t_k) - \mathbf{x}_n(t_k) \|_2\right). \tag{7.37}$$

The *averaging operator* $\operatorname{av}(\cdot)$ is defined as:

$$\operatorname{av}(f) := \frac{1}{N}\sum_{k=1}^{N} f(t_k). \tag{7.38}$$

Solving the minimisation problem of (7.37) is equivalent to computing the eigenvalue decomposition of $\frac{1}{N}\mathbf{X}\mathbf{X}^T$ [27]. Because $\frac{1}{N}\mathbf{X}\mathbf{X}^T$ is a symmetric positive definite matrix there exists an orthogonal matrix $\mathbf{V}_r \in \mathbb{R}^{d\times r}$ and a positive real diagonal matrix $\Lambda_r \in \mathbb{R}^{r\times r}$ such that

$$\frac{1}{N}\mathbf{X}\mathbf{X}^T\mathbf{V}_r = \mathbf{V}_r\Lambda_r. \tag{7.39}$$

The term $\frac{1}{N}\mathbf{X}\mathbf{X}^T$ equals the state covariance matrix. The POD basis is a subset of the eigenvectors of this covariance matrix and is stored by the matrix $\mathbf{V}_r$. The most important POD basis function is the eigenvector corresponding to the first eigenvalue. The truncation degree is determined from the eigenvalue distribution in $\Lambda_r = \operatorname{diag}(\lambda_1, \dots, \lambda_r)$. Based on the commonly adopted ad-hoc criterion, the truncation degree r should at least capture 99% of the total energy. The fraction of the total energy is defined as

$$P_r = \frac{\sum_{i=1}^{r} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$$

The POD basis minimises, in Least Squares sense, (7.37) over all possible bases. Error estimates for the solutions obtained from the reduced model are available in [23]. For circuit models the snapshots can be collected e.g. from a transient simulation with fixed parameters and sources. The reduced model can also be used to approximate the model for different parameters or sources as long as the solution still approximately lies in the projected space. For circuit models with a lot of redundancy the reduced model can have a much smaller dimension. Unfortunately, direct application of POD to circuit models does not work well in practice. Firstly, for Differential Algebraic Equations the Galerkin projection scheme may yield an unsolvable reduced order model. Secondly, the computational effort required to solve the reduced order model and the original model is about the same in nonlinear cases. This is due to the fact that the evaluation costs of the reduced model (7.36) are not reduced at all because $\mathbf{V}_r$ will be a dense matrix in general. For linear time-invariant DAEs of the type

$$\mathbf{C}\dot{\mathbf{x}} + \mathbf{G}\mathbf{x} = \mathbf{s} \tag{7.40}$$

the solvability depends on the matrix pencil $\lambda\mathbf{C}+\mathbf{G}$. If the set of generalised eigenvalues $\{\lambda \in \mathbb{C} : \det(\lambda\mathbf{C} + \mathbf{G}) = 0\}$ is a finite set, it follows that (7.40) has a unique solution. Because the solvability of a nonlinear DAE is hard to analyse, we restrict ourselves to the solvability of the discretised model. We assume that the numerical scheme solves the following nonlinear equation at every time step t_i

$$\lambda_i\mathbf{q}(t_i, \mathbf{x}_i) + \mathbf{j}(t_i, \mathbf{x}_i) = \mathbf{r}_i, \tag{7.41}$$

where $\mathbf{x}_i$ is the numerical approximation of $\mathbf{x}(t_i)$ and $\mathbf{r}_i$ is a known vector that may include the values of the state $\mathbf{x}$ at previous time step(s). The coefficient λ_i depends on the discretisation method used and stepsize control. Note that all Linear Multistep Methods have this property. The solvability condition of the nonlinear numerical model (7.41) depends on the implemented numerical method. Here we employ the Newton method. At every time step, the nonlinear equation (7.42) is solved iteratively for every time step.

$$\mathbf{f}(\mathbf{x}_i) = \lambda_i\mathbf{q}(t_i, \mathbf{x}_i) + \mathbf{j}(t_i, \mathbf{x}_i) - \mathbf{r}_i = \mathbf{0}. \tag{7.42}$$

Let $\mathbf{x}^l$ denote the value of $\mathbf{x}_i$ at l-th iteration step and $\mathbf{J}(\mathbf{x}^l) = \lambda_i\mathbf{C}(t_i, \mathbf{x}_i^l) + \mathbf{G}(t_i, \mathbf{x}_i^l)$ be the Jacobian matrix of the nonlinear function $\mathbf{f}$. In each iteration (7.43) is solved until convergence criteria are met

$$\mathbf{J}(\mathbf{x}^l)(\mathbf{x}^{l+1} - \mathbf{x}^l) = -\mathbf{f}(\mathbf{x}^l). \tag{7.43}$$

Clearly, if the Jacobian matrix $\mathbf{J}(\mathbf{x}^l)$ is invertible for all iteration steps, the nonlinear equation (7.41) is solvable for the time step t_i. For the reduced order model, derived by the Galerkin projection method, the reduced Jacobian matrix is $\mathbf{J}_r(\mathbf{z}^l) = \mathbf{V}_r^T \mathbf{J}(\mathbf{x}_r^l)\mathbf{V}_r$ where $\mathbf{x}_r = \mathbf{V}_r \mathbf{z}_r$. Since the matrix $\mathbf{V}_r$ is generally a rectangular matrix, the reduced Jacobian matrix is not equivalent to a similarity transformation of $\mathbf{J}(\mathbf{x}_r^l)$. Consequently, $\det(\mathbf{J}(\mathbf{x}_r^l)) \neq \det(\mathbf{J}_r(\mathbf{z}^l))$. Thus, $\mathbf{J}_r(\mathbf{z}^l)$ may not be invertible even though $\mathbf{J}(\mathbf{x}_r^l)$ is invertible. We note that this phenomenon is typical for DAEs. For ODEs $\mathbf{C} = \mathbf{I}$ and $\mathbf{J}_r(\mathbf{z}^l) = \mathbf{V}_r^T(\mathbf{I} + \mathbf{G}(t_i, \mathbf{x}_i^l))\mathbf{V}_r = \mathbf{I} + \mathbf{V}_r^T \mathbf{G}(t_i, \mathbf{x}_i^l)\mathbf{V}_r$ is always invertible.

Example 7.1 Consider problem (7.40), where

$$\mathbf{C} = \mathbf{G} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

If we take the basis

$$\mathbf{V} = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

the application of a Galerkin projection as in (7.36) leads to the following unsolvable reduced DAE:

$$0\dot{\mathbf{z}} + 0\mathbf{z} = \mathbf{s}_1.$$

This basis $\mathbf{V}$ can be obtained by POD by choosing proper input functions.

To preserve the solvability of the numerical model, the reduced Jacobian matrix $\mathbf{J}_r(\mathbf{z}^l)$ must be defined differently. Instead of the Galerkin projection (7.36) we consider the DAE (7.35) itself. Discretising this continuous problem leads for each Newton iteration to the linear system

$$\mathbf{J}(\mathbf{x}^l)(\mathbf{V}_r\mathbf{z}^{l+1} - \mathbf{V}_r\mathbf{z}^l) = -\mathbf{f}(\mathbf{V}_r\mathbf{z}^l), \tag{7.44}$$

that can be abbreviated to

$$\mathbf{M}\mathbf{V}_r\mathbf{z}^{l+1} = \mathbf{b}. \tag{7.45}$$

Of course this linear algebraic system may not be solvable if $\mathbf{V}_r$ is not invertible. However, because $\mathbf{V}_r$ is a matrix with full column rank r, it is still possible to solve the least square problem

$$\min \| \mathbf{M}\mathbf{V}_r\mathbf{z}^{l+1} - \mathbf{b} \|_2 . \tag{7.46}$$

Solving the least square problem (7.46) is equivalent to solving (7.47)

$$\underbrace{\mathbf{V}_r^T\mathbf{M}^T\mathbf{M}\mathbf{V}_r}_{\mathbf{M}_r}\mathbf{z}^{l+1} = \mathbf{V}_r^T\mathbf{M}^T\mathbf{b}. \tag{7.47}$$

This linear system can be efficiently solved by a QR factorisation of $\mathbf{M}\mathbf{V}_r$. The model (7.47) is a low dimensional model, the dimension of mass matrix $\mathbf{M}_r$ is equal to the dimension of the reduced state $\mathbf{z}$. If $\mathbf{M}$ is invertible, the invertibility is automatically preserved in the least-square reduced order model (7.47) since $\mathbf{M}_r$ is a symmetric positive definite matrix and therefore invertible.

7.7 The Missing Point Estimation (MPE)

For linear time-invariant systems the current MOR techniques are able to reduce systems of very large sizes by means of Krylov-space methods. For moderate sizes also control-theory methods, like TBR, can be applied which have some superior properties (also error estimates available) [1]. For nonlinear systems, there are attempts to generalise the linear theory, but with little success for large dimensions [47]. Other methods are based on a known trajectory, like Trajectory PieceWise Linear, Proper Orthogonal Decomposition (POD) and Empirical Balanced Truncation (EBT) [2,41,69]. The two last methods have in common that a matrix $\mathbf{V}$ is computed which range contains the dominant part of the solution $\mathbf{x}$. Let the given model be a differential-algebraic equation (DAE) of the type

$$\frac{d}{dt}\mathbf{q}(t,\mathbf{x}) + \mathbf{j}(t,\mathbf{x}) = \mathbf{0}. \tag{7.48}$$

Then the reduced model is constructed by use of Galerkin projection:

$$\frac{d}{dt}\mathbf{V}^T\mathbf{q}(t,\mathbf{V}\mathbf{z}) + \mathbf{V}^T\mathbf{j}(t,\mathbf{V}\mathbf{z}) = \mathbf{0}. \tag{7.49}$$

The original state can be obtained by $\mathbf{x} = \mathbf{V}\mathbf{z}$. Thus indeed it is assumed that $\mathbf{x} \in \mathcal{R}(\mathbf{V})$. If $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{V} \in \mathbb{R}^{d\times r}$ where $r \ll d$ it is clear that the reduced model (7.49) is of much smaller size than the original model (7.48). For LTI systems with $\mathbf{q}(t,\mathbf{x}) = \mathbf{C}\mathbf{x}$ and $\mathbf{j}(t,\mathbf{x}) = \mathbf{G}\mathbf{x} - \mathbf{s}(t)$ it is really possible to reduce the simulation time for small r. In particular if the reduced model is diagonalised, we certainly get a model that is very cheap to solve. For the general case it is much worse because then the evaluation costs are not reduced at all. But if the linear algebra part is dominant, we still can expect any speed-up. Despite the resulting low dimensional model, the computational effort required to solve the reduced order model and the original model is relatively the same in nonlinear cases. It may even occur that the original model is cheaper to evaluate than the reduced order model. The low dimensionality is obtained by means of projection, either by the Galerkin projection method or the least square method. In the projection schemes, the original numerical model must be projected onto the projection space. It implies that the original model must be re-evaluated when the original numerical model is time-varying, which is the general case for nonlinear systems. A consequence is that the evaluation costs for the reduced model are not reduced at all.

Missing Point Estimation (MPE) is a well-known technique that modifies the matrix $\mathbf{V}$ such that only a part of the equations of the original model have to be evaluated. This makes POD applicable for model order reduction of nonlinear DAEs. For POD each Newton iteration the Jacobian matrix $\mathbf{J} = \mathbf{J}(\mathbf{x}^l)$ and the right-hand-side $\mathbf{f} = \mathbf{f}(\mathbf{x}^l)$ of the large-dimensional original DAE (7.48) have to be evaluated. The Missing Point Estimation (MPE) was proposed in [2] as a method to reduce the computational cost of reduced order, nonlinear, time-varying model. The method is inspired by the Gappy-POD approach that was introduced by Everson and Sirovich in [11]. Provided that the original, high-dimensional state $\mathbf{x} \in \mathbb{R}^d$ can be approximated closely by r POD basis functions, where $r \ll d$, it follows that the POD coefficients $\mathbf{z}$ can also be estimated from the knowledge of $r \ll d$ state variables or data points only [27]. Suppose that the POD coefficients are estimated from $g < d$ state variables. Let $\mathbf{P} \in \{0,1\}^{g\times d}$ be a selection matrix of full rank, where $g \ll d$, and define the restricted basis $\tilde{\mathbf{V}}_r$ as

$$\tilde{\mathbf{V}}_r = \mathbf{P}\mathbf{V}_r. \tag{7.50}$$

Note that it always holds that $\mathbf{P}\mathbf{P}^{\mathsf{T}} = \mathbf{I}_g$. Corresponding to the restricted basis $\tilde{\mathbf{V}}_r$, introduce the restricted state $\tilde{\mathbf{x}} \in \mathbb{R}^g$ that is defined as

$$\tilde{\mathbf{x}} = \mathbf{P}\mathbf{x} \tag{7.51}$$

Accordingly, the restricted state $\tilde{\mathbf{x}}_r$ can be approximated by the expansion of the restricted basis $\tilde{\mathbf{V}}_r$

$$\tilde{\mathbf{x}} \approx \tilde{\mathbf{x}}_n = \tilde{\mathbf{V}}_r\mathbf{z}_r$$

The paper [11] showed that the coefficients $\mathbf{z}_r$ for a given restricted state $\tilde{\mathbf{x}}$ and restricted basis $\tilde{\mathbf{V}}$ can be estimated by minimising (7.52).

$$\| \, \tilde{\mathbf{x}} - \tilde{\mathbf{V}}_r\tilde{\mathbf{z}}_r \, \|_2 \tag{7.52}$$

The coefficients $\tilde{\mathbf{z}}_r$ are the POD coefficients estimated from the knowledge of the restricted state $\tilde{\mathbf{x}}$. This idea is then extended for dynamical systems in [2]. This extended approach is referred to as the Missing Point Estimation (MPE) method. In MPE, the model of the original state $\mathbf{x}$ is given. The dynamics of the restricted state $\tilde{\mathbf{x}}$ is also given. Note that in the Gappy-POD approach, the restricted state $\tilde{\mathbf{x}}$ is exact, while here we only consider the dynamical model of the restricted state. Recall that in order to solve the original model, we solve the following equation at every iteration step l

$$\mathbf{J}(\mathbf{x}^l)(\mathbf{x}^{l+1} - \mathbf{x}^l) = -\mathbf{f}(\mathbf{x}^l), \tag{7.53}$$

In MPE, we only solve a part of (7.53). Similar to the formulation of the restricted basis and the restricted state, we multiply both sides of (7.53) by the selection matrix $\mathbf{P} \in \mathbb{R}^{g \times d}$.

$$\mathbf{P}\mathbf{J}(\mathbf{x}^l)(\mathbf{x}^{l+1} - \mathbf{x}^l) = -\mathbf{P}\mathbf{f}(\mathbf{x}^l). \tag{7.54}$$

Note that in the MPE case, we only need the evaluations of the restricted Jacobian matrix $\mathbf{PJ}$ and forcing term $\mathbf{Pf}$. As before in (7.44), the model (7.54) can be rewritten into

$$\mathbf{P}\mathbf{M}\mathbf{V}\mathbf{z}^{l+1} = \mathbf{P}\mathbf{b}. \tag{7.55}$$

Introduce $\mathbf{P}_{nb} = \{0, 1\}^{g_{nb} \times d}$ as another selection matrix. The selection matrix $\mathbf{P}_{nb}$ is introduced in order to include the state variables that contribute to the dynamics of the restricted state $\tilde{\mathbf{x}}$ but do not belong to the restricted state $\tilde{\mathbf{x}}$. Hence, let

$$\tilde{\mathbf{x}}_{nb} = \mathbf{P}_{nb}\tilde{\mathbf{x}} = \{\tilde{\mathbf{x}}, \tilde{\mathbf{x}}_{\mathbf{c}}\},$$

where $\tilde{\mathbf{x}}_c$ are the variables that contribute to the dynamics of $\tilde{\mathbf{x}}$, $G \leq \dim \tilde{\mathbf{x}}_{nb} \leq d$. Let

$$\tilde{\mathbf{M}} = \mathbf{P}\mathbf{M}, \quad \tilde{\mathbf{b}} = \mathbf{P}\mathbf{b}, \tilde{\mathbf{V}}_r^{nb} = \mathbf{P}_{nb}\mathbf{V} \tag{7.56}$$

Solving (7.55) as a least squares problem and rewriting the appropriate terms based on the redefinitions in (7.56) lead to the following MPE reduced order model

$$\underbrace{\tilde{\mathbf{V}}^{\top}_{r^{nb}}\tilde{\mathbf{M}}^{\top}\tilde{\mathbf{M}}\tilde{\mathbf{V}}^{nb}_{n}}_{\mathbf{M}_r}\tilde{\mathbf{z}}^{l+1} = \tilde{\mathbf{V}}^{\top}_{r^{nb}}\tilde{\mathbf{M}}^{\top}\tilde{\mathbf{b}}. \tag{7.57}$$

Upon obtaining the estimated POD coefficients $\tilde{\mathbf{a}}$ from (7.57), the complete state $\mathbf{x}$ can be approximated using the complete POD basis $\mathbf{V}_r$ as $\hat{\mathbf{x}}$ presented in (7.58)

$$\mathbf{x} \approx \hat{\mathbf{x}} = \mathbf{V}_r \tilde{\mathbf{z}}. \tag{7.58}$$

The model (7.57) is solvable if $\mathbf{M}_r$ is invertible. The condition will imply that $\tilde{\mathbf{M}} = \mathbf{PM}$ should be full rank. It is difficult to verify this requirement as $\tilde{\mathbf{M}}$ will change when the operating condition changes. Ideally, a new set of state variables should be chosen whenever the operating condition changes. This problem is simplified by assuming that the selected state variables $\tilde{\mathbf{x}}$ will still be representative for the operating region we are interested in. Instead of imposing the rank condition on $\tilde{\mathbf{M}}$, we impose the condition of full row rank on $\tilde{\mathbf{V}}_r^{nb}$. It means that the selected state $\tilde{\mathbf{x}}$ should comprise of state variables whose dynamics are independent of each other. Missing Point Estimation introduces a new contribution to the error, that is also called the *aliasing error*. In this section, we will use the criterion proposed in [5] and [4] on minimising the aliasing error resulting from using $\tilde{\mathbf{x}}$. This is equivalent to the minimisation of the following norm

$$\| \left(\tilde{\mathbf{V}}_r^T \tilde{\mathbf{V}}_r \right)^{-1} - \mathbf{I}_r \|, \tag{7.59}$$

where $\mathbf{I}_r$ is an identity matrix and

$$\tilde{\mathbf{V}} = \begin{pmatrix} \tilde{\mathbf{v}}_1 & \dots & \tilde{\mathbf{v}}_r \end{pmatrix} = \mathbf{P}\mathbf{V}_r.$$

Note that (7.59) tries to retain as much as possible of the orthogonality of $\tilde{\mathbf{V}}_r$. The loss is due to the selection process. Thus we get the following optimisation problem

$$\begin{array}{rc} \text{find} & \mathbf{P} \\ \text{such that} & \| \left(\mathbf{V}_r^T \mathbf{P}^T \mathbf{P} \mathbf{V}_r \right)^{-1} - \mathbf{I}_r \| < \text{TOL}, \\ \text{subject to} & \mathbf{P}\mathbf{P}^T = \mathbf{I}. \\ & \mathbf{P} \in \{0,1\}^{g \times d}. \end{array} \tag{7.60}$$

There exist various methods to solve this optimisation problem (7.60). With regard to the reduction of overall computational complexities, preference will be generally given to non-combinatorial methods. We will use the iterative version of the greedy algorithm [74]. Note that the constraint ensures that the matrix $\left(\mathbf{V}_r^T \mathbf{P}^T \mathbf{P} \mathbf{V}_r \right)$ is well conditioned. Therefore, it is bounded to above by

$$\text{cond}(\mathbf{V}_r^T \mathbf{P}^T \mathbf{P} \mathbf{V}_r) < \text{TOL}.$$

We saw that the found orthogonal matrix $\mathbf{V}$ by POD was not really optimal to get a reduced model with a low-to-moderate complexity to evaluate. Therefore methods like Missing Point Estimation are necessary to make POD attractive for model reduction of non-linear problems. Note that for POD we compute the following eigendecomposition:

$$\frac{1}{N} \mathbf{X} \mathbf{X}^T = \mathbf{U} \Sigma^2 \mathbf{U}^T,$$

where $\mathbf{U}$ is orthogonal and Σ is diagonally positive real. For Missing Point Estimation we assumed that $\mathbf{P}^T\bar{\mathbf{V}} = \mathbf{P}^T\mathbf{P}\mathbf{V}$ is a good approximation of $\mathbf{V}$ itself, where $\mathbf{P} \in \mathbb{R}^{g\times d}$, $\bar{\mathbf{V}} \in \mathbb{R}^{g\times r}$. Thus

$$\frac{1}{N}\mathbf{X}\mathbf{X}^T \approx \mathbf{P}^T\bar{\mathbf{V}}\Sigma^2\bar{\mathbf{V}}^T\mathbf{P}.$$

Clearly this approximation is only valid if the most unselected elements of $\mathbf{X}\mathbf{X}^T$ equal zero.

7.8 Interpolation of function snapshots

A new alternative compared to MPE is to create an approximate model by using *interpolation of function snapshots* along a given trajectory. Define

$$\bar{\mathbf{q}} = \mathbf{P}\mathbf{q}, \quad \bar{\mathbf{j}} = \mathbf{P}\mathbf{j}, \quad \bar{\mathbf{x}} = \mathbf{P}\mathbf{x},$$

where $\mathbf{P} \in \{0,1\}^{g\times d}$ is a selection matrix with $\mathbf{P}\mathbf{P}^T = \mathbf{I}_g$. Now we assume that

$$\mathbf{q} \approx \mathbf{V}_q\bar{\mathbf{q}}, \quad \mathbf{j} \approx \mathbf{V}_j\bar{\mathbf{j}}, \quad \mathbf{x} \approx \mathbf{V}_x\bar{\mathbf{x}}.$$

Thus we assume that the elements of $\bar{\mathbf{x}}, \bar{\mathbf{q}}, \bar{\mathbf{j}}$ form a basis for all elements of $\mathbf{x}, \mathbf{q}, \mathbf{j}$. This is a very reasonable assumption for models with much redundancy. Consider the collected "snapshots" at the time points t_i of $\mathbf{q}, \mathbf{j}$ and $\mathbf{x}$

$$\begin{array}{rcll}\mathbf{Q} & = & [\mathbf{q}(t_1,\mathbf{x}(t_1))\dots\mathbf{q}(t_s,\mathbf{x}(t_s))] & \in \mathbb{R}^{n\times s},\\ \mathbf{J} & = & [\mathbf{j}(t_1,\mathbf{x}(t_1))\dots\mathbf{j}(t_s,\mathbf{x}(t_s))] & \in \mathbb{R}^{n\times s},\\ \mathbf{X} & = & [\mathbf{x}(t_1)\dots\mathbf{x}(t_s)] & \in \mathbb{R}^{n\times s}.\end{array} \tag{7.61}$$

For a fixed selection matrix $\mathbf{P}$ it is not optimal to compute the singular value decomposition of these matrices $\mathbf{Q}, \mathbf{J}, \mathbf{X}$. Define the matrices with restricted snapshote

$$\bar{\mathbf{Q}} = \mathbf{P}\mathbf{Q} \in \mathbb{R}^{g\times s}, \quad \bar{\mathbf{J}} = \mathbf{P}\mathbf{J} \in \mathbb{R}^{g\times s}, \quad \bar{\mathbf{X}} = \mathbf{P}\mathbf{X} \in \mathbb{R}^{g\times s}.$$

We want to approximate $\mathbf{Q}, \mathbf{J}, \mathbf{X}$ by

$$\hat{\mathbf{Q}} = \mathbf{V}_q\bar{\mathbf{Q}}, \quad \hat{\mathbf{J}} = \mathbf{V}_j\bar{\mathbf{J}}, \quad \hat{\mathbf{X}} = \mathbf{V}_x\bar{\mathbf{X}}.$$

The still unknown matrices $\mathbf{V}_q, \mathbf{V}_j, \mathbf{V}_x \in \mathbb{R}^{d\times g}$ should minimise the residuals of the following equations

$$\hat{\mathbf{Q}} = \mathbf{V}_q\bar{\mathbf{Q}} = \mathbf{Q}, \quad \hat{\mathbf{J}} = \mathbf{V}_j\bar{\mathbf{J}} = \mathbf{J}, \quad \hat{\mathbf{X}} = \mathbf{V}_x\bar{\mathbf{X}} = \mathbf{X}.$$

Because these equations are overdetermined for $g < d$ we have to use least squares. Note that $\mathbf{V}_q$ has to satisfy

$$\bar{\mathbf{Q}}^T\mathbf{V}_q^T = \mathbf{Q}^T$$

and similarly for $\mathbf{V}_j$ and $\mathbf{V}_x$. Thus the least squares solution for $\mathbf{V}_q^T$ equals

$$\mathbf{V}_q^T = \left[\bar{\mathbf{Q}}\bar{\mathbf{Q}}^T\right]^{-1}\bar{\mathbf{Q}}\mathbf{Q}^T.$$

Hence we get the following three best approximations for $\mathbf{V}_q, \mathbf{V}_j, \mathbf{V}_x$ if $\mathbf{P}$ is given

$$\begin{cases} \mathbf{V}_q &= \mathbf{Q}\bar{\mathbf{Q}}^T \left[\bar{\mathbf{Q}}\bar{\mathbf{Q}}^T\right]^{-T}, \\ \mathbf{V}_j &= \mathbf{J}\bar{\mathbf{J}}^T \left[\bar{\mathbf{J}}\bar{\mathbf{J}}^T\right]^{-T}, \\ \mathbf{V}_x &= \mathbf{X}\bar{\mathbf{X}}^T \left[\bar{\mathbf{X}}\bar{\mathbf{X}}^T\right]^{-T}. \end{cases} \tag{7.62}$$

Finally we get the following approximations of $\mathbf{Q}, \mathbf{J}, \mathbf{X}$

$$\begin{cases} \hat{\mathbf{Q}} &= \mathbf{Q}\bar{\mathbf{Q}}^T \left[\bar{\mathbf{Q}}\bar{\mathbf{Q}}^T\right]^{-T} \bar{\mathbf{Q}}, \\ \hat{\mathbf{J}} &= \mathbf{J}\bar{\mathbf{J}}^T \left[\bar{\mathbf{J}}\bar{\mathbf{J}}^T\right]^{-T} \bar{\mathbf{J}}, \\ \hat{\mathbf{X}} &= \mathbf{X}\bar{\mathbf{X}}^T \left[\bar{\mathbf{X}}\bar{\mathbf{X}}^T\right]^{-T} \bar{\mathbf{X}}. \end{cases} \tag{7.63}$$

It follows from (7.63) that

$$\hat{\mathbf{X}}\hat{\mathbf{X}}^T = \mathbf{X}\bar{\mathbf{X}}^T \left[\bar{\mathbf{X}}\bar{\mathbf{X}}^T\right]^{-T} \bar{\mathbf{X}}\bar{\mathbf{X}}^T \left[\bar{\mathbf{X}}\bar{\mathbf{X}}^T\right]^{-1} \bar{\mathbf{X}}\mathbf{X}^T = \mathbf{X}\bar{\mathbf{X}}^T \left[\bar{\mathbf{X}}\bar{\mathbf{X}}^T\right]^{-T} \bar{\mathbf{X}}\mathbf{X}^T, \text{ etc.}$$

In general $\hat{\mathbf{Q}}, \hat{\mathbf{J}}, \hat{\mathbf{X}}$ will preserve the structure of the original model, like sparsity, much better than the matrices obtained by a singular value decomposition. Now we replace the original model (7.48) of size n by the following reduced model of size g:

$$\frac{d}{dt}\mathbf{V}_x^T\mathbf{V}_q\bar{\mathbf{q}}(t, \mathbf{V}_x\bar{\mathbf{x}}) + \mathbf{V}_x^T\mathbf{V}_j\bar{\mathbf{j}}(t, \mathbf{V}_x\bar{\mathbf{x}}) = \mathbf{0}, \quad \bar{\mathbf{x}} \in \mathbb{R}^g. \tag{7.64}$$

The projection matrices $\mathbf{V}_x, \mathbf{V}_q, \mathbf{V}_j \in \mathbb{R}^{d\times g}$ are defined in (7.62). Note that for this approach all elements of $\bar{\mathbf{x}}$ still have their original interpretation. It is even possible to generalise this approach by using different selection matrices $\mathbf{P}_q \in \{0,1\}^{g_q\times d}, \mathbf{P}_j \in \{0,1\}^{g_j\times d}, \mathbf{P}_x \in \{0,1\}^{g_x\times d}$ for $\mathbf{q}, \mathbf{j}$ and $\mathbf{x}$. To get a solvable system it is required that $g_x \leq \min\{g_q, g_j\}$. This approach replaces the original functions $\mathbf{q}, \mathbf{j}$ by $\mathbf{V}_q\bar{\mathbf{q}}, \mathbf{V}_j\bar{\mathbf{j}}$ and makes Missing Point Estimation not necessary. Indeed the evaluation costs of $\mathbf{V}_x^T\mathbf{V}_q\bar{\mathbf{q}}(t, \mathbf{V}_x\bar{\mathbf{x}}) = \mathbf{V}_x^T\mathbf{V}_q\mathbf{P}\mathbf{q}(t, \mathbf{V}_x\mathbf{P}\mathbf{x})$ is about similar as for MPE with $\hat{\mathbf{V}}^T\mathbf{q}(t, \tilde{\mathbf{V}}\bar{\mathbf{x}}) = \bar{\mathbf{V}}^T\mathbf{P}\mathbf{q}(t, \mathbf{P}^T\bar{\mathbf{V}}\bar{\mathbf{x}})$. In both cases only a part of the equations has to be evaluated. The difference is that with MPE for each equation only a part of the unknowns is used. Furthermore, the Jacobian matrix evaluations of MPE are cheaper than for this method. For MPE we had $\hat{\mathbf{V}}^T\mathbf{C}(t, \tilde{\mathbf{V}}\bar{\mathbf{x}})\tilde{\mathbf{V}} = \bar{\mathbf{V}}^T\mathbf{P}\mathbf{C}(t, \mathbf{P}^T\bar{\mathbf{V}}\bar{\mathbf{x}})\mathbf{P}^T\bar{\mathbf{V}}$, that only evaluates a submatrix of $\mathbf{C}$ of size $g \times g$. For the new proposed method we need to evaluate $\mathbf{V}_x^T\mathbf{V}_q\bar{\mathbf{C}}(t, \mathbf{V}_x\bar{\mathbf{x}})\mathbf{V}_x = \mathbf{V}_x^T\mathbf{V}_q\mathbf{P}\mathbf{C}(t, \mathbf{V}_x\mathbf{P}\mathbf{x})\mathbf{V}_x\mathbf{P}$. Now a submatrix of $\mathbf{C}$ of size $g \times n$ has to be evaluated. But in practice the matrices $\mathbf{C}, \mathbf{G}$ are often sparse such that this is not a very large problem. For systems with dense Jacobian matrices we should use a different approximation of $\mathbf{x}$. Instead of $\mathbf{x} \approx \mathbf{V}_x\mathbf{P}_x\mathbf{x}$ we should look at

$$\mathbf{x} \approx \mathbf{P}_x^T\mathbf{W}_x\mathbf{x},$$

where $\mathbf{W}_x \in \mathbb{R}^{g_x\times d}$. For a given snapshot matrix $\mathbf{X} \in \mathbb{R}^{d\times s}$ we get the following equation for $\mathbf{W}_x$

$$\mathbf{P}_x^T\mathbf{W}_x\mathbf{X} = \mathbf{X}.$$

This is again an overdetermined system that can be approximated by

$$\mathbf{P}_x \mathbf{P}_x^T \mathbf{W}_x \mathbf{X} = \mathbf{P}_x \mathbf{X}.$$

Because $\mathbf{P}_x \mathbf{P}_x^T = \mathbf{I}_g$ and $\mathbf{X}$ is invertible it follows that

$$\mathbf{W}_x = \mathbf{P}_x$$

is a selection matrix. Now we get the system

$$\frac{d}{dt} \mathbf{P}_x^T \mathbf{V}_q \bar{\mathbf{q}}(t, \mathbf{P}_x \bar{\mathbf{x}}) + \mathbf{P}_x^T \mathbf{V}_j \bar{\mathbf{j}}(t, \mathbf{P}_x \bar{\mathbf{x}}) = \mathbf{0}, \quad \bar{\mathbf{x}} \in \mathbb{R}^g. \tag{7.65}$$

Indeed the Jacobian matrices of this reduced system are cheap to evaluate because the evaluation of $\mathbf{P}_x^T \mathbf{V}_q \bar{\mathbf{C}}(t, \mathbf{P}_x \bar{\mathbf{x}}) \mathbf{P}_x$ only needs a submatrix of $\mathbf{C}$ of size $g \times g$. The previous method computes two different matrices $\mathbf{V}_q, \mathbf{V}_j$ that span the function values of $\mathbf{q}$ and $\mathbf{j}$. In each timestep a nonlinear system has to be solved of the type:

$$\alpha_n \mathbf{q}(\mathbf{x}_n) + h_n \mathbf{j}(\mathbf{x}_n) = \mathbf{r}_n.$$

An interesting alternative is to construct a matrix $\mathbf{V}_{qj} \in \mathbb{R}^{n \times (g_q + g_j)}$ that spans the dominant part of the range of both matrices. Then we get the following reduced model

$$\frac{d}{dt} \mathbf{W} \bar{\mathbf{q}}(t, \mathbf{V}_x \bar{\mathbf{x}}) + \mathbf{W} \bar{\mathbf{j}}(t, \mathbf{V}_x \bar{\mathbf{x}}) = \mathbf{0}, \quad \bar{\mathbf{x}} \in \mathbb{R}^{g_x}, \tag{7.66}$$

where $\mathbf{W} = \mathbf{V}_x^T \mathbf{V}_{qj} \in \mathbb{R}^{g_x \times (g_q + g_j)}$. If also $g_x = g_q + g_j$ it follows that $\mathbf{W}$ is invertible. Then we get the following equivalent system:

$$\frac{d}{dt} \bar{\mathbf{q}}(t, \mathbf{V}_x \bar{\mathbf{x}}) + \bar{\mathbf{j}}(t, \mathbf{V}_x \bar{\mathbf{x}}) = \mathbf{0}, \quad \bar{\mathbf{x}} \in \mathbb{R}^{g_q + g_j}. \tag{7.67}$$

Although this model is not of optimal size, we do not need the additional matrix multiplication with $\mathbf{W}$, which makes it cheaper to evaluate. Note that this approach also works well to detect the active part for a multirate-partition. A second alternative is to use POD with $\mathbf{V}_x = \mathbf{V} \in \mathbb{R}^{d \times r}$ by use of a singular value decomposition of $\mathbf{X}$. Then we get a reduced model of size r instead of g.

$$\frac{d}{dt} \mathbf{V}^T \mathbf{V}_q \bar{\mathbf{q}}(t, \mathbf{V} \bar{\mathbf{x}}) + \mathbf{V}^T \mathbf{V}_j \bar{\mathbf{j}}(t, \mathbf{V} \bar{\mathbf{x}}) = \mathbf{0}, \quad \bar{\mathbf{x}} \in \mathbb{R}^r.$$

To get a solvable system it is always required that $r \leq g$. The advantage of this approach is that in general $r \ll g$, but a possible disadvantage is that now the elements of $\bar{\mathbf{x}}$ have lost their original interpretation. Also here it is possible to use different selection matrices $\mathbf{P}_q, \mathbf{P}_j$ provided that $r \leq \min\{g_q, g_j\}$. For systems with dense Jacobian matrices it again is possible to approximate $\mathbf{V}$ by use of Missing Point Estimation. Finally we can combine all approaches with the LS-Galerkin MOR method. Galerkin projection can have the drawback that it leads to unsolvable reduced models. This problem might in particular occur for differential-algebraic systems. An alternative could be to consider instead the following overdetermined system for $\bar{\mathbf{x}}$:

$$\frac{d}{dt} \mathbf{V}_q \bar{\mathbf{q}}(t, \mathbf{V}_x \bar{\mathbf{x}}) + \mathbf{V}_j \bar{\mathbf{j}}(t, \mathbf{V}_x \bar{\mathbf{x}}) = \mathbf{0}. \tag{7.68}$$

For each Newton iteration we get the overdetermined linear system

$$\underbrace{\left[\lambda_n \mathbf{V}_q \bar{\mathbf{C}}_{n,k} + \mathbf{V}_j \bar{\mathbf{G}}_{n,k}\right]}_{\mathbf{M}_{n,k}} \mathbf{V}_x \Delta \mathbf{x}_{n,k+1} = -\mathbf{r}_{n,k}.$$

If the rank of this matrix is equal to g_x it is possible to use Least Squares, that minimises the residual. Then we get

$$\mathbf{V}_x^T \mathbf{M}_{n,k}^T \mathbf{M}_{n,k} \mathbf{V}_x \Delta \mathbf{x}_{n,k+1} = -\mathbf{V}_x^T \mathbf{M}_{n,k}^T \mathbf{r}_{n,k}.$$

Note that

$$\mathbf{V}_x^T \mathbf{M}_{n,k}^T \mathbf{M}_{n,k} \mathbf{V}_x = \mathbf{V}_x^T \left[\lambda_n \bar{\mathbf{C}}_{n,k}^T \mathbf{V}_q^T + \bar{\mathbf{G}}_{n,k}^T \mathbf{V}_j^T\right] \left[\lambda_n \mathbf{V}_q \bar{\mathbf{C}}_{n,k} + \mathbf{V}_j \bar{\mathbf{G}}_{n,k}\right] \mathbf{V}_x$$

and

$$\mathbf{V}_x^T \mathbf{M}_{n,k}^T \mathbf{r}_{n,k} = \mathbf{V}_x^T \left[\lambda_n \bar{\mathbf{C}}_{n,k}^T \mathbf{V}_q^T + \bar{\mathbf{G}}_{n,k}^T \mathbf{V}_j^T\right] \mathbf{r}_{n,k}.$$

Thus we can also apply Least Squares only based on evaluations of $\bar{\mathbf{q}}, \bar{\mathbf{j}}, \bar{\mathbf{C}}$ and $\bar{\mathbf{G}}$.
We showed before how the find an optimal approximation of the original matrices $\mathbf{Q}, \mathbf{J}, \mathbf{X}$ of the form $\mathbf{V}_q \mathbf{P}_q \mathbf{Q}$, etc. For a given selection matrix $\mathbf{P}_q$ we derived the optimal choice of $\mathbf{V}_q$. If the sizes g_q, g_j, g_x are fixed, we have to find optimal selection matrices $\mathbf{P}_q, \mathbf{P}_j, \mathbf{P}_x$ for which the errors of $\hat{\mathbf{Q}}, \hat{\mathbf{J}}, \hat{\mathbf{X}}$ are minimal. From (7.63) we can derive the following error bounds

$$\left\{ \begin{array}{rcl} \|\hat{\mathbf{Q}} - \mathbf{Q}\| & \leq & \|\mathbf{Q}\| \|\mathbf{Q}\mathbf{Q}^T \mathbf{P}_q^T \left[\mathbf{P}_q \mathbf{Q}\mathbf{Q}^T \mathbf{P}_q^T\right]^{-1} \mathbf{P}_q - \mathbf{I}_d\|, \\ \|\hat{\mathbf{J}} - \mathbf{J}\| & \leq & \|\mathbf{J}\| \|\mathbf{J}\mathbf{J}^T \mathbf{P}_j^T \left[\mathbf{P}_j \mathbf{J}\mathbf{J}^T \mathbf{P}_j^T\right]^{-1} \mathbf{P}_j - \mathbf{I}_d\|, \\ \|\hat{\mathbf{X}} - \mathbf{X}\| & \leq & \|\mathbf{X}\| \|\mathbf{X}\mathbf{X}^T \mathbf{P}_x^T \left[\mathbf{P}_x \mathbf{X}\mathbf{X}^T \mathbf{P}_x^T\right]^{-1} \mathbf{P}_x - \mathbf{I}_d\|. \end{array} \right. \tag{7.69}$$

Thus the errors are small if $\mathbf{Q}\mathbf{Q}^T \mathbf{P}_q^T \left[\mathbf{P}_q \mathbf{Q}\mathbf{Q}^T \mathbf{P}_q^T\right]^{-1} \mathbf{P}_q \approx \mathbf{I}_d$, etc. For $g_q = q_j = g_x = n$ the optimal choice is $\mathbf{P}_q = \mathbf{P}_j = \mathbf{P}_x = \mathbf{I}_d$ because then

$$\mathbf{Q}\mathbf{Q}^T \mathbf{P}_q^T \left[\mathbf{P}_q \mathbf{Q}\mathbf{Q}^T \mathbf{P}_q^T\right]^{-1} \mathbf{P}_q = \mathbf{Q}\mathbf{Q}^T \left[\mathbf{Q}\mathbf{Q}^T\right]^{-1} = \mathbf{I}_d, \text{ etc.}$$

To obtain lower computational costs it is necessary to find some suitable selection matrices for which these errors are minimised. Note that for suitable matrices it holds that

$$\mathbf{P}_q^T \left[\mathbf{P}_q \mathbf{Q}\mathbf{Q}^T \mathbf{P}_q^T\right]^{-1} \mathbf{P}_q \approx \left[\mathbf{Q}\mathbf{Q}^T\right]^{-1}, \text{ etc.}$$

A complication is that for DAEs in general the matrix $\mathbf{Q}\mathbf{Q}^T$ will be singular. In general it is a hard problem to find an optimal $\mathbf{P}_q$ for fixed g_q. We can approximate it by a greedy method that selects the rows iteratively. Each added row should minimise the errors in (7.69). To this end, for a fixed g, we are looking for a selection matrix $\mathbf{P}$ for which the upperbound of $\|\hat{\mathbf{Q}} - \mathbf{Q}\|$ is minimised. Denote $\mathbf{A} = \mathbf{Q}\mathbf{Q}^T \geq 0$, then we get the following discrete optimisation problem:

$$\min_{\mathbf{P}} \|\mathbf{A}\mathbf{P}^T \left[\mathbf{P}\mathbf{A}\mathbf{A}^T\right]^{-T} \mathbf{P} - \mathbf{I}\| = \min_{\mathbf{P}} \|\mathbf{A}\mathbf{P}^T \left[\mathbf{P}\mathbf{A}\mathbf{P}^T\right]^{-1} \mathbf{P} - \mathbf{I}\|. \tag{7.70}$$

Note that $\mathbf{P}^T \left[\mathbf{P}\mathbf{A}\mathbf{P}^T\right]^{-1} \mathbf{P}$ is an approximation of $\mathbf{A}^{-1}$.

Chapter 8

Applications

In order to test the previous theoretical results we apply it to a large number of test examples both in MATLAB and in Pstar (the in-house analogue circuit simulator provided by NXP Semiconductors). Section 8.1 contains for several examples the results of multirate transient simulations compared with single-rate transient simulation. Section 8.2 shows how nonlinear model reduction can be used to reduce the simulation time of several nonlinear circuit models. We compare all results with respect to the required CPU time, accuracy and overhead costs.

8.1 Multirate experiments

First we show some typical features of the multirate methods for two academic nonlinear test examples. The last two circuit models are real-world circuit designs that have been modelled and simulated with the multirate implementation within Pstar. More information about these numerical experiments can be found in [61, 67].

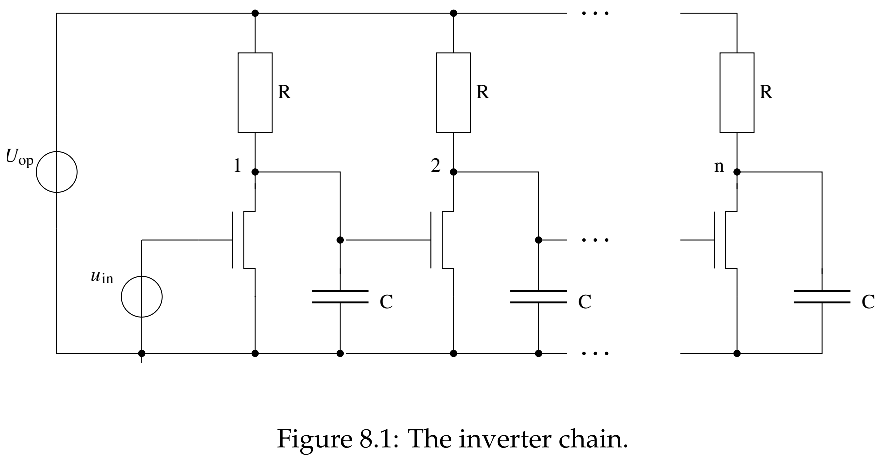


Figure 8.1: The inverter chain.

Consider the circuit model of an inverter chain described in more detail in [7] and of which the diagram is shown in Fig. 8.1. The inverter chain is a concatenation of 500 inverters in this case and is built from the combination of resistors and capacitors. The function of an inverter is to invert and smoothen the incoming signal. The output of an inverter will be a delayed, inverted signal. The time scale of an inverter chain ranges between $30 - 200$ nanoseconds due to the high frequency sampling.
The state of the inverter chain model comprises of 500 nodal voltages and some electrical currents. The excitation signal u_{in} for the inverter chain is depicted in Figure 8.2. The dynamical response of the inverter chain for the first 20 ns is shown in Figure 8.3,

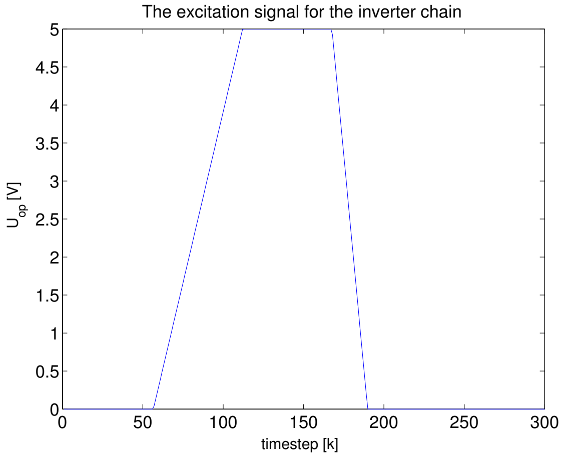


Figure 8.2: The excitation signal u_{in} for the inverter chain

when about 30 nodes are or have been active. The total simulation time is 30 ns. The remaining simulation data will be used as validation data.

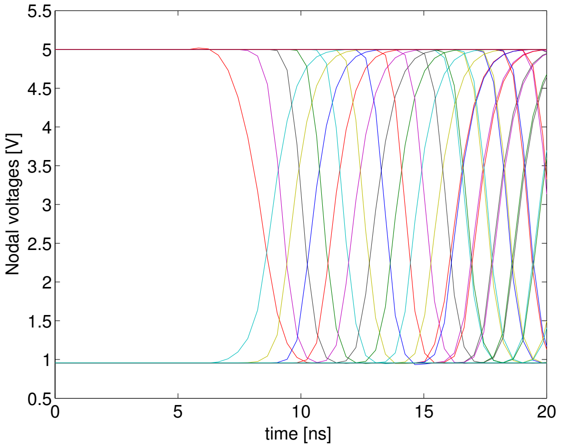


Figure 8.3: The response of the inverter chain

If we excite the first unknown voltage V_1 with a short pulse, a voltage wave is traveling through the chain from left to right. This means that on $[0, 10]$ only the first 8 nodes are activated yet. We applied a BDF Compound-Fast algorithm on $[0, 10]$ with order 1 on the coarse time-grid and order 2 on the refined time-grid. During the compound phase we only looked at the error of the latent part, so $\tau = 0$ in (6.2). During the refinement only the active part, consisting of the 8 activated nodes, is simulated. The tolerance levels were equal to $TOL_L = 1$, $TOL_A = 1$ and $TOL_C = TOL_L$. Because of the latency of the slow part, the solution can be determined by just 5 compound steps and 93 refinement steps, while an uniform BDF-method with $k = 1$ would have used about 93 time-steps. The solution is shown in Figure 8.4. Indeed only the first eight nodal voltages behave fast at $[0, 10]$.

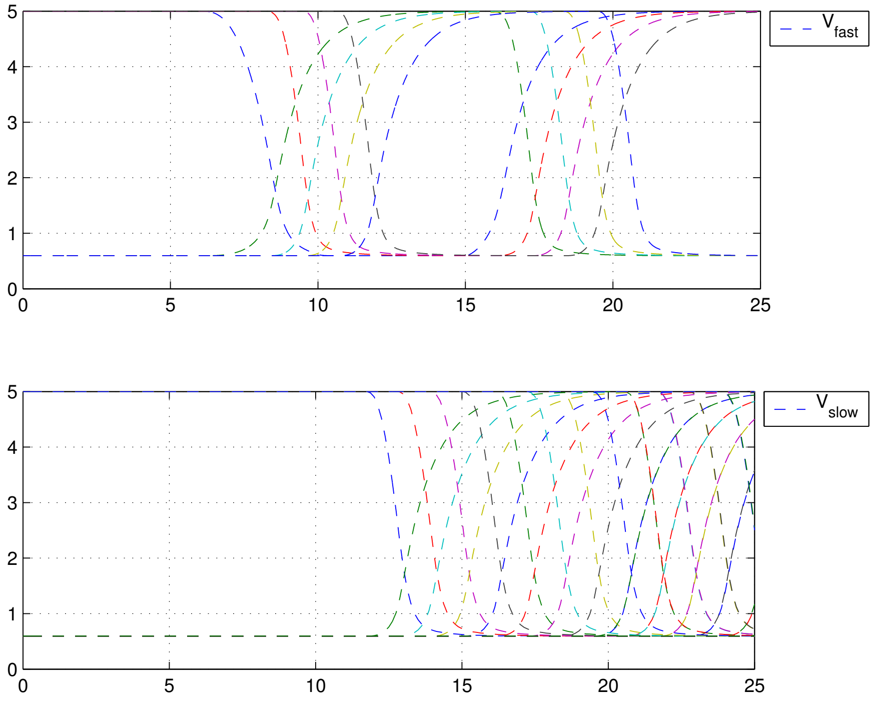


Figure 8.4: Numerical solution of the slow and fast parts for the Inverter chain.

For $TOL = 10^{-2}$, $N = 100$ we also did an experiment on $[0, 75\text{ ns}]$ by several dynamical partitioning algorithms described in Section 6.4. Algorithm I is used with the workload model (6.45) but for different values of α. In all cases at most 4 iterations are performed during a compound step. Algorithm II is used for different values of ϵ_{rel}. All algorithms use $\gamma = 3$ as overlap value. Table 8.1 shows the results. Note that n_i and k_i are the numbers of timesteps and Newton iterations, respectively. Clearly, for each case the number of refinement steps, n_R, is much larger than the number of compound steps, n_C. It is larger than the number of steps for the single-rate version because of error control reasons. In the column below $\mathrm{av}(\frac{d_A}{d})$ the average relative size of the active part is shown. The required CPU time also includes the repartitioning time effort.

Method	α	ϵ_{rel}	n_C	n_R	k_C	k_R	$\mathrm{av}(\frac{d_A}{d})(\%)$	time (s)	S
Single-rate			1340	0	5440	0	0	266	
I	2		82	1651	1008	3415	16	87	3.1
I	$\frac{3}{2}$		94	1663	996	3429	15	86	3.1
II		10^{-1}	166	1953	1313	4034	9	100	2.7
II		10^{-2}	97	2001	1225	4105	16	105	2.5
II		10^{-3}	94	1992	1637	4093	22	133	2.0

Table 8.1: Statistics of single-rate and multirate method using algorithms 1 and 2 of type A for the inverter chain model.

If we compare the algorithms for lower accuracy TOL = 10^{-1} it appears that the method II does not converge in contrast to 1. Also for other experiments it appears that the method I implies better convergence than II. All methods are able to follow the active wave front. Figure 8.5 shows the timepoints per element for case I with $\alpha = \frac{3}{2}$. The two flanks of the traveling wave are easy to observe in the picture.

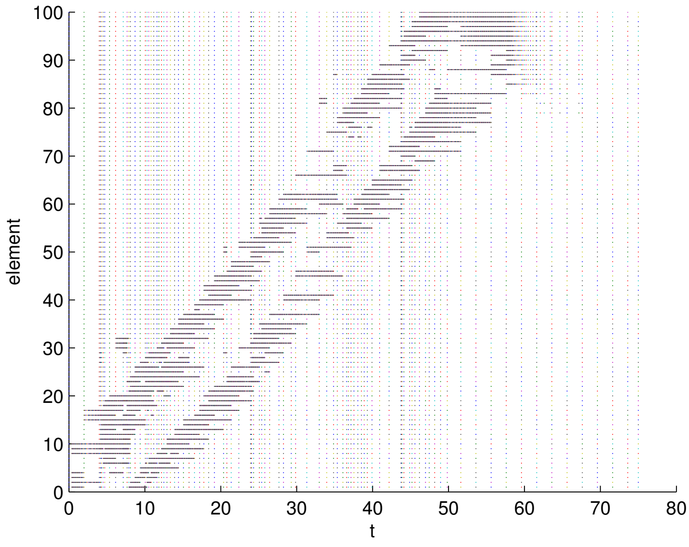


Figure 8.5: Timepoints per element for the inverter chain (case 1 with $\alpha = \frac{3}{2}$).

For small N the speed-up factor is relatively small because of the overhead. But for large N it is indeed possible to get a large speed-up. This is also the case for increasing accuracy.

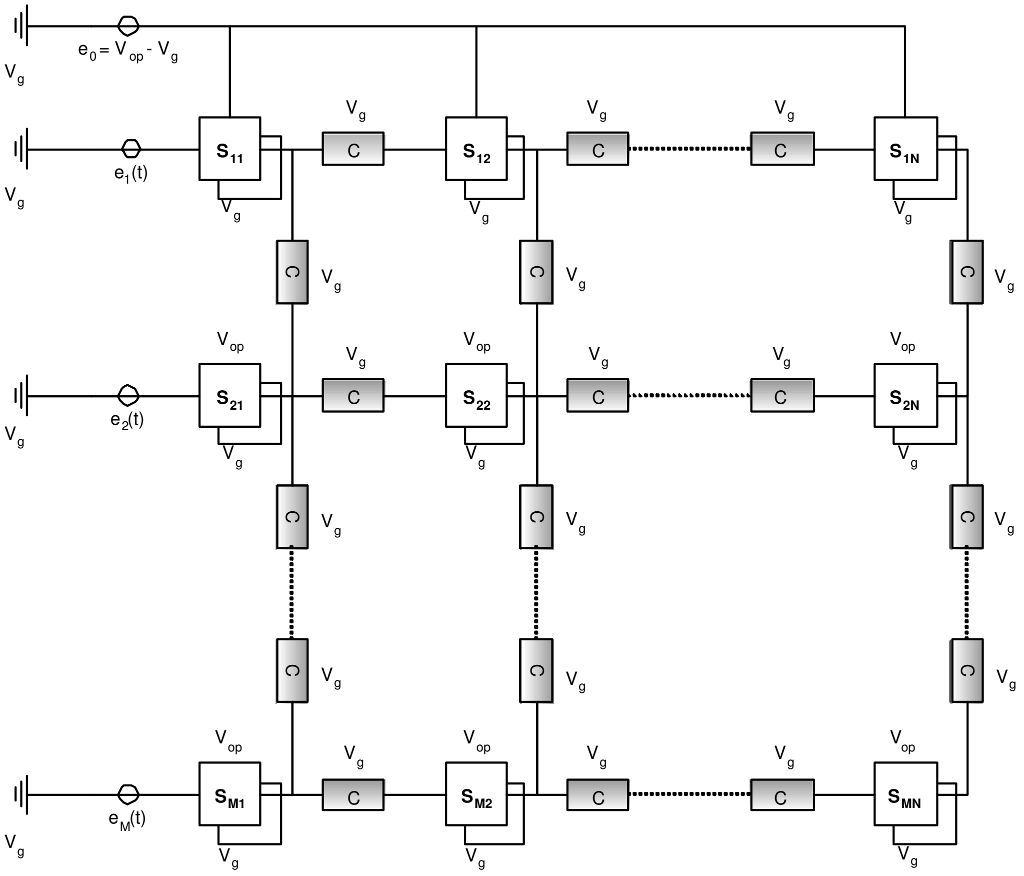


Figure 8.6: Circuit diagram of a (M, N) Matrix Circuit.

Figure 8.6 shows another test example, which is a scalable circuit consisting of $M \times N$ inverter models S_{ij}. The inverters S_{ij} are connected by linear subcircuits C that can be used to decouple the dynamics of the neighbouring currents or voltages.

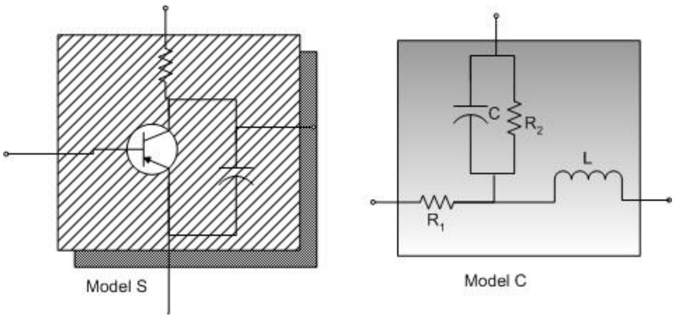


Figure 8.7: Circuit diagrams of the subcircuits S and C.

The circuit is driven by M voltage sources $e_i(t)$ at the left that can have different frequencies. The location of the active part is controlled by the subcircuits of type C and by the voltage sources $e_1, \ldots, e_M$. The source $e_0(t)$ is set to $V_{op} = 5$ V. Furthermore we use the voltage sources $e_i = \frac{5}{2}(1 - \cos(\omega_i t))$, where $\omega_1 = 100 \cdot 10^9$, and for $i > 1$, $\omega_i = 10^9$. We restrict ourselves to the case $M = 5, N = 10$. The subcircuits of type S are inverter models, while the subcircuits of type C are chosen such that the three subcircuits S_{11}, S_{12}, S_{13} are active and nearly decoupled from the other subcircuits. They form an active part because they are activated by the voltage source e_1 with higher frequency than the other voltage sources.

First we do a numerical experiment for $R_1 = R_2 = 10^4\ \Omega, C = 10^{-3}$ F, $L = 10^{-3}$ H. For

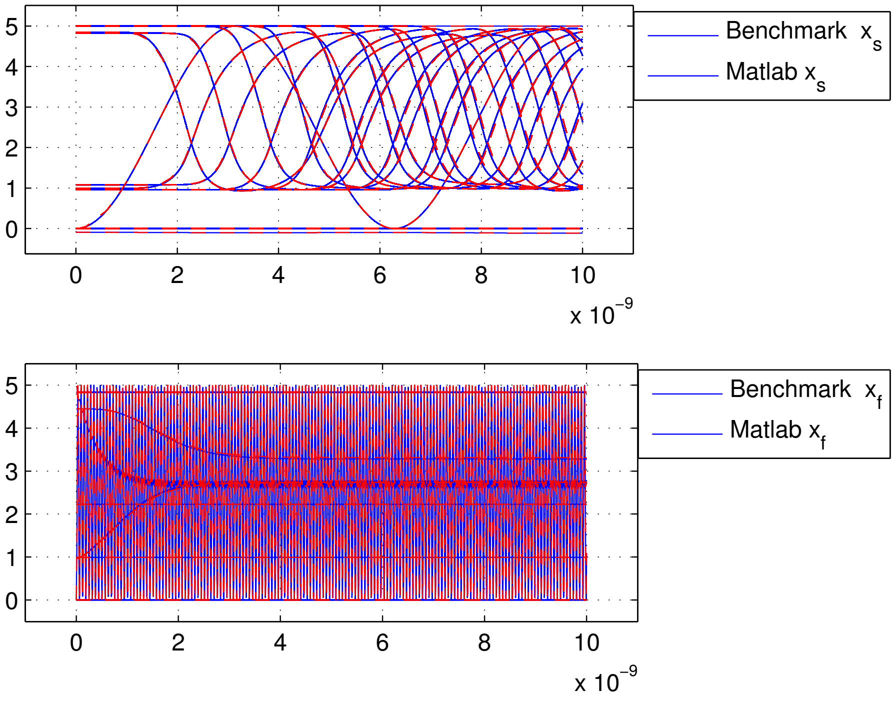


Figure 8.8: Numerical solution of the slow and fast parts for the Matrix circuit.

these values the subcircuits of model C behave like filters for the voltages and currents. We take an Euler Backward multirate simulation on $[0, 10^{-8}]$s with $w = 0.5, \tau = 0$ and tolerance levels $\text{TOL}_C = \text{TOL}_A = 10^{-1}$. We also do a normal Euler Backward (single-rate) simulation with the same tolerance levels. In both cases the timesteps arer automatically controlled based on the tolerance levels. It turns out that the single-rate simulation required 2018 timesteps and a computational time of 5491 seconds, while the multirate simulation just required 71 compound steps, 2810 refinement steps and a computational time of 422 seconds. Therefore we get a speed-up factor $S \approx 13$.

We also carry another experiment, where we compare a BDF2 single-rate method to a

BDF2 multirate method for $\mathrm{TOL_C} = \mathrm{TOL_A} = 10^{-2}$. We use the balance numbers $w = 0.5$ and $w = 10^{-4}$ together with $\tau = 0$. The errors are estimated by comparing the results to the numerical solution of a single-rate BDF2 simulation using higher tolerance levels $\mathrm{TOL_C} = \mathrm{TOL_A} = 10^{-3}$. Thus these single-rate simulation results have a much smaller error and can be considered as a benchmark solution. Table 8.2 and Figure 8.8 illustrate that the BDF Compound-Fast multirate algorithm is able to produce accurate results in an efficient way (speed-up $S > 10$). Table 8.2 also shows that too large values of the balance number decreases the efficiency.

Table 8.2: Statistics of single-rate and multirate BDF2 methods for the Matrix circuit.

method	w	n_C	n_R	comp. time (s)	S	max. error
single-rate		2937		7330		$5.8 \cdot 10^{-2}$
multirate	0.5	111	3765	668	11	$1.8 \cdot 10^{-1}$
multirate	10^{-4}	111	3002	612	12	$1.8 \cdot 10^{-1}$

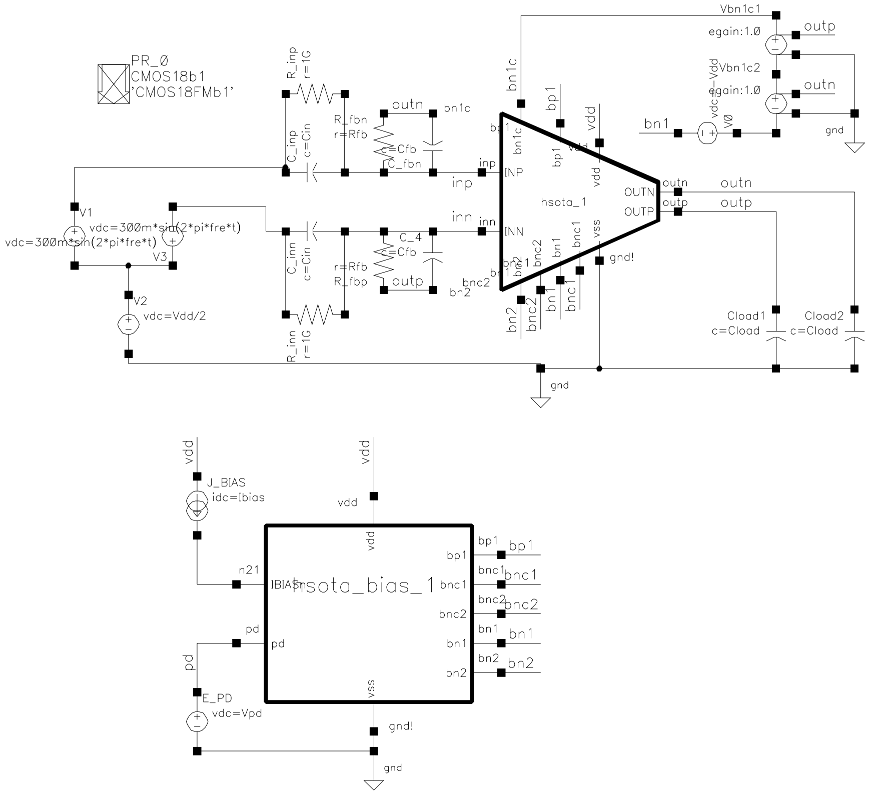


Figure 8.9: The high-speed operational transconductance amplifier.

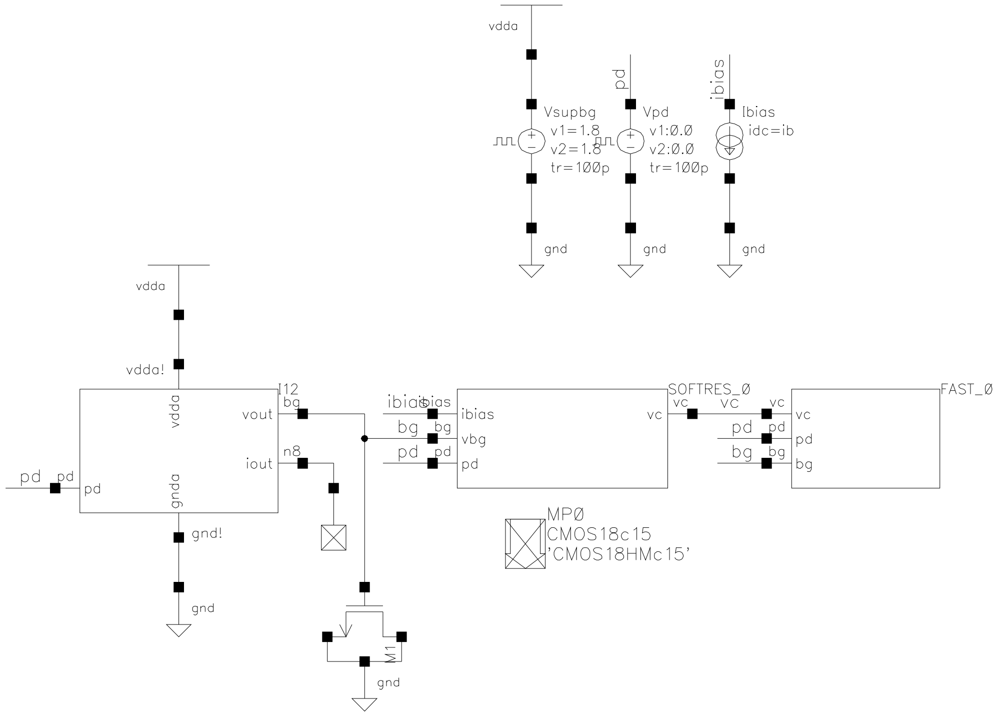


Figure 8.10: The temperature-independent oscillator.

The implementation of the multirate time integration algorithm in Pstar, allows us to obtain results for various circuits. We consider two practical examples, coming from the actual circuit design. The block-diagrams at top-level for the high-speed operational transconductance amplifier (HSOTA) and the temperature-independent oscillator (temp.ind.osc.) are shown in Fig 8.9 and 8.10. An operational transconductance amplifier (OTA) is an operational amplifier (op-amp) of a transconductance type, which means that the input voltage controls an output current by means of the device transconductance. This makes the OTA a voltage-controlled current source, which is in contrast to the conventional op-amp, which is a voltage-controlled voltage source. The temperature-independent oscillator consists of a slow block which makes the input voltage independent of the temperature. Then the result is used in a ring oscillator to get a voltage signal of the right frequency. The oscillator block typically has a much higher activity than the first block.

In the HSOTA and Temp.ind.osc. models there are relatively large bias blocks that practically have constant dynamics. This allows to use a small number of compound steps of much larger size. Numerical results are summarised in Table 8.3. Remind that in Section 6.4 we derived in (6.44) the formula

$$S \approx \frac{1}{\frac{1}{q} + E}.$$

The multirate factor q and the work load ratio E. also have been defined in the same

Section. Clearly, we can only expect a large speed-up factor if $q \gg 1$ and $0 < E \ll 1$. The HSOTA model allows a very large multirate factor $q \approx 207$ but has a rather large work load ratio $E \approx 0.5$. Thus the expected speed-up factor equals just $S \approx \frac{1}{E+\frac{1}{q}} \approx 2$. Because the Temp.ind.osc. model with $q \approx 68$ has a smaller workload ratio, $E \approx 0.12$, its estimated speedup factor is indeed larger.
Only if $q >> 1$ and $0 < E << 1$ we can expect a large speed-up factor. For the HSOTA example the speed-up factor could be increased by using a better partitioning with smaller E and q.

Table 8.3: Multirate results with static partitioning. Notation: d- number of unknowns, N_C- number of compound steps, N_R- number of refinement steps, N_S- number of single-rate steps, d_A- number of active unknowns, S- speed-up factor.

Circuit name	d	N_C	N_R	N_S	q	d_A/d	S
HSOTA	61	68	14092	14068	207	50%	2
Temp.ind.osc.	249	151	10284	7419	68	12%	4.5

Because finding a good partitioning manually can be very difficult, it is better to use a multirate transient simulation together with dynamical partitioning as is described in Section 6.4. Table 8.4 shows the results for the the HSOTA and the charge pump. Note that the HSOTA needs only one repartitioning because the active part is not moving there, while the charge pump needs about 8 repartitionings. The partition is determined by algorithm I in Section 6.4, while also the interpolation errors at the interface are checked to be sufficiently small.

Table 8.4: Multirate results with dynamical partitioning. Notation: d- number of unknowns, N_C- number of compound steps, N_R- number of refinement steps, N_S- number of single-rate steps, d_A- number of active unknowns, R_p- number of repartitionings, S- speed-up factor.

Circuit name	d	N_C	N_R	N_S	q	d_A/d	R_p	S
HSOTA	66	120	13983	13963	117	55%	1	1.6
Temp.ind.osc.	245	172	9408	80	55	11%	8	4.2

8.2 Applications of Model Order Reduction

In this section we will show how POD combined with MPE and TPWL can be used to reduce nonlinear circuit models. First we will consider the reduced order modeling of the inverter chain model, which has already been described in the previous section 8.1. But in contrast to the previous experiments, this time the inverter chain consists of 100 inverters. Because of the additional current unknowns for the two voltage sources, the

state dimension d equals 102.
Data collected during the first 15 ns are used to derive the POD basis $\mathbf{V} \in \mathbb{R}^{n\times k}$. For $N = 251$ samples the snapshot matrix $X \in \mathbb{R}^{d\times N}$ reads:

$$\mathbf{X} = \frac{1}{N}\begin{pmatrix}\mathbf{x}(t_1) & \dots & \mathbf{x}(t_N)\end{pmatrix}$$

The POD basis $\mathbf{V}$ is found by solving the eigenvalue problem (7.39) in Section 7.6. The resulting singular value plot is shown in Figure 8.11. It shows that the snapshot data can be accurately spanned by only the first 20 eigenvectors of the correlation matrix $\frac{1}{N}\mathbf{X}\mathbf{X}^T$. Therefore, we select a POD basis $\mathbf{V}$ that comprises the eigenvectors corresponding to the 20 largest singular values. The POD reduced order model is solved by implementing the least-square formulation in (7.47). To enhance the computational efficiency the MPE method described in section 7.7 is applied using $k = 20$ and $g = 29$.

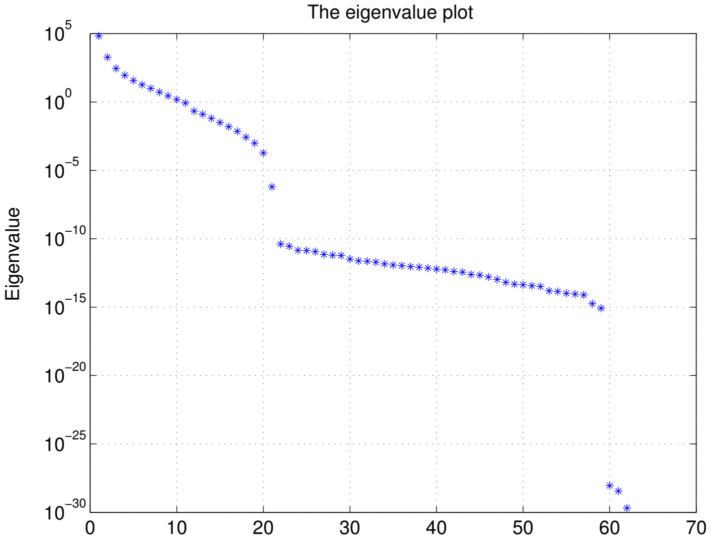


Figure 8.11: The singular value plot corresponding to the POD snapshots

Applying this procedure to the inverter chain model yields $\dim(\tilde{\mathbf{x}}) = 29$. Hence, the MPE reduced order model can be built from less than 28% of the original equations.
Two comparisons are shown here. First, the reconstruction of the original dynamics by the POD and the MPE models. The simulation conditions are the same as the conditions when the POD basis is derived. Figure 8.12 shows the plot of the original, POD, and MPE models. It is clear that the simulation results are very close to each other. Note that the solution obtained from the reduced model also consists of constant parts equal to zero, which do not occur in the solution of the original model. They arise because of the changed steady-state of the reduced model of type (7.36). The maximum of the average absolute error for the POD model is 0.0019 while the maximum of the average absolute error for the MPE model is 0.0020. It is obvious that the MPE model can reconstruct the original dynamics very well despite the fact that the model is built from only 29 equations.

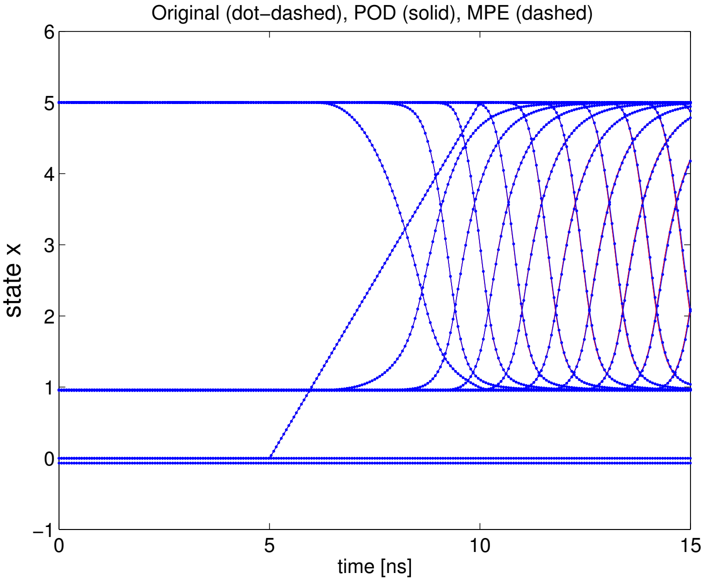


Figure 8.12: The reconstruction of the original dynamics by POD and MPE models

The reduced order models are validated by comparing the dynamics simulated between $15-25$ ns. Recall that the POD basis is derived from the simulation data between $0-15$ ns. The capability of the reduced models in capturing the unknown dynamics will be investigated in this case.

Figure 8.13 shows the comparisons. The POD and MPE models can adequately describe the dynamics of the nodes that are already active during $0-15$ ns. The models fail to capture the dynamics of the nodes or state variables that have been inactive during the first 15 ns. This is understandable as the inactive state variables will be viewed by the POD basis as zero state variables only and it will be extremely difficult, if not impossible, for both reduced models to simulate the conditions when these nodes become active. During this time interval, both MPE and POD models have the same maximum absolute error that occurs on the state variable x_{23}. The maximum absolute error is 3.2152 V. The evaluation costs are decreasing for about 70% when compared to the case without use of MPE.

We also consider the academic diode chain model shown in Fig. 8.14, described through the following equations

$$\begin{aligned} V_1 - U_{in}(10^9 t) &= 0, \\ i_E - g(V_1, V_2) &= 0, \\ g(V_1, V_2) - g(V_2, V_3) - C\frac{dV_2}{dt} - \frac{1}{R}V_2 &= 0, \\ &\vdots \\ g(V_{N-1}, V_N) - g(V_N, V_{N+1}) - C\frac{dV_N}{dt} - \frac{1}{R}V_N &= 0, \\ g(V_N, V_{N+1}) - C\frac{dV_{N+1}}{dt} - \frac{1}{R}V_{N+1} &= 0, \end{aligned} \tag{8.1}$$

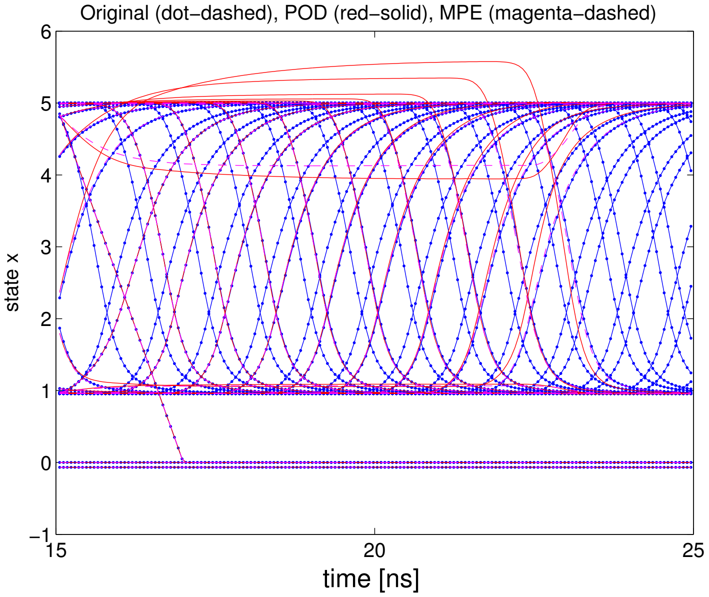


Figure 8.13: The validations by POD and MPE models

where the dioded functionality is described by

$$g(V_a, V_b) = \begin{cases} (I_s e^{\frac{V_a - V_b}{V_T}} - 1) & \text{if } V_a - V_b > 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{8.2}$$

and

$$U_{in}(t) = \begin{cases} 20 & \text{if } t \le 10 \\ 170 - 15t & \text{if } 10 < t \le 11 \\ 5 & \text{if } t > 11 \end{cases} .$$

Fig. 8.15 (left) shows the numerical solution (nodal voltage in each node) of the original model, computed by the Euler Backward method with fixed step sizes of 0.1 ns. At $t = 10^{-8}$s the solution behaves very active because of the sudden switch of the input voltage signal U_{in}. Such behaviour is very typical for nonlinear IC models. The figure on the right indicates the redundancy of the model, as most of the eigenvalues of the correlation matrix $\frac{1}{n}XX^T$ can be neglected. Figure 8.16 shows the relative errors over all nodes in the time interval [0, 70 ns], defined as $\varepsilon_r = \frac{||V\mathbf{z} - \mathbf{x}||}{||\mathbf{x}||}$, for the reduced models of different orders constructed by TPWL (left) and POD (right). Most of the time the relative error of TPWL is lower than the chosen error bound $\varepsilon = 0.025$. Furthermore, for higher order reduced models, a smaller number of LTs is used than for the reduced models with lower order, as the local systems with higher orders are more accurate. E.g. for a reduced model of order 100 we have used 42 LTs and for smaller reduced models 60 LTs. The POD models are, as expected, more accurate than TPWL, but much slower to simulate than the TPWL models (see the corresponding extraction and simulation

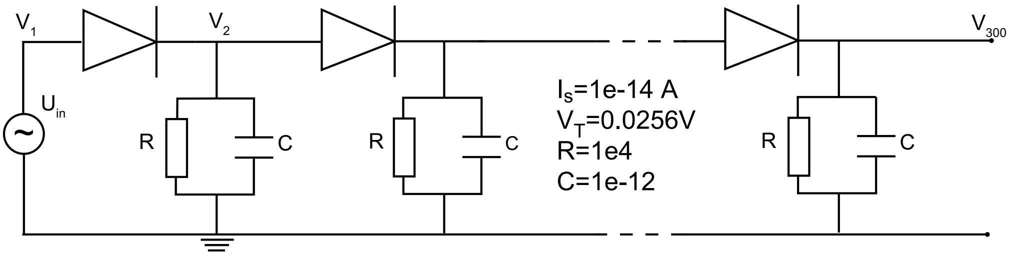


Figure 8.14: The diode chain

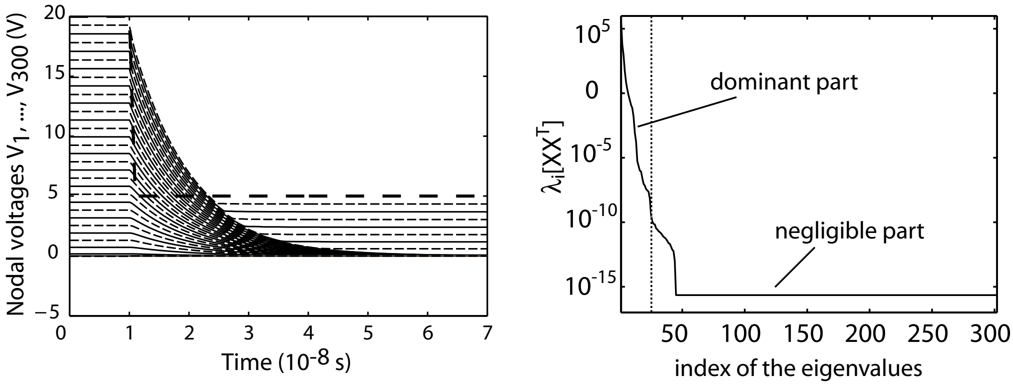


Figure 8.15: Numerical solution of the full-scale nonlinear diode chain model (left) and the singular values of the snapshot matrix **X**.

times in Table 8.5). A significant speed up has been achieved by combining the POD with MPE.

Table 8.5: Comparison of extraction and simulation times in seconds.

Model	r	Extr. time	Sim. time	Model	r	g	Extr. time	Sim. time
Original	302	0	142	POD	10	302	142	168
TPWL	10	290	1.1	POD	25	302	142	182
TPWL	25	285	1.5	POD + MPE	10	32	146	74
TPWL	50	206	2.3	POD + MPE	25	55	151	123

Even combined with MPE, the method POD is still rather expensive because of the higher evaluation costs of the Jacobian matrices. The method TPWL is faster because it approximates the nonlinear model by a piecewise linear model. Therefore, the performance of POD can also be improved by keeping the Jacobian matrices constant for some timesteps. This allows also to reuse the same factorisation, such that also the linear algebra costs are reduced. Table 8.6 shows a comparison of the simulation times with respect to normal POD. It turns out that in this case, the modified version of POD

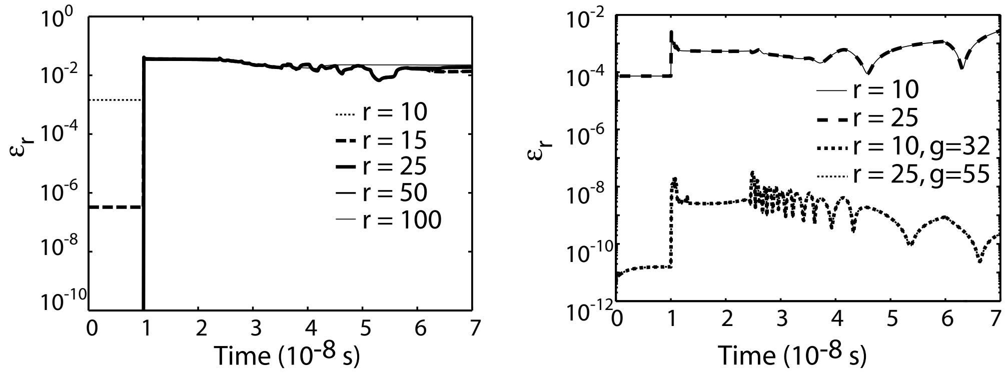


Figure 8.16: Relative errors over all nodes for the reduced models created by TPWL (left) and by POD (right).

is nearly three times faster than normal POD. Note that all simulation times in this table are slightly smaller than in Table 8.5 because we used a more efficient implementation of the BDF method. Figure 8.17 shows that the accuracy of this modified implementation of POD has not been decreased.

Table 8.6: Comparison of simulation times in seconds of normal POD and modified POD (including MPE and modified Newton).

Model	r	g	Sim. time
Original	302		80
POD	10	302	87
POD	20	302	102
modified POD	10	32	28
modified POD	20	40	31

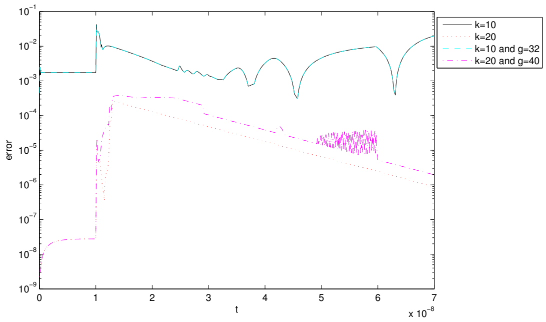


Figure 8.17: Relative errors over all nodes for the reduced models created by the normal and modified POD implementations.

List of frequently used symbols

i	current
v	voltage difference
V	nodal voltage
t	time variable
ω	angular frequency of a source function
$\mathbf{x}(t)$	state vector of circuit
d	dimension of $\mathbf{x}$
$\mathbf{q}(t, \mathbf{x})$	charge function
$\mathbf{j}(t, \mathbf{x})$	electrical current function
$\mathbf{C}(t, \mathbf{x}))$	Jacobian matrix of $\mathbf{q}(t, \mathbf{x})$
$\mathbf{G}(t, \mathbf{x})$	Jacobian matrix of $\mathbf{j}(t, \mathbf{x})$
$\mathbf{f}(t, \mathbf{x})$	right-hand-side of an ODE
$\mathbf{J}(t, \mathbf{x})$	Jacobian matrix of $\mathbf{f}(t, \mathbf{x})$
ν	global index of a DAE
μ_n	local index of a DAE around $t = t_n$
t_n	n-th timepoint of time-grid
h_n	n-th timestep $t_n - t_{n-1}$ of time-grid
k	number of steps of a multistep integration method
p	integration order of an integration method
$\mathbf{x}_n$	numerical approximation of $\mathbf{x}(t)$ at t_n
δ_n	local discretisation error of a numerical scheme
$\hat{\delta}_n$	estimate of δ_n
$\mathbf{d}_n$	scaled local discretisation error of a numerical scheme
$\mathbf{e}_n$	global error of a numerical scheme
$C_{p,n}$	error constant of an pth-order integration method at t_n
$\xi_{n,m}$	dimensionless coefficient that satisfies $t_n - t_{n-m} = \xi_{n,m} h_n$
$\hat{r}_n$	controlled error, often based on the norm of $\hat{\delta}_n$
TOL	tolerance level for a stepsize controller
θ	safety factor
ϵ	reference level θ TOL that is really used by a stepsize controller
$\mathbf{y}(t)$	Predictor polynomial for $\mathbf{x}(t)$
$\mathbf{p}(t)$	Predictor polynomial for $\mathbf{q}(t, \mathbf{x}(t))$
$\mathbf{i}(t)$	Predictor polynomial for $\mathbf{j}(t, \mathbf{x}(t))$
$\mathbf{x}(t)$	Corrector polynomial for $\mathbf{x}(t)$

$\mathbf{q}(t)$	Corrector polynomial for $\mathbf{q}(t, \mathbf{x}(t))$
$\mathbf{j}(t)$	Corrector polynomial for $\mathbf{j}(t, \mathbf{x}(t))$
$l(t)$	Lagrangian basis polynomial
$\bar{\mathbf{Y}}$	Nordsieck matrix corresponding to $\mathbf{y}(t)$
$\bar{\mathbf{P}}$	Nordsieck matrix corresponding to $\mathbf{p}(t)$
$\bar{\mathbf{X}}$	Nordsieck matrix corresponding to $\mathbf{x}(t)$
$\bar{\mathbf{Q}}$	Nordsieck matrix corresponding to $\mathbf{q}(t)$
$\mathbf{l}$	Nordsieck vector corresponding to $l(t)$
$\mathbf{x}_A$	$\mathbf{x}$ for active (fast) part of circuit
$\mathbf{x}_L$	$\mathbf{x}$ for latent (slow) part of circuit
d_A	dimension of $\mathbf{x}_A$
d_L	dimensions of $\mathbf{x}_L$
$\mathbf{B}_A$	selection operator such that $\mathbf{x}_A = \mathbf{B}_A\mathbf{x}$
$\mathbf{B}_L$	selection operator such that $\mathbf{x}_L = \mathbf{B}_L\mathbf{x}$
$\mathbf{q}_A$	$\mathbf{q}$ for active (fast) part of circuit
$\mathbf{j}_A$	$\mathbf{j}$ for active (fast) part of circuit
$\mathbf{q}_L$	$\mathbf{q}$ for latent (slow) part of circuit
$\mathbf{j}_L$	$\mathbf{j}$ for latent (slow) part of circuit
T_n	time-point at the coarse time-grid
H_n	compound or macro step $H_n = T_n - T_{n-1}$
$t_{n,m}$	time-point at the fine time-grid,
$h_{n,m}$	refinement or micro step $h_{n,m} = t_{n,m} - t_{n,m-1}$
$\mathbf{K}$	coupling matrix
w	balance number
q_n	multirate-factor that satisfies $\sum_{m=1}^{q_n} h_{n,m} = H_n$
E	workload ratio
S	speed-up factor
$\hat{q}$	estimate of q
$\hat{E}$	estimate of E
$\hat{S}$	estimate of S
W_C	computational workload per compoundstep
W_R	computational workload per refinement step
W_S	computational workload per single-rate step
$\mathbf{V}$	matrix corresponding to reduced basis
$\mathbf{z}(t)$	state vector of the reduced model such that $\mathbf{x}(t) \approx \mathbf{V}\mathbf{z}(t)$
r	dimension of reduced model

Bibliography

[1] A.C. Antoulas. *Approximation of Large-Scale Dynamical Systems*. Society for Industrial and Applied Mathematics, Philadelphia, 2005.

[2] P. Astrid. *Reduction of process simulation models: a proper orthogonal decomposition approach*. PhD thesis, Eindhoven University of Technology, Department of Electrical Engineering, Eindhoven, 2004.

[3] P. Astrid and A. Verhoeven. Application of Least Squares MPE technique in the reduced order modeling of electrical circuits. In Y. Yamamoto, T. Sugie, and Y. Ohta, editors, *Proceedings of 17th Int. Symposium on Mathematical Theory of Networks and Systems (CDROM)*, pages 1980–1986, Japan, 2006.

[4] P. Astrid and S. Weiland. On the construction of POD models from partial observations. In *Proceedings of the 44th IEEE Conf. on Decision and Control*, pages 2272–2277, Spain, 2005.

[5] P. Astrid, S. Weiland, K. Willcox, and A. Backx. Missing point estimation in models described by proper orthogonal decomposition. In *Proceedings of the 43th IEEE Conf. on Decision and Control*, volume 2, pages 1767–1772, Bahamas, 2004.

[6] A. Bartel. Generalised Multirate: Two ROW-type Versions for Circuit Simulation. Master's thesis, TU Darmstadt & IWRMM Universität Karlsruhe, 2000.

[7] A. Bartel and M. Günther. A multirate W-method for electrical networks in state-space formulation. *Journal of Comput. and Applied Maths.*, 147:411–425, 2002.

[8] P. Benner, V. Mehrmann, and D.C. Sorensen. *Dimension Reduction of Large-Scale Systems*. Springer-Verlag, Berlin/Heidelberg, 2006.

[9] K.E. Brenan, S.L. Campbell, and L.R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. SIAM, Philadelphia, 1996.

[10] E. Ciggaar. The MOESP Algorithm for Linear Time Domain Modelling. Technical Report 123/98, Nat.Lab., Eindhoven, 1998.

[11] R. Everson and L. Sirovich. The Karhunen-Luève procedure for gappy data. *Journal Opt.Soc.Am.*, 12:1657–1664, 1995.

[12] Y. Favennec, M. Girault, and D. Petit. The adjoint method coupled with the modal identification method for nonlinear model reduction. *Inverse problems in Science and Engineering*, 14:153–170, 2006.

[13] J.G. Fijnvandraat, S.H.M.J. Houben, E.J.W. ter Maten, and J.M.F. Peters. Time domain analog circuit simulation. *Journal of Computational and Applied Mathematics*, 185(2):441–459, 2006.

[14] R.W. Freund. Krylov-subspace methods for reduced order modeling in circuit simulation. *Journal of Computational and Applied Mathematics*, 123:395–421, 2000.

[15] C.W. Gear. *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice Hall, New Jersey, USA, 1971.

[16] C.W. Gear and D.R. Wells. Multirate linear multistep methods. *BIT*, 24:484–502, 1984.

[17] G.R. G'omez. *Absolute stability analysis of semi-implicit multirate linear multistep methods*. PhD thesis, Instituto Nacional de Astrofisica, Optica y Electronica, Tonantzintla, Pue, Mexico, 2002.

[18] A. El Guennouni, A. Verhoeven, E.J.W. ter Maten, and T.G.J. Beelen. Aspects of Multirate Time Integration Methods in Circuit Simulation Problems. In A. Di Bucchianico, R.M.M. Mattheij, and M.A. Peletier, editors, *c*, Eindhoven, The Netherlands, 2006. Springer-Verlag.

[19] M. Günther and P. Rentrop. Partitioning and multirate strategies in latent electrical circuits. *International Series of Numerical Mathematics*, 117:33–60, 1994.

[20] K. Gustafsson. Control Theoretic Techniques for Stepsize Selection in Implicit Runge-Kutta Methods. *ACM TOMS*, 20:496–517, 1994.

[21] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I, nonstiff problems*. Springer, Berlin, 1993.

[22] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II, stiff and differential-algebraic problems*. Springer, Berlin, 1993.

[23] C. Homescu, L.R. Petzold, and R Serban. Error estimation for reduced-order models of dynamical systems. *SIAM Journal on Numerical Analysis*, 43:1693–1714, 2005.

[24] S.H.M.J. Houben. Algorithms for Periodic Steady State Analysis on Electric Circuits. Master's thesis, Eindhoven University of Technology, 1999.

[25] S.H.M.J. Houben. *Circuits in motion, The numerical simulation of electrical oscillators*. PhD thesis, Technische Universiteit Eindhoven, 2003.

[26] Z. Ilievski, H. Xu, A. Verhoeven, E.J.W. ter Maten, W.H.A. Schilders, and R.M.M. Mattheij. Adjoint transient sensitivity analysis in circuit simulation. In G. Ciuprina and D. Ioan, editors, *Scientific Computing in Electrical Engineering*, pages 183–190, Sinaia, Romania, 2007. Springer.

[27] M. Kirby. *Geometric data analysis*. John Wiley and Sons, New York, 2001.

[28] A. Kvaerno. Stability of multirate Runge-Kutta schemes. *International Journal of Differential Equations and Applications*, 1:97–105, 2000.

[29] A. Logg. Multi-adaptive Galerkin methods for ODEs I. *SIAM J. Sci. Comp.*, 24:1879–1902, 2003.

[30] A. Logg. Multi-adaptive Galerkin methods for ODEs II. *SIAM J. Sci. Comp.*, 25:1119–1141, 2003.

[31] J. ter Maten, A. Verhoeven, A. El Guennouni, and Th. Beelen. Multirate hierarchical time integration for electronic circuits. In *PAMM, Virtual Annual Meeting Proceedings of the GAMM conference*, pages 819–820, Luxembourg, 2005.

[32] R.M.M. Mattheij and J. Molenaar. *Ordinary differential equations in theory and practice*. SIAM, Philadelphia, 2002.

[33] H. Mauritz and W. Mathis. Integration System as Adaptive Control System. In *Circuits and Systems, Proceedings of ISCAS Conference*, pages 101–104, 1994.

[34] P.B.L. Meijer. *Neural network applications in device and subcircuit modelling for circuit simulation*. PhD thesis, Eindhoven University of Technology, Department of Electrical Engineering, Eindhoven, 1996.

[35] P.B.L. Meijer. Neural Networks for Device and Circuit Modelling. In U. van Rienen, M. Günther, and D. Hecht, editors, *Scientific Computing in Electrical Engineering*, Warnemünde, Germany, 2000. Springer.

[36] A. Odabasioglu, M. Celik, and L.T. Pileggi. PRIMA: Passive Reduced-order Interconnect Macromodeling Algorithm. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17:645–654, 1998.

[37] J.M. Ortega and W.C. Rheinboldt. *Iterative solution of nonlinear equations in several variables*. Academic Press, New York, 1972.

[38] J. Phillips and L.M. Silveira. Poor man's TBR: a simple model reduction scheme. 2:938–943, 2004.

[39] S. J. Polak, W. H. A. Schilders, C. Den Heijer, Wachters, A. J. H., and H. M. Vaes. Automatic problemsize reduction for on-state semiconductor problems. *SIAM Journal on Scientific and Statistical Computing*, 4(3):452–461, 1983.

[40] A. Prothero and A. Robinson. On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations. *Math. of Comp.*, 28:145–162, 1974.

[41] M. Rewienski. *A Trajectory Piecewise-Linear Approach to Model Order Reduction of Nonlinear Dynamical Systems*. PhD thesis, MIT, Massachusets, USA, 2003.

[42] M. Rewienski and J. White. A Trajectory Piecewise-Linear Approach to Model Order Reduction and Fast Simulation of Nonlinear Circuits and Micromachined Devices. In *Proc. Of IEEE/ACM International Conference on Computer Aided-Design*, pages 252–257, San Jose, 2001.

[43] J. Rommes. Jacobi-Davidson methods and preconditioning with applications in pole-zero analysis. Master's thesis, Utrecht University, 2002.

[44] R. Saleh, S. Jou, and A.R. Newton. *Mixed-mode simulation and analog multilevel simulation*. Kluwer Academic Publishers, Boston, 1994.

[45] J. Sand and S. Skelboe. Stability of Backward Euler Multirate Methods and Convergence of Waveform Relaxation. *BIT*, 32:350–366, 1992.

[46] V. Savcenco, W.H. Hundsdorfer, and J.G. Verwer. A multirate time stepping strategy for stiff ordinary differential equations. *BIT*, 47:137–155, 2007.

[47] J.M.A. Scherpen. *Balancing for nonlinear systems*. PhD thesis, Universiteit Twente, Enschede, 1994.

[48] L.F. Shampine. *Numerical solution of ordinary differential equations*. Chapman & Hall, New York, 1994.

[49] S. Sjö. *Analysis of computational algorithms for linear multistep methods*. PhD thesis, Lund University, 1999.

[50] S. Skelboe and P.U. Andersen. Stability properties of backward Euler multirate formulas. *SIAM J. Sci. Stat. Comput.*, 10:1000–1009, 1989.

[51] G. Söderlind. Digital filters in adaptive time-stepping. *ACM Tr. on Math.Softw.*, V:1–24, 2000.

[52] G. Söderlind. Automatic control and adaptive time-stepping. *Numerical Algorithms*, 31:281–310, 2002.

[53] M. Striebel. *Hierarchical mixed multirating for distributed integration of DAE network equations in chip design*. PhD thesis, Bergische Univ. Wuppertal, Wuppertal, Germany, 2006.

[54] M. Striebel and M.G. Günther. Hierarchical mixed multirating in circuit simulation. In G. Ciuprina and D. Ioan, editors, *Scientific Computing in Electrical Engineering*, pages 221 – 228, Sinaia, Romania, 2007. Springer.

[55] T. Stykel. Gramian-based model reduction for descriptor systems. *Mathematics of Control, Signals and Systems*, 16:297–310, 2004.

[56] B. Tasić, A. Verhoeven, E.J.W. ter Maten, and T.G.J. Beelen. Compound BDF multirate transient analysis applied to circuit simulation. In T.E Simos, G. Psihoyios, and Ch. Tsitouras, editors, *International Conference on Numerical Analysis and Applied Mathematics 2006*, pages 480–483, Crete, Greece, 2006. WILEY-VCH.

[57] C. Tischendorf. *Solution of index-2 differential algebraic equations and its application in circuit simulation*. PhD thesis, Humboldt-Universität zu Berlin, Logos Verlag Berlin, 1996.

[58] R.S. Varga. *Matrix iterative analysis*. Springer, Berlin, 2000.

[59] A. Verhoeven. Automatic control for adaptive time stepping in electrical circuit simulation. Master's thesis, Eindhoven University of Technology, 2004.

[60] A. Verhoeven. Multirate with dynamical partitioning for transient analysis applied to Pstar. Technical Report NXP-R-TN 2007/00154, NXP Semiconductors, Eindhoven, 2007.

[61] A. Verhoeven, T.G.J. Beelen, A. El Guennouni, E.J.W. ter Maten, R.M.M. Mattheij, and B. Tasić. Error analysis of BDF Compound-Fast multirate method for differential-algebraic equations. CASA-Report 06-10[1] , 2006. Extended abstract for Copper Mountain conference.

[62] A. Verhoeven, T.G.J. Beelen, M.L.J. Hautus, and E.J.W. ter Maten. Digital linear control theory for automatic stepsize control. In A.M. Anile, G. Ali, and G. Mascali, editors, *Scientific Computing in Electrical Engineering*, volume 9, pages 137–142, Capo d'Orlando, Italy, 2006. Springer.

[63] A. Verhoeven, T.G.J. Beelen, M.L.J. Hautus, and E.J.W. ter Maten. Digital linear control theory for automatic stepsize control in electrical circuit simulation. In A. Di Bucchianico, R.M.M. Mattheij, and M.A. Peletier, editors, *Progress in Industrial Mathematics at ECMI 2004*, pages 199–203, Eindhoven, The Netherlands, 2006. Springer-Verlag.

[64] A. Verhoeven, A. El Guennouni, E.J.W. ter Maten, and R.M.M. Mattheij. Multirate methods for the transient analysis of electrical circuits. In *PAMM, Virtual Annual Meeting Proceedings of the GAMM conference*, pages 821–822, Luxembourg, 2005.

[65] A. Verhoeven, A. El Guennouni, E.J.W. ter Maten, and R.M.M. Mattheij. A general compound multirate method for circuit simulation problems. In A.M. Anile, G. Ali, and G. Mascali, editors, *Scientific Computing in Electrical Engineering*, pages 143–150, Capo d'Orlando, Italy, 2006. Springer.

[66] A. Verhoeven, E.J.W. ter Maten, R.M.M. Mattheij, and B. Tasić. Stability analysis of the BDF slowest first multirate methods. *Int. J. of Computer Mathematics*, 84:895–923, dec 2007.

[67] A. Verhoeven, B. Tasić, T.G.J. Beelen, E.J.W. ter Maten, and R.M.M. Mattheij. Automatic partitioning for multirate methods. In G. Ciuprina and D. Ioan, editors, *Scientific Computing in Electrical Engineering*, pages 229–236, Sinaia, Romania, 2007. Springer.

[68] A. Verhoeven, B. Tasić, T.G.J. Beelen, E.J.W. ter Maten, and R.M.M. Mattheij. BDF Compound-Fast multirate transient analysis with adaptive stepsize control. CASA-Report 07-05[2], January 2007. Submitted.

[69] T. Voss. Model reduction for nonlinear differential algebraic equations. Master's thesis, Bergische Univ. Wuppertal, 2005.

---

[1] `ftp://ftp.win.tue.nl/pub/rana/rana06-10.pdf`

[2] `ftp://ftp.win.tue.nl/pub/rana/rana07-05.pdf`

[70] T. Voss and A. Verhoeven. Speed up for Pstar by using numerical differential formula (NDF) instead of backward differential formula (BDF). Technical Report PR-TN-2005/00161, Philips ED&T/Analogue Simulation, Eindhoven, 2004.

[71] T. Voss, A. Verhoeven, T. Bechtold, and J. ter Maten. Model order reduction for nonlinear differential-algebraic equations in circuit simulation. In L.L. Bonilla, M. Moscoso, and J.M. Vega, editors, *Progress in Industrial Mathematics at ECMI 2006*, Madrid, Spain, 2007. Springer.

[72] J.K. White and A. Sangiovanni-Vincentelli. *Relaxation techniques for the simulation of VLSI circuits*. Kluwer Academic Publishers, Boston, 1987.

[73] M.C.J.van der Wiel. Numerical time integration techniques for circuit simulation in Pstar. Technical Report No. 6668, NAT.LAB., Eindhoven, 1992.

[74] K. Willcox. Unsteady flow sensing and estimation via the gappy proper orthogonal decomposition. *Computers and Fluids*, 35:208–226, 2006.

Index

A-stable, 53
absolutely stable, 23
accepted timestep, 36
active part, 40
adaptive partitioning control, 90
adaptive stepsize control, 33
adaptivity order, 35
aliasing error, 117
Alternating Current (AC) analysis, 13
averaging operator, 112

Backward Difference Formulae (BDF), 24
balance number, 87
balancing transformation, 104
Banach's contraction theorem, 16
bandwidth, 38
BDF Compound-Fast multirate, 44
branch, 9
branch equations, 10

circuit analysis-methods, 13
companion matrix, 55
compound step, 45
Compound-Fast (CF), 43
computational work per multirate step, 91
computational work per single-rate step, 91
computational workload, 91
consistent initial value for DAE, 16
consistent numerical integration method, 22
constitutive relations, 10
control law for stepsizes, 37
controllability function, 104
corrector polynomial, 28
coupling matrix, 83
current-controlled components, 11
current-defined components, 10

decision strategy about switching between single-rate and multirate, 99
design of Integrated Circuits, 3
differential-algebraic equation (DAE), 15
digital linear control theory, 34
Direct current (DC) analysis, 13

electrical circuits, 9
electrical component, 9
elementary stepsize control, 33
equivalent Nordsieck matrices, 26
error filter, 36
error model, 35
error sequence, 33

Fast-Slow (FS), 42
Fastest First, 41
filter, 35
filter order, 36
filter-properties, 37
first order interpolation, 42
free oscillator problem, 14

Galerkin projection, 112
Gauss-Seidel version, 41
Generalised Compound-Fast (GCF), 43
global DAE-index, 18
global error, 21
Gramians, 104
ground, 12

Hierarchical circuit models, 74

implicit interpolation techniques, 77
Implicit multirate method, 44
improper dynamical system, 106

initial partitioning, 95
initial value problem (IVP), 14
Integrated Circuits, 3
interpolation error, 82
interpolation of function snapshots, 118

Jacobi version, 41
Jacobian matrices, 12

Karhunen-Loéve expansion, 112
Kirchhoff’s Current Law (KCL), 10
Kirchhoff’s Voltage Law (KVL), 10
Kronecker decomposition, 18
Krylov subspace methods, 105

Lagrangian basis polynomial, 24
latent part, 40
Linear Multistep Methods (LMM), 23
linear time-invariant (LTI) system, 19
linearisation tuple (LT), 109
Lipschitz continuous, 16
local discretisation error (LDE), 22
local efficiency analysis, 91
local stability, 20
Lyapunov equations, 104

macrostep, 41
matrix pencil, 18
microstep, 41
Missing Point Estimation (MPE), 115
Mixed Compound-Fast (MCF), 43
model order reduction (MOR), 103
Modified Nodal Analysis (MNA), 11
monotonically stable, 53
multirate error analysis, 81
multirate error model, 85
multirate factor, 40
multirate factor estimation, 92
multirate time-integration, 41
multistep method, 21

network, 10
Nodal Analysis, 11
node, 9
Nordsieck data representation, 26
Nordsieck matrix, 26
numerical time integration methods, 21

observability function, 104
one step method, 21
onestep error model, 34
optimal balance number, 87
optimal partitioning, 93
order of adaptivity, 35
ordinary differential equation (ODE), 15

partition level, 102
partitioned system, 40
partitioning, 40
partitioning sequence, 94
period, 14
Periodic steady-state analysis, 14
predictor polynomial, 28
Principal Component Analysis (PCA), 112
proper dynamical system, 106
proper orthogonal decomposition (POD), 112
Prothero-Robinson equation, 53

recurrence relation, 55
reducible, 76
redundancy, 4
refinement phase, 45
rejected timestep, 36
rejection strategy, 100
relaxation
  linear relaxation, 40
  nonlinear relaxation, 41
  waveform relaxation, 41
relaxation number, 81
repartitioning, 90
rescaling and resizing of error vectors, 93
Runge Kutta methods, 21

scaled local discretisation error, 22
selection operators, 40
semi-explicit DAE, 15
Semi-implicit multirate, 42
semi-optimal partitioning, 94
shift-operator, 34
single-rate time-integration, 39
Slow-Fast (SF), 42
Slowest First, 41
smoothness, 33

solvability, 18
stability analysis of
  Euler Backward multirate algorithm, 55
  multirate schemes, 53
  multistep BDF multirate algorithms, 62
  the dynamical system, 19
steady-state, 13
stepsize control, 33
stepsize filter, 36
stepsize sequence, 33
switching strategy between single-rate and multirate, 99
synchronised time-grids, 39

test equation, 54
topology matrix, 10
Trajectory Piecewise Linear (TPWL), 109
transfer function, 105
Transient analysis, 14
transitions of partitioning, 94
transposed Nordsieck vector, 26
two-dimensional test equation, 54
two-point boundary value problem, 14

unconditionally stable, 53

voltage-controlled components, 11
voltage-defined components, 10

Waveform Relaxation (WR), 41
workload ratio, 91

zeroth order extrapolation, 42

Summary

Redundancy Reduction of IC Models by Multirate Time-Integration and Model Order Reduction

Circuit simulation is an essential step within circuit design. Because of the increasing complexity of the Integrated Circuits, electronic companies need fast and accurate simulation software and there is a constant request at the companies to further improve the simulation software. Development of new, more advanced, transient simulation algorithms is an attractive way to increase the performance of this software. Mathematics is the basis to analyze the convergence properties.

The objective of this PhD research is to increase the performance of Pstar, the in-house analog circuit simulator at Philips and now of NXP Semiconductors, while properties like accuracy and robustness are maintained. In particular the convergence and stability properties of newly developed multirate time-integration algorithms is studied. Usually circuit models are large systems of differential-algebraic equations that are derived from Kirchhoff's conservation laws for currents and voltages and the constitutive relations for the electronic components. For a transient analysis one traditionally uses implicit time-integration schemes, like Backward Difference Formulae (BDF). All these schemes discretise the time on one time-grid. In contrast multirate algorithms use more than one time-grid and compute the slowly time-varying state elements only at coarsely distributed time-points, while the fastly time-varying state elements are computed at finer distributed timepoints. This makes a multirate algorithm potentially much faster for circuits with large low-frequency parts. There are many types of multirate time-integration methods that may differ in the order of the slow and fast integration and the treatment of the interface variables. We used a direct extension of the BDF scheme combined with Lagrange interpolation of the same order.

The standard theory for multistep methods does not hold anymore for multirate algorithms. Therefore we look at properties like stability and convergence in more detail. It turns out that the method is stable if the partitioned subsystems are individually stable and if the coupling is sufficiently weak. The discretisation error for a multirate method also contains an interpolation error due to the slow unknowns at the interface. This error component is not needed for ordinary multistep methods. It is possible to control this error by independent control of the coarse and fine macro and micro time-steps, respectively. The interpolation error and the coarse discretisation error is controlled by

the macro stepsize, while the micro stepsize controls the fine discretisation errors for the fast state part. For multirate it is necessary to partitioning the system into a slow and a fast part. Therefore a part of the research is spent to the development and analysis of automatic partitioning algorithms. The underlying problem is a discrete optimisation problem, that can be handled by greedy-like methods. It is also possible to change the partitioning dynamically during the simulation, which is useful for moving active parts. All algorithms are implemented in Matlab; they work satisfactorily when tested for a variety of circuit models. Furthermore a multirate implementation including error control and dynamical partitioning is created in the circuit simulator Pstar itself.

Besides multirate time-integration also model order reduction is studied, which transforms the large data models into smaller and simpler models, that still give the proper accuracy, but that are much cheaper to solve. Because IC models are nonlinear, nonlinear reduction techniques are considered in particular, like POD. In particular we focused on the problem to reduce the evaluation costs of these reduced models.

A proper use of multirate and model order reduction is able to speed up transient simulation in general and is significantly faster (more than an order) for redundant circuit models, while the accuracy and robustness are maintained. Redundancy occurs if the state elements have many correlations, or if the sampled state signal has correlations in time.

Samenvatting

Redundantie Reductie van IC Modellen door Multirate Tijd-Integratie en Model Orde Reductie

Circuit simulatie is een belangrijk element binnen het ontwerpen van Integrated Circuits. Vanwege de toenemende complexiteit van deze circuits heeft de elektronische industrie behoefte aan snelle en nauwkeurige simulatie software en is er een constante vraag naar verdere verbetering ervan. De ontwikkeling van nieuwe geavanceerde transient simulatie algoritmen is een aantrekkelijke manier om deze software te verbeteren. Wiskunde is de basis waarmee de convergentie eigenschappen kunnen geanalyseerd worden.

Het doel van dit PhD onderzoek is om de simulatietijd van Pstar te verkleinen, zonder dat eigenschappen als nauwkeurigheid en robuustheid achteruit gaan. Pstar is de eigen circuit simulator van NXP Semiconductors en wordt ook gebruikt door Philips. We bestuderen in het bijzonder de convergentie en stabiliteitseigenschappen van nieuw ontwikkelde multirate tijdsintegratie algoritmen. Gewoonlijk geven circuit modellen grote stelsels differentiaal-algebraïsche vergelijkingen, welke afgeleid worden van de wetten van Kirchhoff voor de stromen en spanningen en de constitutieve vergelijkingen voor de elektrische componenten. Voor een transient analyse gebruikt men gewoonlijk impliciete tijdsintegratie methoden, zoals de Backward Difference Formulae (BDF), die de tijd discredtiseren op één tijdrooster. In tegenstelling tot deze methoden gebruiken multirate algoritmen meer dan één tijdrooster en worden de langzaam in de tijd variërende toestandselementen alle op grof verdeelde tijdpunten berekend, terwijl de snel in de tijd variërende toestandselementen op alle tijddiscretisatiepunten berekend worden. Dit maakt een multirate algoritme potentieel veel sneller voor circuits met grote laag-frequente delen. Er zijn veel soorten multirate-methoden die verschillen in de volgorde waarmee de langzame en snelle delen geïntegreerd worden en in de behandeling van de variabelen op de randen tussen de verschillende delen. We gebruiken een directe uitbreiding van de BDF methode gecombineerd met Lagrange interpolatie van dezelfde orde.

De bestaande theorie voor meerstaps methoden geldt niet meer voor multirate algoritmen. Daarom analyseren we opnieuw eigenschappen zoals stabiliteit en convergentie. Het blijkt dat de methode stabiel is als de subsystemen afzonderlijk stabiel zijn en als de koppeling voldoende klein is. De discretisatiefout van een multirate methode bevat

ook een interpolatiefout vanwege de koppeling met de rand. Deze foutcomponent komt niet voor bij gewone meerstaps methoden. Het is mogelijk om deze fout te regelen door onafhankelije regeling van de macro- en minitijdstappen. De interpolatiefout en de discretisatiefout op het grove rooster worden bepaald door de macro stapgrootte, terwijl de micro stapgrootte de discretisatiefout van het snelle deel op het verfijnde tijdrooster bepaalt. Voor multirate is het nodig om het stelsel te partitioneren in een langzaam en een snel deel. Daarom is een deel van het onderzoek besteed aan de ontwikkeling en analyse van automatische partitioneer-algoritmen. Het onderliggende probleem is een discreet optimalisatieprobleem dat benaderd kan worden met greedy-achtige methoden. Het is ook mogelijk om de partitie te veranderen tijdens de simulatie, wat nuttig is voor zich verplaatstende activiteit. Alle algoritmen zijn geïmplementeerd in Matlab; ze bljken goed te werken voor een aantal geteste circuit modellen. Bovendien is multirate inclusief foutcontrole en dynamische partitionering geïmplementeerd in de circuit simulator Pstar zelf.

Behalve multirate tijdsintegratie wordt model orde reductie bestudeerd. Hierbij wordt een groot wiskundig model getransformeerd in een klein en eenvoudig model. Zo'n gereduceerd model moet nog steeds nauwkeurig genoeg zijn, maar veel goedkoper te simuleren. Omdat IC modellen niet-lineair zijn, worden in het bijzonder niet-lineaire reductie technieken bestudeerd, zoals POD. Ook beschouwen we het probleem om de evaluatiekosten van deze gereduceerde modellen te reduceren.

Een goed gebruik van multirate en model orde reductie kan in het algemeen een transient simulatie versnellen en is (zeker meer dan een orde) sneller voor redundante circuit modellen, terwijl de nauwkeurigheid en robuustheid worden behouden. Redundantie treedt op als de toestandselementen veel onderlinge correlaties hebben of als de gediscretiseerde toestand correlaties in de tijd heeft.

Curriculum vitae

The author of this thesis was born July 18th, 1980, in Utrecht, the Netherlands. He finished his preuniversity education at the Van Lodenstein College Amersfoort in 1998. The name of this school refers to Jodocus van Lodenstein, who was a Dutch reformed preacher of the seventeenth century.
Arie started studying Applied Mathematics at Eindhoven University of Technology, and received the propaedeutic diploma a year later. In 2004, Arie graduated with honors. His master's thesis, written under the supervision of prof.dr.ir. M.L.J. Hautus, was entitled *Automatic control for adaptive time stepping in electrical circuit simulation*. This project was already supervised by dr. E.J.W. ter Maten from Philips Research.
In January 2004 Arie started working as a PhD student in the Scientific Computing Group (CASA since 2004) of prof.dr. R.M.M. Mattheij at Eindhoven University of Technology. This PhD project was a cooperation with the same department of Philips Research (NXP Semiconductors since 2006). The author studied how the transient simulation can be improved by use of multirate time integration and model order reduction. This research has led to several papers and finally to this thesis.
Arie married Cobi Boer June 9th, 2005 and moved to Best. They have one son, Johan, who was born March 26th, 2007. The author will take on a position as scientific programmer at VORtech Computing at Delft.
He can still be reached via email at `Arie.Verhoeven@na-net.ornl.gov`.

Acknowledgements

During the time that I was a PhD student, I have had support from many people. Therefore, I would like to express my gratitude towards them. First of all, I would like to thank Bob Mattheij and Jan ter Maten for the coordination of the project, but also for the mathematical support and many other things. Also the other members of the reading committee, Michael Günther, Jacquelien Scherpen and Wil Schilders, are thanked for their comments.
I am greatly indebted to all colleagues who motivated and helped me a lot during my research, like Patricia Astrid, Tamara Bechtold, Ahmed El Guennouni, Joost Rommes, Valeriu Savcenco, Michael Striebel, Bratislav Tasić and Thomas Voss.
Of course I am also grateful to all other members of the CASA group of the Eindhoven University of Technology for their hospitality and support. Furthermore I would like to thank the company NXP Semiconductors because of the opportunity to apply mathematical ideas to practical examples from industry. In particular the people from the Design Methods group of NXP Semiconductors (ED&T/AS of Philips Research in previous times) are thanked, like Theo Beelen, Jos Dohmen, Marcel Sevat, etc. Last, but not least, I would like to thank my wife Cobi and other family members for their support. Finally I will end by citing some Dutch verses of ds.J. van Lodenstein.

Wij zijn niet dan voor de Heer',
En ons heil is in Zijn eer;
En Zijn eer in al Zijn werken.
Als Hij ons dan zinken doet,
Kunnen wij Zijn lof maar sterken;
Wat Hij doet is even goed.

Wijsheid zonder eind of paal
Zijn Zijn wegen altemaal.
Zijn zij zuurheid, zijn ze zoetheid,
Laat ons altijd zwijgen stil,
Want de wezenlijke Goedheid
Maakt het goed met dat Zij 't wil.

Arie Verhoeven
Best, November 2007