LUND UNIVERSITY

Adaptive time-integration for goal-oriented and coupled problems

Meisrimel, Peter

2021

Document Version:
Publisher's PDF, also known as Version of record

Link to publication

Citation for published version (APA):
Meisrimel, P. (2021). *Adaptive time-integration for goal-oriented and coupled problems.* Lund University.

Total number of authors:
1

General rights
Unless other specific re-use rights are stated the following general rights apply:
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

Read more about Creative commons licenses: https://creativecommons.org/licenses/

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

– CENTRUM SCIENTIARUM MATHEMATICARUM –

Adaptive time-integration for goal-oriented and coupled problems

PETER MEISRIMEL

Lund University
Faculty of Science
Centre for Mathematical Sciences
Numerical Analysis

Printed by Media-Tryck, Lund 2021 NORDIC SWAN ECOLABEL 3041 0903

Doctoral Theses in Mathematical Sciences 2021:7

ISBN 978-91-7895-930-3
ISSN 1404-0034

Adaptive time-integration for goal-oriented and coupled problems

Adaptive time-integration for goal-oriented and coupled problems

by Peter Meisrimel

LUND
UNIVERSITY

Thesis for the degree of Doctor of Philosophy in Numerical Analysis
Thesis advisors: Prof. Dr. Philipp Birken, Prof. Dr. Claus Führer
Faculty opponent: Prof. Dr. Andreas Dedner

To be presented, with the permission of the Faculty of Science of Lund University, for public criticism in the lecture hall Hörmandersaalen at the Centre for Mathematical Sciences, Sölvegatan 18A, Lund, on Thursday, the 2nd of September 2021 at 14:00.

Organization
LUND UNIVERSITY

Centre for Mathematical Sciences
Box 118
SE–221 00 LUND
Sweden

Document name
DOCTORAL DISSERTATION

Date of disputation
2021-09-02

Sponsoring organization

Author(s)
Peter Meisrimel

Title and subtitle
Adaptive time-integration for goal-oriented and coupled problems

Abstract
We consider efficient methods for the partitioned time-integration of multiphysics problems, which commonly exhibit a multiscale behavior, requiring independent time-grids. Examples are fluid structure interaction in e.g., the simulation of blood-flow or cooling of rocket engines, or ocean-atmosphere-vegetation interaction. The ideal method for solving these problems allows independent and adaptive time-grids, higher order time-discretizations, is fast and robust, and allows the coupling of existing subsolvers, executed in parallel.

We consider Waveform relaxation (WR) methods, which can have all of these properties. WR methods iterate on continuous-in-time interface functions, obtained via suitable interpolation. The difficulty is to find suitable convergence acceleration, which is required for the iteration converge quickly.

We develop a fast and highly robust, second order in time, adaptive WR method for unsteady thermal fluid structure interaction (FSI), modelled by heterogeneous coupled linear heat equations. We use a Dirichlet-Neumann coupling at the interface and an analytical optimal relaxation parameter derived for the fully-discrete scheme. While this method is sequential, it is notably faster and more robust than similar parallel methods.

We further develop a novel, parallel WR method, using asynchronous communication techniques during time-integration to accelerate convergence. Instead of exchanging interpolated time-dependent functions at the end of each time-window or iteration, we exchange time-point data immediately after each timestep. The analytical description and convergence results of this method generalize existing WR theory.

Since WR methods allow coupling of problems in a relative black-box manner, we developed adapters to PDE-subsolvers implemented using `DUNE` and `FEniCS`. We demonstrate this coupling in a thermal FSI test case.

Lastly, we consider adaptive time-integration for goal-oriented problems, where one is interested in a quantity of interest (QoI), which is a functional of the solution. The state-of-the-art method is the dual-weighted residual (DWR) method, which is extremely costly in both computation and implementation. We develop a goal oriented adaptive method based on local error estimates, which is considerably cheaper in computation. We prove convergence of the error in the QoI for tolerance to zero under a controllability assumption. By analyzing global error propagation with respect to the QoI, we can identify possible issues and make performance predictions. Numerical results verify these results and show our method to be more efficient than the DWR method.

Key words
time-adaptivity, goal-oriented problems, error estimation, coupled problem, waveform relaxation, asynchronous methods, conjugate heat transfer

Classification system and/or index terms (if any)

Supplementary bibliographical information

Language
English

ISSN and key title
1404-0034

ISBN
978-91-7895-930-3 (print)
978-91-7895-929-7 (pdf)

Recipient's notes

Number of pages
176

Price

Security classification

I, the undersigned, being the copyright owner of the abstract of the above-mentioned dissertation, hereby grant to all reference sources the permission to publish and disseminate the abstract of the above-mentioned dissertation.

Signature _____ Date 2021-06-01

DOKUMENTDATABLAD enl SIS 61 41 21

Adaptive time-integration for goal-oriented and coupled problems

by Peter Meisrimel

LUND
UNIVERSITY

Funding information: This work in thesis is made possible due to the support of the Crafoord Foundation under grant number 20150681 and the Swedish e-science collaboration eSSENCE.

Numerical Analysis
Centre for Mathematical Sciences
Lund University
Box 118
SE-221 00 Lund
Sweden

www.maths.lu.se

Doctoral Thesis in Mathematical Sciences 2021:7
ISSN: 1404-0034

ISBN: 978-91-7895-930-3 (print)
ISBN: 978-91-7895-929-7 (pdf)
LUNFNA-1009-2021

© Peter Meisrimel, 2021

Printed in Sweden by Media-Tryck, Lund University, Lund 2021

Abstract

We consider efficient methods for the partitioned time-integration of multiphysics problems, which commonly exhibit a multiscale behavior, requiring independent time-grids. Examples are fluid structure interaction in e.g., the simulation of blood-flow or cooling of rocket engines, or ocean-atmosphere-vegetation interaction. The ideal method for solving these problems allows independent and adaptive time-grids, higher order time-discretizations, is fast and robust, and allows the coupling of existing subsolvers, executed in parallel.

We consider Waveform relaxation (WR) methods, which can have all of these properties. WR methods iterate on continuous-in-time interface functions, obtained via suitable interpolation. The difficulty is to find suitable convergence acceleration, which is required for the iteration converge quickly.

We develop a fast and highly robust, second order in time, adaptive WR method for unsteady thermal fluid structure interaction (FSI), modelled by heterogeneous coupled linear heat equations. We use a Dirichlet-Neumann coupling at the interface and an analytical optimal relaxation parameter derived for the fully-discrete scheme. While this method is sequential, it is notably faster and more robust than similar parallel methods.

We further develop a novel, parallel WR method, using asynchronous communication techniques during time-integration to accelerate convergence. Instead of exchanging interpolated time-dependent functions at the end of each time-window or iteration, we exchange time-point data immediately after each timestep. The analytical description and convergence results of this method generalize existing WR theory.

Since WR methods allow coupling of problems in a relative black-box manner, we developed adapters to PDE-subsolvers implemented using `DUNE` and `FEniCS`. We demonstrate this coupling in a thermal FSI test case.

Lastly, we consider adaptive time-integration for goal-oriented problems, where one is interested in a quantity of interest (QoI), which is a functional of the solution. The state-

of-the-art method is the dual-weighted residual (DWR) method, which is extremely costly in both computation and implementation. We develop a goal oriented adaptive method based on local error estimates, which is considerably cheaper in computation. We prove convergence of the error in the QoI for tolerance to zero under a controllability assumption. By analyzing global error propagation with respect to the QoI, we can identify possible issues and make performance predictions. Numerical results verify these results and show our method to be more efficient than the DWR method.

Popular Science Summary

The foundation of modeling physical phenomena is their description by mathematical models, most prominently *differential equations* (*DE*). *DE*s are used to model phenomena in all natural sciences, engineering disciplines and more: Classical mechanical engineering applications such as construction or vehicle design, electrical field and circuit modelling, simulation of chemical reactions, climate and vegetation simulation, infectious disease modelling and many more, are all based on *DE*s, albeit each with their own nuances and flavors.

*DE*s relate rate of change to a given state and known inputs. Solving a *DE* means to determine future states based on an initial state. An example is Newton's second law: *Force* = *mass* $\times$ *acceleration*. Acceleration is the rate of change of velocity, which is the rate of change of position. Solving this *DE* means to determine future position based on a known initial position, force and mass.

Analytical solutions, obtained via solving the *DE* by hand, are extremely rare. Instead, with the rise of computers and computing power, we solve them using *numerical methods*, algorithms designed to approximate their solutions. This enables the modelling of highly complex physical phenomena. In particular, product design and development based on numerical models is typically faster, cheaper and safer than traditional methods using physical prototypes.

While artificial intelligence and machine learning are very hot topics right now, these are unlikely to replace physical based modeling using *DE*s, since machine learning is mostly used for finding patterns in data. However, most physical phenomena are well described by *DE*s, the difficulty is to accurately, yet efficiently solve them. Thus, the development of *efficient numerical methods* is of key importance, since it enables e.g., more precise weather forecasts and climate change predictions. Additionally, current development on increased computing power involves parallel machines and making use of them requires suitable numerical methods.

This thesis consists of three parts centered around *numerical methods* for *DE*s. One solves

these in a step-wise manner, yielding a solution at fixed time-points, e.g. weather data every 15 minutes. Smaller steps give a more accurate result, but increase the cost of computing the whole solution. Another approach is to *adaptively* choose stepsizes. That is choosing them as large as possible and as small as necessary to reduce computational cost, while retaining a target accuracy. *Adaptive* approaches are based on estimating the accuracy and then choosing a suitable stepsize.

In the first part of the thesis we developed and analyzed *adaptive* methods in the context of *goal-oriented problems*. In practice, one solves *DE*s to answer questions such as "How much energy does this turbine produce?", "How fuel efficient is this vehicle design?", "Is component X sufficiently cooled?" or "How fast are the polar caps melting?". Here, the goal is the computation of a *quantity of interest (QoI)* (e.g., average energy produced, maximum temperature, total ice loss per year) based on the solution of the *DE*, to answer the relevant question.

We derived an *adaptive* method for the solution of *goal-oriented problems* that chooses stepsizes to control the accuracy in the *QoI*. Additionally, we established mathematical conditions for when such an approach is sensible, with guidelines to predict performance gains or losses. Our new method shows notable performance improvements in various test cases.

The remaining part of the thesis concerns *coupled problems*. These are physical phenomena described by a multitude of connected *DE*s. An example is thermal fluid-structure interaction in the cooling of rocket nozzles. This involves simulating the interaction between extremely hot combustion products, the solid nozzle structure, liquid coolant within the structure and the ambient air outside. Another example is climate simulation, featuring connected *DE*s for atmosphere/ocean dynamics + chemistry, vegetation, cloud and ice formation, and possibly more.

There are different approaches in designing simulation software for coupled problems. The *monolithic approach* is to view the coupled problem as a whole and to design and implement a tailor-made algorithm to solve it. However, changes or additions to the coupled problem require changes to the whole algorithm. Additionally, designing an efficient algorithm requires the combined expertise of doing so for all subproblems.

We consider the so-called *partitioned approach*, which aims to solve a coupled problem using solvers for the subproblems combined with a suitable coupling method. Partitioned coupling methods typically involve an iterative process in which the subproblems need to be solved multiple times each. The advantage of the partitioned approach is that it allows re-using existing specialized codes for the subsolvers. In particular, this does not necessarily require a lot of specialized knowledge on the subproblems.

The re-use of software is a pre-requisite for sustainable software development and being able to jointly work in larger groups. This approach makes it far easier to exchange subsolvers

or add new components to a coupled problems.

We studied *Waveform Relaxation (WR)* methods, an iterative partitioned coupling method that allows a large degree of independence in solving the subproblems. This is highly desirable when coupling different physical phenomena that require different numerical methods or stepsizes for solving them.

First we developed a *WR* method for solving dynamic heterogeneous coupled heat equations. That is, modeling the time-dynamics of heat exchange between different materials as e.g., present in the fluid-structure interaction when cooling rocket nozzles. The resulting method performs better than existing methods, by using independent and *adaptive* stepsizes when solving the subproblems, resolving the coupling reliably in very few iterations.

Secondly, we developed a novel parallel *WR* method, which requires fewer iterations than classical parallel *WR* methods, by utilizing asynchronous communication techniques. The resulting method shows promising performance results. Its analytical description and theoretical results generalize existing theory on *WR* methods.

Lastly, we developed software for solving coupled problems using *WR* methods. In particular, we are able to couple *DE*s implemented using certain free open-source packages that provide a vast range of tools for solving complex *DE*s at relative ease. With this, we are able to couple different *DE* subsolvers almost at a "plug-and-play" principle, to create solvers for multiphysics problems.

Scientific publications

This thesis is based on the following publications, referred to by their Roman numerals:

I **Goal Oriented Time Adaptivity Using Local Error Estimates**

P. Meisrimel, P. Birken
Algorithms 13.5 (2020): 113

II **On Goal Oriented Time Adaptivity using Local Error Estimates**

P. Meisrimel, P. Birken
PAMM 17.1 (2017): 849-850

III **A time adaptive multirate Dirichlet–Neumann waveform relaxation method for heterogeneous coupled heat equations**

P. Meisrimel, A. Monge, P. Birken
In final preparation for submission.

IV **Waveform Relaxation with asynchronous time-integration**

P. Meisrimel, P. Birken
In final preparation for submission.

V **Waveform Iteration with asynchronous communication**

P. Meisrimel, P. Birken
PAMM 19.1 (2019): e201900410

Author contributions

Since the papers have multiple authors, I here describe my contributions to each.

Paper I and Paper II: Goal Oriented Time Adaptivity Using Local Error Estimates

I performed the analysis, proofs and numerical experiments. The global error propagation analysis in Section 4.3 in Paper I was my original idea. All coding, except for the implementation of the linear heat equation test case, was done by me. I was the main author of the paper.

Paper III: A time adaptive multirate Dirichlet–Neumann waveform relaxation method for heterogeneous coupled heat equations

The original idea for the methods in the paper is from Azahar Monge and Philipp Birken. I came up with the key ideas for advancing the development to the methods presented in the paper. My contribution to the writing was in the technical description of the methods, experiments and numerical results. All numerical experiments were designed and performed by me. The implementation of the methods was done by me, building upon an earlier code from Azahar Monge. I was the main author of the paper.

Paper IV and Paper V: Waveform Relaxation with asynchronous time-integration

I designed and implemented the algorithms and numerical experiments, and performed the analysis presented in the paper. The analytical description of the method and the variable relaxation algorithm were my original idea. I received help in the implementation of the `DUNE` test case. I was the main author of the paper.

Acknowledgements

This thesis is the result of my academic journey and I want to thank all those that made this possible and supported me along the way.

First and most importantly thanks to my supervisor Philipp Birken for all his guidance and patience. I'm very happy to have a motivated and involved supervisor as you are. Secondly, thanks to my co-supervisor Claus Führer for his shared knowledge and opening new perspectives by constantly challenging me. Also a big thanks to Gustaf Söderlind, who also co-supervised me during the first half of this thesis, helping to guide this work and sharing knowledge in many long discussions.

Next I want to thank the whole Numerical analysis group (past and present) for making this work environment feel like a second home: Alexandros, Azahar, Carina, Carmen, Christian, Claus, Dara, Emil, Erik, Eskil, Fatemeh, Joachim, Julio, Lea, Måns, Monika, Philipp, Robert, Stefan, Tony and Viktor. A particular thanks to my academic sisters Azahar and Lea for being a large support through the woes of PhD student life. Also a big thanks to Robert Klöfkorn for all the development in `DUNE` to facilitate the final numerical experiments.

I also want to thank my family for being encouraging and having given me the freedom to learn and get to where I am today. Thanks to my friends, in Sweden and abroad: Adrian, Anna, Dorothea, Erik, Frankie, Kamilla, Lisa, Moritz, Nicolas, Ngoc, Vitali, Xiaoli and Ylva, for all the fun, games and adventures helping to keep my life in balance.

A thanks to all contributors of Wikipedia, Stackoverflow and Stackexchange, for helping contributing to these amazing resources of knowledge, solutions to programming issues, LaTeX and more.

And finally I want to thank my love Zuleita Ho and our furry companions Guma and Kuro, being the best home-office companions during the pandemic and generally making everything else about my life more enjoyable.

Funding

This work in thesis is made possible due to the support of the Crafoord Foundation under grant number 20150681 and the Swedish e-science collaboration eSSENCE.

Contents

1 Introduction 1

1.1 Goal oriented adaptivity . . . 2
1.2 Coupled problems . . . 3
1.2.1 Waveform Relaxation . . . 4
1.2.2 Thermal Fluid Structure interaction . . . 5
1.2.3 Waveform Relaxation with asynchronous time-integration . . . 6
1.3 Organization of thesis . . . 7

2 Time-adaptivity using local error estimates 9

2.1 Time adaptivity based on local error estimates . . . 10
2.1.1 Error estimation . . . 12
2.1.2 Timestep controller . . . 13
2.1.3 Convergence in the solution . . . 14
2.2 On time invariance and asymptotic correctness . . . 16
2.2.1 Time invariance . . . 16
2.2.2 Asymptotic correctness . . . 21

3 Goal-oriented time-adaptivity using local error estimates 23

3.1 Goal oriented time adaptivity based using local error estimates . . . 24
3.1.1 Error estimate and timestep controller . . . 25
3.1.2 Convergence in the quantity of interest . . . 25
3.2 Numerical results and conclusions . . . 27

4 Waveform Relaxation 29

4.1 Continuous Waveform relaxation method . . . 30
4.2 Time-discrete Waveform relaxation method . . . 31
4.3 Convergence Analysis . . . 32
4.3.1 Continuous iteration . . . 34
4.3.2 Time-discrete iteration . . . 36
4.4 Relaxation . . . 39
4.5 Windowing . . . 40
4.6 Implementation . . . 40
4.6.1 Adapter . . . 43

4.6.2 Discussion . . . 44

5 Dirichlet-Neumann Waveform Relaxation for heterogeneous coupled heat equations 45
5.1 Dirichlet-Neumann Waveform relaxation . . . 46
5.1.1 Semi-discrete iteration . . . 46
5.2 Time-discretization . . . 49
5.2.1 Implicit Euler . . . 49
5.2.2 SDIRK2 . . . 51
5.2.3 Multi-rate and adaptive partitioned time-integration . . . 53
5.3 Optimal relaxation parameter . . . 53
5.4 Summary of numerical results . . . 54

6 Waveform Relaxation with asynchronous time-integration 57
6.1 Waveform relaxation with asynchronous communication . . . 57
6.2 Convergence analysis . . . 59
6.2.1 Time-discrete WR with asynchronous communication . . . 59
6.2.2 Continuous WR with asynchronous communication . . . 61
6.3 Summary of numerical results . . . 63

7 Conclusions and future work 65
7.1 Summary and conclusions . . . 65
7.2 Future work . . . 67

8 Appendix 69
8.1 Linear multistep methods . . . 69
8.2 On (block)normal matrices . . . 69

Bibliography 73

Paper I: Goal Oriented Time Adaptivity Using Local Error Estimates 81

Paper II: On Goal Oriented Time Adaptivity using Local Error Estimates 105

Paper III: A time adaptive multirate Dirichlet–Neumann waveform relaxation method for heterogeneous coupled heat equations 109

Paper IV: Waveform Relaxation with asynchronous time-integration 131

Paper V: Waveform Iteration with asynchronous communication 155

Chapter 1

Introduction

Modern simulations typically involve modelling problems comprised of a multitude of connected systems with different physics. Examples are coupled mechanical systems or fluid-structure interaction in e.g., simulation of blood flow in large arteries [16] or cooling of rocket engines [37, 38]. Another example is climate simulation, which involves simulation of the ocean and atmosphere, the chemistry within them, as well as modelling of vegetation, cloud formation and more.

When simulating such *multiphysics problems* one requires different computational methods to solve the subproblems, commonly handled by different software. Additionally, multiphysics problems commonly exhibit multirate/multiscale behavior. I.e., processes operate on different spatial or temporal scales, requiring separate treatment. Consequently, the development and analysis of methods to orchestrate the coupling of heterogeneous codes and numerical methods is an important research topic.

Parellelization is an absolute necessity when performing large computational tasks. The standard method to solve partial differential equations (PDEs) in parallel, is via the parallelization of matrix vector products or using Domain decomposition [61, 82] techniques. These correspond to subdividing the computational domain and each processor solves the problem on a part of it. In multiphysics problems with different codes for each subproblem, the evident idea is to solve the subproblems in parallel.

When solving differential equations using discrete time-integration, constant stepsizes are the simplest and most straight-forward. However, to achieve computationally efficient methods, one wants to use stepsizes as large as possible and as small as necessary. This is particularly necessary if the differential equations have processes operating on different scales, as is common in multiphysics problems. With *adaptive methods* one automatically chooses stepsizes based on estimates to the accuracy of the solution.

Thus, the ideal method for solving multiphysics problems executes different subsolver codes in parallel, allowing independent and adaptive time-grids, and is both fast and robust. I.e., it works well for a wide range of problems. This task has been formulated as a major challenge in computational sciences [11, 20, 65].

Additionally, practical simulations come with the objective of answering questions such as: How fuel efficient is a vehicle design (lift over drag)? How much energy does a certain turbine design produce [89]? Where to place wind/tidal turbines for maximal energy output [22]? What is the expected average global temperature increase? What is the total ice loss of the (ant)artic [29]? Is a building/room sufficiently heated/ventilated?

These are called *goal oriented problems*. In addition to solving the differential equations modelling the relevant physical processes, one computes a *quantity of interest (QoI)* to answer the original question. Using adaptive methods, one aims to control the accuracy in the QoI, rather than the whole solution.

This thesis consists of three separate parts, corresponding to Paper I , Paper III and Paper IV . (Paper II and Paper V represent early stage results of Paper I resp. Paper IV .) Paper I presents a local error based goal-oriented time-adaptive method. The two remaining parts concern Waveform relaxation (WR) methods for the partitioned time-integration of coupled multiphysics problems. In Paper III we developed a highly robust WR method for the particular application of *thermal fluid structure interaction*. Paper IV presents a novel parallel WR method utilizing asynchronous communication. We now present the individual topics in more detail.

1.1 Goal oriented adaptivity

The standard approach for adaptivity in goal oriented problems is the dual weighted residual (DWR) method [4, 7, 60]. It is most commonly used in the context of space-adaptivity for PDEs, but there is also work specifically on time-adaptivity [48, 49, 78]. DWR uses the adjoint (dual) problem to perform a posteriori adaptivity to achieve error bounds in the QoI.

In the time-dependent case, the adjoint problem is a terminal value problem, i.e., an initial value problem (IVP) backwards in time. A single iteration of the DWR method consists of solving both the original IVP and its adjoint problem on a given time-grid. Then, one constructs the error estimate (error bound for linear problems) in the QoI using both solutions, which may have considerable storage requirements. The error estimate contains local information used to refine the time-grid. This procedure is repeated until a grid is found, where the error estimate fulfills a target error bound. This algorithm is very costly not only in terms of computation and possibly storage, but also in implementation, since adjoints

are specific to the IVP and the QoI.

With Paper I , we investigate easy to implement and computationally cheaper methods for goal oriented time-adaptivity. Specifically, we consider local error based methods [73], in which time-grids are constructed during time-integration, using local information. This approach does not yield estimates of the global error, but is computationally cheap and works well in practice. We develop a goal oriented adaptive method using estimates of the local error in the QoI and show convergence in the QoI under a controllability condition.

We numerically verify our convergence results using a variety of tests. A performance comparison shows that local error based methods are much more efficient than the DWR method for dissipative problems. Our new goal oriented adaptive method shows good performance in most cases and significant speedups in some. Based on analyzing error propagation w.r.t. the QoI, we can predict the performance of our new method w.r.t. classical, non goal-oriented local error based adaptive methods. It turns out to be relatively easy to predict bad performance, but harder to predict performance improvements.

1.2 Coupled problems

Multiphysics simulations as a research field has emerged from the necessity of simulating coupled problems in various scientific areas. As such, there is no established standard of notation and we largely follow [20].

Consider the following *coupled problem*:

$$\dot{\mathbf{v}}(t) = \mathbf{g}(t, \mathbf{v}(t), \mathbf{w}(t)), \qquad \mathbf{v}(0) = \mathbf{v}_0 \in \mathbb{R}^{d_v}, \tag{1.1a}$$

$$\dot{\mathbf{w}}(t) = \mathbf{h}(t, \mathbf{v}(t), \mathbf{w}(t)), \qquad \mathbf{w}(0) = \mathbf{w}_0 \in \mathbb{R}^{d_w}, \tag{1.1b}$$

with $t \in [0, T_f]$. Here, (1.1a) and (1.1b) represent two systems of ordinary differential equations (ODEs), resp. semi-discrete PDEs, corresponding to different physical phenomena. We call (1.1) the *monolithic problem* and refer to the individual problems as *subproblems*. W.l.o.g., we consider two coupled systems, since the extension to more systems is largely straight-forward from a theoretical point-of-view.

We particularly consider coupled problems with a *strong bidirectional surface coupling*. I.e., the subproblems have a strong interaction via a lower dimensional interface, as e.g., in thermal fluid structure interaction. An example for a *weak coupling* is within climate simulation. There, ocean and atmosphere chemistry are slow processes and have only a weak effect on the underlying fluid-dynamics.

The *monolithic approach* is to directly solve (1.1), using methods tailored to the specific problem. In the *partitioned approach* one solves (1.1) by solving the subproblems in con-

junction with a suitable coupling method, commonly called a *partitioned time-integration method*, which facilitates the information exchange via the interface. With suitable coupling methods, the subproblems can be solved using independent grids and numerical methods. Strongly coupled problems require *strongly coupled methods*, which is also called an *implicit coupling*. This typically requires an iteration on the interface solutions, to avoid stability problems [14]. This iteration requires repeated solutions of the subproblems for varying interface data.

Partitioned methods enable the re-use of existing specialized software for solving the subproblems. These may be highly efficient and sophisticated solvers with problem specific discretizations, preserving e.g., mass, entropy or positivity. Additionally, open source libraries such as `deal-II` [2], `DUNE` [6], `FEniCS` [41], `Firedrake` [62], `OpenFOAM` [35], `SU2` [19], or also commercial packages, enable the creation of single-physics PDE solvers at relative ease.

Using suitable adapters [84] this allows for the construction of multiphysics simulations almost at a "plug-and-play" principle, see Figure 1.1. Examples are the coupling libraries `preCICE` [13] or `MpCCI` [68]. These performs partitioned time-integration, accelerated using a quasi-Newton method or classical over-relaxation, and have a variety of adapters for open source or commercial libraries, c.f. [13, 64, 84] resp. [68].

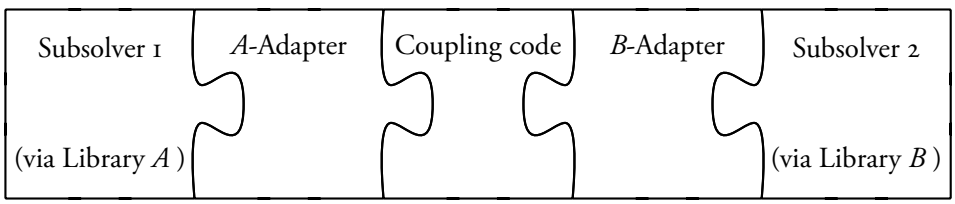

Figure 1.1: Coupling principle for the partitioned approach. Here, the coupling code executes the partitioned time-integration, handling the information exchange between the subproblems. Adapters to subsolvers implemented via a certain library are largely reusable.

Note that there exists no unified interface standard for coupling PDE solvers. The FMI-Standard [53] for the co-simulation [1] of coupled systems is widely used for coupled mechanical systems. However, co-simulation is a loose coupling scheme and not suitable for strongly coupled problems.

1.2.1 Waveform Relaxation

We consider Waveform Relaxation (WR), an iterative partitioned time-integration method, in which subproblems exchange *continuous interface functions* in time. These are obtained via suitable interpolation methods and enable the use of independent time-grids (includ-

ing adaptivity) and time-integration methods. Thus, WR methods have few requirements on the subproblems to be coupled and are suitable for the partitioned coupling of surface coupled multiphysics problems. They are less suited for multiphysics problems with a *volumetric coupling*, i.e., subproblems interacting on the whole computational domain, as e.g., in magnetohydrodynamics [80], due to the storage required in saving the continuous function in time.

WR methods were originally used for systems of ODEs in circuit simulation [40] and for the first time to solve time dependent PDEs in [23, 36]. They are also used for coupled systems of (partial) differential algebraic equations ((P)DAEs) [52, 69]. In the literature, WR is found under a variety of names, often depending on the subfield, the most common ones are: Waveform relaxation/iteration, dynamic iteration/relaxation and Picard(-Lindelöf) iteration.

WR methods require convergence acceleration to achieve fast convergence rates. The canonical way is to weight updates with relaxation parameters. Optimal relaxation parameters are *highly problem specific* and suitable relaxation may be required to obtain a convergent method in the first place [57]. Other acceleration techniques involve using an additional convolution relaxation term [34, 63] or Krylov-subspace acceleration [43], see [43] for a wider overview of different acceleration techniques. However, many of these are not applicable in the partitioned approach. Another option is to use black-box acceleration methods such as quasi-Newton methods [18], which have been successfully applied for WR methods [67].

1.2.2 Thermal Fluid Structure interaction

Conjugate heat transfer, thermal interaction between solids and liquids, plays an important role in many applications and its simulation has proved essential [5]. Examples for thermal fluid structure interaction are cooling of gas-turbine blades, thermal anti-icing systems of airplanes [12], supersonic reentry of vehicles from space [30, 47], gas quenching, which is an industrial heat treatment of metal work-pieces [28, 79] or the cooling of rocket nozzles [37, 38].

Thermal fluid structure interaction (FSI) is a strongly surface coupled problem. A characteristic feature is a significant jump in the material coefficients at the interface, which makes for a natural geometry based splitting to obtain the coupled problem, which is also known as Domain decomposition [61, 82]. The subproblems are commonly coupled via a Dirichlet-Neumann (DN) coupling at the interface [61, 82]. That is, the subproblems use Dirichlet resp. Neumann boundary conditions w.r.t. the interface and provide the other subproblem with the suitable interface temperature or heat flux.

Consequently WR is a suitable method for solving this coupled problem and has first been considered for the heat equation in [23, 36]. Recent work considers the continuous Dirichlet-Neumann Waveform relaxation (DNWR) and Neumann-Neumann Waveform relaxation (NNWR) methods [24, 39, 45] with homogeneous materials. DNWR solves the subproblems in sequence. NNWR consists of first solving two Dirichlet problems, followed by two Neumann problems in parallel, in each iteration. In [54, 57] a fully discrete NNWR method is derived along with a fully discrete optimal relaxation parameter (for 1D, implicit Euler and heterogeneous materials). While the iteration converges quickly, observed convergence rates are far from achieving theoretical peak convergence rates.

With Paper III we continue the research in [54, 57], in order to develop a robust and fast partitioned time-integration method for heterogeneous coupled heat equations. We improve upon the NNWR method [57] and derive the corresponding fully discrete DNWR method, including optimal relaxation. We show that the DNWR method is both fast and *robust*. I.e., the derived optimal relaxation parameters for 1D and implicit Euler yields near optimal convergence acceleration in 2D, the second order SDIRK2 scheme, as well as the multirate and adaptive cases. In particular, our DNWR method is shown to be notably more robust than the NNWR method. Overall, we obtain a fast, robust, multirate and time adaptive partitioned solver for unsteady conjugate heat transfer.

1.2.3 Waveform Relaxation with asynchronous time-integration

The main flavors of WR methods are Gauss-Seidel WR (GS WR), solving subproblems in sequence, and Jacobi WR, solving subproblems in parallel. GS WR convergence is typically faster than for Jacobi WR, due to increased information exchange. If the parallelism of Jacobi WR can outweigh a slower convergence rate is highly problem dependent due to relaxation.

Standard WR methods exchange continuous time-dependent functions, resp. the discrete interpolants after performing time-integration of a subproblem on the interval $[0, T_f]$. With Paper IV we develop a novel parallel WR method with a faster convergence rate than classical parallel WR methods. Our ansatz is to increase communication by already exchanging information during time-integration. That is, instead of communicating whole discrete functions, we communicate the solutions of each new timestep and directly incorporate all new information in the interpolants.

We facilitate the communication via *One-sided communication*, which is asynchronous. Unlike the more common *Point-to-Point* communication, this does not require function calls on the receiving processor and is thus most suitable for non-matching time-grids.

We prove convergence for our WR method with asynchronous time-integration, using ele-

mentary techniques. The analytical description and convergence results of this method generalize existing WR theory. We present an algorithm for optimal relaxation for two coupled problems. Asynchronous communication is not deterministic and in our algorithm we need to determine the realized communication at runtime to choose optimal relaxation.

Numerical results confirm theoretical convergence results and show good performance results for subproblems with approximately equal computational load. For our numerical experiments we coupled subsolvers implemented using `DUNE` [6] and `FEniCS` [41]. Here, we developed adapters to facilitate the coupling in a black-box manner w.r.t. space discretizations. We demonstrate this coupling via a thermal FSI problem, where the structure is solved using a finite element discretization implemented in `FEniCS` and the fluid model is solved using a discontinuous Galerkin discretization implemented in `DUNE`.

Here, we use optimal relaxation parameters based on the results of Paper III , with which the WR method converges with very few iterations. This demonstrates that the optimal relaxation parameters derived in Paper III yield good convergence acceleration for conjugate heat transfer modelled using the compressible Euler equations.

1.3 Organization of thesis

The chapters in this thesis correspond to the topics presented in this introduction. Chapter 2 discusses general adaptive time-integration using local error estimates and Chapter 3 extends this to goal oriented problems, summarizing Paper I . Chapters 4 - 6 concern WR method for the partitioned time-integration of coupled problems. Here, Chapter 4 discusses classical WR, basic convergence results, implementation and adapters for coupling the open source packages `DUNE` and `FEniCS`. Chapter 5 presents the DNWR method from Paper III and Chapter 6 presents WR with asynchronous time-integration from Paper IV .

Chapter 2

Time-adaptivity using local error estimates

Consider the following initial value problem (IVP):

$$\dot{\mathbf{u}}(t) = \mathbf{f}(t, \mathbf{u}(t)), \quad t \in [0, T_f], \quad \mathbf{u}(0) = \mathbf{u}_0, \quad \mathbf{f} : [0, T_f] \times \mathbb{R}^d \to \mathbb{R}^d, \tag{2.1}$$

with $\mathbf{f}$ continuous in the first component and *Lipschitz-continuous* in the second component, i.e., $\forall\, \mathbf{x}, \mathbf{y} \in \mathbb{R}^d \exists\, L > 0 : \|\mathbf{f}(t, \mathbf{x}) - \mathbf{f}(t, \mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|$. This guarantees existence of a unique solution via the Picard-Lindelöf theorem [27].

When performing discrete time-integration, the time-grid $0 = t_0 < \ldots < t_N = T_f$ can greatly impact the accuracy of the solution. The simplest approach is to use an equidistant time-grid with stepsizes $\Delta t_n := t_{n+1} - t_n$ constant. However, this requires stepsizes to be chosen to address the smallest timescales, which is inefficient for problems with processes on varying time-scales.

We instead use non-equidistant time-grids obtained by an *adaptive method*, which we say consists of a *time-integration scheme* for (2.1), an *error estimator* and a *timestep controller*. An adaptive method requires an initial stepsize Δt_0 and a tolerance TOL, which is used by the timestep controller, as its inputs.

We consider adaptive methods using local error estimates. In each timestep we estimate the local time-integration error and choose the following stepsize to set the local time-integration error to equal to the tolerance TOL.

This adaptive approach works well in practice, in that it does result in time-grids with small stepsizes when necessary and large ones when possible. It does not yield global error bounds and is by no means optimal in the sense of minimizing the error for a given amount of steps.

Another advantage is that it is "light-weight" in terms of implementation. It can be applied almost in a black-box manner, is not problem specific and does not requiring a lot of extra tools.

Definition 2.1. *An adaptive method for* (2.1) *is called convergent, if*

$$\|\mathbf{u}(T_f) - \mathbf{u}_N\| \to 0, \quad \textit{for} \quad TOL \to 0,$$

where $\mathbf{u}_N \approx \mathbf{u}(T_f)$.

The two standard classes of time-integration schemes are *one-step methods*, most prominently Runge-Kutta methods, but also exponential integrators [31, 81], and *linear multistep methods* (LMM), see Appendix 8.1. For one-step methods there exists a solid theory on convergence for the variable stepsize case [25, 71] and there are inherently no zero-stability problems. With LMM, one has to consider zero-stability, especially in the case of variable stepsizes. There are no proofs on stability and convergence for arbitrary variable step sizes, but they exist for smoothly varying stepsizes [27, pp.402], [77].

The here considered stepsize adaptivity is also commonly referred to as h-refinement, since stepsizes are commonly denoted by h. Another option is p-refinement, which is to instead adapt the order of the time-integration method to meet a target accuracy, or to preserve stability. With LMM it is common to use both h and p-refinement, while one-step methods typically only use h-refinement.

Overview over chapter

The core of this chapter is the derivation and convergence proof of classical local error based time-adaptivity, which is also shown in Paper I . Here, we additionally discuss the variants obtained from (not) using *local extrapolation* or *error per (unit) step* when constructing the error estimate and timestep controller. In particular, Section 2.2 discusses the aspects of *time-invariance* and *asymptotic correctness* in adaptive timestep control.

Notation in this chapter is largely identical to Paper I , differences are as follows: The default time-interval is denoted by $[0, T_f]$, rather than $[t_0, t_e]$, and the tolerance in controllers is denoted by TOL instead of τ.

2.1 Time adaptivity based on local error estimates

The following method is the standard in ODE solvers using one-step methods. It is based on local error estimates and does not yield global error bounds or near optimal grids, but works well in practice, is computationally cheap and convergent under a smoothness assumption.

The results from this section for one-step methods and the deadbeat controller (2.10) are in principal classic [71]. Here, we present a new convergence proof, following the same principle techniques as in [71], but separating requirements on the error estimate, timesteps, and Δt_0, for one-step methods. This prepares convergence proofs for general controllers and estimates, and we use it to show convergence in the QoI for the goal oriented adaptive method in Section 3.1. We first introduce the relevant notation.

Definition 2.2. *The flow [76] of an IVP* (2.1) *is the map*

$$\mathcal{M}^{t,\Delta t} : \mathbf{u}(t) \to \mathbf{u}(t+\Delta t),$$

where $t \in [0, T_f]$ *and* $t + \Delta t \leq T_f$ *for* $\Delta t > 0$.

The flow is the solution operator for $\mathbf{u}(t)$. To numerically solve an IVP means to approximate the flow by a *numerical flow map* $\mathcal{N}^{t,\Delta t} : \mathbb{R}^d \to \mathbb{R}^d$ given by a numerical scheme, commonly called a time-integration scheme. The solution after a timestep is written in the form

$$\mathbf{u}_{n+1} = \mathcal{N}^{t_n,\Delta t_n}\mathbf{u}_n.$$

The existence of a unique solution guarantees the existence of the flow map $\mathcal{M}^{t,\Delta t}$. We define the *global error* by

$$\begin{aligned}\mathbf{e}_{n+1} &:= \mathbf{u}_{n+1} - \mathbf{u}(t_{n+1}) \\ &= \underbrace{(\mathcal{N}^{t_n,\Delta t_n} - \mathcal{M}^{t_n,\Delta t_n})\mathbf{u}_n}_{\text{global error increment}} + \underbrace{\mathcal{M}^{t_n,\Delta t_n}\mathbf{u}_n - \mathcal{M}^{t_n,\Delta t_n}\mathbf{u}(t_n)}_{\text{global error propagation}}. \end{aligned} \tag{2.2}$$

The flow map and the dynamics of global error propagation are usually not known. However, the global error increment has a known structure. Given a sufficiently smooth right-hand side $\mathbf{f}$ in (2.1), the *local error* of a scheme $\mathcal{N}^{t,\Delta t}$ of order p is

$$\ell_n := \left(\mathcal{N}^{t_n,\Delta t_n} - \mathcal{M}^{t_n,\Delta t_n}\right)\mathbf{u}_n = \Delta t_n^{p+1}\phi(t_n, \mathbf{u}_n) + \mathcal{O}(\Delta t_n^{p+2})$$

with the *principal error function* ϕ, c.f. [27]. Here, $\mathcal{O}$ is defined as follows:

$$\mathbf{e}(t) = \mathcal{O}(t^q), \quad \text{iff} \quad \exists\, C > 0,\ \delta > 0 : \|\mathbf{e}(t)\| \leq Ct^q, \quad \forall\ 0 < t \leq \delta.$$

The approach for local error based time-adaptivity is to estimate the global error increment, which is equal to the local error of the time-integration scheme used to propagate the solution. Based on this estimate, we control the timesteps Δt_n, aiming to keep the global error increment at the target tolerance TOL. This approach does not consider global error propagation and consequently yields no global error bounds.

2.1.1 Error estimation

Definition 2.3. *A local error estimate $\hat{\boldsymbol{\ell}}_n \approx \boldsymbol{\ell}_n$ is called asymptotically correct [70, 74], if*

$$\lim_{\Delta t_n \to 0} \frac{\|\hat{\boldsymbol{\ell}}_n\|}{\|\boldsymbol{\ell}_n\|} = 1.$$

I.e., a local error estimate is asymptotically correct, if it correctly estimates the leading Δt term. We now present the most common techniques to estimate local errors for one-step methods.

Embedded methods

First, we consider *embedded Runge-Kutta* (ERK) schemes [27]. The following applies to the use of two schemes of different order, ERK schemes are very efficient in this, since one can obtain two different solutions by using different weights.

Assume two schemes $\mathcal{N}_+^{t,\Delta t}, \mathcal{N}_-^{t,\Delta t}$ with orders p and $\hat{p}$, $p > \hat{p}$. We define the local errors

$$\boldsymbol{\ell}_n^- := (\mathcal{N}_-^{t_n,\Delta t_n} - \mathcal{M}^{t_n,\Delta t_n})\mathbf{u}_n = \Delta t_n^{\hat{p}+1}\boldsymbol{\phi}_-(t_n, \mathbf{u}_n) + \mathcal{O}(\Delta t_n^{\hat{p}+2}), \tag{2.3}$$

$$\boldsymbol{\ell}_n^+ := (\mathcal{N}_+^{t_n,\Delta t_n} - \mathcal{M}^{t_n,\Delta t_n})\mathbf{u}_n = \Delta t_n^{p+1}\boldsymbol{\phi}_+(t_n, \mathbf{u}_n) + \mathcal{O}(\Delta t_n^{p+2}), \tag{2.4}$$

with principal error functions $\boldsymbol{\phi}_-$ and $\boldsymbol{\phi}_+$. The difference of the two local errors, resp. solutions, yields the local error estimate

$$\hat{\boldsymbol{\ell}}_n := \boldsymbol{\ell}_n^- - \boldsymbol{\ell}_n^+ = (\mathcal{N}_-^{t_n,\Delta t_n} - \mathcal{N}_+^{t_n,\Delta t_n})\mathbf{u}_n = \Delta t_n^{\hat{p}+1}\boldsymbol{\phi}_-(t_n, \mathbf{u}_n) + \mathcal{O}(\Delta t_n^{\hat{p}+2}). \tag{2.5}$$

It is asymptotically correct w.r.t. (2.3), differing only in terms of order $\mathcal{O}(\Delta t_n^{p+1})$ and higher. Similarly, $\hat{\boldsymbol{\ell}}_n$ is not asymptotically correct w.r.t. (2.4).

Since we require two schemes for the error estimate, we can choose with which to propagate the solution. The lower order scheme $\mathcal{N}_-^{t,\Delta t}$ yields an asymptotically correct error estimate, but a less accurate solution. However, using the higher order solution obtained from $\mathcal{N}_+^{t,\Delta t}$ does not yields asymptotically correct error estimate. This case is also known as *local extrapolation* [70] and is our default case.

Richardson extrapolation

Second, we consider *Richardson extrapolation*, which estimates the error by the difference of the solution of a single step of length Δt_n and two consecutive steps with stepsize $\Delta t_n/2$

each. With the time-integration scheme $\mathcal{N}^{t,\Delta t}$ the error estimate is

$$\hat{\boldsymbol{\ell}}_n = \mathcal{N}^{t_n,\Delta t_n}\mathbf{u}_n - \mathcal{N}^{t_n+\Delta t_n/2,\Delta t_n/2}\left(\mathcal{N}^{t_n,\Delta t_n/2}\mathbf{u}_n\right).$$

It is generally not asymptotically correct. This can easily be seen in the following example.

Example 2.4. *Consider the autonomous problem*

$$\dot{\mathbf{u}}(t) = \lambda\mathbf{u}(t), \quad \mathbf{u}(0) = \mathbf{u}_0, \quad t \in [0, T_f], \quad \lambda \in \mathbb{R} \setminus \{0\}.$$

The explicit Euler scheme for this problem is $\mathcal{N}^{\Delta t_n} = 1 + \lambda\Delta t_n$*. This yields*

$$\mathcal{N}^{\Delta t_n/2} \circ \mathcal{N}^{\Delta t_n/2} = \left(1 + \frac{\lambda\Delta t_n}{2}\right)^2 = 1 + \lambda\Delta t_n + \frac{(\lambda\Delta t_n)^2}{4},$$

which is not a second order accurate approximation to the flow $\mathcal{M}^{t_n,\Delta t_n} = \mathrm{e}^{\lambda\Delta t_n}$*.*

While this scheme does not yield an asymptotically correct error estimate and is computationally expensive, it is versatile in terms of application, since it does not require any other time-integration schemes.

2.1.2 Timestep controller

We now discuss the construction of a timestep controller for the local error estimate. W.l.o.g., we do this for the asymptotically incorrect error estimate obtained of an ERK method and local extrapolation.

We model the global error increment using the principal error function of the local error estimate (2.5), which is

$$\mathbf{m}_n := \Delta t_n^{\hat{p}+1}\boldsymbol{\phi}_-(t_n, \mathbf{u}_n). \tag{2.6}$$

Based on this model, the local error in the next step is

$$\begin{aligned}
\mathbf{m}_{n+1} &= \Delta t_{n+1}^{\hat{p}+1}\boldsymbol{\phi}_-(t_{n+1}, \mathbf{u}_{n+1}) = \Delta t_{n+1}^{\hat{p}+1}\boldsymbol{\phi}_-(t_n, \mathbf{u}_n) + \mathcal{O}(\Delta t_n\,\Delta t_{n+1}^{\hat{p}+1}) && (2.7)\\
&= \left(\frac{\Delta t_{n+1}}{\Delta t_n}\right)^{\hat{p}+1}\mathbf{m}_n + \mathcal{O}(\Delta t_n\,\Delta t_{n+1}^{\hat{p}+1}) && (2.8)\\
&= \left(\frac{\Delta t_{n+1}}{\Delta t_n}\right)^{\hat{p}+1}\hat{\boldsymbol{\ell}}_n + \mathcal{O}(\Delta t_n\,\Delta t_{n+1}^{\hat{p}+1}). && (2.9)
\end{aligned}$$

Here, we aim to choose Δt_{n+1} such that $\|\mathbf{m}_{n+1}\| = TOL$. This is commonly known as the *error per step* (*EPS*) approach [71]. An alternative is the *error per unit step* (*EPUS*) approach,

aiming for $\|\mathbf{m}_{n+1}\| = \Delta t_{n+1}\, TOL$. Here we consider both approaches, starting with EPS. Applying it to (2.7) - (2.9) gives

$$TOL = \left(\frac{\Delta t_{n+1}}{\Delta t_n}\right)^{\hat{p}+1} \|\hat{\boldsymbol{\ell}}_n\| + \mathcal{O}(\Delta t_n\, \Delta t_{n+1}^{\hat{p}+1}).$$

Neglecting higher order terms and using this as a rule for computing Δt_{n+1} gives the well-known *deadbeat controller*

$$\Delta t_{n+1} = \Delta t_n \left(\frac{TOL}{\|\hat{\boldsymbol{\ell}}_n\|}\right)^{1/(\hat{p}+1)}. \tag{2.10}$$

Similarly, the EPUS approach yields

$$\Delta t_{n+1} = \Delta t_n \left(\frac{TOL}{\|\hat{\boldsymbol{\ell}}_n\|/\Delta t_n}\right)^{1/\hat{p}}, \tag{2.11}$$

which is also a deadbeat controller, but with a different error estimate and exponent. These controllers requires suitable initial stepsizes Δt_0. Choosing a good initial stepsize with little computational effort is a separate topic [86], here we use simple choices for Δt_0.

In practice one typically bounds the rate by which timesteps change as follows [27, pp. 168]

$$\Delta t_{n+1} = \Delta t_n \, \min(f_{\max}, \max(f_{\min}, \text{ind})), \quad \text{ind} = \left(\frac{TOL}{\tilde{\ell}_n}\right)^{1/p^*}.$$

Here $\tilde{\ell}_n$ and p^* take a forms corresponding to (2.10) or (2.11). The aim is to provide more computational stability by preventing too large stepsize changes. In practice, this does not take effect for $TOL \to 0$.

Two additional common practices are as follows [27, pp. 168]: First, the usage of a *safety factor* $0 < \theta < 1$, replacing TOL by $\theta \cdot TOL$, a common choice is $\theta = 0.9$. Second, rejection and repetition of timesteps, if the local error estimate exceeds the target tolerance. The repeated timestep typically uses a fixed fraction of the original stepsize.

Advantages and disadvantage of using the different approaches of error estimation and controller construction are discussed further in Section 2.2.

2.1.3 Convergence in the solution

A time-integration scheme, error estimate, timestep controller and initial stepsize Δt_0 form an *adaptive method*, for which we now analyse the global error (2.2). W.l.o.g., we do this for the EPS based controller (2.10) and local extrapolation.

We first consider the case of variable, non-adaptive timesteps. Here, we have the following Lemma:

Lemma 2.5. *([25, pp. 68]) Consider Problem* (2.1) *and a scheme $\mathcal{N}^{t,\Delta t}$ of order p. Assume a grid $0 = t_0 < \cdots < t_N = T_f$ with timesteps $\Delta t_n = t_{n+1} - t_n$ given by a stepsize function $\theta : [0, T_f] \to [\theta_{min}, 1]$, $\theta_{min} > 0$ such that for a given base step size ΔT one has*

$$\Delta t_n = \theta(t_n)\Delta T.$$

Then, the global error (2.2) *fulfills $\|\mathbf{e}_n\| = \mathcal{O}(\Delta T^p)$.*

Assuming $\phi_-^{\min} := \min_{t\in[0,T_f]} \|\boldsymbol{\phi}_-(t, \mathbf{u}(t))\| > 0$, we define reference timesteps Δt_n^{ref} by a stepsize function θ^{ref} and ΔT_{ref} as follows

$$\theta^{\text{ref}}(t_n) := \left(\frac{\phi_-^{\min}}{\|\boldsymbol{\phi}_-(t_{n-1}, \mathbf{u}(t_{n-1}))\|}\right)^{1/(\hat{p}+1)}, \quad \theta^{\text{ref}}(0) = \left(\frac{TOL}{c_0^{\text{ref}}}\right)^{1/(\hat{p}+1)}, \ (c_0^{\text{ref}} > 0),$$
$$\Delta T_{\text{ref}} = \left(\frac{TOL}{\min\{c_0^{\text{ref}},\ \phi_-^{\min}\}}\right)^{1/(\hat{p}+1)}. \tag{2.12}$$

These variable, non-adaptive timesteps meet the requirements of Lemma 2.5, yielding $\|\mathbf{e}_n\| = \mathcal{O}(\Delta T_{\text{ref}}^p) = \mathcal{O}(TOL^{p/(\hat{p}+1)})$. Now, one can observe that Lemma 2.5 still holds for a grid perturbed by

$$\Delta t_n = \theta(t_n)\Delta T + \mathcal{O}(\Delta T^2), \tag{2.13}$$

see [71]. This is straight-forward to show using the scaling ansatz $\theta^*(t_n) := \theta(t_n)/c + \mathcal{O}(\Delta T)$ with $\Delta T^* := c\Delta T$. To prove convergence for the adaptive method with controller (2.10), it thus suffices to show that the adaptive timesteps are a sufficiently small perturbation of the reference steps Δt_n^{ref}.

Theorem 2.6. *Consider problem* (2.1) *and an adaptive method consisting of: A pair of schemes $\mathcal{N}_+^{t,\Delta t}$, $\mathcal{N}_-^{t,\Delta t}$ with orders p, $\hat{p}$, with $p > \hat{p}$, error estimator* (2.3)*, deadbeat controller* (2.10)*, and $\Delta t_0 = (TOL/c_0)^{1/(\hat{p}+1)} + \mathcal{O}(TOL^{2/(\hat{p}+1)})$, $c_0, \Delta t_0 > 0$. If the principal error function $\boldsymbol{\phi}_-$ to $\mathcal{N}_-^{t,\Delta t}$ fulfills*

$$\min_{t\in[0,T_f]} ||\boldsymbol{\phi}_-(t, \mathbf{u}(t))|| > 0, \tag{2.14}$$

then the adaptive method is convergent with $\|\mathbf{e}_n\| = \mathcal{O}(TOL^{p/(\hat{p}+1)})$.

Proof. See Theorem 1 in Paper I . □

The assumption (2.14) in Theorem 2.6 is a requirement on controllability in the asymptotic regime. The global error is not controllable by the local error if its estimate vanishes at some time-point t. In practice, this can be prevented by manually enforcing a maximal stepsize.

Theorem 2.6 provides a structure with which one can prove similar results for different controllers, e.g., PID controllers [75]. To prove convergence, it is sufficient to show $\Delta t_n^{\text{new cont}} = \Delta t_n^{\text{deadbeat}} + \mathcal{O}((\Delta t_n^{\text{deadbeat}})^2)$.

Remark 2.7. *In Theorem 2.6 we considered EPS + local extrapolation. Using the EPUS based controller* (2.11) *and possibly no local extrapolation, analogously yields the following convergence rates:*

- *EPS:* $\|\mathbf{e}_n\| = \mathcal{O}(TOL^{\hat{p}/(\hat{p}+1)})$,
- *EPUS:* $\|\mathbf{e}_n\| = \mathcal{O}(TOL^{\hat{p}/\hat{p}})$,
- *EPUS + Local extrapolation:* $\|\mathbf{e}_n\| = \mathcal{O}(TOL^{p/\hat{p}})$.

One typically uses schemes with orders $p = \hat{p} + 1$. With an asymptotically correct error estimates, this keeps the extra cost for error estimation minimal. In the local extrapolation setting, this keeps the degree of asymptotic incorrectness minimal. The case of $p = \hat{p} + 1$ yields $\|\mathbf{e}_n\| = \mathcal{O}(TOL)$ for EPS + Local extrapolation and EPUS, which is also called *tolerance proportionality* [72].

2.2 On time invariance and asymptotic correctness

We introduced the property of asymptotic correctness and presented both the EPS and EPUS approach for constructing timestep controllers. The EPUS approach is not invariant to a re-scaling of time. In this section we analyze this in more depth and quantify the effect on timesteps and global error.

2.2.1 Time invariance

Consider problem (2.1) and re-scale time to $s = \alpha t$, $\alpha > 0$. The resulting problem is

$$\dot{\mathbf{y}}(s) = \mathbf{f}^*(s, \mathbf{y}(s)), \quad \mathbf{y}(0) = \mathbf{u}_0, \quad s \in [0, S_f], \tag{2.15}$$

with $\mathbf{y}(s) = \mathbf{u}(s/\alpha)$, $S_f = \alpha\, T_f$ and

$$\mathbf{f}^*(s, \mathbf{y}(s)) = \frac{1}{\alpha}\mathbf{f}\,(s/\alpha, \mathbf{u}(s/\alpha)). \tag{2.16}$$

We denote quantities related to the scaled problem with a superscript * and its timesteps by Δs_n.

Definition 2.8. *Consider the IVP* (2.1) *and time-scaling* (2.15) - (2.16), $\alpha > 0$. *An adaptive method is called time-invariant, if* $\Delta s_n = \alpha \Delta t_n$ *for all timesteps.*

Considering this property is not new, see e.g. [71], which states that an EPS approach results in a time-invariant adaptive method, while EPUS does not. The latter part is intuitive, since the size of a unit step changes under scaling. Here, we verify this and analyse how it affects timesteps and solution in the EPUS case.

Time-invariance of Runge-Kutta schemes

First, we verify that the given time-integration scheme is time-invariant. That is, using appropriately scaled stepsizes, the original and scaled IVPs yield the same solutions. Here, we consider Runge-Kutta schemes

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \Delta t_n \sum_{i=1}^{s} b_i \mathbf{k}_i, \quad \text{where} \quad \mathbf{k}_i = \mathbf{f}\left(t_n + c_i \Delta t_n, \mathbf{u}_n + \Delta t_n \sum_{j=1}^{s} a_{i,j} \mathbf{k}_j \right).$$

The scaled problem yields

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta s_n \sum_{i=1}^{s} b_i \mathbf{k}_i^* = \mathbf{y}_n + \Delta t_n \sum_{i=1}^{s} b_i \alpha \mathbf{k}_i^*$$

and

$$\begin{aligned} \mathbf{k}_i^* &= \mathbf{f}^* \left(s_n + c_i \Delta s_n, \mathbf{y}_n + \Delta s_n \sum_{j=1}^{s} a_{i,j} \mathbf{k}_j^* \right) \\ &\overset{(2.16)}{=} \frac{1}{\alpha} \mathbf{f} \left(t_n + c_i \Delta t_n, \mathbf{y}_n + \Delta t_n \sum_{j=1}^{s} a_{i,j} (\alpha \mathbf{k}_j^*) \right). \end{aligned}$$

From $\alpha \mathbf{k}_i^* = \mathbf{k}_i$, we see that the solutions are identical given $\mathbf{u}_0 = \mathbf{y}_0$. In practice, results can vary slightly due to inaccuracies in solving nonlinear and linear systems.

Time-invariance of controllers

Next, we analyze the timestep controllers (2.10) and (2.11) on time-invariance. We do this in the asymptotic regime $TOL \to 0$, by comparing the reference timesteps (2.12). In the

EPS case, this yields

$$\Delta s_n^{\text{ref}} = \left(\frac{TOL}{\|\phi^*(s_n, \mathbf{y}(s_n))\|} \right)^{1/(\hat{p}+1)}, \tag{2.17}$$

where ϕ^* is the principal error function corresponding to local errors of the scaled IVP. Given sufficient smoothness, the principal error functions for a time-integration scheme of order p are

$$\phi(t_n, \mathbf{u}(t_n)) = c\,\mathbf{f}^{(p)}(t_n, \mathbf{u}(t_n)) \quad \text{and} \quad \phi^*(s_n, \mathbf{y}(s_n)) = c^*\,\mathbf{f}^{*(p)}(s_n, \mathbf{y}(s_n)),$$

cf. [27, pp. 156], where c, c^* are constants. For ERK methods with appropriately scaled timesteps the solutions and thus local errors are identical. This yields the following relation

$$c\,\mathbf{f}^{(p)}(t_n, \mathbf{u}(t_n)) = c^*\,\mathbf{f}^{*(p)}(s_n, \mathbf{y}(s_n)) \overset{(2.16)}{=} c^*\,\alpha^{-1}\mathbf{f}^{(p)}(s_n/\alpha, \mathbf{u}(s_n/\alpha))$$

Basic differentiation rules then yield $c = \alpha^{p+1}c^*$ and thus

$$\phi^*(s_n, \mathbf{y}(s_n)) = \alpha^{-(\hat{p}+1)}\phi(t_n, \mathbf{u}(t_n)). \tag{2.18}$$

Inserting this into the stepsize formula (2.17) gives

$$\Delta s_n^{\text{ref}} = \alpha \Delta t_n^{\text{ref}},$$

thus, the deadbeat controller (2.10) from the EPS approach is time-invariant. Similarly, the EPUS based approach yields the following reference timesteps

$$\Delta t_n^{\text{ref}} = \left(\frac{TOL}{\|\phi(t_n, \mathbf{u}(t_n))\|} \right)^{1/\hat{p}}, \quad \text{resp.} \quad \Delta s_n^{\text{ref}} = \left(\frac{TOL}{\|\phi^*(s_n, \mathbf{y}(s_n))\|} \right)^{1/\hat{p}},$$

and (2.18) gives

$$\Delta s_n^{\text{ref}} = \alpha^{(\hat{p}+1)/\hat{p}} \Delta t_n^{\text{ref}}. \tag{2.19}$$

This shows that the EPUS based controller is not time-invariant and how scaling affects the stepsizes in the asymptotic regime.

Number of steps under scaling

Since the EPS approach yields a time-invariant adaptive method, the remainder focuses on the EPUS approach. In particular, we analyse how time-scaling affects the total number of timesteps N resp. N^* and the global errors.

We define the average stepsizes

$$\bar{\Delta} t := T_f/N \quad \text{and} \quad \bar{\Delta} s := S_f/N^*,$$

and assume they fulfill the relation (2.19). This yields

$$\alpha^{1/\hat{p}} N^* \approx N, \tag{2.20}$$

i.e., time-scaling has an effect on the number of timesteps an adaptive method takes to solve a given problem. This effect is less pronounced for higher order methods.

Global error under scaling

With the global error, we need to look at the form of $\mathbf{e}_n$, before it is reduced to $\|\mathbf{e}_n\| = \mathcal{O}(\Delta T^p)$. In [25, pp. 60] it is given due to the bound

$$\|\mathbf{e}_n\| \leq D \Delta T^p \frac{\mathrm{e}^{L T_f} - 1}{L}.$$

Here, L is the Lipschitz-constant w.r.t. the second argument of $\mathbf{f}$ in (2.1), i.e.,

$$\|\mathbf{f}(t, \mathbf{v}) - \mathbf{f}(t, \mathbf{w})\| \leq L \|\mathbf{v} - \mathbf{w}\|, \quad \forall\ \mathbf{v}, \mathbf{w} \in \mathbb{R}^d, \quad \mathbf{v} \neq \mathbf{w}.$$

The constant D is the max-norm of the principal error function of the global error increment

$$D := \max_{t \in [0, T_f]} \|\boldsymbol{\phi}(t, \mathbf{u}(t))\|.$$

The constants L and D are affected by scaling as follows:

$$D^* \overset{(2.18)}{=} \alpha^{-(p+1)} D, \quad L^* \overset{(2.16)}{=} L/\alpha.$$

This yields the following global error bound

$$\|\mathbf{e}_n^*\| \leq D^* \Delta S^p \frac{\mathrm{e}^{L^* S_f} - 1}{L^*} = \alpha^{p/\hat{p}} D\ \Delta T^p \frac{\mathrm{e}^{L T_f} - 1}{L}.$$

Assuming the global errors have similar distances to their respective bounds, we get

$$\|\mathbf{e}_n^*\| \approx \alpha^{p/\hat{p}} \|\mathbf{e}_n\|. \tag{2.21}$$

This suggests that scaling affects the global error with at least the order of the scaling, as $p/\hat{p} > 1$ due to $p > \hat{p}$.

Numerical test

We test the relations (2.20) and (2.21) using the following linear test problem

$$\dot{\mathbf{u}}(t) = \mathbf{A}\,\mathbf{u}(t), \quad \mathbf{A} = \begin{pmatrix} -1 & 1 \\ 0 & -5 \end{pmatrix}, \quad t \in [0, 2], \quad \mathbf{u}(0) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{2.22}$$

We solve both the original and scaled problems, and use (2.20) and (2.21) to predict N^* and $\|\mathbf{e}_n^*\|$ under scaling with $\alpha \in \{0.01, 100\}$.

We use the classical Runge-Kutta scheme of order $p = 4$ with an embedded third order (for autonomous problems) solution given by $\hat{\mathbf{b}} = \frac{1}{3}(1,\ 1,\ 0,\ 1)^T$, local extrapolation, $\Delta t_0 = TOL^{1/(\hat{p}+1)}\,T_f$ and the controller (2.11).

Figure 2.1 shows the predictions (2.20) and (2.21) are accurate for $TOL \to 0$. Based on these results we make the following Conjecture, which seems to hold based on numerical experiments, but we have not been able to prove it yet.

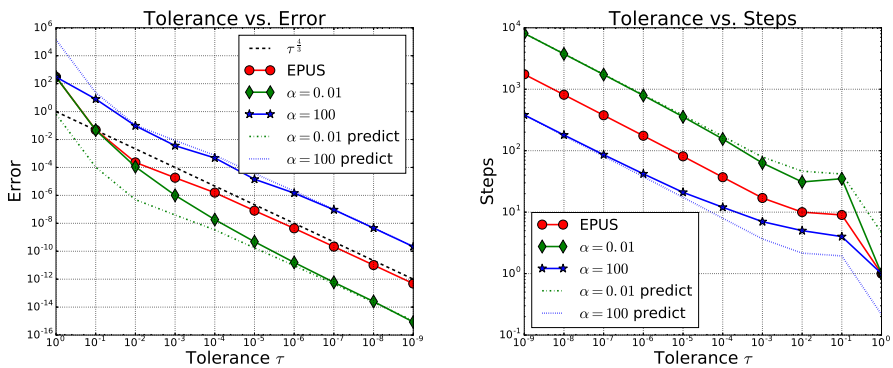

Figure 2.1: Tolerance vs. Error (left) and Tolerance vs. Steps (right) when solving problem (2.22) for various time-scalings.

Conjecture 2.9. *Given a linear IVP* $\dot{\mathbf{u}}(t) = \mathbf{A}\mathbf{u}(t)$, $t \in [0, T_f]$ *and* $\mathbf{u}(0) = \mathbf{u}_0$, *the relations* (2.20) *and* (2.21) *hold with equality for* $TOL \to 0$.

A more interesting quantity is the grid quality, measured in error vs. steps. Assume $\|\mathbf{e}_n\| = c_1 \Delta T^p = c_2 N^{-p}$, with $c_1,\ c_2 > 0$, by Conjecture 2.9 the error from the scaled problem fulfills

$$\|\mathbf{e}_n^*\| \overset{(2.21)}{=} \|\mathbf{e}_n\| \alpha^{p/\hat{p}} = c_1 \Delta T^p \alpha^{p/\hat{p}} = c_2 N^{-p} \alpha^{p/\hat{p}} \overset{(2.20)}{=} c_2 N^{*-p}, \quad TOL \to 0.$$

As a result, we expect the grid quality in terms of error vs. steps to follow the same parameterized curve. This result is seen in Figure 2.2, suggesting that the efficiency of an adaptive

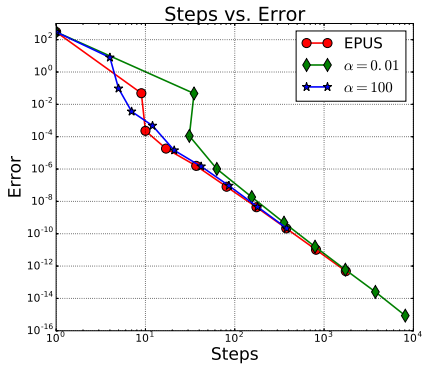

Figure 2.2: Steps vs. Error when solving problem (2.22) for various time-scalings.

method in terms of error over steps is not affected by time-scaling.

However, it is not entirely clear how grid quality relates to computational time, which for most practical purposes is more interesting. With implicit schemes ones needs to solve (non)linear problems during time-integration. While the conditioning of the (non)linear problems with smaller stepsizes is better, it is hard to make a statement about the total computational cost of time-integration under scaling. This would be subject to further research.

2.2.2 Asymptotic correctness

In Section 2.1.1 we introduced *asymptotic correctness* as a property of error estimates, which we now discuss in more detail. When using local extrapolation, the global error increment (2.2) is the local error of the higher order scheme (order p), i.e.,

$$\ell_n := \Delta t_n^{p+1} \phi_+(t_n, \mathbf{u}_n) + \mathcal{O}(\Delta t_n^{p+2}),$$

where ϕ_+ is the principal error function. We chose the error model (2.6) and thus effectively make the following approximation

$$\phi_+ \approx \phi_-, \tag{2.23}$$

neglecting higher order terms. To analyse the quality of this approximation, we consider the linear, autonomous case

$$\dot{\mathbf{u}}(t) = \mathbf{A}\,\mathbf{u}(t).$$

The principal error functions are based on higher order derivatives of the right-hand side and method dependent constants, which we denote by c and c_-. As examples, the explicit

Euler scheme has $c = 1/2$ and Crank-Nicolson scheme has $c = -1/12$. In the linear case the principal error functions are

$$\phi(t_n, \mathbf{u}_n) = c\,\mathbf{A}^{p+1}\mathbf{u}_n \quad \text{and} \quad \phi_-(t_n, \mathbf{u}_n) = c_-\,\mathbf{A}^{\hat{p}+1}\mathbf{u}_n.$$

Given c and c_- of similar magnitude, the approximation (2.23) is particularly bad if $\mathbf{A}$ has coefficients small or large in magnitude. In general terms we have one of the three following cases:

$$\|c\,\mathbf{A}^{p+1}\mathbf{u}_n\| = \|c_-\,\mathbf{A}^{\hat{p}+1}\mathbf{u}_n\|, \text{ the error estimate is asymptotically correct.}$$

$$\|c\,\mathbf{A}^{p+1}\mathbf{u}_n\| < \|c_-\,\mathbf{A}^{\hat{p}+1}\mathbf{u}_n\|, \text{ the error is overestimated.} \tag{2.24}$$

$$\|c\,\mathbf{A}^{p+1}\mathbf{u}_n\| > \|c_-\,\mathbf{A}^{\hat{p}+1}\mathbf{u}_n\|, \text{ the error is underestimated.} \tag{2.25}$$

Neglecting the unlikely first case, overestimation of the global error increment leads to a timestep smaller than necessary, resulting in a higher precision at increased computational cost. Underestimation results in larger timesteps and global error increments exceeding the target tolerance.

In problems with strong global error dampening, the consequences of underestimating the local error are not severe, as the larger global error increments are dampened. In general, error underestimation can be problematic, as it may leave dynamics of the system under-resolved.

The criteria (2.24) and (2.25) are already in the linear case not usable in practice. Considering not overly stiff nonlinear problems we can make the rule of thumb that for fast processes, the error will be underestimated and for slow processes, the error will be overestimated. Here, slow (fast) processes are characterized by small (large) derivatives due small (large) values in $\mathbf{A}$.

Yet, the question is: Is asymptotic correctness important or not? Should one use local extrapolation? Taking aside asymptotic considerations, the purpose of time-adaptivity is to choose small timesteps when necessary and large timesteps when possible. Except for very stiff problems, this is achieved when using local extrapolation.

Furthermore, one should note that the grid in the limit of $TOL \to 0$, obtained from an asymptotically correct error estimate, is by no means optimal. It is uncertain if a grid from local extrapolation is better or worse. However, the order gain from propagating the numerical solution with the higher order method is certain and likely to outweigh the benefit of more accurate error estimates. As a consequence, we focus on the case of using local extrapolation for EPS based controller.

Chapter 3

Goal-oriented time-adaptivity using local error estimates

We now consider the IVP (2.1) in combination with the QoI

$$\mathcal{J}(\mathbf{u}) := \int_0^{T_f} j(t, \mathbf{u}(t)) \mathrm{d}t, \quad j : [0, T_f] \times \mathbb{R}^d \to \mathbb{R}. \tag{3.1}$$

Here, j is also called the *density function* and may correspond to integrals in space, if the original IVP is a semi-discrete PDE. The examples from Section 1.1 qualify if one considers unsteady flows.

We consider embedded Runge-Kutta schemes for time-integration and approximate $\mathcal{J}$ by $\mathcal{J}_h$ using quadrature:

$$\mathcal{J}_h(\mathbf{u}_h) := \sum_{n=0}^{N-1} \Delta t_n \sum_{k=0}^{s} \sigma_k j\left(t_n^{(k)}, \mathbf{u}_n^{(k)}\right) \approx \int_0^{T_f} j(t, \mathbf{u}(t)) \, \mathrm{d}t = \mathcal{J}(\mathbf{u}).$$

Here, $\mathbf{u}_h \approx \mathbf{u}$ denotes the discrete solution. The quadrature nodes and weights are $t_n^{(k)}$ and σ_k, with $\mathbf{u}_n^{(k)} \approx \mathbf{u}(t_n^{(k)})$.

We derive a goal oriented adaptive method, using local error estimates in the QoI. The goal is to be more efficient than both the dual-weighted residual (DWR) method and the non goal oriented adaptive method using local error estimates, by having a cheap error estimate and focusing on direct error contributions to the QoI. The only additional computational costs, w.r.t. the classical adaptive approach from Chapter 2, are evaluations of j when computing the error estimate. This makes it notably less intrusive to implement compared to the DWR method and suitable for partitioned approaches to coupled systems.

Definition 3.1. *An adaptive method for* (2.1), (3.1) *is called convergent in the QoI, if*

$$|\mathcal{J}(\mathbf{u}) - \mathcal{J}_h(\mathbf{u}_h)| \to 0, \quad \textit{for} \quad TOL \to 0.$$

For our new adaptive method, we show convergence in the QoI under a controllability assumption with additional requirements on the timesteps.

Overview over chapter

This chapter corresponds to the material presented in Paper I . Here, we sketch the main idea and present the key results. Paper I additionally includes discussion on similar approaches used in the literature. In particular, with our new results we are able to explain their performance results and issues.

See Paper I or [50] for a brief summary of the DWR method in the context of time-dependent problems, or see [4, 7, 60] for a comprehensive overview.

3.1 Goal oriented time adaptivity based using local error estimates

We now consider the goal oriented setting (3.1) for problem (2.1) and are only interested in the QoI $\mathcal{J}(\mathbf{u})$. First, we establish the connection between convergence rates in the solution and the QoI.

Theorem 3.2. *Consider problem* (2.1), (3.1) *with j sufficiently smooth, a grid* $0 = t_0 < \ldots < t_N = T_f$ *with timesteps* $\Delta t_n = \mathcal{O}(TOL^{1/q})$*, and an approximation* $\mathcal{J}_h \approx \mathcal{J}$ *by a quadrature scheme of order r. For an approximation* $\mathbf{u}_h \approx \mathbf{u}$ *with*

$$\mathbf{u}_h(t) - \mathbf{u}(t) = \mathcal{O}(TOL^{p/q}), \quad \forall t \in [0, T_f],$$

the error in the QoI fulfills $e^{\mathcal{J}} := |\mathcal{J}(\mathbf{u}) - \mathcal{J}_h(\mathbf{u}_h)| = \mathcal{O}(TOL^{\min(r,p)/q})$*,* $TOL \to 0$*.*

Proof. The idea is to split the error in the QoI as follows

$$e^{\mathcal{J}} \leq \underbrace{|\mathcal{J}(\mathbf{u}) - \mathcal{J}_h(\mathbf{u})|}_{\text{quadrature error}} + \underbrace{|\mathcal{J}_h(\mathbf{u}) - \mathcal{J}_h(\mathbf{u}_h)|}_{\text{time-integration error}}$$

and treat both errors separately. See Theorem 2 in Paper I for the full proof. □

3.1.1 Error estimate and timestep controller

Based on the idea of splitting the error into time-integration and quadrature error contributions, we construct local error estimates for both step-wise contributions to the error in the QoI. Following the same principles as in Chapter 2, using local extrapolation, this yields the following local error estimates:

$$\begin{aligned} \text{Time-integration:} \quad & \hat{\ell}_n^{\mathcal{J}_h} := j\left(t_{n+1}, \mathcal{N}_-^{t_n,\Delta t_n}\mathbf{u}_n\right) - j\left(t_{n+1}, \mathcal{N}_+^{t_n,\Delta t_n}\mathbf{u}_n\right), \\ \text{Quadrature:} \quad & \hat{\ell}_n^{\mathcal{J}_Q} := \left|\hat{\mathcal{J}}_{h,n}^{r}(\mathbf{u}_h) - \hat{\mathcal{J}}_{h,n}^{\hat{r}}(\mathbf{u}_h)\right|. \end{aligned} \tag{3.2}$$

Here,

$$\hat{\mathcal{J}}_{h,\,n}^{\,r}(\mathbf{u}) := \Delta t_n \sum_{k=0}^{s} \sigma_k j\left(t_n^{(k)}, \mathbf{u}(t_n^{(k)})\right),$$

is an r-th order quadrature scheme applied to the solution on the time-interval $[t_n, t_{n+1}]$. With $\hat{r} < r$, this follows the local extrapolation principle for constructing the error estimate using an additional lower order scheme.

The resulting deadbeat-type timestep controller is

$$\Delta t_{n+1} = \Delta t_n \left(\frac{TOL}{\left|\hat{\ell}_n^{\mathcal{J}_h}\right| + \left|\hat{\ell}_n^{\mathcal{J}_Q}\right|}\right)^{1/(\hat{p}+1)}. \tag{3.3}$$

Here, one can choose to use only one of the two error estimates.

See Paper I for the detailed derivation of the error estimates and controller. Furthermore, see [50] for details on the non local extrapolation and error per unit step variants.

3.1.2 Convergence in the quantity of interest

We show convergence in the QoI for the adaptive timesteps (3.3) following the structure from Section 2.1.3. Consider

$$\psi(t) = \left|\frac{\partial j(t, \mathbf{u}(t))}{\partial \mathbf{u}}\, \boldsymbol{\phi}_-(t, \mathbf{u}(t))\right| + \left|\varphi_-(t, \mathbf{u}(t))\right|, \tag{3.4}$$

where $\boldsymbol{\phi}_-$ is the principle error function of the lower order time-integration scheme, and φ_- is the analogous principle error function of the lower order quadrature scheme. We assume

$$\psi_{\min} := \min_{t \in [0, T_f]} \psi(t) > 0, \tag{3.5}$$

i.e., the combined error estimates do not vanish. We define non-adaptive reference timesteps $\Delta t_n^{\text{ref}} = \theta^{\text{ref}}(t_n)\Delta T_{\text{ref}}$ as follows:

$$\theta^{\text{ref}}(t_n) := \left(\frac{\psi_{\min}}{\psi(t_{n-1})}\right)^{1/(\hat{p}+1)}, \quad \theta^{\text{ref}}(0) := \left(\frac{TOL}{c_0^{\text{ref}}}\right)^{1/(\hat{p}+1)}, \ (c_0^{\text{ref}} > 0),$$
$$\Delta T_{\text{ref}} := \left(\frac{TOL}{\min\{c_0,\ \psi_{\min}\}}\right)^{1/(\hat{p}+1)}.$$

We now show that our adaptive timesteps (3.3) are a sufficiently small perturbation (2.13) of the above reference steps. This yields convergence in the solution by Lemma 2.5 and then convergence in the QoI by Theorem 3.2.

Theorem 3.3. *Consider problem* (2.1), (3.1) *and assume j to be sufficiently smooth. Assume an adaptive method consisting of: Time-integration schemes $\mathcal{N}_+^{t,\Delta t}$, $\mathcal{N}_-^{t,\Delta t}$ with orders p, $\hat{p}$, $p > \hat{p}$, quadrature schemes of order r, $\hat{r} = \hat{p}$, and $r > \hat{r}$, a scheme $\hat{\mathcal{N}}^{t,\Delta t}$ to obtain solutions of order $p-1$ for intermediate points, the error estimates* (3.2), *controller* (3.3), *and $\Delta t_0 = (TOL/c_0)^{1/(\hat{p}+1)} + \mathcal{O}(TOL^{2/(\hat{p}+1)})$, $c_0, \Delta t_0 > 0$. If the principle error functions corresponding to the lower order schemes fulfill* (3.5), *then $e^{\mathcal{J}} = |\mathcal{J}(\mathbf{u}) - \mathcal{J}_h(\mathbf{u}_h)| = \mathcal{O}(TOL^{\min(r,\,p)/(\hat{p}+1)})$.*

Proof. See Theorem 3 in Paper I . □

The assumption (3.4) is a controllability requirement in the asymptotic regime, ensuring non-zero error estimates. Using both the quadrature and time-integration error estimate makes the adaptive method more robust, as $\psi_{\min} = 0$ requires a common zero in both principle error functions.

In Paper I , Section 4.3, we analyze the global error dynamics w.r.t. the error in the QoI and its zero-set to make a qualitative statement about the obtained grid. With this, we establish guidelines to predict grid quality and thus performance of the goal-oriented adaptive method.

Consider a QoI with a nontrivial zero-set. Global errors can accumulate in the zero-set of the QoI and then be propagated to the image of the QoI, since local error based adaptive methods do not control global error propagation. The resulting error increases are not controllable using local error based methods. Advection dominated PDEs with spatially local QoIs are particularly prone to this type of behavior. See Paper I , Section 4.3 for the details.

3.2 Numerical results and conclusions

See Section 5 in Paper I for the numerical experiments and results. Here, we provide a short summary of the results.

Similar to the classical local error based adaptive method from Chapter 2, we derived a simple and easy to implement goal oriented local error estimator. We showed convergence and determined convergence rates of the error in the QoI. The expected convergence orders from Theorems 2.6, 3.2 and 3.3 are achieved in experiments. Using both the quadrature and time error estimate in the goal oriented controller (3.3) yields a more robust method than using only one error component.

We test the analysis on global error dynamics w.r.t. the QoI in a scalar test problem with a multitude of different QoIs, and a convection dominated Convection-Diffusion problem, with a spatially local QoI and source term. The DWR method produces high quality grids (small error using few timesteps), but is computationally very costly and thus less efficient than local error based methods. It is easy to predict and explain when the goal oriented method performs poorly, but it is harder to predict performance gains w.r.t. the classical time-adaptive adaptive from Chapter 2.

The goal oriented method is easily applicable to coupled problems, as demonstrated using a system of two coupled heterogeneous heat equations, c.f. Chapter 5. There, the QoI is the heat flux between the two subproblems. The goal oriented method shows performance improvements when compared with classical local error based adaptivity using norms, see Figure 3.1.

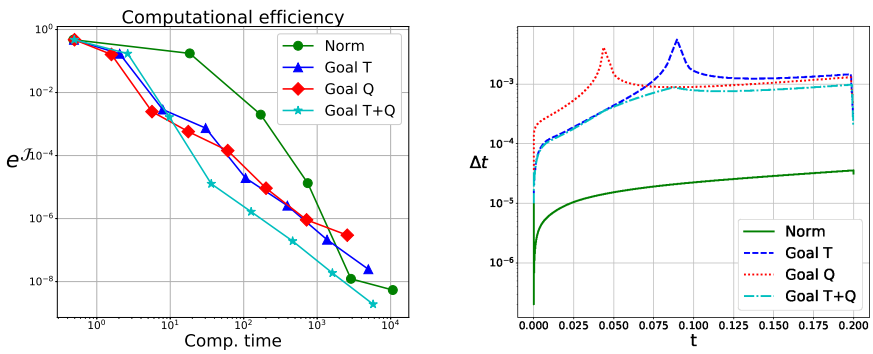

Figure 3.1: Left: Computational efficiency comparison of goal oriented adaptive method with classical norm based local error control. "Goal T" and "Goal Q" refer to the controller (3.3), using only the time -or quadrature error estimate. Using both estimates yields a better results. Right: Stepsizes over time for $TOL = 10^{-5}$. The spike shaped increases in stepsize when using only the time -or quadrature estimate, indicate possible zeroes in the corresponding principle error functions, c.f. (3.4).

In this test case, we do not consider the DWR method, since it is not suited for partitioned approaches for solving coupled problems. There, one would first need to derive the monolithic adjoint problem. Then, one needs to a find suitable coupling conditions (e.g., a Dirichlet-Neumann coupling, c.f. Chapter 5) to formulate the adjoint problem as a coupled problem. Finally, one requires a suitable partitioned time-integration method and subsolvers to solve the coupled adjoint problem.

Chapter 4

Waveform Relaxation

Relaxation refers to iterative methods, which commonly use weighting of updates by so called *relaxation parameters* to accelerate convergence. The name *"Waveform relaxation"* (WR) originates from the initial application in the context of electronic circuit simulation [40]: "This method essentially uses an iterative relaxation scheme [...] in which the elements of the relaxation are waveforms of unknown variables." and : "...the waveform of a signal is the shape of its graph as a function of time, ..."[1]. In a more general sense, WR is relaxation of time-dependent functions.

Overview over chapter

This chapter serves as an introduction to WR methods for the following chapters. We first introduce the time-continuous and time-discrete WR methods in the general nonlinear setting. Then, convergence is analyzed in the linear setting in Section 4.3. Afterwards we discuss relaxation and the successive application of WR on time-windows in Sections 4.4 resp. 4.5. These theoretical results are not novel and largely based on [32, 33, 34, 58, 59, 85, 87].

Section 4.6 discusses our framework for technical implementation, in particular w.r.t. coupling subsolvers implemented using `DUNE` [10] and `FEniCS` [41]. The numerical tests utilizing this coupling are presented in Paper IV .

[1]Source: `https://en.wikipedia.org/wiki/Waveform`, Accessed: 7. December 2020

4.1 Continuous Waveform relaxation method

Consider problem (1.1) and let $\mathbf{v}^{(k)}$ and $\mathbf{w}^{(k)}$ be given. A single iteration of a general continuous Waveform Relaxation method consists of solving two differential equations and performing two relaxation steps as follows:

$$\dot{\hat{\mathbf{v}}}^{(k+1)}(t) = \mathbf{g}\left(t, \hat{\mathbf{v}}^{(k+1)}(t), \mathbf{w}_*^{(k)}(t)\right), \quad \hat{\mathbf{v}}^{(k+1)}(0) = \mathbf{v}_0, \quad t \in [0, T_f], \tag{4.1a}$$

$$\mathbf{v}^{(k+1)} = \mathbf{v}^{(k)} + \mathbf{\Theta}_v(\hat{\mathbf{v}}^{(k+1)} - \mathbf{v}^{(k)}), \tag{4.1b}$$

$$\dot{\hat{\mathbf{w}}}^{(k+1)}(t) = \mathbf{h}\left(t, \mathbf{v}_*^{(k)}(t), \hat{\mathbf{w}}^{(k+1)}(t)\right), \quad \hat{\mathbf{w}}^{(k+1)}(0) = \mathbf{w}_0, \quad t \in [0, T_f], \tag{4.1c}$$

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \mathbf{\Theta}_w(\hat{\mathbf{w}}^{(k+1)} - \mathbf{w}^{(k)}), \tag{4.1d}$$

with nonsingular diagonal matrices $\mathbf{\Theta}_v \in \mathbb{R}^{d_v \times d_v}$, $\mathbf{\Theta}_w \in \mathbb{R}^{d_w \times d_w}$ for relaxation. Extensions to more than two systems are straight-forward, c.f. [85]. Here, one iteration consists of solving two differential equations over the time-interval $[0, T_f]$ and two relaxation steps. The specific WR method is defined by the choices for $\mathbf{v}_*^{(k)}$ and $\mathbf{w}_*^{(k)}$.

The most common WR methods are *Gauss-Seidel* (GS) and *Jacobi* WR, c.f., [85, 87]. Continuous GS WR is given by

$$\mathbf{v}_*^{(k)} = \mathbf{v}^{(k+1)} \quad \text{and} \quad \mathbf{w}_*^{(k)} = \mathbf{w}^{(k)}. \tag{4.2}$$

GS WR is sequential, solving (4.1a) - (4.1d) in order, which makes it sensitive to the order of the systems in (1.1).

The Gauss-Seidel iteration with relaxation is also commonly referred to "Successive Over Relaxation" (SOR) [34, 90]. We do not differentiate between SOR WR and GS WR, since relaxation is essential in obtaining good convergence rates. Furthermore, no relaxation, i.e., $\mathbf{v}^{(k+1)} = \hat{\mathbf{v}}^{(k+1)}$ corresponds to trivial relaxation with $\mathbf{\Theta}_v = \mathbf{I}$.

Jacobi WR is given by

$$\mathbf{v}_*^{(k)} = \mathbf{v}^{(k)} \quad \text{and} \quad \mathbf{w}_*^{(k)} = \mathbf{w}^{(k)}, \tag{4.3}$$

allowing parallel computation of (4.1a), (4.1b) and (4.1c), (4.1d).

With $\mathbf{u} = (\mathbf{v}^T, \mathbf{w}^T)^T \in \mathbb{R}^{d_v + d_w}$, the iteration is typically terminated via one of the following two criteria:

$$\|\mathbf{u}^{(k+1)} - \mathbf{u}^{(k)}\| < TOL_{WR}, \quad \text{or} \quad \|\mathbf{u}^{(k+1)}(T_f) - \mathbf{u}^{(k)}(T_f)\| < TOL_{WR}. \tag{4.4}$$

I.e., if the update is smaller than a given tolerance. The update can be measured in a function (semi)norm or also commonly in a vector (semi)norm at $t = T_f$, where updates

tend to be the largest. One commonly uses a seminorm, measuring the update only in the subset of unknowns of $\mathbf{v}$ resp. $\mathbf{w}$ that $\mathbf{h}$ resp. $\mathbf{g}$ in (1.1) depend on.

Algorithms 4.1 and 4.2 show pseudocodes of the continuous GS and Jacobi WR methods, the latter one in a parallel version. The trivial initial guesses are $\mathbf{v}^{(0)} \equiv \mathbf{v}_0$ resp. $\mathbf{w}^{(0)} \equiv \mathbf{w}_0$.

Pseudocode: Continuous GS WR

1: $\mathbf{w}^{(0)} \leftarrow$ initial guess
2: **for** $k = 0, \ldots, k_{\max} - 1$ **do**
3: $\quad \hat{\mathbf{v}}^{(k+1)} \leftarrow$ Solve (4.1a)
4: $\quad \mathbf{v}^{(k+1)} \leftarrow$ Relaxation (4.1b)
5: $\quad \hat{\mathbf{w}}^{(k+1)} \leftarrow$ Solve (4.1c)
6: $\quad \mathbf{w}^{(k+1)} \leftarrow$ Relaxation (4.1d)
7: $\quad$ Check (4.4), `break` if `true`
8: **end for**

Algorithm 4.1: Pseudocode of the continuous GS WR algorithm. Here, $\mathbf{v}_*^{(k)}$ and $\mathbf{w}_*^{(k)}$ in (4.1) are given by (4.2). In the steps 3 and 5 one needs to exactly solve a differential equation over the time-interval $[0, T_f]$.

Pseudocode: Continuous Jacobi WR

Process 0 (**p**0)	**Process** 1 (**p**1)
1: $\mathbf{w}^{(0)} \leftarrow$ initial guess	$\mathbf{v}^{(0)} \leftarrow$ initial guess
2: **for** $k = 0, \ldots, k_{\max} - 1$ **do**	**for** $k = 0, \ldots, k_{\max} - 1$ **do**
3: $\quad \hat{\mathbf{v}}^{(k+1)} \leftarrow$ Solve (4.1a)	$\quad \hat{\mathbf{w}}^{(k+1)} \leftarrow$ Solve (4.1c)
4: $\quad \mathbf{v}^{(k+1)} \leftarrow$ Relaxation (4.1b)	$\quad \mathbf{w}^{(k+1)} \leftarrow$ Relaxation (4.1d)
5: $\quad \mathbf{v}^{(k+1)} \rightarrow$ Send to **p**1	$\quad \mathbf{w}^{(k+1)} \rightarrow$ Send to **p**0
6: $\quad \mathbf{w}^{(k+1)} \leftarrow$ Recv. from **p**1	$\quad \mathbf{v}^{(k+1)} \leftarrow$ Recv. from **p**0
7: $\quad$ Check (4.4), `break` if `true`	$\quad$ Check (4.4), `break` if `true`
8: **end for**	**end for**

Algorithm 4.2: Pseudocode of the continuous Jacobi WR algorithm. Here, $\mathbf{v}_*^{(k)}$ and $\mathbf{w}_*^{(k)}$ in (4.1) are given by (4.3). In step 3 one needs to exactly solve differential equations over the time-interval $[0, T_f]$.

4.2 Time-discrete Waveform relaxation method

When using discrete time-integration to solve (4.1a) and (4.1c), we want to use independent time-grids and time-integration schemes. This is since the two subproblems in (1.1) may correspond to different physical phenomena, requiring different time-integration schemes and stepsizes. We enable this by using interpolants of the respective discrete solutions in the right-hand sides, which can then be evaluated in all $t \in [0, T_f]$. We denote discrete

solutions by

$$\underline{\mathbf{v}}^{(k)} := \{\mathbf{v}_n^{(k)}\}_{n=0,\ldots,N_v^{(k)}} \quad \text{and} \quad \underline{\mathbf{w}}^{(k)} := \{\mathbf{w}_n^{(k)}\}_{n=0,\ldots,N_w^{(k)}},$$

on time-grids

$$0 = t_0^{(v),(k)} < \ldots < t_{N_v^{(k)}}^{(v),(k)} = T_f \quad \text{and} \quad 0 = t_0^{(w),(k)} < \ldots < t_{N_w^{(k)}}^{(w),(k)} = T_f.$$

The interpolants are continuous functions:

$$\mathcal{I}(\underline{\mathbf{v}}^{(k)}) \in \mathcal{C}\left([0, T_f]; \mathbb{R}^{d_v}\right) \quad \text{and} \quad \mathcal{I}(\underline{\mathbf{w}}^{(k)}) \in \mathcal{C}\left([0, T_f]; \mathbb{R}^{d_w}\right).$$

Here, we omit the time-grids as input to the interpolants for ease of notation. We obtain a time-discrete WR method by using discrete time-integration to solve

$$\dot{\hat{\mathbf{v}}}^{(k+1)}(t) = \mathbf{g}\left(t, \hat{\mathbf{v}}^{(k+1)}(t), \mathcal{I}(\underline{\mathbf{w}}_*^{(k)})(t)\right), \quad \hat{\mathbf{v}}^{(k+1)}(0) = \mathbf{v}_0, \quad t \in [0, T_f], \tag{4.5a}$$

$$\dot{\hat{\mathbf{w}}}^{(k+1)}(t) = \mathbf{h}\left(t, \mathcal{I}(\underline{\mathbf{v}}_*^{(k)})(t), \hat{\mathbf{w}}^{(k+1)}(t)\right), \quad \hat{\mathbf{w}}^{(k+1)}(0) = \mathbf{w}_0, \quad t \in [0, T_f], \tag{4.5b}$$

and choosing $\underline{\mathbf{v}}_*^{(k)}$, $\underline{\mathbf{w}}_*^{(k)}$ in accordance with e.g., (4.2) or (4.3).

We consider *polynomial interpolation*. Then, relaxation is performed in the discrete data-points as follows

$$\mathbf{v}_n^{(k+1)} = (\mathbf{I} - \boldsymbol{\Theta}_{\boldsymbol{v}})\,\mathcal{I}(\underline{\mathbf{v}}^{(k)})(t_n^{(v),(k+1)}) + \boldsymbol{\Theta}_{\boldsymbol{v}}\hat{\mathbf{v}}_n^{(k+1)}, \quad n = 1, \ldots, N_v^{(k+1)}, \tag{4.6a}$$

$$\mathbf{w}_n^{(k+1)} = (\mathbf{I} - \boldsymbol{\Theta}_{\boldsymbol{w}})\,\mathcal{I}(\underline{\mathbf{w}}^{(k)})(t_n^{(w),(k+1)}) + \boldsymbol{\Theta}_{\boldsymbol{w}}\hat{\mathbf{w}}_n^{(k+1)}, \quad n = 1, \ldots, N_w^{(k+1)}. \tag{4.6b}$$

Pseudocodes of the time-discrete GS and Jacobi WR algorithms are shown in Algorithm 4.3 and 4.4. We use interpolation with evaluation at run-time. I.e., we define the interpolants once using fixed data-structures for $\underline{\mathbf{v}}_*$ and $\underline{\mathbf{w}}_*$. We then update $\underline{\mathbf{v}}_*$ and $\underline{\mathbf{w}}_*$ during the iteration, affecting the results of subsequent evaluations.

4.3 Convergence Analysis

Similar to [32, 33, 58, 59] we analyze convergence in the linear setting. Consider the following *monolithic system*

$$\mathbf{B}\dot{\mathbf{u}}(t) + \mathbf{A}\mathbf{u}(t) = \mathbf{f}(t), \quad \mathbf{u}(0) = \mathbf{u}_0 \in \mathbb{R}^d, \quad t \in [0, T_f < \infty], \quad \mathbf{B} \text{ nonsingular}, \tag{4.7}$$

Pseudocode: Time-discrete GS WR

$\underline{\mathbf{v}}^{(0)}$ discrete initial guess + initialize $\underline{\mathbf{v}}_*$ and $\mathcal{I}(\underline{\mathbf{v}}_*)$
$\underline{\mathbf{w}}^{(0)}$ discrete initial guess + initialize $\underline{\mathbf{w}}_*$ and $\mathcal{I}(\underline{\mathbf{w}}_*)$
for $k = 0, \dots, k_{\max} - 1$ **do**
    $\underline{\mathbf{w}}_* \leftarrow \underline{\mathbf{w}}^{(k)}$ Update interpolant
    $\underline{\hat{\mathbf{v}}}^{(k+1)} \leftarrow$ Discr. solve (4.5a)
    $\underline{\mathbf{v}}^{(k+1)} \leftarrow$ Relaxation (4.6a)
    $\underline{\mathbf{v}}_* \leftarrow \underline{\mathbf{v}}^{(k+1)}$ Update interpolant
    $\underline{\hat{\mathbf{w}}}^{(k+1)} \leftarrow$ Discr. solve (4.5b)
    $\underline{\mathbf{w}}^{(k+1)} \leftarrow$ Relaxation (4.6b)
    Check (4.4), `break` if `true`
end for

Algorithm 4.3: Pseudocode of the time-discrete GS WR method. Here, $\mathbf{v}_*^{(k)}$ and $\mathbf{w}_*^{(k)}$ in (4.5) are given by (4.2). Here, one can use a trivial initial guess for $\underline{\mathbf{v}}^{(0)}$, since it is not used in the iteration.

Pseudocode: Time-discrete Jacobi WR

Process 0 (**p**0)	**Process** 1 (**p**0)
$\underline{\mathbf{w}}^{(0)}$ discrete initial guess	$\underline{\mathbf{v}}^{(0)}$ discrete initial guess
Initialize $\underline{\mathbf{w}}_*$ and $\mathcal{I}(\underline{\mathbf{w}}_*)$	Initialize $\underline{\mathbf{v}}_*$ and $\mathcal{I}(\underline{\mathbf{v}}_*)$
for $k = 0, \dots, k_{\max} - 1$ **do**	**for** $k = 0, \dots, k_{\max} - 1$ **do**
$\underline{\mathbf{w}}_* \leftarrow \underline{\mathbf{w}}^{(k)}$ Update interpolant	$\underline{\mathbf{v}}_* \leftarrow \underline{\mathbf{v}}^{(k)}$ Update interpolant
$\underline{\hat{\mathbf{v}}}^{(k+1)} \leftarrow$ Discr. solve (4.5a)	$\underline{\hat{\mathbf{w}}}^{(k+1)} \leftarrow$ Discr. solve (4.5b)
$\underline{\mathbf{v}}^{(k+1)} \leftarrow$ Relaxation (4.6a)	$\underline{\mathbf{w}}^{(k+1)} \leftarrow$ Relaxation (4.6b)
$\underline{\mathbf{v}}^{(k+1)} \rightarrow$ Send to **p**1	$\underline{\mathbf{w}}^{(k+1)} \rightarrow$ Send to **p**1
$\underline{\mathbf{w}}^{(k+1)} \leftarrow$ Recv. from **p**1	$\underline{\mathbf{v}}^{(k+1)} \leftarrow$ Recv. from **p**1
Check (4.4), `break` if `true`	Check (4.4), `break` if `true`
end for	**end for**

Algorithm 4.4: Pseudocode of the time-discrete Jacobi WR method. Here, $\mathbf{v}_*^{(k)}$ and $\mathbf{w}_*^{(k)}$ in (4.5) are given by (4.3).

with $\mathbf{B}, \mathbf{A} \in \mathbb{R}^{d \times d}$ and $\mathbf{f}$ Lipschitz-continuous. Here, one can express classical WR method such as Jacobi and GS WR via constant splittings [33, 58]

$$\mathbf{B} = \mathbf{M}_B - \mathbf{N}_B \quad \text{and} \quad \mathbf{A} = \mathbf{M}_A - \mathbf{N}_A, \quad \mathbf{M}_B \text{ nonsingular}, \tag{4.8}$$

and the iteration

$$\begin{aligned} \mathbf{M}_B \dot{\mathbf{u}}^{(k+1)}(t) + \mathbf{M}_A \mathbf{u}^{(k+1)}(t) &= \mathbf{N}_B \dot{\mathbf{u}}^{(k)}(t) + \mathbf{N}_A \mathbf{u}^{(k)}(t) + \mathbf{f}(t), \\ \mathbf{u}^{(k+1)}(0) &= \mathbf{u}_0, \quad t \in [0, T_f]. \end{aligned} \tag{4.9}$$

The particular splitting (4.8) depends on the WR method, e.g., Jacobi or GS WR, and include relaxation. We omit dependencies of the splitting matrices on the relaxation matrices for readability.

Consider for example the following system of ODEs:

$$\underbrace{\begin{pmatrix}\mathbf{B}_1 & \mathbf{B}_2\\ \mathbf{B}_3 & \mathbf{B}_4\end{pmatrix}}_{\mathbf{B}}\underbrace{\begin{pmatrix}\dot{\mathbf{v}}(t)\\ \dot{\mathbf{w}}(t)\end{pmatrix}}_{\dot{\mathbf{u}}(t)}+\underbrace{\begin{pmatrix}\mathbf{A}_1 & \mathbf{A}_2\\ \mathbf{A}_3 & \mathbf{A}_4\end{pmatrix}}_{\mathbf{A}}\underbrace{\begin{pmatrix}\mathbf{v}(t)\\ \mathbf{w}(t)\end{pmatrix}}_{\mathbf{u}(t)}=\underbrace{\begin{pmatrix}\mathbf{f}_1(t)\\ \mathbf{f}_2(t)\end{pmatrix}}_{\mathbf{f}(t)},\quad \underbrace{\begin{pmatrix}\mathbf{v}(0)\\ \mathbf{w}(0)\end{pmatrix}}_{\mathbf{u}(0)}=\underbrace{\begin{pmatrix}\mathbf{v}_0\\ \mathbf{w}_0\end{pmatrix}}_{\mathbf{u}_0},$$

with $t \in [0, T_f]$ and $\mathbf{B}_1$, $\mathbf{B}_4$ nonsingular. Then, Jacobi WR, without relaxation, is given by

$$\begin{aligned}&\begin{pmatrix}\mathbf{B}_1 & \mathbf{0}\\ \mathbf{0} & \mathbf{B}_4\end{pmatrix}\dot{\mathbf{u}}^{(k+1)}(t)+\begin{pmatrix}\mathbf{A}_1 & \mathbf{0}\\ \mathbf{0} & \mathbf{A}_4\end{pmatrix}\mathbf{u}^{(k+1)}(t)\\ =&\begin{pmatrix}\mathbf{0} & -\mathbf{B}_2\\ -\mathbf{B}_3 & \mathbf{0}\end{pmatrix}\dot{\mathbf{u}}^{(k)}(t)+\begin{pmatrix}\mathbf{0} & -\mathbf{A}_2\\ -\mathbf{A}_3 & \mathbf{0}\end{pmatrix}\mathbf{u}^{(k)}(t)+\mathbf{f}(t),\end{aligned}\tag{4.10}$$

with $t \in [0, T_f]$ and $\mathbf{u}^{(k+1)}(0) = \mathbf{u}_0$. The inherent parallelism of this method is reflected by the block-diagonal structure of the matrices on the left-hand side.

Consider the additive block decomposition $\mathbf{B} = \mathbf{D}_B + \mathbf{L}_B + \mathbf{U}_B$, analogous for $\mathbf{A}$, into block diagonal and lower & upper triangular parts. Then, Jacobi WR is given by $\mathbf{M}_B = \mathbf{D}_B$ and $\mathbf{M}_A = \mathbf{D}_A$, see (4.10). Analogously, GS WR is given by the block-splittings $\mathbf{M}_B = \mathbf{D}_B + \mathbf{L}_B$ and $\mathbf{M}_A = \mathbf{D}_A + \mathbf{L}_A$.

The structures of these splittings are identical to GS and Jacobi methods for solving linear equation systems [90]. There, one solves the linear equation system

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \quad \mathbf{A} \in \mathbb{R}^{d\times d},\ \mathbf{x}, \mathbf{b} \in \mathbb{R}^d$$

via a splitting $\mathbf{A} = \mathbf{M}_A - \mathbf{N}_A$, with $\mathbf{M}_A$ nonsingular, and the iteration

$$\mathbf{x}_{k+1} = \mathbf{M}_A^{-1}\left(\mathbf{N}_A\mathbf{x}_k + \mathbf{b}\right), \quad k = 0, 1, \ldots,$$

given an initial guess $\mathbf{x}_0 \in \mathbb{R}^d$. As such, WR can be seen as the function space extension [34].

We can now determine the convergence properties of continuous WR methods by analyzing the iteration (4.9). Similarly, time-discrete WR methods are described via time-discretizations of (4.9).

4.3.1 Continuous iteration

The solution of the monolithic problem (4.7) is

$$\mathbf{u}(t) = \mathrm{e}^{-\mathbf{B}^{-1}\mathbf{A}t}\mathbf{u}_0 + \int_0^t \mathrm{e}^{\mathbf{B}^{-1}\mathbf{A}(s-t)}\mathbf{B}^{-1}\mathbf{f}(s)\,\mathrm{d}s.\tag{4.11}$$

Applying this solution formula to (4.9) yields

$$\mathbf{u}^{(k+1)}(t) = \mathrm{e}^{\mathbf{M}_B^{-1}\mathbf{M}_A t}\mathbf{u}_0 + \int_0^t \mathrm{e}^{\mathbf{M}_B^{-1}\mathbf{M}_A(s-t)}\mathbf{M}_B^{-1}\left(\mathbf{N}_B\dot{\mathbf{u}}^{(k)}(s) + \mathbf{N}_A\mathbf{u}^{(k)}(s) + \mathbf{f}(s)\right)\mathrm{d}s.$$

Here, we replace the $\dot{\mathbf{u}}^{(k)}(s)$ term using integration by parts:

$$\begin{aligned}\int_0^t \mathrm{e}^{\mathbf{M}_B^{-1}\mathbf{M}_A(s-t)}\mathbf{M}_B^{-1}\mathbf{N}_B\dot{\mathbf{u}}^{(k)}(s)\mathrm{d}s = &\left[\mathrm{e}^{\mathbf{M}_B^{-1}\mathbf{M}_A(s-t)}\mathbf{M}_B^{-1}\mathbf{N}_B\mathbf{u}^{(k)}(s)\right]_0^t \\ &- \int_0^t \mathrm{e}^{\mathbf{M}_B^{-1}\mathbf{M}_A(s-t)}\mathbf{M}_B^{-1}\mathbf{M}_A\mathbf{M}_B^{-1}\mathbf{N}_B\mathbf{u}^{(k)}(s)\mathrm{d}s.\end{aligned}$$

Then, basic rearrangement yields the solution

$$\mathbf{u}^{(k+1)}(t) = \mathbf{K}\mathbf{u}^{(k)}(t) + \int_0^t \boldsymbol{\mathcal{K}}_c(t-s)\mathbf{u}^{(k)}(s)\,\mathrm{d}s + \boldsymbol{\varphi}(t), \tag{4.12}$$

with

$$\begin{aligned}\mathbf{K} &= {\mathbf{M}_B}^{-1}\mathbf{N}_B, \\ \boldsymbol{\mathcal{K}}_c(t) &= \mathrm{e}^{-{\mathbf{M}_B}^{-1}\mathbf{M}_A t}{\mathbf{M}_B}^{-1}\left(\mathbf{N}_A - \mathbf{M}_A{\mathbf{M}_B}^{-1}\mathbf{N}_B\right), \\ \boldsymbol{\varphi}(t) &= \mathrm{e}^{-{\mathbf{M}_B}^{-1}\mathbf{M}_A t}(\mathbf{I} - {\mathbf{M}_B}^{-1}\mathbf{N}_B)\mathbf{u}_0 + \int_0^t \mathrm{e}^{\mathbf{M}_B^{-1}\mathbf{M}_A(s-t)}\mathbf{M}_B^{-1}\mathbf{f}(s)\mathrm{d}s.\end{aligned}$$

By construction via the splittings (4.8), the monolithic solution (4.11) is a fixed point to (4.12). Consider the error

$$\mathbf{e}^{(k)} := \mathbf{u}^{(k)} - \mathbf{u}, \tag{4.13}$$

where $\mathbf{u}$ is the monolithic solution (4.11). Taking the difference between (4.12) and (4.11) yields

$$\mathbf{e}^{(k+1)}(t) = \mathbf{K}\mathbf{e}^{(k)}(t) + \int_0^t \boldsymbol{\mathcal{K}}_c(t-s)\mathbf{e}^{(k)}(s)\mathrm{d}s, \quad \mathbf{e}^{(k+1)}(0) = \mathbf{0}, \quad t \in [0, T_f].$$

The source term $\mathbf{f}$ from (4.7) does not appear here, i.e., it does not affect the WR error.

Let a vector norm $\|\cdot\|$ be given. For the following theorem, we define the (vector) function norm $\|\mathbf{e}\|_{[0,t]} := \sup_{\tau\in[0,t]}\|\mathbf{e}(\tau)\|$ and (matrix) function norm $\|\mathbf{A}\|_{[0,t]} := \sup_{\tau\in[0,t]}\|\mathbf{A}(\tau)\|$, based on the induced matrix norm.

Theorem 4.1. *([33]) Consider the iteration* (4.9) *for problem* (4.7) *with splittings* (4.8) *and* $T_f < \infty$. *Then, the error* (4.13) *fulfills*

$$\|\mathbf{e}^{(k)}\|_{[0,t]} \leq \left(\sum_{j=0}^{k}\binom{k}{j}\|\mathbf{K}\|^{k-j}\|\boldsymbol{\mathcal{K}}_c\|_{[0,t]}^{j}\frac{t^j}{j!}\right)\|\mathbf{e}^{(0)}\|_{[0,t]}, \tag{4.14}$$

where $\|\mathbf{e}^{(k)}\|_{[0,t]} := \sup_{\tau\in[0,t]}\|\mathbf{e}^{(k)}(\tau)\|$.

Also see Theorem 6.3, which generalizes this result.

The term $\|\mathcal{K}_c\|^j_{[0,t]}\frac{t^j}{j!}$ decreases super-linearly for $j \to \infty$, but can lead to large bounds for large t and small k.

Corollary 4.2. *Consider splittings* (4.8)*,* $T_f < \infty$ *and* $\rho(\mathbf{K}) = \rho(\mathbf{M}_B^{-1}\mathbf{N}_B) < 1$*. Then, the iteration* (4.9) *converges to the solution* (4.11)*.*

The continuous WR operator is defined by

$$\mathcal{W} : \mathcal{C}([0, T_f]) \to \mathcal{C}([0, T_f]), \quad \mathbf{u} \to \mathbf{K}\mathbf{u} + \mathcal{K}_c \star \mathbf{u} + \boldsymbol{\varphi},$$

with the *convolution*

$$\mathcal{K}_c \star \mathbf{u}(t) = \int_0^t \mathcal{K}_c(t-s)\mathbf{u}^{(k)}(s)\,\mathrm{d}s. \tag{4.15}$$

The spectrum of $\mathcal{W}$ is given by $\mathbf{K}$ and is independent of $T_f < \infty$, since the convolution has a zero spectrum [33]. However, this no longer holds for $T_f = \infty$. In [33, 85] an example with convergence for $T_f < \infty$ and $\rho(\mathcal{W}) > 1$ for $T_f = \infty$ is shown. I.e., the spectrum of the WR operator with $T_f = \infty$ is not the limit of the finite time WR operator for $T_f \to \infty$.

It is shown in [42], that the WR operator (for $\mathbf{B} = \mathbf{M}_B = \mathbf{I}$) is nonnormal and its convergence behavior is better described by its *Pseudospectrum* [83] for large T_f. In particular, the Pseudospectrum of the finite time WR operator converges to the infinite time one [42].

4.3.2 Time-discrete iteration

Similar to [32, 59], we consider the time-discrete case using convergent and zero-stable linear multistep methods (LMM), see Appendix 8.1, for constant and matching stepsizes. Since we consider the linear system of ODEs (4.7), LMM and the linear splittings (4.8) for WR, we get the relation visualized in Figure 4.1. I.e., one can either apply the WR splitting to (4.7) to create the iteration (4.9), and then discretize in time, or first discretize the monolithic problem and then apply the splittings (4.8) in the resulting linear system.

Here, we first discretize in time, which gives

$$\sum_{\ell=0}^{m}(a_\ell\mathbf{B} + b_\ell\Delta t\mathbf{A})u_{n+\ell} = \Delta t\sum_{\ell=0}^{m} b_\ell\mathbf{f}(t_{n+\ell}). \tag{4.16}$$

Then, we use the splittings (4.8) to construct the iteration

$$\sum_{\ell=0}^{m}(a_\ell\mathbf{M}_B + b_\ell\Delta t\,\mathbf{M}_A)\mathbf{u}_{n+\ell}^{(k+1)} = \sum_{\ell=0}^{m}(a_\ell\mathbf{N}_B + b_\ell\Delta t\,\mathbf{N}_A)\mathbf{u}_{n+\ell}^{(k)} + \Delta t\sum_{\ell=0}^{m} b_\ell\mathbf{f}(t_{n+\ell}). \tag{4.17}$$

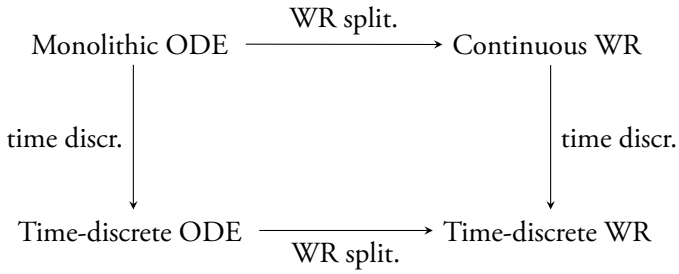

Figure 4.1: Diagram to visualize the relationship between the monolithic and time-discrete ODE resp. WR. Given a linear ODE, linear time-integration method and matching time-discretizations, this diagram is commutative.

By construction via the splittings (4.8), the solution to (4.16) is a fixed-point to (4.17). We define the time-discrete WR error

$$\mathbf{e}_n^{(k)} := \mathbf{u}_n - \mathbf{u}_n^{(k)}. \tag{4.18}$$

The difference of (4.16) and (4.17) yields

$$\sum_{\ell=0}^{m} (a_\ell \mathbf{M}_B + b_\ell \Delta t \mathbf{M}_A) \mathbf{e}_{n+\ell}^{(k+1)} = \sum_{\ell=0}^{m} (a_\ell \mathbf{N}_B + b_\ell \Delta t \mathbf{N}_A) \mathbf{e}_{n+\ell}^{(k)}. \tag{4.19}$$

The starting values $\mathbf{u}_\ell^{(k+1)}$ define the starting errors $\mathbf{e}_\ell^{(k+1)}$, $\ell = 0, \ldots, m-1$. We denote the above matrices as follows:

$$\mathbf{C}_\ell := a_\ell \mathbf{M}_B + b_\ell \Delta t \mathbf{M}_A, \quad \mathbf{D}_\ell := a_\ell \mathbf{N}_B + b_\ell \Delta t \mathbf{N}_A, \quad \ell = 0, \ldots, m. \tag{4.20}$$

In the following Theorem we consider convergence in terms the vector

$$\underline{\mathbf{e}}^{(k)} := (\mathbf{e}_m^{(k)}, \ldots, \mathbf{e}_N^{(k)})^T \in \mathbb{R}^{d(N-m+1)},$$

representing the discrete solution all in time-points, not including starting values.

Theorem 4.3. *([32]) Consider iteration* (4.17) *with*

$$\mathbf{C}_m \textit{ nonsingular}, \quad \rho(\mathbf{C}_m^{-1}\mathbf{D}_m) < 1,$$

exact starting values $\mathbf{u}_i^{(k+1)} = \mathbf{u}_i$, $i = 0, \ldots, m-1$, $\forall k \geq 0$ *and an initial guess* $\underline{\mathbf{e}}^{(0)} \in \mathbb{R}^{d(N-m+1)}$. *Then, the iteration converges to the solution of* (4.16).

Proof. Consider the so-called "all-at-once" system

$$\underbrace{\begin{pmatrix} \mathbf{C}_m & \mathbf{0} & \dots & \dots & \dots & \mathbf{0} \\ \vdots & \mathbf{C}_m & \ddots & & & \vdots \\ \mathbf{C}_0 & & \ddots & \ddots & & \vdots \\ \mathbf{0} & \ddots & & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & & \ddots & \mathbf{0} \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{C}_0 & \dots & \mathbf{C}_m \end{pmatrix}}_{=:\underline{\mathbf{C}}} \underline{\mathbf{e}}^{(k+1)} = \underbrace{\begin{pmatrix} \mathbf{D}_m & \mathbf{0} & \dots & \dots & \dots & \mathbf{0} \\ \vdots & \mathbf{D}_m & \ddots & & & \vdots \\ \mathbf{D}_0 & & \ddots & \ddots & & \vdots \\ \mathbf{0} & \ddots & & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & & \ddots & \mathbf{0} \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{D}_0 & \dots & \mathbf{D}_m \end{pmatrix}}_{=:\underline{\mathbf{D}}} \underline{\mathbf{e}}^{(k)},$$

c.f. (4.20), with $\underline{\mathbf{C}}, \underline{\mathbf{D}} \in \mathbb{R}^{(d(N-m+1))\times(d(N-m+1))}$. Both block-lower triangular and block-Toeplitz structures are preserved under inversion and multiplication with $\underline{\mathbf{D}}$. Thus, the iteration matrix $\underline{\mathbf{C}}^{-1}\underline{\mathbf{D}}$ is block-lower triangular and block-Toeplitz, with $\mathbf{C}_m^{-1}\mathbf{D}_m$ on its block diagonal, which gives

$$\rho\left(\underline{\mathbf{C}}^{-1}\underline{\mathbf{D}}\right) = \rho\left(\mathbf{C}_m^{-1}\mathbf{D}_m\right). \tag{4.21}$$

As such, $\rho(\mathbf{C}_m^{-1}\mathbf{D}_m) < 1$ is a sufficient criterion for convergence of the iteration. □

The assumption of $\mathbf{C}_m$ nonsingular is a solvability assumption on the occurring linear systems in (4.17) resp. (4.19). The result of Theorem 4.3 is consistent with the continuous result of Corollary 4.2, in that assumptions and asymptotic convergence rate align for $\Delta t \to 0$. Furthermore, it is sufficient to look at a single timestep to determine the spectral radius, which is consistent with the continuous asymptotic convergence rate being independent of T_f, if $T_f < \infty$.

The iteration matrix is not normal, c.f. Theorem 8.5 in Appendix 8.2. Thus, the spectral radius is the asymptotic convergence rate for $k \to \infty$. In particular, this can lead to an initial error growth, which is consistent with the continuous error bound in Theorem 4.1. The degree of non-normality can be quantified by analyzing the Pseudospectrum [42, 83].

Similar to the continuous iteration, which can be described via the convolution (4.15), the discrete iteration is a discrete convolution [32].

Definition 4.4. *Let f, g defined on $\mathbb{Z}$. A discrete convolution is defined by*

$$(f \star g)[n] := \sum_{i=-\infty}^{\infty} f[n-i]\, g[i].$$

In our case, we have compact support and the relationship

$$\underline{\mathbf{e}}_n^{(k+1)} = \left(\left(\underline{\mathbf{C}}^{-1}\underline{\mathbf{D}}\right) * \underline{\mathbf{e}}^{(k)}\right)_n = \sum_{i=1}^{n} \left(\underline{\mathbf{C}}^{-1}\underline{\mathbf{D}}\right)_{n-i+1,\,1} \underline{\mathbf{e}}_i, \quad n = m, \dots, N,$$

due to the Toeplitz structure of $\underline{\mathbf{C}}^{-1}\underline{\mathbf{D}}$. Here, $\underline{\mathbf{e}}_n^{(k+1)} \in \mathbb{R}^d$ and $\left(\underline{\mathbf{C}}^{-1}\underline{\mathbf{D}}\right)_{i,j} \in \mathbb{R}^{d\times d}$ refer to blocks.

Consider $\mathbf{B} = \mathbf{M}_B = \mathbf{I}$ and an explicit LMM characterized by $b_m = 0$. This yields $\mathbf{D}_m = \mathbf{0}$. As such, the iteration matrix has the spectral radius 0, i.e., convergence is super-linear. In particular, $\underline{\mathbf{C}}^{-1}\underline{\mathbf{D}}$ is nil-potent and the iteration yields the discrete monolithic solution after at most N iterations, where N is the number of timesteps.

4.4 Relaxation

We now consider relaxation in the linear setting from Section 4.3. Then, the first step for both Jacobi and GS WR, omitting the $\mathbf{f}$ term, is

$$\begin{aligned}\mathbf{B}_1\dot{\hat{\mathbf{v}}}^{(k+1)}(t) + \mathbf{A}_1\hat{\mathbf{v}}^{(k+1)}(t) &= -\mathbf{B}_2\dot{\mathbf{w}}^{(k)}(t) - \mathbf{A}_2\mathbf{w}^{(k)}(t),\\ \hat{\mathbf{v}}^{(k+1)}(0) &= \mathbf{v}_0, \quad t \in [0, T_f],\\ \mathbf{v}^{(k+1)} &= \mathbf{v}^{(k)} + \boldsymbol{\Theta}_v(\hat{\mathbf{v}}^{(k+1)} - \mathbf{v}^{(k)}).\end{aligned}$$

Substitution to eliminate the intermediate solution $\hat{\mathbf{v}}^{(k+1)}$, c.f. [34], yields

$$\begin{aligned}\mathbf{B}_1\boldsymbol{\Theta}_v^{-1}\dot{\mathbf{v}}^{(k+1)} + \mathbf{A}_1\boldsymbol{\Theta}_v^{-1}\mathbf{v}^{(k+1)} &= \mathbf{B}_1\left(\boldsymbol{\Theta}_v^{-1} - \mathbf{I}\right)\dot{\mathbf{v}}^{(k)} + \mathbf{A}_1\left(\boldsymbol{\Theta}_v^{-1} - \mathbf{I}\right)\mathbf{v}^{(k)}\\ -\mathbf{B}_2\dot{\mathbf{w}}^{(k)}(t) - \mathbf{A}_2\mathbf{w}^{(k)}(t), &\quad \mathbf{v}^{(k+1)} = \mathbf{0}, \quad t \in [0, T_f].\end{aligned}$$

The second steps for both Jacobi and GS WR work analogously. Defining

$$\boldsymbol{\Theta} := \begin{pmatrix}\boldsymbol{\Theta}_v & \mathbf{0}\\ \mathbf{0} & \boldsymbol{\Theta}_w\end{pmatrix}$$

and using the same additive block-decompositions from Section 4.3, we get Jacobi WR with the splittings $\mathbf{M}_B = \mathbf{D}_B\boldsymbol{\Theta}^{-1}$ and $\mathbf{M}_A = \mathbf{D}_A\boldsymbol{\Theta}^{-1}$. Similarly, we get GS WR by adding $\mathbf{L}_B$ resp. $\mathbf{L}_A$ to $\mathbf{M}_B$ resp. $\mathbf{M}_A$. This shows that the splitting description of WR methods from Section 4.3 includes relaxation.

The error bound (4.14) consists of a linear and a super-linear component. In choosing relaxation we focus on the linear term $\mathbf{K}$, since it is independent of t and constitutes the asymptotic convergence rate. Choosing relaxation to minimize the impact of the super-linear term can lead to better observed convergence rates for large T_f, by reducing possible initial error growth associated with the super-linear term. This approach is pursued in [34, 63], using an additional convolution relaxation term.

The canonical relaxation is $\boldsymbol{\Theta} = \Theta\mathbf{I}$, $\Theta \in \mathbb{R} \setminus \{0\}$ and yields $\mathbf{M}_B(\Theta)$ and $\mathbf{M}_A(\Theta)$ with e.g., $\mathbf{M}_B(\Theta) = \frac{1}{\Theta}\mathbf{D}_B + \mathbf{L}_B$ in the GS case. Then, the optimal relaxation parameter Θ_{opt}

is given by

$$\Theta_{opt}^{cont} = \text{argmin}_{\Theta \in \mathbb{R} \setminus \{0\}} \rho \left(\mathbf{M}_B^{-1}(\Theta) \mathbf{N}_B(\Theta) \right).$$

In the time-discrete setting, we have the result (4.21), i.e., in a given time-discretization, one only needs to consider a single timestep to determine the spectral radius of the iteration matrix, we get

$$\Theta_{opt}^{discr} = \text{argmin}_{\Theta \in \mathbb{R} \setminus \{0\}} \rho \left(\mathbf{C}_m^{-1}(\Theta) \mathbf{D}_m(\Theta) \right).$$

In particular, Θ_{opt}^{discr} depends on Δt and $\Theta_{opt}^{discr} \to \Theta_{opt}^{cont}$ for $\Delta t \to 0$.

4.5 Windowing

As shown in Theorem 4.1, the WR error bound critically depends on t. In practice, a large T_f can lead to observed convergence rates slower than the asymptotic ones, or even an initial error growth. Thus, a common technique [59] is to divide $[0, T_f]$ into *windows* $0 = T_0 < \ldots < T_M = T_f$ and successively perform WR on them. Sufficiently small time-windows can prevent initial error growth due to the super-linear term in (4.14), resp. reduce degree of non-normality of $\underline{\mathbf{C}}^{-1}\underline{\mathbf{D}}$.

The initial value in the time-window $[T_m, T_{m+1}]$ is $\mathbf{u}(T_m)$, the final value of the previous time-window. The initial value for each time window includes an additional error due to termination of the iteration when the update is below a given tolerance, see (4.4). With TOL_{WR} sufficiently small, this error is negligible w.r.t. the time-integration error.

Time-point relaxation [8] is the case when stepsize and window size are equal, i.e., one timestep per time-window. Thus, it is a special case of WR. Additionally, using only a single iteration per time-window constitutes an *operator splitting*, c.f. [20]. In these two variants, replacing the single timestep per time-window by several steps is referred to *subcycling* [20].

Lastly, as a practical consideration, the use of time-windows can greatly reduce the memory cost of storing the interpolants.

4.6 Implementation

Here we provide an overview the implementation framework used, see [51] for the code. We implemented the WR algorithms in `C++`, since the WR methods in Chapter 6 resp. Paper IV require asynchronous `MPI` functionality only available in `C++` or `Fortran`.

We use a `Waveform` class for the interpolants from Section 4.2. Evaluation at a time-point t performs the interpolation. Otherwise, the class has a variety of functions for computing updates, relaxation and convenient data handling.

WR methods pose few restrictions on the problems to be coupled and we choose an implementation framework with minimal requirements on the subproblems. The subproblems resp. subsolvers only interact via the subset of unknowns of $\mathbf{v}$ or $\mathbf{w}$ that the respective other right-hand side in (1.1) depends on. We call these the *exchange variables* and differentiate between the *input* and *output* variables for a given subproblem.

Since WR methods do not require access to the internal unknowns of a subsolver, we treat spatial discretizations of PDE subsolvers as black-boxes. We use exchange variables in the form of real vectors of fixed length. With PDE subsolvers this corresponds to exchanging e.g., temperatures or heat fluxes in a fixed set of points in space.

We represent a subsolver by the following *abstract* `WRproblem` class:

```
class WRproblem{
protected:
/* ... */
public:
    // implementation required
    virtual void get_u0(double *uout);
    virtual void do_step(double t, double dt, double *uout,
   Waveform *WF) = 0;
    virtual void create_checkpoint()   = 0;
    virtual void reset_to_checkpoint() = 0;

    // implementation optional
    virtual double get_norm_factor(){return 1;};
    virtual void callback_iteration(){};
    virtual void callback_window(){};
    virtual void callback_finalize(){};
};
```

The purpose of this class is to define the minimal necessary functionality a subsolver needs to implement for partitioned time-integration using WR methods. The above functions allow the WR algorithm to control time-integration. It is designed to enable a black-box coupling.

Figure 4.2 shows the architecture for the parallel Jacobi WR method run with two processors, one for each subsolver. With sequential WR methods, such as GS WR, the relations between the WR code the `WRproblem` instances are analogous.

The role of an adapter is to be the connecting piece to libraries such as `DUNE` or `FEniCS`.

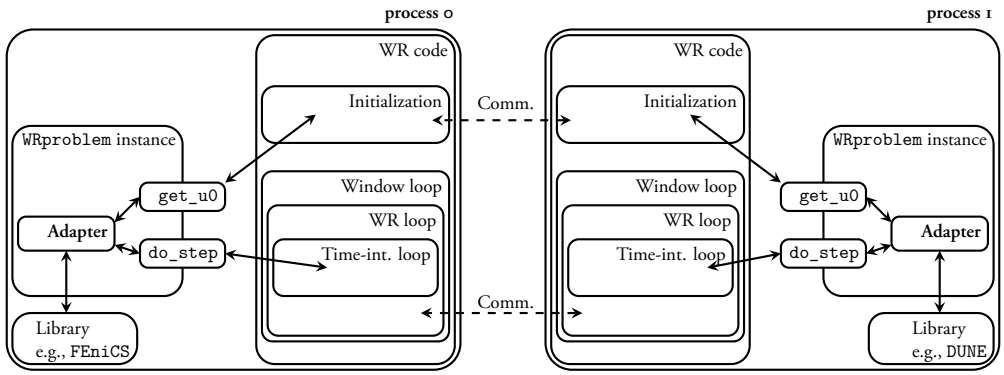

Figure 4.2: Our architecture of a parallel WR method, e.g., Jacobi WR, run on two processors. Not shown in this figure: Termination check (4.4), creation and restoration of checkpoints, and calls of callback functions.

We discuss details on the adapter in Section 4.6.1.

A parallel WR algorithm is executed on each processor and called with an instance of the `WRproblem` class each. In the initialization of the WR method the processors exchange necessary information on in -and output lengths. Additionally, `get_u0` is called to obtain an initial guess of the output exchange variables. These initial guesses are exchanged and used to initialize the interpolant on the respective other processor.

The WR loop is described by Algorithm 4.4 and time-integration is performed by repeated calls of the `do_step` function, which facilitates a single step of time-integration. We currently assume fixed time-grids with constant, but not necessarily matching stepsizes. As such, t and Δt are known in advance for every timestep. The output `uout` is the output exchange variable at $t + \Delta t$. Here, the input exchange variable is represented by a *single* interpolant resp. `Waveform` instance. This does not allow the exchange of intermediate solutions obtained in the time-integration of a subproblem when using e.g., RK schemes.

Note that with the WR methods shown in this chapter, the `WRproblem` class could instead use a function for performing time-integration on the whole time-window. Our choice of controlling time-integration from the WR loop was done to be consistent with the WR algorithms presented in Chapter 6 resp. Paper IV , which require this functionality.

Not shown in Figure 4.2 are the `create_checkpoint` and `reset_to_checkpoint` functions, used to create and restore checkpoints of the internal unknowns. A checkpoint is created at the beginning of each time-window and is restored at the end of each WR iteration, unless the termination criterion is met.

In the termination criterion (4.4) we compute the 2-norm of the update in either a single, or both exchange variables combined, at the (window) end-point. By using an additional weighting factor provided by the `WRproblem` class, we can create e.g., interface $\mathcal{L}^2$ norms.

The `callback` functions are invoked at the end of each iteration, time-window and upon finalizing the simulation. These can be used to e.g., (re)set time-integration related parameters or to write internal states of a given problem to disk for visualization.

4.6.1 Adapter

Here, we present the role of the adapter in the context of coupling subproblems solved with the open source packages `DUNE` [6] and `FEniCS` [41]. Our adapter for e.g., `DUNE` is not a universal piece of code to couple any conceivable PDE solved with `DUNE`, but rather a collection of tools for solving common issues to enable the coupling.

The libraries `DUNE` and `FEniCS` offer convenient and efficient tools for constructing PDE subsolvers, in particular space discretizations. The solutions resp. unknowns are stored using library specific data-structures. The core role of the adapter is to handle conversion from and to these data-structures, since we use real vectors in the data exchange. Obtaining the output exchange variable of a given subproblem requires construction of the solution in the set of points used for the interface information exchange. The input requires conversion of the discrete interface data to the suitable internal data-structures. With `DUNE` and `FEniCS` this consists of interpolating the input data to discrete function spaces. Note that both of these operations are performed with every call of `do_step`.

The development and maintenance of an adapter requires developer rather than mere user level knowledge of the given libraries to be coupled [84]. The `DUNE` adapter has been developed in close collaboration with Robert Klöfkorn [17]. Our `FEniCS` adapter has been developed jointly with Benjamin Rodenberg [66] and has been further developed to a universal adapter for the coupling of `FEniCS` based subsolvers in `preCiCe` [64].

When trying to couple an existing subsolver code, one first needs to ensure all involved libraries support the coupling, e.g., all instances of `MPI_COMM_WORLD` must be replaced by suitable communicators (currently `MPI_COMM_SELF`). The next step is to restructure the given subsolver code to fit the abstract `WRproblem` class, processing inputs and providing outputs using the correct data-structures. Another likely step is to re-structure existing time-integration routines, such that they can be called in a step-wise manner using the `do_step` function.

Lastly, using external libraries might require crossing the programming language barrier. In the case of `DUNE` and `FEniCS` this means to embed `Python` in `C++`, i.e., call `Python` functions from `C++`. The most straight-forward way is to use the same principle `WRproblem` class structure in e.g., `Python` and call the respective functions using the `C++` to interface to `Python`. The latter part is largely independent of the actual problem to be coupled. Here, both `DUNE` and `FEniCS` use the same code for this embedding.

4.6.2 Discussion

With adaptive time-grids we want subsolvers to choose their own stepsizes. Then, a subsolver requires T_f, resp. the endpoint of the current time-window, and needs to output the stepsize used. With adaptive grids, interpolants are defined by a variable number of time -and data-points. Then, one requires data-structures that allow efficient allocation of extra memory at runtime. Alternatively, one estimates an upper bound for the number of timesteps and allocates sufficient memory.

Changing the WR methods from this chapter to work with adaptive time-grids is straightforward. However, our main use of the classical WR methods presented in this chapter is as reference for the WR methods presented in Chapter 6 resp. Paper IV . There, the use of adaptive time-grids is not straight-forward. Since optimal relaxation depends on the stepsizes used, see Section 4.4, we would then want the relaxation parameters to be provided by a function of the `WRproblem` class, rather than as input parameters.

Our current implementation, as shown in Figure 4.2, does not support parallelism in the subsolvers. Changing the code to do so would foremost require a `WRproblem` instance to take a `MPI_Comm` instance as input during initialization. This communicator can then be used to construct and store the space discretization using multiple processors, which is standard functionality in `DUNE` and `FEniCS`.

The difficulty is to facilitate efficient and scalable communication via the interface. E.g., `preCiCe` establishes fixed communication channels between the processors of the subsolvers whose discretizations include the coupling interface [13, 67]. With surface couplings, this greatly reduces the number of processors involved in communication between the subproblems. However, it severely restricts possibilities for spatial adaptivity and dynamic load-balancing on the subsolvers.

Chapter 5

Dirichlet-Neumann Waveform Relaxation for heterogeneous coupled heat equations

Consider the linear heat equation on a domain $\Omega \subset \mathbb{R}^d$ as follows:

$$\begin{aligned} \alpha(\mathbf{x})\partial_t u(t,\mathbf{x}) - \nabla\cdot(\lambda(\mathbf{x})\nabla u(t,\mathbf{x})) = 0, \quad (t,\mathbf{x}) &\in (0,T_f]\times\Omega, \\ u(t,\mathbf{x}) = 0, \quad (t,\mathbf{x}) &\in [0,T_f]\times\partial\Omega, \\ u(0,\mathbf{x}) = u_0(\mathbf{x}), \quad \mathbf{x} &\in \Omega. \end{aligned} \tag{5.1}$$

Here, λ is the thermal conductivity of the material. The thermal diffusivity D is defined by

$$D = \lambda/\alpha, \quad \text{with} \quad \alpha = \rho c_p,$$

with density ρ and specific heat capacity c_p. In particular, we consider $\Omega = \Omega_1 \cup \Omega_2$, with $\alpha(\mathbf{x})$ and $\lambda(\mathbf{x})$ constant on the subdomains Ω_1 and Ω_2. We thus denote $\alpha_m = \alpha\big|_{\Omega_m}$, $\lambda_m = \lambda\big|_{\Omega_m}$ and $u_m = u\big|_{\Omega_m}$, $m = 1,2$.

Problem (5.1) is equivalent to

$$\alpha_m\partial_t u_m(t,\mathbf{x}) - \lambda_m\Delta u_m(t,\mathbf{x}) = 0, \quad (t,\mathbf{x}) \in (0,T_f]\times\Omega_m, \tag{5.2a}$$

$$u_m(t,\mathbf{x}) = 0, \quad (t,\mathbf{x}) \in [0,T_f]\times\Omega_m\setminus\Gamma, \tag{5.2b}$$

$$u_m(0,\mathbf{x}) = u_0(\mathbf{x}), \quad \mathbf{x}\in\Omega_m, \quad m = 1,2. \tag{5.2c}$$

$$u_1(t,\mathbf{x}) = u_2(t,\mathbf{x}), \quad (t,\mathbf{x}) \in (0,T_f]\times\Gamma, \tag{5.2d}$$

$$\lambda_1\nabla u_1(t,\mathbf{x})\cdot\mathbf{n}_1 = -\lambda_2\nabla u_2(t,\mathbf{x})\cdot\mathbf{n}_2, \quad (t,\mathbf{x}) \in (0,T_f]\times\Gamma, \tag{5.2e}$$

in a weak sense [61, Chap.7]. Here, (5.2) is commonly referred to as the *unsteady transmission problem*, with *transmission conditions* (5.2d) and (5.2e) at the interface $\Gamma = \partial\Omega_1 \cap \partial\Omega_2$.

We thus consider the following two coupled problems: The Dirichlet problem defined by (5.2a) - (5.2c) for $m = 1$ and (5.2d), and the Neumann problem defined by (5.2a) - (5.2c) for $m = 2$ and (5.2e).

Overview over chapter

In this chapter, resp. Paper III , we develop a fast and highly robust, time-adaptive WR method for the specific application of solving heterogeneous coupled heat equations. In particular, we consider the GS type Dirichlet-Neumann Waveform relaxation (DNWR) method, while also comparing results to the closely related parallel Neumann-Neumann Waveform relaxation (NNWR) method.

In Paper III we first derive the continuous, and then semi-discrete DNWR method. We then discretize the individual subproblems using the implicit Euler resp. second order SDIRK2 methods to obtain fully discrete DNWR methods. By individually discretizing the subproblems of the semi-discrete DNWR method, the use of non-matching and adaptive time-grids is straight-forward.

Here, we present a different approach to deriving the same methods, which provides more insight into important details when constructing higher order partitioned schemes. We construct the fully discrete DNWR methods following the same approach used in the time-discrete WR analysis in Section 4.3.2. That is, we first derive a fully discrete method solving the monolithic problem (5.1) and then apply suitable splittings (4.8) to obtain a fully discrete DNWR method. This approach guarantees that solution to the fully discrete monolithic problem is a fixed point of the resulting iteration, for matching stepsizes. Here, we follow the notation of Chapter 4, which slightly differs from Paper III .

5.1 Dirichlet-Neumann Waveform relaxation

In the WR approach to DNWR we require the semi-discrete problem as a starting point. See Section 3 in Paper III for the continuous (space and time) DNWR method.

5.1.1 Semi-discrete iteration

Assume a space discretization in which the unknowns correspond to the nodal values of a subdivision (e.g., triangulation) of Ω. Then, splittings (4.8) of the semi-discrete PDE

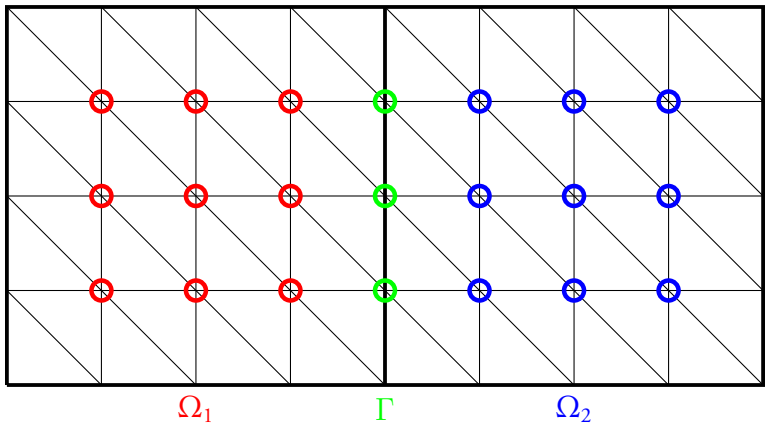

Figure 5.1: Example for triangulations of Ω_1 and Ω_2 with matching nodes on Γ. The different colored circles mark the interior resp. interface nodes.

corresponds to a decomposition of the spatial domain Ω.

Here, the jump in the coefficients $\alpha(\mathbf{x})$ resp. $\lambda(\mathbf{x})$ defines a natural decomposition and we thus consider a subdivision of Ω that aligns at Γ. I.e., subdivisions of Ω_1 and Ω_2 with matching nodes on Γ. Furthermore, the union of the subdivisions of Ω_1 and Ω_2 is a subdivision of Ω without hanging nodes. An example is shown in Figure 5.1.

Consider $\mathbf{u} = \left(\mathbf{u}_I^{(1)^T}, \mathbf{u}_I^{(2)^T}, \mathbf{u}_\Gamma^T \right)^T \in \mathbb{R}^{d_1+d_2+s}$ as the unknowns on Ω. Here d_m, $m = 1, 2$ is the number of interior unknowns in Ω_m and s the number of unknowns on the interface. Then, a general semi-discretization of (5.1) can be written as

$$\mathbf{B}\dot{\mathbf{u}}(t) + \mathbf{A}\mathbf{u}(t) = \mathbf{0}, \quad \mathbf{u}(0) = \mathbf{u}_0, \quad t \in [0, T_f], \tag{5.3}$$

with

$$\mathbf{B} = \begin{pmatrix} \mathbf{B}_{II}^{(1)} & \mathbf{0} & \mathbf{B}_{I\Gamma}^{(1)} \\ \mathbf{0} & \mathbf{B}_{II}^{(2)} & \mathbf{B}_{I\Gamma}^{(2)} \\ \mathbf{B}_{\Gamma I}^{(1)} & \mathbf{B}_{\Gamma I}^{(2)} & \mathbf{B}_{\Gamma\Gamma}^{(1)} + \mathbf{B}_{\Gamma\Gamma}^{(2)} \end{pmatrix}, \quad \mathbf{A} = \begin{pmatrix} \mathbf{A}_{II}^{(1)} & \mathbf{0} & \mathbf{A}_{I\Gamma}^{(1)} \\ \mathbf{0} & \mathbf{A}_{II}^{(2)} & \mathbf{A}_{I\Gamma}^{(2)} \\ \mathbf{A}_{\Gamma I}^{(1)} & \mathbf{A}_{\Gamma I}^{(2)} & \mathbf{A}_{\Gamma\Gamma}^{(1)} + \mathbf{A}_{\Gamma\Gamma}^{(2)} \end{pmatrix}. \tag{5.4}$$

For the consideration of the splitting, we neglect relaxation, i.e., $\mathbf{\Theta} = \mathbf{I}$. Then, the semi-discrete DNWR method is given by the iteration

$$\mathbf{M}_B\dot{\mathbf{u}}^{(k+1)}(t) + \mathbf{M}_A\mathbf{u}^{(k+1)}(t) = \mathbf{N}_B\dot{\mathbf{u}}^{(k)}(t) + \mathbf{N}_A\mathbf{u}^{(k)}(t), \quad \mathbf{u}(0) = \mathbf{u}_0, \quad t \in [0, T_f], \tag{5.5}$$

c.f. (4.9), with the splittings

$$\mathbf{M}_B = \begin{pmatrix} \mathbf{B}_{II}^{(1)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{II}^{(2)} & \mathbf{B}_{I\Gamma}^{(2)} \\ \mathbf{B}_{\Gamma I}^{(1)} & \mathbf{B}_{\Gamma I}^{(2)} & \mathbf{B}_{\Gamma\Gamma}^{(2)} \end{pmatrix}, \ \mathbf{M}_A = \begin{pmatrix} \mathbf{A}_{II}^{(1)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{II}^{(2)} & \mathbf{A}_{I\Gamma}^{(2)} \\ \mathbf{A}_{\Gamma I}^{(1)} & \mathbf{A}_{\Gamma I}^{(2)} & \mathbf{A}_{\Gamma\Gamma}^{(2)} \end{pmatrix}. \tag{5.6}$$

This result can be summarized with the diagram shown in Figure 5.2. It shows that with our meshes, the solution of the semi-discrete PDE converges to the solution of the monolithic PDE for $\Delta x \to 0$, and by construction, the solution of the semi-discrete PDE is a fixed-point to the semi-discrete DNWR method.

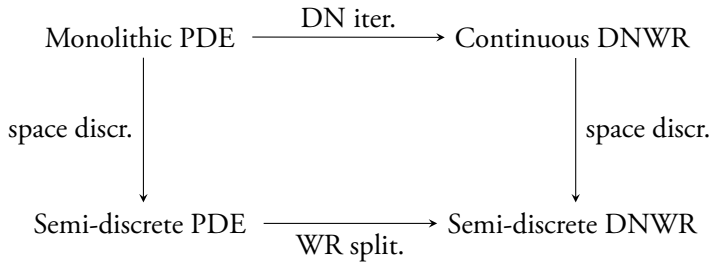

Figure 5.2: Commutative diagram to visualize the relationship between the continuous and semi-discrete PDEs -and DNWR methods. This guarantees that the solution of semi-discrete PDE is a fixed point to the semi-discrete DNWR method. The different steps are commutative due to the triangulations of Ω_1 and Ω_2 sharing the same nodes and interface unknowns on Γ.

In Paper III we use the common Domain Decomposition approach [82, pp.3] of using Green's formula to approximate the semi-discrete heat-flux in a finite element setting as follows:

$$\lambda_1 \int_\Gamma (\nabla u_1 \cdot \mathbf{n}_1)\phi_j \, \mathrm{d}S = \lambda_1 \int_{\Omega_1} (\Delta u_1 \phi_j + \nabla u_1 \nabla \phi_j) \mathrm{d}\mathbf{x}$$
$$= \alpha_1 \int_{\Omega_1} \frac{\mathrm{d}}{\mathrm{d}t} u_1 \phi_j \, \mathrm{d}\mathbf{x} + \lambda_1 \int_{\Omega_1} \nabla u_1 \nabla \phi_j \, \mathrm{d}\mathbf{x},$$

with basis functions ϕ_j to a suitable finite element space. This results in the semi-discrete heat flux

$$\mathbf{q}^{(k+1)}(t) = \mathbf{B}_{\Gamma I}^{(1)} \dot{\mathbf{u}}_I^{(1),(k+1)}(t) + \mathbf{A}_{\Gamma I}^{(1)} \mathbf{u}_I^{(1),(k+1)}(t) + \mathbf{B}_{\Gamma\Gamma}^{(1)} \dot{\mathbf{u}}_\Gamma^{(k)}(t) + \mathbf{A}_{\Gamma\Gamma}^{(1)} \mathbf{u}_\Gamma^{(k)}(t). \tag{5.7}$$

Here, we get the same semi-discrete heat flux by gathering $\mathbf{u}_I^{(1),(k+1)}$ and $\mathbf{u}_\Gamma^{(k)}$ terms, including derivatives, in the third block row of the iteration (5.5).

Paper III also discusses the related Neumann-Neumann Waveform relaxation (NNWR) method [39, 56, 57], which is based on the same subproblems. It first solves a Dirichlet problem on each sub-domain, followed by a Neumann problem on each sub-domain, in parallel. It is not clear how to derive the NNWR method on a WR splitting basis.

5.2 Time-discretization

We now discuss time-discretizations using implicit Euler and the second order SDIRK2 scheme. Since both the problems and time-integration schemes are linear, we get the same type of relationship as shown in Figure 4.1. In the context of the DNWR method, the analogue relationships are shown in Figure 5.3. Here, we follow the same approach as

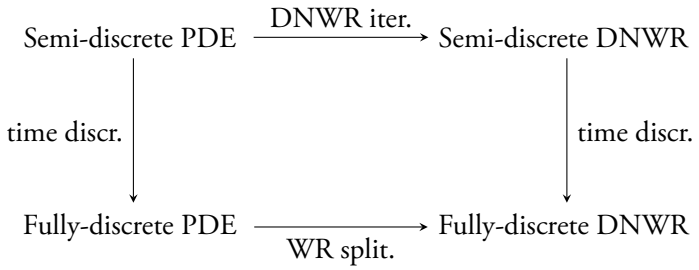

Figure 5.3: Diagram to visualize the relationship between the semi -and fully-discrete PDEs -and DNWR methods. With carefully chosen time-integration schemes, c.f. Section 5.2.1, 5.2.2, and matching time-grids, this is a commutative diagram.

in Section 4.3.2, by first discretizing in time and then applying the splittings (5.6) to the resulting linear systems. By construction, this guarantees a fully-discrete DNWR method with the solution of the fully-discrete PDE as its fixed-point.

However, the resulting method is *not partitioned*, since it is based on a time-discretization of the full semi-discrete PDE. I.e., it uses matching time-grids, relying on the time-point solutions of the respective other sub-domain. We demonstrate how to obtain the corresponding partitioned method, allowing independent time-grids and adaptivity, by example for the implicit Euler method.

5.2.1 Implicit Euler

The implicit Euler scheme applied to (5.3) is

$$(\mathbf{B} + \Delta t\, \mathbf{A})\mathbf{u}_{n+1} = \mathbf{B}\mathbf{u}_n$$

with matrices (5.4). Applying the splittings (5.6) gives the iteration

$$(\mathbf{M}_B + \Delta t\, \mathbf{M}_A)\mathbf{u}_{n+1}^{(k+1)} - \mathbf{M}_B\mathbf{u}_n^{(k+1)} = (\mathbf{N}_B + \Delta t\, \mathbf{N}_A)\mathbf{u}_n^{(k)} - \mathbf{N}_B\mathbf{u}_n^{(k)},$$

with $\mathbf{u}_0^{(k+1)} = \mathbf{u}_0$. We start the partitioning by splitting this system along the following marked lines

$$\mathbf{M}_B = \left(\begin{array}{c|cc} \mathbf{B}_{II}^{(1)} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{B}_{II}^{(2)} & \mathbf{B}_{I\Gamma}^{(2)} \\ \mathbf{B}_{\Gamma I}^{(1)} & \mathbf{B}_{\Gamma I}^{(2)} & \mathbf{B}_{\Gamma\Gamma}^{(2)} \end{array}\right), \quad \mathbf{N}_B = \left(\begin{array}{c|cc} \mathbf{0} & \mathbf{0} & \mathbf{B}_{I\Gamma}^{(1)} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}_{\Gamma\Gamma}^{(1)} \end{array}\right),$$

$\mathbf{M}_A$, $\mathbf{N}_A$ analogous, to create two separate iterations for the internal unknowns of Ω_1 and Ω_2 plus Γ. In this, we replace all off-diagonal terms by evaluations of suitable interpolants of the corresponding solutions. The iteration for the internal unknowns in Ω_1 is

$$\begin{aligned} &\left(\mathbf{B}_{II}^{(1)} + \Delta t \mathbf{A}_{II}^{(1)}\right) \mathbf{u}_{I,n+1}^{(1),(k+1)} \\ &= \mathbf{B}_{II}^{(1)} \mathbf{u}_{I,n}^{(1),(k+1)} - \left(\mathbf{B}_{I\Gamma}^{(1)} + \Delta t_n \mathbf{A}_{I\Gamma}^{(1)}\right) \mathcal{I}(\underline{\mathbf{u}}_\Gamma^{(k)})(t_{n+1}) + \mathbf{B}_{I\Gamma}^{(1)} \mathcal{I}(\underline{\mathbf{u}}_\Gamma^{(k)})(t_n). \end{aligned}$$

Here we replaced $\mathbf{u}_{\Gamma,n}^{(k)}$ and $\mathbf{u}_{\Gamma,n+1}^{(k)}$ by evaluations of the interpolant $\mathcal{I}(\underline{\mathbf{u}}_\Gamma^{(k)})$. Analogous replacements for the iteration corresponding to the internal unknowns of Ω_2 and Γ yield:

$$\begin{aligned} &\begin{pmatrix} \mathbf{B}_{II}^{(2)} + \Delta t \mathbf{A}_{II}^{(2)} & \mathbf{B}_{I\Gamma}^{(2)} + \Delta t \mathbf{A}_{I\Gamma}^{(2)} \\ \mathbf{B}_{\Gamma I}^{(2)} + \Delta t \mathbf{A}_{\Gamma I}^{(2)} & \mathbf{B}_{\Gamma\Gamma}^{(2)} + \Delta t \mathbf{A}_{\Gamma\Gamma}^{(2)} \end{pmatrix} \begin{pmatrix} \mathbf{u}_{I,n+1}^{(2),(k+1)} \\ \mathbf{u}_{\Gamma,n+1}^{(k+1)} \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{B}_{II}^{(2)} & \mathbf{B}_{I\Gamma}^{(2)} \\ \mathbf{B}_{\Gamma I}^{(2)} & \mathbf{B}_{\Gamma\Gamma}^{(2)} \end{pmatrix} \begin{pmatrix} \mathbf{u}_{I,n}^{(2),(k+1)} \\ \mathbf{u}_{\Gamma,n}^{(k+1)} \end{pmatrix} + \Delta t \begin{pmatrix} \mathbf{0} \\ -\mathcal{I}(\underline{\mathbf{q}}^{(k+1)})(t_{n+1}) \end{pmatrix}. \end{aligned}$$

Here, the interpolated heat-flux $\mathcal{I}(\underline{\mathbf{q}}^{(k+1)})$ is based on the discrete heat fluxes

$$\begin{aligned} \mathbf{q}_{n+1}^{(k+1)} = &\left(\mathbf{B}_{\Gamma I}^{(1)} / \Delta t_n + \mathbf{A}_{\Gamma I}^{(1)}\right) \mathbf{u}_{I,n+1}^{(1),(k+1)} - \mathbf{B}_{\Gamma I}^{(1)} / \Delta t_n \mathbf{u}_{I,n}^{(1),(k+1)} \\ &+ \left(\mathbf{B}_{\Gamma\Gamma}^{(1)} / \Delta t_n + \mathbf{A}_{\Gamma\Gamma}^{(1)}\right) \mathcal{I}(\underline{\mathbf{u}}_\Gamma^{(k)})(t_{n+1}) - \mathbf{B}_{\Gamma\Gamma}^{(1)} / \Delta t_n \mathcal{I}(\underline{\mathbf{u}}_\Gamma^{(k)})(t_n). \end{aligned}$$

which is a discretization of (5.7), where the derivatives are approximated by backward differences, as defined by the implicit Euler scheme. The computation of the discrete heat-flux uses evaluations of the interpolant $\mathcal{I}(\underline{\mathbf{u}}_\Gamma^{(k)})$ rather than the discrete solutions, since heat-fluxes and interface temperatures are computed on potentially different time-grids.

The resulting partitioned method exchanges information in the form of the interface temperatures $\underline{\mathbf{u}}_\Gamma^{(k)}$ and the heat flux $\underline{\mathbf{q}}^{(k+1)}$. We require an initial heat-flux $\mathbf{q}_0^{(k+1)}$ for interpolation, which we obtain by (5.7), approximating derivatives using forward differences. See Algorithm 1 in Paper III for a pseudocode of this method.

5.2.2 SDIRK2

See Section 6.2 in Paper III for the definition of the SDIRK2 scheme. Applying it to (5.3) yields the iteration

$$\begin{pmatrix} \mathbf{B} + a\Delta t\,\mathbf{A} & \mathbf{0} \\ -\beta\mathbf{B} & \mathbf{B} + a\Delta t\,\mathbf{A} \end{pmatrix} \begin{pmatrix} \mathbf{U}_1 \\ \mathbf{u}_{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{B}\mathbf{u}_n \\ (1-\beta)\mathbf{B}\mathbf{u}_n \end{pmatrix},$$

with $a = 1 - \frac{1}{2}\sqrt{2}$, $\beta = (1-a)/a$. Here $\mathbf{U}_1$ is the stage solution. We apply the splittings (5.6) to all blocks to construct a fully discrete DNWR SDIRK2 method as follows

$$\begin{aligned} &\begin{pmatrix} \mathbf{M}_B + a\Delta t\,\mathbf{M}_A & \mathbf{0} \\ -\beta\mathbf{M}_B & \mathbf{M}_B + a\Delta t\,\mathbf{M}_A \end{pmatrix} \begin{pmatrix} \mathbf{U}_1^{(k+1)} \\ \mathbf{u}_{n+1}^{(k+1)} \end{pmatrix} - \begin{pmatrix} \mathbf{M}_B\mathbf{u}_n^{(k+1)} \\ (1-\beta)\mathbf{M}_B\mathbf{u}_n^{(k+1)} \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{N}_B + a\Delta t\,\mathbf{N}_A & \mathbf{0} \\ -\beta\mathbf{N}_B & \mathbf{N}_B + a\Delta t\,\mathbf{N}_A \end{pmatrix} \begin{pmatrix} \mathbf{U}_1^{(k)} \\ \mathbf{u}_{n+1}^{(k)} \end{pmatrix} - \begin{pmatrix} \mathbf{N}_B\mathbf{u}_n^{(k)} \\ (1-\beta)\mathbf{N}_B\mathbf{u}_n^{(k)} \end{pmatrix}, \end{aligned}$$

with $\mathbf{u}_0^{(k+1)} = \mathbf{u}_0$. This method is an iteration on both the time-point solutions $\mathbf{u}_{n+1}^{(k)}$ and the stage solutions $\mathbf{U}_1^{(k)}$. Partitioning this method as shown for the implicit Euler scheme then yields an iteration on the interface temperatures and heat fluxes for both the stage values and time-points. We denote the stage interface temperatures by $\underline{\mathbf{U}}_{1,\Gamma}^{(k)}$, the stage heat fluxes by $\underline{\mathbf{q}}_1^{(k)}$, and the respective time-point solutions by $\underline{\mathbf{u}}_\Gamma^{(k)}$ and $\underline{\mathbf{q}}_2^{(k)}$. Note that we require interpolants for each of these 4 solutions.

The SDIRK2 stage solution is equivalent to an implicit Euler step with stepsize $a\Delta t_n$ and thus first order accurate. With sufficiently accurate interpolation of the second order accurate time-point solutions, the interpolants of the stage solution and stage flux appear redundant. Thus, we consider the following replacements:

$$\mathcal{I}(\underline{\mathbf{U}}_{1,\Gamma}^{(k)}) \leftarrow \mathcal{I}(\underline{\mathbf{u}}_\Gamma^{(k)}) \quad \text{and} \quad \mathcal{I}(\underline{\mathbf{q}}_1^{(k+1)}) \leftarrow \mathcal{I}(\underline{\mathbf{q}}_2^{(k+1)}), \tag{5.8}$$

i.e., replacing evaluations of the stage solution resp. stage flux interpolants by the corresponding interpolants based on the time-point solutions. This yields a total of 4 methods by applying none, one or both of the above replacements. *The fully-discrete DNWR SDIRK2 method in Paper III is the one obtained by only replacing* $\mathcal{I}(\underline{\mathbf{U}}_{1,\Gamma}^{(k)})$.

First, we consider convergence of these 4 fully-discrete DNWR methods for $TOL_{WR} \to 0$ and a fixed Δt. We want the methods to converge to the solution of the fully-discrete monolithic PDE. By construction, only the *"no repl."* method should achieve this, since any replacements introduce an additional error in dependence of Δt. Figure 5.4 shows exactly this result.

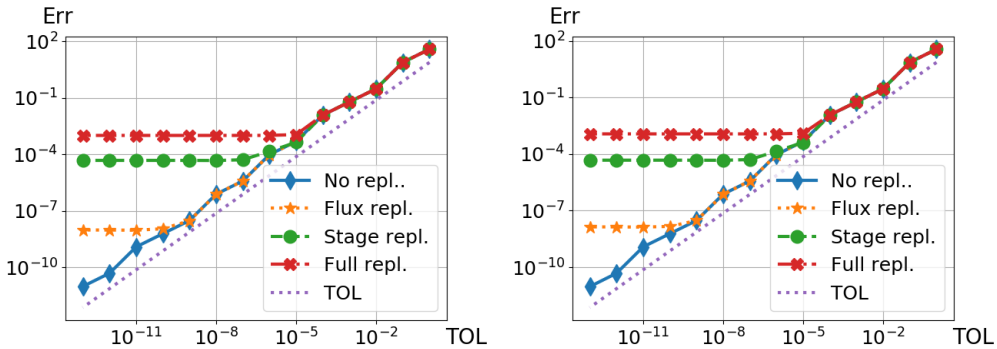

Figure 5.4: WR splitting error (discrete $\mathcal{L}^2$ norm on Ω) w.r.t. the fully-discrete DNWR SDIRK2 method for $N = 100$ timesteps and using $\Theta = 0.6$ for relaxation, c.f. Section 5.3. **Left**: 1D, $\Delta x = 1/51$. **Right**: 2D, $\Delta x = 1/33$.

We now consider convergence of the given fully-discrete DNWR methods to the solution of the semi-discrete PDE, c.f. Figure 5.3, for $TOL_{WR}, \Delta t \to 0$. To this end, we measure the error of the fully-discrete DNWR SDIRK2 methods w.r.t. the solution of the semi-discrete PDE, by using $TOL_{WR} = 10^{-13}$ and a monolithic reference solution with sufficiently small Δt. Figure 5.5 shows that using both replacements (5.8) results in order reduction.

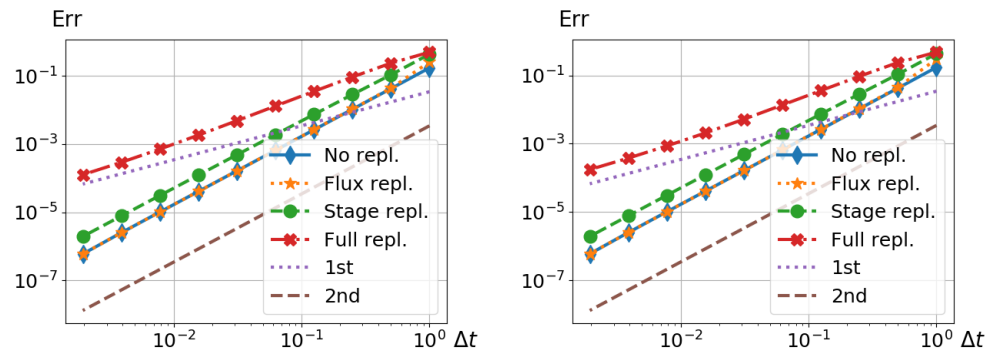

Figure 5.5: Combined time-discretization and WR splitting error of the fully-discrete DNWR SDIRK2 methods with $TOL_{WR} = 10^{-13}$ and $\Theta = 0.6$. **Left**: 1D, $\Delta x = 1/51$. **Right**: 2D, $\Delta x = 1/33$.

The "stage repl." method presented in Paper III is second order convergent. While it is evidently not the one with the smallest error, it is more robust than the other second order convergent variants. That is, observed convergence rates in the multirate and time-adaptive setting are faster. This is discussed in more detail at the end of Section 5.3.

5.2.3 Multi-rate and adaptive partitioned time-integration

After the partitioning, it is straight-forward to use non-matching time-grids, including time-adaptivity, as presented in Chapter 2, when solving the subproblems. Paper III shows details on the controllers, Δt_0 and how to choose TOL in a timestep-controller (2.10) w.r.t. TOL_{WR} in the WR termination criterion, c.f. (4.4). Algorithm 2 in Paper III shows a pseudocode for the time-adaptive DNWR SDIRK2 method.

5.3 Optimal relaxation parameter

In the DNWR method we perform relaxation with $\Theta \in \mathbb{R}$ on $\mathbf{u}_\Gamma^{(k)}$ only, i.e.,

$$\mathbf{u}_{\Gamma,n+1}^{(k+1)} \leftarrow \Theta \mathbf{u}_{\Gamma,n+1}^{(k+1)} + (1-\Theta)\mathbf{u}_{\Gamma,n+1}^{(k)}$$

As shown in Section 4.4, one only needs to consider a single timestep to determine Θ_{opt}. Using the general space discretization from Section 5.1.1, one can determine the iteration matrix w.r.t. $\mathbf{u}_\Gamma^{(k)}$ using the Schur-complement [82, pp.5], which is

$$\mathbf{u}_{\Gamma,n+1}^{(k+1)} = \left(-\Theta \underbrace{\mathbf{S}^{(2)^{-1}}\mathbf{S}^{(1)}}_{=:\mathbf{\Sigma}} + (1-\Theta)\mathbf{I} \right) \mathbf{u}_{\Gamma,n+1}^{(k)} + \boldsymbol{\psi},$$

with $\mathbf{S}^{(m)}$ defined by

$$\begin{aligned}\mathbf{S}^{(m)} := &\left(\mathbf{B}_{\Gamma\Gamma}^{(m)} + \Delta t\, \mathbf{A}_{\Gamma\Gamma}^{(m)}\right) \\ &+ \left(\mathbf{B}_{\Gamma I}^{(m)} + \Delta t\, \mathbf{A}_{\Gamma I}^{(m)}\right) \left(\mathbf{B}_{II}^{(m)} + \Delta t\, \mathbf{A}_{II}^{(m)}\right)^{-1} \left(\mathbf{B}_{I\Gamma}^{(m)} + \Delta t\, \mathbf{A}_{I\Gamma}^{(m)}\right).\end{aligned}$$

In the 1D case $\mathbf{\Sigma}$ is a scalar, since the interface is a single point. Then, optimal relaxation is given by

$$\Theta_{opt}^{discr} = \frac{1}{|1+\mathbf{\Sigma}|}.$$

For linear finite elements on a uniform space discretization in 1D and an implicit Euler time-discretization, an analytical expression for $\mathbf{\Sigma}$ was computed in [54].

This derivation most notably requires the inverse of the tridiagonal matrix $\mathbf{B}_{II}^{(m)} + \Delta t\, \mathbf{A}_{II}^{(m)}$. While this inverse is both large and dense, both matrices multiplied with the inverse are mostly zero. Using the known Toeplitz structure of $\mathbf{B}_{II}^{(m)} + \Delta t\, \mathbf{A}_{II}^{(m)}$, one can exactly formulate its inverse via an Eigendecomposition, yielding an exact expression for $\mathbf{S}^{(m)}$. See [54, 55] for the detailed steps and Paper III for the resulting expression. In particular,

this result is consistent with analysis of the 1D continuous DNWR method for constant material coefficients [24, 45].

In the multirate case, i.e., equidistant but non-matching time-grids, we use Θ_{opt}^{discr} based on the larger stepsize. This strategy was determined on a heuristic basis, see Paper III for the experiments. In the time-adaptive case, we compute the average stepsize on each grid and then use Θ_{opt}^{discr} based on the larger average stepsize. Here, Θ_{opt}^{discr} needs to be re-computed with every iteration, since time-grids may change.

In the previously discussed fully-discrete DNWR SDIRK2 methods, one additionally needs to perform relaxation on the stage interface temperature $\mathbf{U}_{1,\Gamma}^{(k)}$. The computation of this stage solution is equivalent to a timestep of the implicit Euler scheme with the stepsize $\alpha\,\Delta t_n$ and we thus know the optimal relaxation parameter.

However, relaxation in the multirate or time-adaptive case no longer has an exact analytical basis. Consequently, convergence acceleration for the stage solution and the time-point solution can differ. For the SDIRK2 methods utilizing interpolants of the stage solution, this resulted in worse observed convergence rates in the multirate case and failure of the time-adaptive method. There, a worse convergence acceleration for the stage solution $\underline{\mathbf{U}}_{1,\Gamma}^{(k)}$ lead to the local error estimate becoming independent of Δt_n. This causes the stepsize control to fail since the local errors do not vanish for $\Delta t_n \to 0$. The method presented in Paper III avoids this problem, replacing $\mathcal{I}(\underline{\mathbf{U}}_{1,\Gamma}^{(k)})$ by $\mathcal{I}(\underline{\mathbf{u}}_{\Gamma}^{(k)})$, only requiring relaxation for $\underline{\mathbf{u}}_{\Gamma}^{(k)}$.

5.4 Summary of numerical results

See Section 9 in Paper III for details on the numerical experiments and results. The numerical results for material combinations of air, water and steel can be summarized as follows:

All presented methods (DNWR/NNWR + implicit Euler/SDIRK2 + adaptivity) are shown to retain their orders of convergence w.r.t. time-integration, i.e., for $k \to \infty$ and $\Delta t \to 0$ (resp. $TOL \to 0$ for timestep-adaptivity).

Now, we consider the convergence rates of the WR. Since Θ_{opt} is derived for 1D and implicit Euler, the important question is the *robustness* of Θ_{opt}. I.e., if the *observed convergence rates* match the theoretical ones, and how fast the methods converge in the 2D, SDIRK2 and the multirate resp. time-adaptive setting. The main results are shown in Figures 5.6 and 5.7, where we measure the convergences rate via the average update reduction rate.

The observed 1D convergences rates align extremely well with the expected rates for both DNWR and NNWR. Observed 2D convergence rates for DNWR are generally slower than

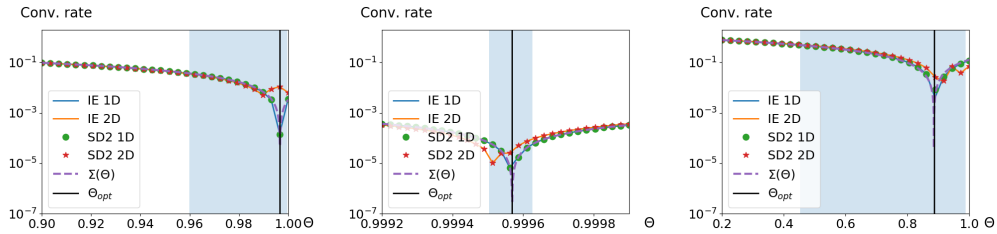

Figure 5.6: Left: Air-water. Centre: Air-steel. Right: Water-steel. Observed convergence rates for DNWR algorithm, see Paper III for details.

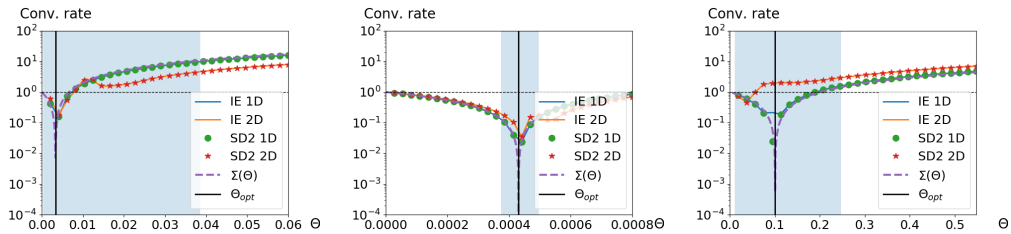

Figure 5.7: Left: Air-water. Centre: Air-steel. Right: Water-steel. Observed convergence rates for DNWR algorithm, see Paper III for details.

the 1D ones, but Θ_{opt} is shown to be near optimal. The multirate and singlerate (matching stepsizes) convergence rates turned out to be indistinguishable. Similarly implicit Euler and SDIRK2 convergence rates are almost identical. *Here it is important to note that DNWR is convergent regardless of the choice of* Θ *in our examples*. Furthermore, the results appear to extend well into non-square geometries, as shown for $|\Omega_1| = 9|\Omega_2|$, see Paper III .

The observed NNWR 2D convergence rates align well with the 1D ones in the air-water and air-steel test-cases, but the method diverges with Θ_{opt} in the water-steel case. Unlike for DNWR, the NNWR method requires a well chosen Θ to be convergent and the range of convergent Θ is extremely narrow. In particular, it can be smaller than the range defined by the spatial and temporal limits of the analytical expression for Θ_{opt}^{discr}, which is highlighted in blue in Figures 5.6 and 5.7.

On average, the observed DNWR convergence rates are at least one order of magnitude faster then the corresponding NNWR rates. These experiments show that DNWR is notably more robust than the NNWR method.

Finally, we compared performance of the multirate and time-adaptive DNWR methods, in terms of error vs. number of timesteps. See Figure 5.8 for the results in one test case, which shows the adaptive method is more efficient.

The time-adaptive method is generally favored, due to the difficulty of finding suitable stepsize ratios and suitable TOL_{WR} for the termination criterion in the multirate setting.

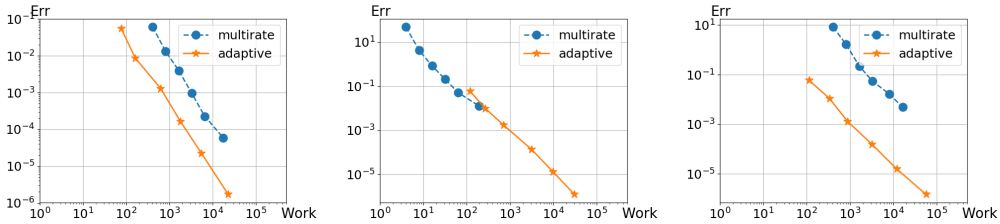

Figure 5.8: Left: Air-water. Centre: Air-steel. Right: Water-steel. DNWR work over error comparison for 2D test case, see Paper III for details on the computational setup.

With the time-adaptive method, both the termination criterion and stepsizes are controlled via a single tolerance and suitable stepsizes are determined automatically. In our comparison we based the multirate stepsizes on the material parameters, aiming for comparable *CFL* numbers. Tests with two different initial conditions result in notably different stepsize ratios from the time-adaptive method, showing the adaptive approach to be more robust.

Chapter 6

Waveform Relaxation with asynchronous time-integration

We construct a novel and inherently parallel WR method with more information exchange than Jacobi WR. Our ansatz to increase communication is to exchange the results of each timestep directly after computation. Any new information is directly incorporated by updating the interpolants, affecting their subsequent evaluations. This increases the information exchange and thus enhances convergence rates. We resolve the communication using asynchronous *One-sided-communication* in `MPI`.

Overview over chapter

See Section 4 in Paper IV for a brief introduction to One-sided communication using `MPI`, which is asynchronous. Here, we present the principle idea and analytical description of our new WR method. Additionally, we present the convergence results, which generalize Theorems 4.1 and 4.3. Paper IV shows the algorithm for practical relaxation as well as the numerical experiments, which feature the `DUNE-FEniCS` coupling discussed in Section 4.6.

6.1 Waveform relaxation with asynchronous communication

We use interpolation with evaluation at run-time as described in Section 4.2. I.e., we define the interpolants once using fixed data-structures corresponding to $\underline{\mathbf{v}}_*$ and $\underline{\mathbf{w}}_*$. Then, any updates to the interpolant data directly affect subsequent evaluations. By exposing $\underline{\mathbf{v}}_*$ and $\underline{\mathbf{w}}_*$ via `MPI_Window` objects, one can remotely update interpolants.

We start with the parallel Algorithm 4.4 and modify it to increase communication. We move relaxation and communication to the time-step level, i.e., into the time-stepping loop. We asynchronously communicate new time-point solutions, remotely updating the corresponding values in $\underline{\mathbf{v}}_*$ resp. $\underline{\mathbf{w}}_*$ on the other process, using `MPI_Put`.

Algorithm 6.1 shows the pseudocode for two coupled problems, with a different number of timesteps $N_v \neq N_w$ for each subproblem.

Pseudocode: WR with asynchronous time-integration

	Process 0 (**p**0)		**Process** 1 (**p**1)
1:	$\underline{\mathbf{w}}^{(0)}$ discrete initial guess		$\underline{\mathbf{v}}^{(0)}$ discrete initial guess
2:	Initialize $\underline{\mathbf{w}}_* \leftarrow \underline{\mathbf{w}}^{(0)}$ and $\mathcal{I}(\underline{\mathbf{w}}_*)$		Initialize $\underline{\mathbf{v}}_* \leftarrow \underline{\mathbf{v}}^{(0)}$ and $\mathcal{I}(\underline{\mathbf{v}}_*)$
3:	Expose $\underline{\mathbf{w}}_*$ via `MPI_Window`		Expose $\underline{\mathbf{v}}_*$ via `MPI_Window`
4:	**for** $k = 0, \ldots, k_{\max} - 1$ **do**		**for** $k = 0, \ldots, k_{\max} - 1$ **do**
5:	**for** $n = 1, \ldots, N_v$ **do**		**for** $n = 1, \ldots, N_w$ **do**
6:	$\hat{\mathbf{v}}_n^{(k+1)} \leftarrow$ Solve* (4.5a)		$\hat{\mathbf{w}}_n^{(k+1)} \leftarrow$ Solve* (4.5b)
7:	$\mathbf{v}_n^{(k+1)} \leftarrow$ Relaxation (4.6a)		$\mathbf{w}_n^{(k+1)} \leftarrow$ Relaxation (4.6b)
8:	$\mathbf{v}_n^{(k+1)} \rightarrow$ `Put` to $\underline{\mathbf{v}}_*$ on **p**1		$\mathbf{w}_n^{(k+1)} \rightarrow$ `Put` to $\underline{\mathbf{w}}_*$ on **p**0
9:	**end for**		**end for**
10:	Sync. + Termination check		Sync. + Termination check
11:	**end for**		**end for**

Algorithm 6.1: New proposed method using asynchronous communication during time-integration. Note that obtaining the initial guesses involves communication. Solve* denotes a single discrete timestep in solving (4.5a) resp. (4.5b), with the interpolants defined in Line 2 in the right-hand sides.

Our new method is defined by

$$(\underline{\mathbf{v}}_*^{(k)})_n = \begin{cases} \mathbf{v}_n^{(k+1)}, & \text{if available} \\ \mathbf{v}_n^{(k)}, & \text{else} \end{cases}, \tag{6.1}$$

$\underline{\mathbf{w}}_*^{(k)}$ analogous. Since interpolants are remotely updated, availability is determined by the present data when evaluating the interpolant. Unlike with GS and Jacobi WR, our $\underline{\mathbf{v}}_*^{(k)}$ and $\underline{\mathbf{w}}_*^{(k)}$ can vary for different timesteps and with k, since asynchronous communication is not deterministic. Due to remote updates, evaluations of the interpolants at the same time-point but at different real-life times can differ. The analogous continuous WR method is (4.1) with general $\underline{\mathbf{v}}_*^{(k)}$ and $\underline{\mathbf{w}}_*^{(k)}$ varying in both time and with k.

It is possible that we obtain Jacobi or GS WR. Since optimal relaxation matrices can notably differ for Jacobi and GS WR, constant relaxation is unlike to achieve optimal convergence

acceleration in our new method. We instead consider relaxation varying with t and k, i.e.,

$$\begin{aligned}\mathbf{v}^{(k+1)}(t) &= \mathbf{v}^{(k)}(t) + \boldsymbol{\Theta}_v^{(k+1)}(t)\left(\hat{\mathbf{v}}^{(k+1)}(t) - \mathbf{v}^{(k)}(t)\right),\\ \mathbf{w}^{(k+1)}(t) &= \mathbf{w}^{(k)}(t) + \boldsymbol{\Theta}_w^{(k+1)}(t)\left(\hat{\mathbf{w}}^{(k+1)}(t) - \mathbf{w}^{(k)}(t)\right),\end{aligned} \tag{6.2}$$

where $\boldsymbol{\Theta}_v^{(k+1)}(t)$ and $\boldsymbol{\Theta}_w^{(k+1)}(t)$ are non-singular diagonal matrices. In Section 7 of Paper IV we present an improved algorithm with variable relaxation based on the realized communication. There, one needs to deduce $\underline{\mathbf{v}}_*^{(k)}$ and $\underline{\mathbf{w}}_*^{(k)}$, in each timestep, from the realized communication, to then pick appropriate relaxation for every time-point solution.

6.2 Convergence analysis

Similar to Section 4.3, we analyze convergence in the linear setting for problem (4.7). The fixed $\underline{\mathbf{v}}_*^{(k)}$ and $\underline{\mathbf{w}}_*^{(k)}$ in Chapter 4 correspond to the iteration (4.9) with fixed splittings (4.8). Here, the variable $\underline{\mathbf{v}}_*^{(k)}$ and $\underline{\mathbf{w}}_*^{(k)}$ analogously correspond to the iteration

$$\begin{aligned}&\mathbf{M}_B^{(k+1)}(t)\dot{\mathbf{u}}^{(k+1)}(t) + \mathbf{M}_A^{(k+1)}(t)\mathbf{u}^{(k+1)}(t)\\ &= \mathbf{N}_B^{(k+1)}(t)\dot{\mathbf{u}}^{(k)}(t) + \mathbf{N}_A^{(k+1)}(t)\mathbf{u}^{(k)}(t) + \mathbf{f}(t), \quad \mathbf{u}^{(k+1)}(0) = \mathbf{u}_0, \quad t \in [0, T_f].\end{aligned} \tag{6.3}$$

with the variable splittings

$$\mathbf{B} = \mathbf{M}_B^{(k)}(t) - \mathbf{N}_B^{(k)}(t), \quad \mathbf{A} = \mathbf{M}_A^{(k)}(t) - \mathbf{N}_A^{(k)}(t), \quad \mathbf{M}_B^{(k)}(t) \text{ nonsingular}, \tag{6.4}$$

for all $t \in [0, T_f]$, $k > 0$. Similar to the constant splitting case, this formulation does include variable relaxation (6.2). We omit the dependency on the relaxation matrices for readability.

6.2.1 Time-discrete WR with asynchronous communication

We describe Algorithm 6.1 by an iteration analogous to (4.17), using convergent and zero-stable LMM with constant and matching stepsizes. Here, the splitting matrices (4.8) can differ for each ℓ, n and k. This yields the iteration

$$\begin{aligned}&\sum_{\ell=0}^{m}\left(a_\ell \mathbf{M}_{B,n,\ell}^{(k+1)} + b_\ell \Delta t \mathbf{M}_{A,n,\ell}^{(k+1)}\right)\mathbf{u}_{n+\ell}^{(k+1)}\\ =&\sum_{\ell=0}^{m}\left(a_\ell \mathbf{N}_{B,n,\ell}^{(k+1)} + b_\ell \Delta t \mathbf{N}_{A,n,\ell}^{(k+1)}\right)\mathbf{u}_{n+\ell}^{(k)} + b_\ell \Delta t \mathbf{f}(t_{n+\ell}).\end{aligned} \tag{6.5}$$

Here, the concrete matrices $\mathbf{M}_{B,n,\ell}^{(k+1)}$ and $\mathbf{M}_{A,n,\ell}^{(k+1)}$ are, for each ℓ, n and k, determined by $\underline{\mathbf{v}}_*^{(k)}$ and $\underline{\mathbf{w}}_*^{(k)}$, as emerging from the realized communication in e.g., Algorithm 6.1, including relaxation. These matrices fulfill the splitting property $\mathbf{B} = \mathbf{M}_{B,n,\ell}^{(k+1)} - \mathbf{N}_{B,n,\ell}^{(k+1)}$, $\mathbf{A}$ analogous, by which (4.16) is a fixed point to (6.5).

The iteration (6.5) differs from a time-discretization of (6.3) in that e.g., both $\mathbf{M}_{B,n,\ell}^{(k+1)}$ and $\mathbf{M}_{B,n+1,\ell-1}^{(k+1)}$ correspond to $\mathbf{M}_B^{(k+1)}(t_{n+\ell})$, but may differ. This is due the remote update of the interpolant data, by which evaluations of the same time-point, but at different real-life times can differ.

By taking the difference between (4.16) and (6.5), we see that the discrete WR error (4.18) fulfills

$$\sum_{\ell=0}^{m} \mathbf{C}_{n,\ell}^{(k+1)} \mathbf{e}_{n+\ell}^{(k+1)} = \sum_{\ell=0}^{m} \mathbf{D}_{n,\ell}^{(k+1)} \mathbf{e}_{n+\ell}^{(k)}, \tag{6.6}$$

with

$$\mathbf{C}_{n,\ell}^{(k)} := a_\ell \mathbf{M}_{B,n,\ell}^{(k)} + b_\ell \, \Delta t \mathbf{M}_{A,n,\ell}^{(k)}, \quad \mathbf{D}_{n,\ell}^{(k)} := a_\ell \mathbf{N}_{B,n,\ell}^{(k)} + b_\ell \, \Delta t \mathbf{N}_{A,n,\ell}^{(k)}.$$

The starting values $\mathbf{u}_\ell^{(k+1)}$ define the starting errors $\mathbf{e}_\ell^{(k+1)}$, $\ell = 0, \ldots, m-1$.

Theorem 6.1. *Let the splittings*

$$\mathbf{B} = \mathbf{M}_{B,n,\ell}^{(k)} - \mathbf{N}_{B,n,\ell}^{(k)} \quad \textit{and} \quad \mathbf{A} = \mathbf{M}_{A,n,\ell}^{(k)} - \mathbf{N}_{A,n,\ell}^{(k)}, \quad \mathbf{M}_{B,n,\ell}^{(k)} \textit{ nonsingular},$$

with

$$\mathbf{C}_{n,m}^{(k)} \textit{ nonsingular}, \quad \| {\mathbf{C}_{n,m}^{(k)}}^{-1} \mathbf{D}_{n,m}^{(k)} \| < 1, \quad n = m, \ldots, N \tag{6.7}$$

and an initial guess

$$\underline{\mathbf{e}}^{(0)} := \left({\mathbf{e}_m^{(0)}}^T, \ldots, {\mathbf{e}_N^{(0)}}^T \right)^T \in \mathbb{R}^{d(N-m+1)}$$

be given. Then, the iteration (6.5) *converges to the solution of* (4.16).

Proof. This theorem requires a proof that is structurally different to the one of Theorem 4.3, since the iteration matrix varies with k. Consider the general iteration $\mathbf{e}^{(k+1)} = \mathbf{A}^{(k)} \mathbf{e}^{(k)}$. Here, $\rho(\mathbf{A}^{(k)}) < 1$ for all k, is not sufficient to guarantee $\lim_{k\to\infty} \|\mathbf{e}^{(k)}\| = 0$.

The proof of this theorem is based on a contraction argument utilizing the specific structure of the iteration matrices. See Theorem 6.1 in Paper IV for the full proof. □

For constant splittings, this result coincides with Theorem 4.3. The assumption of $\mathbf{C}_{n,m}^{(k)}$ nonsingular in (6.7) is a solvability assumption on the occurring linear systems in (6.6).

Remark 6.2. *Consider Algorithm 6.1 with matching time-grids, i.e., $N = N_v = N_w$ and constant relaxation. Then, the splitting matrices $\mathbf{M}^{(k)}_{B,n,\ell}$ resp. $\mathbf{M}^{(k)}_{A,n,\ell}$ match those of Jacobi or GS WR for each ℓ, n, k, due to matching $(\underline{\mathbf{v}}^{(k)}_*)_n$ and $(\underline{\mathbf{w}}^{(k)}_*)_n$, for each n, see* (6.1)*. Thus, one only needs to consider three distinct cases for the matrices* (6.7)*. Each one of these is a convergence requirement for either Jacobi or GS WR. Consequently, time-discrete convergence of Jacobi and GS WR (in either ordering of* (1.1)*) means* (6.7) *is met.*

6.2.2 Continuous WR with asynchronous communication

In the previous section, we considered the convergence of the fully discrete WR iteration (6.5) for Δt fixed for $k \to \infty$, which gives (4.16). Here, we instead discuss the convergence of the continuous iteration (6.3) for $k \to \infty$. Before doing so, we would like to point out that we cannot guarantee that we obtain (6.3) from (6.5) in the limit $\Delta t \to 0$. The reason is that the splittings chosen and thus the matrices $\mathbf{M}^{(k+1)}_{B,n,\ell}$ resp. $\mathbf{M}^{(k+1)}_{A,n,\ell}$ in (6.5) can change with every time step. This would yield $\mathbf{M}^{(k)}_B$, $\mathbf{M}^{(k)}_A$ discontinuous everywhere in the limit and (6.3) would not be well defined.

The typical scenario for (6.5), as implemented via Algorithm 6.1, is that splittings match those of Jacobi WR until one subsolver is at least one timestep ahead of another subsolver. From then on, the splitting matrices match those of GS WR. I.e., for a given k, $\mathbf{M}^{(k)}_B$ and $\mathbf{M}^{(k)}_A$ are piece-wise constant, with a single discontinuity.

We thus assume that the limit has only a finite number of jumps and consider (6.3) in a piece-wise sense with piece-wise Lipschitz-continuous data. This guarantees existence of a piece-wise solution of (6.3) for all $k > 0$. Additionally, we assume that splitting matrices corresponding to the same time-point, in the same iteration, are identical. E.g., in (6.5) both $\mathbf{M}^{(k+1)}_{B,n,\ell}$ and $\mathbf{M}^{(k+1)}_{B,n+1,\ell-1}$ correspond to $t_{n+\ell}$. This can be guaranteed by implementation, storing interpolant evaluations. Now we analyze the convergence properties of (6.3) under these assumptions.

Consider (4.7) in a piece-wise sense. Its solution with $\mathbf{A}$, $\mathbf{B}$, $\mathbf{f}$ time-dependent and piece-wise Lipschitz-continuous is

$$\mathbf{u}(t) = \mathrm{e}^{-C(t)} \left(\mathbf{u}_0 + \int_0^t \mathrm{e}^{C(s)} \mathbf{B}^{-1}(s) \mathbf{f}(s) \mathrm{d}s \right), \tag{6.8}$$

where

$$C(t) = \int_0^t \mathbf{B}^{-1}(s) \mathbf{A}(s) \mathrm{d}s.$$

We apply this solution formula to (6.3). Replacing $\dot{\mathbf{u}}^{(k)}$ via integration by parts and performing lengthy, but straight-forward rearrangements similar to the fixed splitting case in

Section 4.3.1, yield the solution

$$\mathbf{u}^{(k+1)}(t) = \mathbf{K}^{(k+1)}(t)\mathbf{u}^{(k)}(t) + \int_0^t \boldsymbol{\mathcal{K}}_c^{(k+1)}(s)\mathbf{u}^{(k)}(s)\mathrm{d}s + \boldsymbol{\varphi}^{(k+1)}(t), \tag{6.9}$$

with

$$\begin{aligned}
\mathbf{K}^{(k)}(t) &= \mathbf{M}_B^{(k)^{-1}}(t)\mathbf{N}_B^{(k)}(t),\\
\mathbf{C}^{(k)}(t) &= \int_0^t \mathbf{M}_B^{(k)^{-1}}(s)\mathbf{M}_A^{(k)}(s)\mathrm{d}s,\\
\boldsymbol{\mathcal{K}}_c^{(k)}(t) &= \mathrm{e}^{\mathbf{C}^{(k)}(s)-\mathbf{C}^{(k)}(t)}\left(\mathbf{M}_B^{(k)^{-1}}(s)\mathbf{N}_A(s) - \frac{\mathrm{d}}{\mathrm{d}s}\left(\mathrm{e}^{\mathbf{C}^{(k)}(s)}\right)\mathbf{K}^{(k)}(s) - \frac{\mathrm{d}}{\mathrm{d}s}\mathbf{K}^{(k)}(s)\right),\\
\boldsymbol{\varphi}^{(k)}(t) &= \mathrm{e}^{-\mathbf{C}^{(k)}(t)}\left(\left(\mathbf{I}-\mathbf{K}^{(k)}(0)\right)\mathbf{u}_0 + \int_0^t \mathrm{e}^{\mathbf{C}^{(k)}(s)}\mathbf{M}_B^{(k)^{-1}}(s)\mathbf{f}(s)\mathrm{d}s\right),
\end{aligned} \tag{6.10}$$

where

$$\frac{\mathrm{d}}{\mathrm{d}s}\left(\mathrm{e}^{\mathbf{C}^{(k)}(s)}\right) = \int_0^1 \mathrm{e}^{\alpha\mathbf{C}^{(k)}(s)}\frac{\mathrm{d}\mathbf{C}^{(k)}(s)}{\mathrm{d}s}\mathrm{e}^{(1-\alpha)\mathbf{C}^{(k)}(s)}\mathrm{d}\alpha,$$

c.f. [88].

Taking the difference between (6.9) and (6.8) yields the following relation for the WR error (4.13):

$$\mathbf{e}^{(k+1)}(t) = \mathbf{K}^{(k+1)}(t)\mathbf{e}^{(k)}(t) + \int_0^t \boldsymbol{\mathcal{K}}_c^{(k+1)}(s)\mathbf{e}^{(k)}(s)\mathrm{d}s, \quad \mathbf{e}^{(k+1)}(0) = \mathbf{0}, \quad t \in [0, T_f].$$

In the following Theorem, we use the same function norms as in Theorem 4.1.

Theorem 6.3. *Let splittings* (6.4) *with* $\mathbf{M}_B^{(k)}$, $\mathbf{M}_A^{(k)}$ *and* $\mathbf{e}^{(0)}$ *piece-wise Lipschitz-continuous, and a finite set of discontinuities over all* $k > 0$ *be given. Then, the error* (4.13) *fulfills*

$$\|\mathbf{e}^{(k)}\|_{[0,t]} \leq \left(\sum_{j=0}^{k}\binom{k}{j}K^{\max}(t)^{k-j}\mathcal{K}_c^{\max}(t)^j\frac{t^j}{j!}\right)\|\mathbf{e}^{(0)}\|_{[0,t]},$$

where $K^{\max}(t) := \sup_{k\in\mathbb{N}}\|\mathbf{K}^{(k)}\|_{[0,t]}$ *and* $\mathcal{K}_c^{\max}(t) := \sup_{k\in\mathbb{N}}\|\boldsymbol{\mathcal{K}}_c^{(k)}\|_{[0,t]}$, *c.f.* (6.10).

Proof. See Theorem 6.2 in Paper IV . □

This result is consistent with constant splitting result in Theorem 4.1 and the time-discrete result of Theorem 6.1 for $\Delta t \to 0$, under the aforementioned assumptions. The asymptotic convergence rate for $k \to \infty$ is bounded by $\|K^{\max}\|_{[0,t]}$.

6.3 Summary of numerical results

See Paper IV for the full details on the experiment descriptions and numerical results. Here, we provide a brief summary.

We developed a novel parallel WR method, using asynchronous communication on the time-integration level. Its analytical description and convergence proofs, in the continuous and time-discrete setting for linear problems, generalizes existing WR theory. In Section 7 of Paper IV we present the algorithm for choosing optimal variable relaxation at runtime.

Our first test case is the problem from Chapter 5 resp. Paper III , here, on matching time-grids. This test case is conform with the setup in the time-discrete convergence analysis. In the variable relaxation algorithm we additionally require relaxation parameters for Jacobi WR and GS WR in the opposite ordering of the subproblems. We derive these using the established relationships between heat-flux and interface temperature from [54], which are also presented Paper III , Section 5.3.

The numerical results demonstrate that our new method is convergent, see Figure 6.1, numerically verifying Theorem 6.1. The convergence rates are slower than for GS WR, but faster than for Jacobi WR. Since our method is parallel, the resulting performance, in terms of error over (wall-clock) runtime, is slightly better than for the classical WR methods.

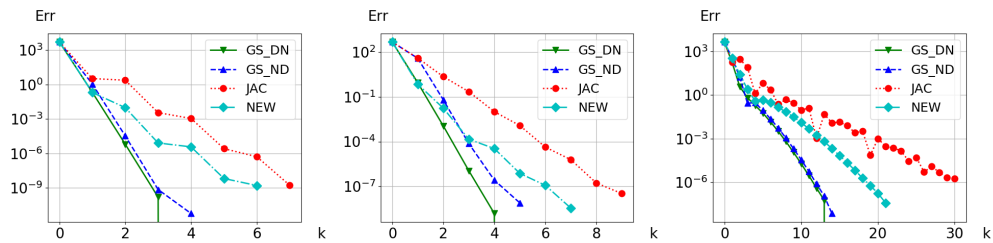

Figure 6.1: Left to right: Air-steel, air-water and water-steel. k marks the number of iterations until (4.4), with $TOL_{WR} = 10^{-10}$, is met. We measure the error of the interface temperature w.r.t. a lower tolerance reference solution.

The second experiment is a gas quenching test case in 2D, in which we model the cooling of a hot steel plate using pressurized air, see Figure 6.2. The solid steel plate is modeled by the nonlinear heat equation and the air is modelled via the compressible Euler equations. The subproblems are coupled via a Dirichlet-Neumann approach. The solid solver uses a finite element discretization implemented in `FEniCS` [41] and the fluid solver uses a 1st order finite volume discretization implemented in `DUNE` [6, 17]. While this test case is outside the scope of our time-discrete analysis, our methods shows rapid convergence of the iteration, and demonstrates the `DUNE-FEniCS` coupling.

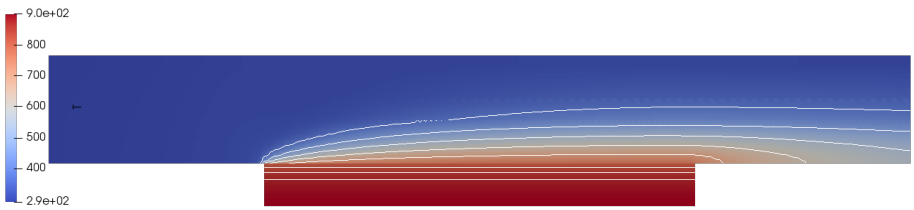

Figure 6.2: Gas quenching test case, cooling of a hot steel plate (bottom) with pressurized air.

Chapter 7

Conclusions and future work

7.1 Summary and conclusions

Goal oriented time-adaptivity

Similar to previously proposed methods, we derived a simple and easy to implement goal oriented local error estimator. Implementation comes with little to no extra work and is well-suited for partitioned approaches to coupled systems.

For the resulting goal oriented adaptive method we show convergence and determine convergence rates w.r.t. the error in the QoI. To obtain tolerance proportionality, sufficiently high order solutions in all quadrature evaluation points are needed. The proof is constructive and gives rise to a principle to formally show convergence for closely related stepsize controllers.

Performance of the goal oriented method deteriorates if local errors of underresolved processes accumulate in the zero-set of the density function and later shift into its image. Convection dominated problems are particularly prone to this type of behavior. It is less problematic for dissipative problems, where slow and global processes dominate the error.

Numerical experiments designed to test these guidelines show them to work well and confirm the results on convergence rates. The tests show that bad performance of the goal oriented method can be predicted and thus avoided. For dissipative test cases, the DWR method is notably more expensive and outperformed by local error based method. The goal oriented adaptive method shows good performance in most cases and significant speedups in some.

Waveform relaxation

We presented a implementation framework for WR, which enables partitioned coupling of PDE subsolvers with different space discretizations. In particular, we developed adapters to enable the coupling of subsolvers implemented in `DUNE` and `FEniCS`.

Dirichlet-Neumann Waveform Relaxation for heterogeneous coupled heat equations

We derived first and second order, multirate resp. time-adaptive DNWR methods for heterogeneous coupled linear heat equations. The optimal relaxation parameter Θ_{opt} for WR coincides with the one for the time-point relaxation DN iteration (one timestep per time-window). We experimentally show how to adapt Θ_{opt} in the multirate case. The observed convergence rates using an analytical Θ_{opt} for 1D implicit Euler are shown to be very robust, yielding fast convergence rates for a second order method and in 2D, for various material combinations and multirate settings on long time intervals.

The same tests for the related parallel NNWR method, using identical Dirichlet and Neumann subsolvers, using an analogous analytical Θ_{opt} for 1D implicit Euler, show a lack of robustness. Convergence rates are slower than for the DNWR method and the iteration diverges in some cases.

The time-adaptive DNWR method is experimentally shown to be favorable over the multirate DNWR method, i.e., constant but non-matching Δt, due ease of use and superior performance. The latter is due to the resulting stepsizes being more suitably chosen than those of the multirate DNWR method. Overall, we obtain a fast, robust, time-adaptive (on each domain), partitioned time-integration method for unsteady conjugate heat transfer.

Waveform relaxation with asynchronous time-integration

We developed a novel parallel WR method, using asynchronous communication on the time-integration level. We present an analytical description of our new method and convergence proofs in the continuous and time-discrete setting for linear problems. This generalizes existing WR theory.

We discuss algorithms for the necessary technical implementation using One-sided communication in `MPI`. This particularly includes an algorithm for choosing optimal variable relaxation for two coupled problems.

In numerical tests we demonstrate convergence of our method. For two problems with an approximately equal computational workload, our new method shows an improved per-

formance over classical WR methods. This is due an improved convergence rate combined with solving the subproblems in parallel. In case of not well load-balanced problems, our new method reverts to classical GS WR.

We demonstrate the functionality of the `DUNE-FEniCS` coupling in a gas quenching test case. The problem consists of the compressible Euler equations and the nonlinear linear heat equation, coupled using a Dirichlet-Neumann approach. Here we use the same relaxation parameters as in Chapter 5, leading to rapid convergence of the iteration.

7.2 Future work

Goal oriented time-adaptivity

It would be interesting to test the proposed goal oriented method in the context of PDE-constrained optimization. There, one repeatedly solves a given PDE for varying parameters. This could lead to considerable speedups for the optimization process as a whole.

Our analysis is based on a specific type of QoI. It is natural to try to extend our results to more general QoIs, e.g., point-wise evaluations in time.

One could further generalize this work by using a density function j_{est} in error estimation, that is not the density function j from the QoI. This adds a degree of freedom in constructing the goal oriented method and can be used to overcome possible weaknesses, such as vanishing error estimates. This requires further testing and gives rise to the question on how to choose j_{est}.

Waveform Relaxation

Aside from adding the features discussed in Section 4.6.2, i.e., time-adaptive grids and parallelism in space for the subsolvers, application to more diverse examples is desired. The gas quenching example can be refined by using non-cartesian and possibly adaptive spatial grids, and including viscous fluxes, i.e., using the compressible Navier-Stokes equations.

Dirichlet-Neumann Waveform Relaxation for heterogeneous coupled heat equations

With both non-matching domain sizes and SDIRK2 one can derive Θ_{opt} following the same principles as in [54]. While this is likely extremely tedious, it would be interesting to see differences to the Θ_{opt} formula presented in Section 5.3 resp. Paper III .

As remarked at the end of Section 5.1.1, there is no evident way of constructing the NNWR method via splittings. By possibly introducing some redundancy in the original system, one might be able to find a corresponding splitting.

The principle recipe for constructing the fully-discrete DNWR SDIRK2 method can be used in conjunction with higher-order time-integration schemes such as SDIRK3. Lastly, it is not yet clear why the replacement of both the stage flux and stage solution for the SDIRK2 method, c.f. Section 5.2.2, only results in a first order accurate method.

Waveform relaxation with asynchronous time-integration

We extended the linear WR theory from Chapter 4 to variable splittings, on finite time-intervals. Since there already exists theory on the infinite time case and nonlinear WR [9, 46], it would be interesting to extend these to variable splittings.

It would be highly desirable to extend the variable relaxation algorithm, see Section 7 in Paper IV , to include adaptive time-grids. This likely results in a technically very involved method. There, another aspect to consider is how to choose optimal relaxation. In Paper III we used the heuristic approach of choosing relaxation based on the maximum of the average stepsizes between both time-grids. Yet, at runtime we have incomplete information about the time-grids. As such, we need to develop strategies for choosing optimal relaxation based on local information about the time-grids. Any insights gained from working strategies should also be applicable to the adaptive DNWR method from Paper III .

Our new method is, similar to Jacobi WR, a parallel in time method. As such, finding suitable load-balancing techniques is necessary for competitive parallel performance. In the context of simulations with parallelism in space on each subsolver, a suitable allocation of computational resources among the subsolvers can help with load-balancing.

WR is a *two-stage algorithm* [21], most notable by two the nested *for-loops* in Algorithm 6.1. The *outer-stage* is the WR iteration loop over k and the *inner-stage* is the time-integration loop.

Our method is a *inner-stage asynchronous WR method*, since we kept the synchronization point of the outer loop and use asynchronous communication in the inner time-integration stage. *Outer-stage asynchronous WR* has been considered in [21, 46], showing potential for speed-ups. These two concepts for using asynchronous communication in WR methods can be combined to construct a inner -and outer stage asynchronous WR method. Finding and implementing suitable cancellation criteria is nontrivial in outer-stage asynchronous methods, c.f. [3, 15, 44]. This is likely even more difficult with asynchronous time-integration.

Chapter 8

Appendix

8.1 Linear multistep methods

See e.g. [27, Chap.3.2] for a comprehensive introduction to linear multistep methods (LMM).

Definition 8.1. *A m-step LMM for* (2.1) *defined by coefficients* a_ℓ, b_ℓ, $\ell = 0, \ldots, m$ *and*

$$\sum_{\ell=0}^{m} a_\ell \, \mathbf{u}_{n+\ell} = \Delta t \sum_{\ell=0}^{m} b_\ell \, \mathbf{f}(t_{n+\ell}, \mathbf{u}_{n+\ell}). \tag{8.1}$$

We assume $a_k \neq 0$ and either $a_0 \neq 0$ or $b_0 \neq 0$. A m-step LMM requires m initial values $\mathbf{u}_i$, $i = 0, \ldots, m-1$, typical obtained by using LMM with an increasing number of steps or one-step methods. The coefficients a_ℓ, b_ℓ, $\ell = 0, \ldots, m$ define the so called *characteristic polynomials*

$$\rho(z) = \sum_{\ell=0}^{m} a_\ell z^\ell \quad \text{and} \quad \sigma(z) = \sum_{\ell=0}^{m} b_\ell z^\ell.$$

Definition 8.2. *A consistent LMM is called zero-stable if all roots of* ρ *fulfill* $|z| \leq 1$ *and roots with* $|z| = 1$ *are simple.*

8.2 On (block)normal matrices

The following results are included for the sake of completeness. They are not novel and likely found in many standard textbooks on Linear Algebra.

The following is based on the Wikipedia article on normal matrices[1].

Lemma 8.3. *If* $\mathbf{R} \in \mathbb{R}^{n\times n}$ *is upper triangular and normal, i.e.,* $\mathbf{R}\mathbf{R}^T = \mathbf{R}^T\mathbf{R}$*, then* $\mathbf{R}$ *is diagonal.*

Proof. $\mathbf{R}$ has the columns $\mathbf{R}\,\mathbf{e}_i$ and rows $\mathbf{R}^T\mathbf{e}_i$. It is diagonal, if

$$|r_{i,i}| = \|\mathbf{R}\,\mathbf{e}_i\|_2 = \|\mathbf{R}^T\mathbf{e}_i\|_2, \quad \forall\ i = 1,\ldots,n. \tag{8.2}$$

We have

$$|r_{1,1}|^2 = \|\mathbf{R}\,\mathbf{e}_1\|_2^2 = \mathbf{e}_1^T\mathbf{R}^T\mathbf{R}\mathbf{e}_1 \overset{\mathbf{R}\text{ normal}}{=} \mathbf{e}_1^T\mathbf{R}\mathbf{R}^T\mathbf{e}_1 = \|\mathbf{R}^T\mathbf{e}_1\|_2^2.$$

This means the first row is zero, except for the diagonal element. We then get (8.2) via induction. □

Theorem 8.4. *Consider* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *regular.* $\mathbf{A}$ *is diagonalizeable iff* $\mathbf{A}$ *is normal.*

Proof. Consider $\mathbf{A}$ diagonalizeable, i.e. $\mathbf{A} = \mathbf{V}^T\mathbf{D}\mathbf{V}$ with $\mathbf{V}$ orthonormal and $\mathbf{D}$ diagonal, then

$$\mathbf{A}^T\mathbf{A} = (\mathbf{V}^T\mathbf{D}\mathbf{V})^T\mathbf{V}^T\mathbf{D}\mathbf{V} = \mathbf{V}^T\mathbf{D}\mathbf{V}(\mathbf{V}^T\mathbf{D}\mathbf{V})^T = \mathbf{A}\mathbf{A}^T.$$

Consider $\mathbf{A}$ normal. For any regular matrix there exists a decomposition $\mathbf{A} = \mathbf{W}^T\mathbf{R}\,\mathbf{W}$ with $\mathbf{W}$ orthonormal and $\mathbf{R}$ upper triangular. We have

$$\begin{aligned}
\mathbf{A}^T\mathbf{A} &= (\mathbf{W}^T\mathbf{R}\,\mathbf{W})^T\mathbf{W}^T\mathbf{R}\,\mathbf{W} = \mathbf{W}^T\mathbf{R}^T\mathbf{R}\,\mathbf{W},\\
\mathbf{A}\mathbf{A}^T &= \mathbf{W}^T\mathbf{R}\,\mathbf{W}(\mathbf{W}^T\mathbf{R}\,\mathbf{W})^T = \mathbf{W}^T\mathbf{R}\mathbf{R}^T\mathbf{W},\\
&\Rightarrow \mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T.
\end{aligned}$$

That is, the matrix $\mathbf{R}$ obtained via the Schur-decomposition is normal. By Lemma 8.3, we know that triangular normal matrices are diagonal, meaning the Schur-decomposition is in fact a diagonalization. □

Theorem 8.5. *Let* $\mathbf{R} \in \mathbb{R}^{n\times n}$ *upper block-triangular.* $\mathbf{R}$ *is normal, iff* $\mathbf{R}$ *is block diagonal with normal diagonal blocks.*

Proof. Consider

$$\mathbf{R} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0} & \mathbf{C} \end{pmatrix}.$$

[1] `https://en.wikipedia.org/wiki/Normal_matrix`, accessed: 2019-08-20.

$\mathbf{R}$ normal implies

$$\mathbf{B}\mathbf{B}^T = \mathbf{A}^T\mathbf{A} - \mathbf{A}\mathbf{A}^T.$$

Using $\|\cdot\|_F^2 = \text{trace}(\cdot)$, where $\|\cdot\|_F$ is the Frobenius norm, we get

$$\text{trace}(\mathbf{A}^T\mathbf{A} - \mathbf{A}\mathbf{A}^T) = 0 = \|\mathbf{B}\mathbf{B}^T\|_F^2 \Leftrightarrow \mathbf{B}\mathbf{B}^T = \mathbf{0}$$

and thus $\mathbf{B} = \mathbf{0}$. Similarly this implies that $\mathbf{A}$ and $\mathbf{C}$ are normal. This extends to matrices with more blocks by applying the same argument to different groupings of blocks.

Showing the reverse direction is straight forward. □

Bibliography

Bibliography

[1] C. Andersson. *Methods and tools for co-simulation of dynamic systems with the functional mock-up interface*. Dissertation, Lund University, 2016.

[2] D. Arndt, W. Bangerth, B. Blais, T. C. Clevenger, M. Fehling, A. V. Grayver, T. Heister, L. Heltai, M. Kronbichler, M. Maier, P. Munch, J.-P. Pelteret, R. Rastak, I. Tomas, B. Turcksin, Z. Wang, and D. Wells. The deal. II library, version 9.2. *Journal of Numerical Mathematics*, 2020.

[3] J. M. Bahi, S. Contassot-Vivier, R. Couturier, and F. Vernier. A decentralized convergence detection algorithm for asynchronous iterative algorithms. *IEEE Transactions on Parallel and Distributed Systems*, 16(1):4–13, 2005.

[4] W. Bangerth and R. Rannacher. *Adaptive finite element methods for differential equations*. Birkhäuser, 2013.

[5] A. Banka. Practical Applications of CFD in heat processing. *Heat Treating Progress*, 5 (5):44–46, 2005.

[6] P. Bastian, M. Blatt, A. Dedner, N. A. Dreier, C. Engwer, R. Fritze, C. Gräser, C. Grüninger, D. Kempf, R. Klöfkorn, M. Ohlberger, and O. Sander. The DUNE framework: Basic concepts and recent developments. *Computers and Mathematics with Applications*, 81:75–112, 2021. ISSN 08981221. doi: 10.1016/j.camwa.2020.06.007.

[7] R. Becker and R. Rannacher. An optimal control approach to a posteriori error estimation in finite element methods. *Acta Numerica 2001*, 10:1–102, 2001.

[8] A. Bellen, Z. Jackiewicz, and M. Zennaro. Time-point relaxation Runge-Kutta for ordinary differential equations. *J. Comp. Appl. Math.*, 45:121–137, 1993.

[9] M. Bjørhus. A note on the convergence of discretized dynamic iteration. *BIT Numerical Mathematics*, 35(June 1994):291–296, 1995.

[10] M. Blatt, A. Burchardt, A. Dedner, C. Engwer, J. Fahlke, B. Flemisch, C. Gersbacher, C. Gräser, F. Gruber, C. Grüninger, D. Kempf, R. Klöfkorn, T. Malkmus, S. Müthing, M. Nolte, M. Piatkowski, and O. Sander. The Distributed and Unified Numerics Environment, Version 2.4. *Archive of Numerical Software*, 4(100):13–29, 2016.

[11] D. L. Brown, J. Bell, D. Estep, B. Hendrickson, S. Keller-McNulty, D. Keyes, J. T. Oden, L. Petzold, and M. Wright. Applied Mathematics at the U.S. Department of Energy: Past, Present and a View to the Future. *Bulletin of the American Mathematical Society*, 56:539–541, 2008.

[12] J. M. Buchlin. Convective heat transfer and infrared thermography (IRTh). *Journal of Applied Fluid Mechanics*, 3(1):55–62, 2010.

[13] H. J. Bungartz, F. Lindner, B. Gatzhammer, M. Mehl, K. Scheufele, A. Shukaev, and B. Uekermann. preCICE – A fully parallel library for multi-physics surface coupling. *Computers and Fluids*, 141:250–258, 2016.

[14] P. Causin, J. F. Gerbeau, and F. Nobile. Added-mass effect in the design of partitioned algorithms for fluid-structure problems. *Computer Methods in Applied Mechanics and Engineering*, 194(42-44):4506–4527, 2005.

[15] J. C. Charr, R. Couturier, and D. Laiymani. A decentralized and fault tolerant convergence detection algorithm for asynchronous iterative algorithms. *Journal of Supercomputing*, 53(2):269–299, 2010.

[16] P. Crosetto, P. Reymond, S. Deparis, D. Kontaxakis, N. Stergiopulos, and A. Quarteroni. Fluid-structure interaction simulation of aortic blood flow. *Computers and Fluids*, 43(1):46–57, 2011.

[17] A. Dedner and R. Klöfkorn. Extendible and efficient python framework for solving evolution equations with stabilized discontinuous galerkin methods. *arXiv*, 2020. ISSN 23318422.

[18] J. E. Dennis, Jr and J. J. Moré. Quasi-Newton Methods, motivation and theory. *SIAM Review*, 19(1):46–89, 1977.

[19] T. D. Economon, F. Palacios, S. R. Copeland, T. W. Lukaczyk, and J. J. Alonso. SU2: An open-source suite for multiphysics simulation and design. *AIAA Journal*, 54(3): 828–846, 2016.

[20] D. E. et al. Keyes. Multiphysics simulations: Challenges and opportunities. *International Journal of High Performance Computing Applications*, 27(1):4–83, 2013.

[21] A. Frommer and D. B. Szyld. On asynchronous iterations. *Journal of Computational and Applied Mathematics*, 123(1-2):201–216, 2000.

[22] S. W. Funke, P. E. Farrell, and M. D. Piggott. Tidal turbine array optimisation using the adjoint approach. *Renewable Energy*, 63:658–673, 2014.

[23] M. J. Gander and A. M. Stuart. Space-time continuous analysis of Waveform Relaxation for the heat equation. 19(6):2014–2031, 1998.

[24] M. J. Gander, F. Kwok, and B. C. Mandal. Dirichlet-Neumann and Neumann-Neumann waveform relaxation algorithms for parabolic problems. *Electronic Transactions on Numerical Analysis*, 45:424–456, 2016.

[25] C. W. Gear. *Numerical initial value problems in ordinary differential equations*. Prentice Hall PTR, 1971.

[26] M. Geveler and S. Turek. How applied sciences can accelerate the energy revolution - A pleading for energy awareness in scientific computing. *Newsletter of the European Community on Computational Methods in Applied Sciences*, 2017.

[27] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving ordinary differential equations I*. Springer, Berlin, 1993.

[28] U. Heck, U. Fritsching, and K. Bauckhage. Fluid flow and heat transfer in gas jet quenching of a cylinder. *International Journal of Numerical Methods for Heat & Fluid Flow*, 2001.

[29] M. M. Helsen, W. J. Van De Berg, R. S. Van De Wal, M. R. Van Den Broeke, and J. Oerlemans. Coupled regional climate-ice-sheet simulation shows limited Greenland ice loss during the Eemian. *Climate of the Past*, 9(4):1773–1788, 2013.

[30] M. Hinderks and R. Radespiel. Investigation of hypersonic gap flow of a reentry nosecap with consideration of fluid structure interaction. In *44th AIAA Aerospace Sciences Meeting and Exhibit*, 2006.

[31] M. Hochbruck and A. Ostermann. Exponential integrators. *Acta Numerica*, 19(2010): 209–286, 2010.

[32] J. Janssen and S. Vandewalle. Multigrid waveform relaxation on spatial finite element meshes: the discrete-time case. *SIAM Journal on Scientific Computing*, 17(1):133–155, 1996.

[33] J. Janssen and S. Vandewalle. Multigrid Waveform Relaxation on Spatial Finite Element Meshes : The Continuous-Time Case. *SIAM Journal on Numerical Analysis*, 33 (2):456–474, 1996.

[34] J. Janssen and S. Vandewalle. On SOR waveform relaxation methods. *SIAM Journal on Numerical Analysis*, 34(6):2456–2481, 1997.

[35] H. Jasak and A. Jemcov. OpenFOAM: A C++ library for complex physics simulations. *International Workshop on Coupled Methods in Numerical Dynamics*, m:1–20, 2007.

[36] R. Jeltsch and B. Pohl. Waveform relaxation with overlapping splittings. *SIAM J. Sci. Comput.*, 16(1):40–49, 1995.

[37] D. Kowollik, V. Tini, S. Reese, and M. Haupt. 3D fluid–structure interaction analysis of a typical liquid rocket engine cycle based on a novel viscoplastic damage model. *International journal for numerical methods in engineering*, 94(13):1165–1190, 2013.

[38] D. S. C. Kowollik, P. Horst, and M. C. Haupt. Fluid-structure interaction analysis applied to thermal barrier coated cooled rocket thrust chambers with subsequent local investigation of delamination phenomena. 4:617–636, 2013.

[39] F. Kwok. Neumann–Neumann Waveform Relaxation for the Time-Dependent Heat Equation. *Domain Decomposition Methods in Science and Engineering XXI*, pages 189–198, 2014.

[40] E. Lelarasmee. *The Waveform Relaxation Method for time domain analysis of large scale integrated circuits: Theory and Applications*. Phd thesis, U.C. Berkeley, 1982.

[41] A. Logg, K.-A. Mardal, and G. Wells. *Automated solution of differential equations by the finite element method: The FEniCS book*, volume 84. Springer, 2012.

[42] A. Lumsdaine and D. Wu. Spectra and Pseudospectra of Waveform Relaxation operators. *SIAM J. Sci. Comput.*, 18(1):286–304, 1997.

[43] A. Lumsdaine and D. Wu. Krylov subspace acceleration of waveform relaxation. *SIAM Journal on Numerical Analysis*, 41(1):90–111, 2003.

[44] F. Magoulès and G. Gbikpi-Benissan. JACK: an asynchronous communication kernel library for iterative algorithms. *Journal of Supercomputing*, 73(8):3468–3487, 2017.

[45] B. C. Mandal. *Convergence analysis of substructuring Waveform Relaxation methods for space-time problems and their application to Optimal Control Problems*. PhD thesis, Université de Genève, 2014.

[46] S. Martin. *Parallel asynchrone Waveform-Relaxation für Anfangswertprobleme*. Diplomarbeit, Bergische University Wuppertal, 1999.

[47] R. Mehta. Numerical computation of heat transfer on reentry capsules at mach 5. In *43rd AIAA Aerospace Sciences Meeting and Exhibit*, 2005.

[48] D. Meidner and T. Richter. Goal-oriented error estimation for the fractional step theta scheme. *Computational Methods in Applied Mathematics*, 14(2):203–230, 2014.

[49] D. Meidner and T. Richter. A posteriori error estimation for the fractional step theta discretization of the incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 288:45–59, 2015.

[50] P. Meisrimel. *Goal Oriented Time Adaptivity using Local Error Estimates*. Licentiate thesis, Lund University, 2018.

[51] P. Meisrimel. Waveform relaxation with asynchronous time-integration, 2021. URL `https://gitlab.maths.lth.se/PeterMeisrimel/asynch-WFR`.

[52] U. Miekkala. Dynamic iteration methods applied to linear DAE systems. *Journal of Computational and Applied Mathematics*, 25(2):133–151, 1989.

[53] Modelica Association Project. Functional Mock-up Interface Standard. URL `https://fmi-standard.org/`.

[54] A. Monge. *Partitioned methods for time-dependent thermal fluid-structure interaction*. Ph.d., Lund University, 2018.

[55] A. Monge and P. Birken. On the convergence rate of the Dirichlet–Neumann iteration for unsteady thermal fluid–structure interaction. *Computational Mechanics*, pages 1–17, 2017.

[56] A. Monge and P. Birken. Towards a Time Adaptive Neumann-Neumann Waveform Relaxation Method for Thermal Fluid-Structure Interaction. *Proceedings of the 25th Domain Decomposition conference*, pages 1–8, 2018.

[57] A. Monge and P. Birken. A Multirate Neumann–Neumann Waveform Relaxation Method for Heterogeneous Coupled Heat Equations. *SIAM Journal on Scientific Computing*, 41(5):S86–S105, 2019.

[58] O. Nevanlinna. Remarks on Picard-Lindelöf Iteration, Part I. *BIT Numerical Mathematics*, 29(April 1988):328–346, 1989.

[59] O. Nevanlinna. Remarks on Picard-Lindelöf Iteration, Part II. *BIT Numerical Mathematics*, 29(April 1988):535–562, 1989.

[60] S. Prudhomme. A Posteriori Error Estimates of Quantities of Interest. *Encyclopedia of Applied and Computational Mathematics*, pages 1–5, 2015.

[61] A. Quarteroni and V. Alberto. *Domain decomposition methods for partial differential equations*. Oxford University Press, 1999.

[62] F. Rathgeber, D. A. Ham, L. Mitchell, M. Lange, F. Luporini, A. T. McRae, G. T. Bercea, G. R. Markall, and P. H. Kelly. Firedrake: Automating the finite element method by composing abstractions. *ACM Transactions on Mathematical Software*, 43 (3), 2016.

[63] M. W. Reichelt, J. K. White, and J. Allen. Optimal convolution SOR acceleration of Waveform Relaxation with application to paralla simulation of semiconductor devices. *SIAM Journal on Scientific Computing*, 16(5):1137–1158, 1995.

[64] B. Rodenberg, I. Desai, R. Hertrich, A. Jaust, and B. Uekermann. FEniCS-preCICE: Coupling FEniCS to other Simulation Software. *arXiv preprint*, 2021. URL `http://arxiv.org/abs/2103.11191`.

[65] U. Rüde, K. Willcox, L. C. McInnes, and H. De Sterck. Research and education in computational science and engineering. *SIAM Review*, 60(3):707–754, 2018.

[66] B. Rüth and P. Meisrimel. FEniCS-preCICE adapter. URL `https://github.com/precice/fenics-adapter`.

[67] B. Rüth, B. Uekermann, M. Mehl, P. Birken, A. Monge, and H. J. Bungartz. Quasi-Newton waveform iteration for partitioned surface-coupled multiphysics applications. *International Journal for Numerical Methods in Engineering*, pages 1–22, 2020.

[68] F. SCAI. MpCCI 4.6.1-1 Documentation. Technical Report March 21, Fraunhofer SCAI, 2021.

[69] S. Schöps, H. De Gersem, and A. Bartel. A cosimulation framework for multirate time integration of field/circuit coupled problems. *IEEE Transactions on Magnetics*, 46(8):3233–3236, 2010.

[70] L. F. Shampine. Local Extrapolation in the Solution of Ordinary Differential Equations. *Mathematics of Computation*, 27(121):91, 1973.

[71] L. F. Shampine. The step sizes used by one-step codes for ODEs. *Applied Numerical Mathematics*, 1(1):95–106, 1985.

[72] L. F. Shampine. Tolerance proportionality in ODE codes. *Numerical Methods for Ordinary Differential Equations*, pages 118–136, 1989.

[73] L. F. Shampine. Error Estimation and Control for ODEs. *J. Sci. Comput.*, 25(1):3–16, 2005.

[74] L. F. Shampine and H. A. Watts. Comparing error estimators for Runge-Kutta methods. *Mathematics of computation*, 25(115):445–455, 1971.

[75] G. Söderlind. Digital filters in adaptive time-stepping. *ACM Transactions on Mathematical Software*, 29(1):1–26, 2003.

[76] G. Söderlind. The logarithmic norm. History and modern theory. *BIT Numerical Mathematics*, 46(3):631–652, 2006.

[77] G. Söderlind, I. Fekete, and I. Faragó. On the zero-stability of multistep methods on smooth nonuniform grids. *BIT Numerical Mathematics*, pages 1–19, 2018.

[78] C. Steiner and S. Noelle. On adaptive timestepping for weakly instationary solutions of hyperbolic conservation laws via adjoint error control. *International journal for numerical methods in biomedical engineering*, 26(6):790–806, 2010.

[79] P. Stratton, I. Shedletsky, and M. Lee. Gas quenching with helium. *Solid State Phenomena*, 118, 2006.

[80] R. Teyssier and B. Commerçon. Numerical Methods for Simulating Star Formation. *Frontiers in Astronomy and Space Sciences*, 6(July), 2019.

[81] M. Tokman. A new class of exponential propagation iterative methods of Runge-Kutta type (EPIRK). *Journal of Computational Physics*, 230(24):8762–8778, 2011.

[82] A. Toselli and O. B. Widlund. *Domain Decomposition Methods – Algorithms and Theory*. Springer, 2005.

[83] L. N. Trefethen and M. Embree. *Spectra and pseudospectra: the behavior of nonnormal matrices and operators*. Princeton University Press, 2005.

[84] B. Uekermann, H. J. Bungartz, L. Cheung Yau, G. Chourdakis, and A. Rusch. Official preCICE Adapters for Standard Open-Source Solvers. *Proceedings of the 7th GACM Colloquium on Computational Mechanics for Young Scientists from Academia*, 2017.

[85] S. Vandewalle. *Parallel multigrid waveform relaxation for parabolic problems*. Springer-Verlag, Berlin, 2013.

[86] H. A. Watts. Starting step size for an ODE solver. *Journal of Computational and Applied Mathematics*, 9(2):177–191, 1983.

[87] J. K. White and A. Sangiovanni-Vincentelli. *Relaxation Techniques for the Simulation of VLSI Circuits*. Kluwer Academic Publishers, Boston, 1987.

[88] R. M. Wilcox. Exponential operators and parameter differentiation in quantum physics. *Journal of Mathematical Physics*, 8(4):962–982, 1967.

[89] W. Xudong, W. Z. Shen, W. J. Zhu, J. N. Sørensen, and C. Jin. Shape optimization of wind turbine blades. *Wind Energy*, 12(8):781–803, 2009.

[90] D. M. Young. *Iterative solution of large linear systems*. Academic Press, 1971.