Parallel Space-Time Multilevel Methods with Application to Electrophysiology

Theory and Implementation

Doctoral Dissertation submitted to the
Faculty of Informatics of the Università della Svizzera Italiana
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

presented by
Pietro Benedusi

under the supervision of
Prof. Rolf Krause

May 2020

Dissertation Committee

Prof. Ilia Horenko	Università della Svizzera Italiana, Switzerland
Prof. Michael Multerer	Università della Svizzera Italiana, Switzerland
Prof. Martin J.Gander	Université de Genève, Switzerland
Prof. Jacob Schroder	University of New Mexico, USA
Prof. Stefano Serra-Capizzano	Università degli Studi dell'Insubria, Italy

Dissertation accepted on 12 May 2020

Research Advisor	PhD Program Director
Prof. Rolf Krause	**Prof. Walter Binder**

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Pietro Benedusi
Lugano, 12 May 2020

Non m'importa di mostrare di aver avuto ragione, ma di stabilire se l'ho avuta. E vi dico: lasciate ogni speranza, o voi che vi accingete a osservare. [...] Se qualche scoperta seconderà le nostre previsioni, la considereremo con speciale diffidenza.

Meine Absicht ist nicht, zu beweisen, daß ich bisher recht gehabt habe, sondern: herauszufinden, ob. Ich sage: laßt alle Hoffnung fahren, ihr, die ihr in die Beobachtung eintretet. [...] Und was wir zu finden wünschen, das werden wir, gefunden, mit besonderem Mißtrauen ansehen.

My intention is not to prove that hitherto I have been right; but to discover whether I am right. I say: abandon all hope, you who enter the realm of observation. [...] And what we wish to find, if we do find it, we shall regard with especial distrust.

Bertolt Brecht, "Vita di Galileo"

Abstract

The goal of this thesis is to design and study an efficient strategy to solve possibly non-linear parabolic partial differential equations on massively parallel machines.

Traditionally, when solving time-dependent problems, time stepping methods are used to advance the solution in time. These techniques are inherently sequential and therefore they introduce a bottleneck in the overall computational scalability. To overcome this limitation, we focus on the design of time parallel solvers.

To achieve parallel efficiency in both space and time, we employ a multilevel space-time finite element discretization, coupled with parallel block preconditioners. We use continuous finite elements to discretize in space and, for stability reasons, we adopt discontinuous finite elements in the time dimension. In space, in particular, we consider the generic finite element framework of isogeometric analysis.

We consider a space-time multilevel method, based on a hierarchy of non-nested meshes, created using a semi-geometric approach. With this technique, we can automatically generate space-time coarse spaces, starting from a single fine spatial mesh, in any dimension and in the presence of complex geometries.

Through a detailed spectral analysis, we can design convenient preconditioners for space-time operators and give estimates of their conditioning, with respect to problem, discretization and multigrid parameters.

We numerically investigate how different iterative solution strategies, coarsening strategies and spectral based preconditioners, can affect the overall convergence and robustness of our multilevel approach. Finally, we run strong and weak scalability experiments, mostly focusing on time parallelism. In this analysis, we consider two model problems: the heat equation, possibly anisotropic or with jumping coefficients, and the monodomain equation, a non-linear reaction-diffusion model arising from the study of the electrical activation of the human heart.

Acknowledgements

During my studies, I learned that teamwork is fundamental in order to deal with the challenges of contemporary research in science. For this reason, I would like to sincerely thank many people, whose contributions have been invaluable to this work.

First, I thank my advisor, Professor Rolf Krause, for his guidance, insightful observations and advice. He has inspired me during the past years with his sincere enthusiasm, rigorous reasoning and with his lively, often challenging, lectures. I would like to acknowledge the opportunity that I have been given to expand my knowledge by forming constructive scientific collaborations and participating in major academic symposia.

I would like to express my sincere gratitude also to Patrick Zulian for his support, tutoring and practical mindset. Without his advice and the computational tools he designed this thesis would have been very different.

Moreover, I am incredibly grateful to Carlo Garoni; he has been inspiring, for his outstanding passion, positivity and *astuzia*. He taught me a lot about mathematical rigour, elegance and academic practices and his contribution and guidance have been essential for the results presented in Chapter 3.

I would also like to thank Professor Serra-Capizzano, for his stimulating lectures, for his contagious enthusiasm and the insightful discussions we had.

My sincere thanks also go to Paola Ferreri for the work on the implementation of IgA matrices, to Alena Kopaničáková for her frequent assistance with the Utopia solvers, to Simone Pezzuto for his clear explanations about mathematical electrophysiology and to Seif Ben Bader for providing me with the anisotropic diffusion tensors described in Appendix A.

I would like to offer a special thank to Maria Nestola, for her help and encouragement when nothing was working.

I am also particularly grateful to Michael Minion; he agreed to host me at the Berkeley Lab as a research visitor; his scientific attitude and his spirit are exceptional and his hospitality was truly special.

My thanks are extended to the staff of the Decanato of informatics, for their

friendly and consistent assistance and to Toby Simpson for the numerous, scientific or more colloquial, sometimes animated, always thought-provoking, discussions.

Last but not the least, I am most grateful to my parents and Virgilia, for their unconditional support. They believed in me in these years enhancing, in the first place, my intellectual curiosity.

Contents

Chapter 1

Introduction

In the near future exascale machines, i.e. machines capable of 10^{18} floating point operations per second, will probably enter in service[1]. In Figure 1.1 an extrapolation of the current trend in supercomputing performance is illustrated[2] and in Figure 1.2, the growing relevance of multi-core architectures in recent years is shown[3].

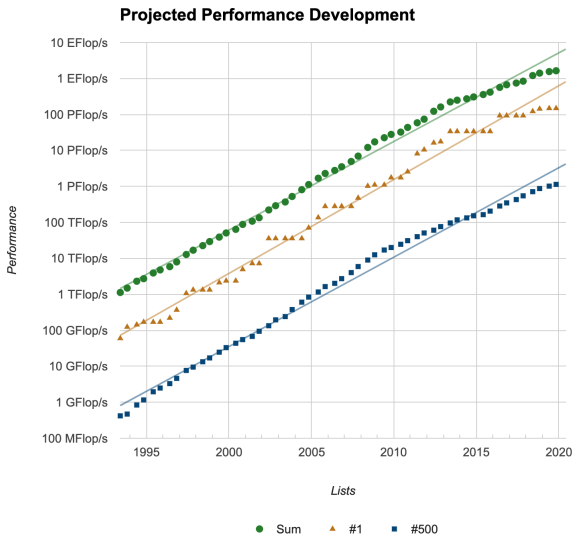

Figure 1.1. Historical data of performance from the "top500" machine list. Exascale systems are expected to appear in the next years (yellow line).

[1] According to the Chinese national plan for high performance computing, see for example: `https://www.top500.org/news/china-reveals-third-exascale-prototype`

[2] Source: `https://www.top500.org/statistics/perfdevel/`

[3] Source: `https://www.karlrupp.net/2015/06/40-years-of-microprocessor-trend-data`

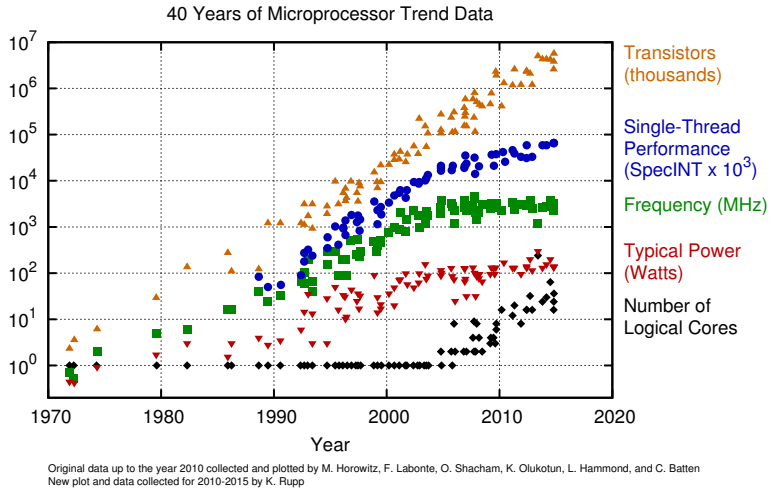

Figure 1.2. The increment in computing performance in the last two decades is due to the use of additional cores, as clock frequency is stagnating. To exploit these resources, highly parallel algorithms have to be designed.

Exploiting the capabilities of such massively parallel systems is important but it is not straightforward; algorithms with optimal complexity and excellent scalability must be designed to minimize the "time-to-solution" of computationally intensive problems, such as the solution of time dependent partial differential equations (PDEs). When dealing with parallel solvers for PDEs, the solution process is usually parallelized in space using domain decomposition techniques, until saturation. Considering the technology trend, the development of new parallel methods has become essential. For time dependent PDEs, promising candidates to improve algorithmic scalability are the parallel-in-time methods. In this class of algorithms, we exploit concurrency in the time direction in addition to the spatial one. In fact, time stepping is traditionally a sequential process and a remarkable bottleneck for computational scalability. So, when parallelization in space saturates due to communication costs, additional speedup can still be achieved with parallel-in-time methods.

In particular, parallelization in time becomes relevant if a very fine temporal grid is needed. For example in multi-scale processes, the resolution in space, and consequently in time, must be very high to resolve microscopic structures. Another example where high accuracy in temporal integration is needed is the case of highly oscillatory PDEs. For these reasons parallel-in-time techniques got more and more attention from the academic community in the past decade, cf.

Figure 1.3[4].

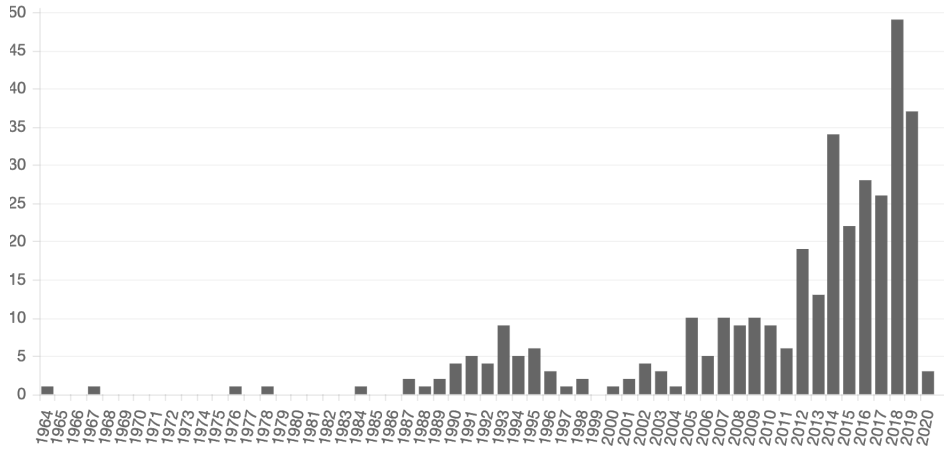

Figure 1.3. Number of publications per year in the field of parallel-in-time methods; February 2020.

However, parallelization in time can be a challenging task, as, for many physical processes, the time direction is governed by a causality principle and back-propagation is an ill-posed problem. Numerical information flows in just one direction through the temporal domain: from past to future and not the other way round. In particular, irreversible processes, like diffusion, are ill conditioned to back propagation. For this reason, time integration is traditionally a sequential process. Generally, we refer to this class of problems as IVP (Initial Value Problems), in contrast to BVP (Boundary Value Problems), where the solution is constrained on the whole boundary.

1.1 Brief history and state of the art of parallel-in-time methods

The seminal paper of this field was written more then 50 years ago by J. Nivergelt [1964]: a shooting type parallel solver for ODEs. Since this visionary contribution, a large number of different strategies have been studied to solve ODEs with some degree of concurrency.

In Nievergelt's paper a rough and cheap predictor was used to get initial values in different time intervals. Then multiple fine integrators were used along

[4]Source: `https://www.parallelintime.org/references/index.html`

the temporal domain: some redundancy in the computations was introduced, but the fine integrator could run in parallel. The fine results were combined trough an interpolation technique to get the desired solution.

The multilevel setting, i.e. the idea of using a coarse sequential predictor and subsequently applying fine integrators in parallel, even introducing some redundancy in the computations, is the *fil rouge* of the majority of the following methods.

For example Lions et al. [2001] invented the popular two-level Parareal algorithm; a shooting type algorithm again, where the continuity of the solution is enforced trough an iterative process.

A more recent development of Parareal is PFASST (Parallel FAS in Space-Time) by Emmett and Minion [2012] that uses the SDC (Spectral Deferred Correction) algorithm [Dutt et al., 2000] as the time integrator coupled with a multilevel strategy in time. In particular, SDC is an iterative method based on the integral form of an IVP (Picard form) where spectral integration is used. PFASST got a certain attention form the community and more recent developments, such as a fault tolerant version [Speck and Ruprecht, 2016], have been studied.

The equivalence between these parallel-in-time algorithms and other multilevel strategies (multigrid) has been investigated, for example by Gander et al. [2018a] and Bolten et al. [2017].

Space-time multigrid was used for solving parabolic problems by Hackbusch [1984] and later on by Horton and Vandewalle [1995]. Other more recent results on this topic are: MGRIT by Falgout et al. [2014], a multigrid reduction in time method, and the work by Gander and Neumüller [2016] developed for parabolic PDEs. In the latter paper the authors use a space-time finite element discretization with discontinuous Galerkin (DG) in time. Then they solve the arising linear system with a parallel smoother (Block Jacobi) inside a multigrid cycle. Space-time multilevel discretizations are of particular interest for us, because our approach, that will be presented in the following sections, belongs to this class.

On the other hand, some direct solvers for time parallel integration have been designed, such as the ParaExp algorithm by Gander and Güttel [2013], especially suited for hyperbolic problems. This method is only suitable for linear problems and it is based on an overlapping domain decomposition.

Another class of parallel-in-time algorithms that has to be mentioned is the "parallelization across the method" one. See for example the work by Burrage [1997] where the author takes advantage of the fact that some time stepping methods consist of independent computations when evolving two successive time steps. These algorithms are usually easier to implement but do not provide a

significant speedup. Some examples are the extrapolation methods by Horton and Knirsch [1992] and the Parallel Runge-Kutta methods by Sommeijer [1993].

An extensive review of parallel-in-time methods can be found in Gander [2015].

1.2 Space-time discretization

Space-time methods are becoming more and more relevant for the scientific community, as the computational capabilities are increasing. In particular space-time discretizations, compared to standard time-stepping techniques enable, somehow automatically, full space-time parallelism for massively parallel architectures. The main idea of space-time formulations is to treat the temporal dimension as an additional spatial one and to assemble a large space-time system to be solved in parallel.

Space-time discretizations possess other advantages aside of the concurrency capabilities: they can enable local time-stepping (see Krause and Krause [2016]) as well as space-time adaptivity as in Adelaide et al. [2002]; Steinbach [2015]; Langer, Matculevich and Repin [2016] and they can efficiently deal with moving domains (see Langer, Moore and Neumüller [2016]) or time-periodic problems as in Benedusi et al. [2016].

On the other hand, space-time methods have some drawbacks. Large linear systems need to be stored, and memory limitations can occur. For this reason this approach is reasonable if memory is distributed among many processors. Additionally, for 3D problems, 4D meshes and data structures might be necessary. The usage of 4D meshes can be avoided if a tensor product structure between space and time is used, as the assembly becomes purely algebraic.

Space-time methods have been used with finite difference discretizations, for example by Benedusi et al. [2016], but mostly in the variational context. The first contribution on space-time finite elements, applied to a parabolic problem, was by Ladyzhenskaia et al. [1968]. Space-time variational formulations have been considered for a variety of applications: in fluid dynamics by Shakib and Hughes [1991], in mechanics by Betsch and Steinmann [2001] or in fluid structure interaction by Tezduyar et al. [2006] and finite elements formulations as isogeometric analysis (IgA) in Langer, Matculevich and Repin [2016]; Benedusi et al. [2018b] or DG methods by Abedi et al. [2006]; Miller and Haber [2008].

Moreover, many doctoral thesis in recent years have been focusing on space-time discretization [Krause, 2013] and correspondent multilevel solvers [Neumüller, 2013; McDonald, 2016; Andreev, 2012; Sudirham, 2005].

When dealing with space-time finite elements the time direction needs spe-

cial care: to ensure that the information flows in the positive time direction, a particular choice of basis in time is often used. Discontinuous Galerkin with an "upwind" flow is a common choice; see, for example, [Klaij et al., 2006; Sudirham et al., 2006].

1.3 Discontinuous Galerkin in time

Variational time-stepping methods are receiving increasing interest by the scientific community especially in the context of space–time adaptivity, see for example the work by Eriksson and Johnson [1991]; Schmich and Vexler [2008]. In fact they provide arbitrary order of convergence, good stability, and, crucially, they may allow local time stepping [Gander and Halpern, 2013] in different areas of the spatial domain and p-adaptivity in time.

Discontinuous Galerkin methods in particular have been widely used to discretize the time direction in the space-time setting. They have been employed for a variety of problems, as convection/advection/diffusion equations or Navier-Stokes equations. For example see the work of Jamet [1978]; Klaij et al. [2006]; Li and Wiberg [1998]; Sudirham et al. [2006]; Feistauer et al. [2011]; Besier and Rannacher [2012]; Gander and Neumüller [2016]. The use of DG discretization in time has been first introduced by Lasaint and Raviart [1974] for the discretization of a neutron transport equation. In this paper the authors showed that, for finite elements of order q, the method is strongly A-stable, has convergence order $2q + 1$ on the nodes and is equivalent to an implicit Runge-Kutta time stepper with q intermediate steps. The first analysis on DG methods as time stepping techniques was provided by Delfour et al. [1981] and Eriksson et al. [1985] followed by the work of Schieweck [2010]; Zhao and Wei [2014]; Thomée [1984]. Since then, specialized solvers have been introduced, for example by Smears [2016]; Richter et al. [2013]; Hussain et al. [2011]. A *priori* and *posteriori* error analysis have been also provided; e.g. see Thomée [1984]; Eriksson and Johnson [1991, 1995]; Schötzau and Wihler [2010]. See Shu [2014] for a recent survey on the topic.

On the other hand standard continuous Galerkin (CG) methods have also been employed for time integration, since they were initially studied by Hulme [1972b,a]. See, for example, Betsch and Steinmann [2000, 2001] for a more recent application in mechanics. Other examples of the use of space-time CG can be found in Hulbert and Hughes [1990]; Langer, Moore and Neumüller [2016]; Steinbach [2015]. Using CG results in a $(q + 1)$-order method. Aziz and Monk [1989] showed that for $q = 1$, CG is equivalent to a Crank–Nicolson scheme

with time averaged data. In the work by Langer, Moore and Neumüller [2016] isogeometric analysis (IgA) is used with the particular choice of a time-upwind test functions, that results in a term analogous to the addition of an artificial diffusion in time. For a comprehensive error analysis and comparative study between CG and DG we refer to Sabawi [2018].

1.4 Models and applications

In this thesis, we focus on diffusion and reaction-diffusion problems, testing non trivial setting such as complex geometries, anisotropic or jumping coefficients and non-linearity.

Diffusive processes, modeled by the heat equation, are a standard benchmark problem for parallel-in-time solvers. An additional reactive term can be used to model signal propagation through excitable media. In particular, we consider the monodomain equation, a standard tool used in cardiac modeling.

A realistic simulation of the human heart is a challenging topic of research in computational medicine. This organ can be particularly difficult to simulate effectively since many different physical processes take place at different scales: it is a multi-physics and a multi-scale problem. In particular, the cardiac functionality is regulated by the contributions of different mechanisms from electrophysiology, solid mechanics and fluid dynamics. In this work, we are mostly interested in the electrophysiological model describing how the electrical activation spreads in the cardiac tissue.

On the other hand, the fluid-dynamics and the mechanical models deal with the muscle contractions and deformations responsible for pumping the blood into the cardio-vascular system. These processes are not considered here.

The space-time framework is well suited for this problem (see for example Krause and Krause [2016] or the work by Sachetto Oliveira et al. [2018]) for the following reasons:

- The dynamics take place in a small region of the space-time domain: this suggests to use suitable space-time adaptive strategies.
- Employing space-time parallelism can reduce the time-to-solution, that can easily become intractable for realistic 3D simulations.

We consider a second application in computational medicine: the permeation of chemicals trough the human skin, that can be modeled as a diffusion process with jumping or discontinuous coefficients.

1.5 Outline and contributions

Chapter 2 is dedicated to the description of the mathematical model, its variational formulation and discretization, motivated by a comparative study along side with error analysis.

In Chapter 3 we introduce the symbol theory and present new results concerning the asymptotical spectral distribution of space-time operators. Such a distribution is validated trough multiple numerical experiments and is used to design suitable space-time preconditioners and estimate the conditioning of space-time operators. Such conditioning exhibits very peculiar features and its analysis can be profitable in practice.

In Chapter 4 the applications are described and the corresponding models are introduced. We show, with numerical examples, how the choice of model parameters affects the underlying solution and the well-posedness of the discrete problems.

In Chapter 5 linear and non-linear solution strategies for the space-time discrete problem are introduced. In particular, we consider a semi-geometric space-time multigrid method that uses PGMRES as smoother and semi-coarsening strategies, discussed in the light of the spectral tools derived in Chapter 3. We design an efficient preconditioner for the GMRES solver, combining algebraic manipulations, and a special multigrid as preconditioner.

Finally, Chapter 6 contains multiple numerical experiments to characterize the convergence, robustness and scaling of the proposed methods. We consider fairly complex geometries, anisotropy and realistic parameters. We also report, for the first time, a preliminary comparison between a space-time multigrid method and PFASST, for both the linear and the non-linear problems. This chapter also contains a description of our implementation with its main functionalities in a parallel framework.

For convenience, a summary of the used notation is provided before the bibliography.

Chapter 2

Model and Discretization

Let $\Omega \subset \mathbb{R}^d$ be the spatial domain and $T \in \mathbb{R}^+$ the final time. We consider the following reaction diffusion equation:

$$\begin{cases} \partial_t u(t,\mathbf{x}) - \nabla \cdot K(\mathbf{x}) \nabla u(t,\mathbf{x}) + r(u(t,\mathbf{x})) = f(t,\mathbf{x}), & (t,\mathbf{x}) \in (0,T) \times \Omega, \\ u(t,\mathbf{x}) = 0, & (t,\mathbf{x}) \in (0,T) \times \Gamma_D, \\ \mathbf{n}_{\mathbf{x}} \cdot (K(\mathbf{x}) \nabla u) = 0, & (t,\mathbf{x}) \in (0,T) \times \Gamma_N, \\ u(0,\mathbf{x}) = u_0(\mathbf{x}), & \mathbf{x} \in \Omega, \end{cases} \tag{2.1}$$

where $K(\mathbf{x}) \in \mathbb{R}^{d \times d}$ is the diffusion coefficient matrix, symmetric positive definite (s.p.d.) for all $\mathbf{x}$, $r : \mathbb{R} \to \mathbb{R}$ is a possibly non-linear reaction term and $u_0(\mathbf{x})$ is the initial condition. We impose homogeneous Dirichlet (Neumann) boundary conditions in Γ_D (Γ_N) with $\Gamma_D \cup \Gamma_N = \partial\Omega$, both for simplicity and because, in the linear case, the inhomogeneous case reduces to the homogeneous one by considering a lifting of the boundary data. Setting $r \equiv 0$, equation (2.1) becomes a pure diffusive problem that is often used as a numerical benchmark in the parallel-in-time literature and for space-time solvers. Moreover this choice becomes essential for the analytical results contained in Chapter 3. On the other hand, the action of a reactive term is relevant in many applications. We provide a significant example in Chapter 4. In the next section we provide a short description of the time discretization, somehow unusual, that we are going to use in Section 2.2–2.4, where the space-time weak form, discretization and assembly are presented. In Section 2.5 some relevant error estimates from the literature are collected.

2.1 Time discretization preliminaries

In this section we introduce and compare multiple finite element formulations for the time integration of the generic initial value problem (IVP):

$$\begin{cases} u'(t) = f(u,t) \quad \text{with} \quad t \in [0,T], \\ u(0) = u_0. \end{cases} \tag{2.2}$$

Instead of traditional time stepping techniques we consider a Galerkin approach to discretize (2.2). For a more comprehensive discussion we refer to Eriksson et al. [1996].

2.1.1 Continuous Galerkin weak form

Let us first consider a standard continuous Galerkin (CG) weak formulation: assuming the solution $u(t)$ to be sufficiently regular over $[0,T]$, we multiply (2.2) by a sufficiently smooth test function $v(t) \in \mathscr{W}_{\text{CG}} \subseteq C^0([0,T])$, s.t. $v(0) = 0$, and we integrate over $[0,T]$:

$$\int_0^T u'(t)v(t)\mathrm{d}t = \int_0^T f(u,t)v(t)\mathrm{d}t, \quad \forall v \in \mathscr{W}_{\text{CG}}. \tag{2.3}$$

2.1.2 Discontinuous Galerkin weak form

In the discontinuous case (DG) the solution $u(t)$ can have jumps in $(0,T)$. To illustrate this technique let us consider $u(t)$ in the form

$$u(t) = \begin{cases} u_1(t), & \text{for} \quad t \in [0,t_1], \\ u_2(t), & \text{for} \quad t \in [t_1,T], \end{cases} \tag{2.4}$$

with $t_1 \in (0,T)$ and considering Remark 2.1. The weak derivative of (2.4) is defined as the function $u'(t)$ satisfying

$$\int_0^T u'(t)v(t)\mathrm{d}t = -\int_0^T u(t)v'(t)\mathrm{d}t, \tag{2.5}$$

for every test function $v(t) \in C_0^\infty([0,T])$. Equation (2.5) can be written, using integration by parts, as

$$\begin{aligned}
\int_0^T u'(t)v(t)\mathrm{d}t &= -\int_0^T u(t)v'(t)\mathrm{d}t \\
&= -\int_0^{t_1} u_1(t)v'(t)\mathrm{d}t - \int_{t_1}^T u_2(t)v'(t)\mathrm{d}t \\
&= \int_0^{t_1} u_1'(t)v(t)\mathrm{d}t - u_1(t)v(t)\Big|_0^{t_1} + \int_{t_1}^T u_2'(t)v(t)\mathrm{d}t - u_2(t)v(t)\Big|_{t_1}^T \\
&= \int_0^{t_1} u_1'(t)v(t)\mathrm{d}t - u_1(t_1)v(t_1) + \int_{t_1}^T u_2'(t)v(t)\mathrm{d}t + u_2(t_1)v(t_1) \\
&= \int_0^{t_1} u_1'(t)v(t)\mathrm{d}t + \int_{t_1}^T u_2'(t)v(t)\mathrm{d}t + v(t_1)(u_2(t_1) - u_1(t_1)).
\end{aligned}$$

Therefore

$$u'(t) = \begin{cases} u_1'(t), & \text{for} \quad t \in [0,t_1), \\ (u_2(t) - u_1(t))\delta_{t_1}(t), & \text{for} \quad t = t_1, \\ u_2'(t), & \text{for} \quad t \in (t_1,T], \end{cases} \tag{2.6}$$

where δ_{t_1} is the Dirac delta function. We can generalize this argument to a generic partition of the time axis $0 = t_0 < \cdots < t_n < t_{n+1} < \cdots < t_N = T$. Defining $u(t_n^-)$ and $u(t_n^+)$ for the left/right values of $u(t_n)$, we get the DG weak formulation in $I_n = [t_n, t_{n+1}]$ for the problem (2.2):

$$\int_{t_n}^{t_{n+1}} u'(t)v(t)\mathrm{d}t + (u(t_n^+) - u(t_n^-))v(t_n) = \int_{t_n}^{t_{n+1}} f(u,t)v(t)\mathrm{d}t \tag{2.7}$$

for every sufficiently smooth test function $v(t)$. Equivalently, integrating by parts in (2.7), we obtain the standard DG formulation:

$$-\int_{t_n}^{t_{n+1}} u(t)v'(t)\mathrm{d}t + u(t_{n+1}^-)v(t_{n+1}) - u(t_n^-)v(t_n) = \int_{t_n}^{t_{n+1}} f(u,t)v(t)\mathrm{d}t \tag{2.8}$$

highlighting an *upwind flux* given by the $u(t_n^-)v(t_n)$ term that couples I_n with I_{n-1}. We impose $u(t_0^-) = u_0$ to enforce the initial condition.

2.1.3 Discretization

We consider the following approximation spaces

$$\mathscr{W}_{\mathrm{DG}(q)} = \left\{ v : v|_{[t_n,t_{n+1}]} \in \mathbb{P}_q \text{ for all } n = 0, ..., N-1 \right\}, \tag{2.9}$$

$$\mathscr{W}_{\mathrm{CG}(q)} = \left\{ v \in C^0([0,T]) : v|_{[t_n,t_{n+1}]} \in \mathbb{P}_q \text{ for all } n = 0, ..., N-1 \right\}, \tag{2.10}$$

where $\mathbb{P}_q$ is the space of polynomials of degree less than or equal to q.

Remark 2.1. *Note that the generic element $v \in \mathscr{W}_{DG(q)}$ is not a function from $[0,\mathrm{T}]$ to $\mathbb{R}$ in the true sense of this word, because it takes two values at the nodes. However, for simplicity we will refer to each $v \in \mathscr{W}_{DG(q)}$ as a function without further specifications.*

On each interval I_n, we construct the approximation u_h in the nodal form,

$$u_h(t) = \sum_{m=0}^{q} u_{n,m} \ell_{n,m}(t) \quad \text{for} \quad t \in [t_n, t_{n+1}], \tag{2.11}$$

where $\left\{ \ell_{n,m}(t) \right\}$ is the basis of Lagrange polynomials of degree q for the $q+1$ grid points $\left\{ t_{n,m} \right\}_{m=0,...,q}$ over the interval I_n. In practice we use right Gauss-Radau nodes where just the right endpoint is included:

$$t_n < t_{n,0} < t_{n,1} < \cdots < t_{n,q} = t_{n+1}.$$

This choice is motivated in Remarks 2.2 and 2.5 but is not essential.

Remark 2.2. *As in the original work of Lasaint and Raviart [1974], the numerical approximation for $u(t_{n+1})$ is given by $u_{n+1} = u_h(t_{n+1}^-) = u_{n,q}$. This choice produces the most accurate result w.r.t. other possible choices as $u_n = u_h(t_n^+)$ or $u_n = (u_h(t_n^+) + u_h(t_n^-))/2$. Moreover it is coherent with the initial condition $u_h(0) = u(t_0^-) = u_0$. See Figure 2.1 for a notation summary.*

For simplicity let us consider the discretization of the Dahlquist test IVP i.e. (2.2) with $f(t, u_h(t)) = \lambda u_h(t)$ and $\lambda \in \mathbb{R}$. Transforming I_n into the reference interval $\tau \in [-1, 1]$, we consider the corresponding Lagrangian basis functions $\{\ell_0(\tau), ..., \ell_q(\tau)\}$ in the $q+1$ right Gauss-Radau nodes $-1 < \tau_0 < \tau_1 < \cdots < \tau_q = 1$. We can write the discretization of (2.8) as

$$K_{[q]}\mathbf{u}_{n+1} - \lambda \frac{\Delta t_n}{2} M_{[q]}\mathbf{u}_{n+1} = J_{[q]}\mathbf{u}_n, \tag{2.12}$$

where $\mathbf{u}_{n+1} = [u_{n,0}, u_{n,1}, \ldots, u_{n,q} = u_{n+1}]^T$, $\Delta t_n = t_{n+1} - t_n$ and $M_{[q]}, K_{[q]}, J_{[q]} \in \mathbb{R}^{(q+1)\times(q+1)}$ are given by

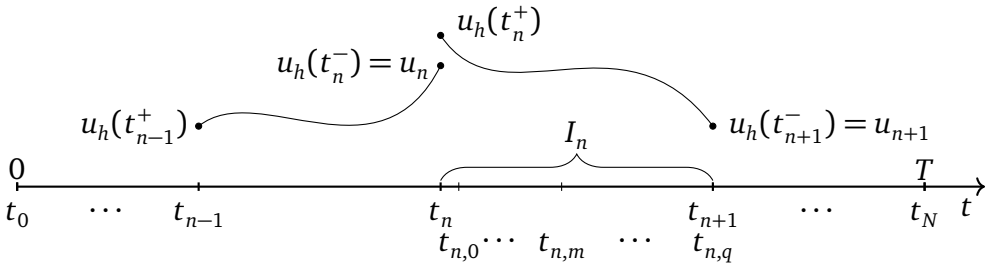

Figure 2.1. The discretization of the time axis and the corresponding notation is shown. The Gauss-Radau points $t_{n,m}$ are depicted for the interval I_n.

$$M_{[q]} = \left[\int_{-1}^{1} \ell_i(\tau)\ell_j(\tau)\mathrm{d}\tau \right]_{i,j=0}^{q}, \tag{2.13}$$

$$K_{[q]} = \left[-\int_{-1}^{1} \ell_i'(\tau)\ell_j(\tau)\mathrm{d}\tau + \ell_i(1)\ell_j(1) \right]_{i,j=0}^{q}, \tag{2.14}$$

$$J_{[q]} = \left[\ell_i(-1)\ell_j(1) \right]_{i,j=0}^{q}, \tag{2.15}$$

and $\mathbf{u}_0 = [0, 0, \ldots, u_0]^T$. Coupling all the equations from (2.12) in a monolithic form, we finally get the linear problem in $\mathbb{R}^{N\times(q+1)}$:

$$\begin{bmatrix} A_{[q]} & & & \\ -J_{[q]} & A_{[q]} & & \\ & \ddots & \ddots & \\ & & -J_{[q]} & A_{[q]} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_N \end{bmatrix} = \begin{bmatrix} J_{[q]}\mathbf{u}_0 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \tag{2.16}$$

with $A_{[q]} = K_{[q]} - \lambda\frac{\Delta t_n}{2}M_{[q]}$. Let us collect some properties of DG methods.

Lemma 2.3. *(Stability.) The DG method* (2.7) *for IVP* (2.2) *is A-stable.*

Proof. See Theorem 2 in Lasaint and Raviart [1974]. □

Lemma 2.4. *(Superconvergence.) Denote by u_h the DG approximation of degree q for the IVP* (2.2) *($f(t) \in C^{2q+1}$) in the interval $I_n = [t_n, t_{n+1}]$, and $\{t_{n,m}\}_{m=0\ldots q}$ the $q+1$ Gauss-Radau points in I_n. Then we have the local truncation error:*

$$u(t_{n,m}) - u_h(t_{n,m}) = \mathcal{O}\left(\Delta t^{q+2}\right), \quad 0 \le m \le q-1,$$

with $\Delta t = \max_n\{\Delta t_n\}$ and at the end point $u_h(t_{n+1}^-) = u_h(t_{n,q})$,

$$u(t_{n+1}) - u_h(t_{n+1}^-) = \mathcal{O}\left(\Delta t^{2q+2}\right).$$

Proof. See Theorem 5 in Adjerid et al. [2002]. □

As a time stepping method, the DG method has two attractive features: Lemma 2.3 provides excellent stability (*A*-stability) and Lemma 2.4 high-order accuracy (the global error is of order $2q+1$ in the nodes).

Remark 2.5. *The Gauss-Radau quadrature requires $q+1$ nodes to reproduce exactly all polynomials of degree $2q$, see Abramowitz and Stegun [1992]. Therefore, due to the superconvergence property given by Lemma 2.4, the right Gauss-Radau points are the proper choice for the integration of the right hand side f. In fact, if the more popular Gauss-Lobatto points are used, one order of accuracy may be lost.*

See Figure 2.2 for an experimental comparison between CG and DG convergence.

Remark 2.6. *The DG discretization, in comparison with the CG one, requires additional $N-2$ degrees of freedom to be stored because the internal nodes are duplicated. On the other hand, its higher accuracy compensates this drawback.*

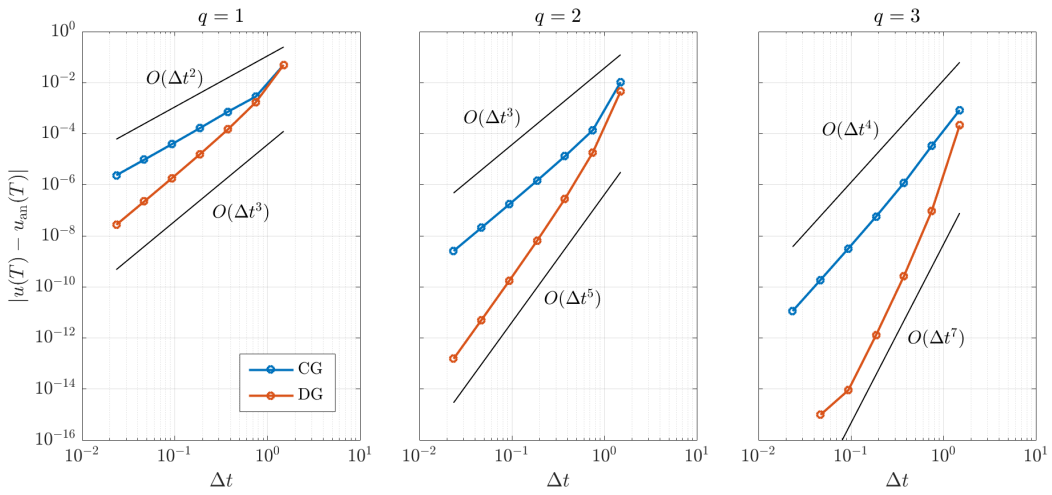

Figure 2.2. Convergence plots of solutions of (2.2) with $f(u,t) = -u(t)$ and $u_0 = 1$ in case of DG and CG discretizations for $q = \{1,2,3\}$. The convergence rates are the expected ones: $2q+1$ in the DG case and $q+1$ for CG. The error w.r.t. the analytical solution u_{an} is computed at the final time $T = 3$.

It can be noticed that the DG discretization for $q = 0$ is equivalent to the one obtained by the implicit Euler method. Moreover solving (2.16) using a block Gauss-Seidel method would be equivalent to standard sequential time stepping. But we are interested in the parallel solutions strategies for the system (2.16);

for this reason we perform some preliminary numerical experiments for parallel block solution methods; results are collected in Figures 2.3–2.4.

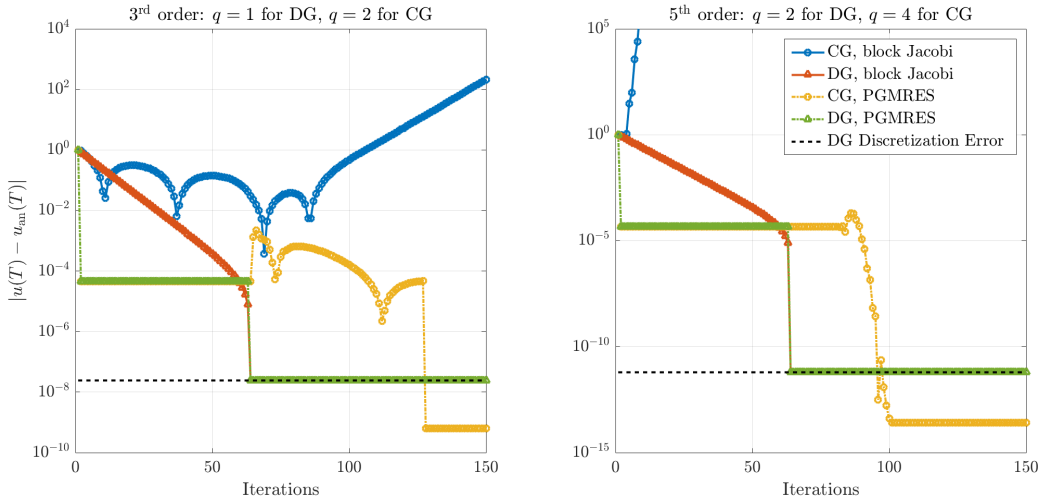

Figure 2.3. Convergence of multiple iterative solvers applied to (2.16) and to its continuous counterpart for a 3^{rd} order method (left) and a 5^{th} order one (right). We used $N = 64$ time steps, $T = 10$ and $\lambda = -1$. For the block Jacobi solver, used also to precondition GMRES, one block per time step is used. Interestingly, we can observe a rapid decrease of the error after a single PGMRES iteration when DG is used. This behavior can be understood in terms of clustering of the eigenvalues of the corresponding discrete operators, shown in Figure 2.4.

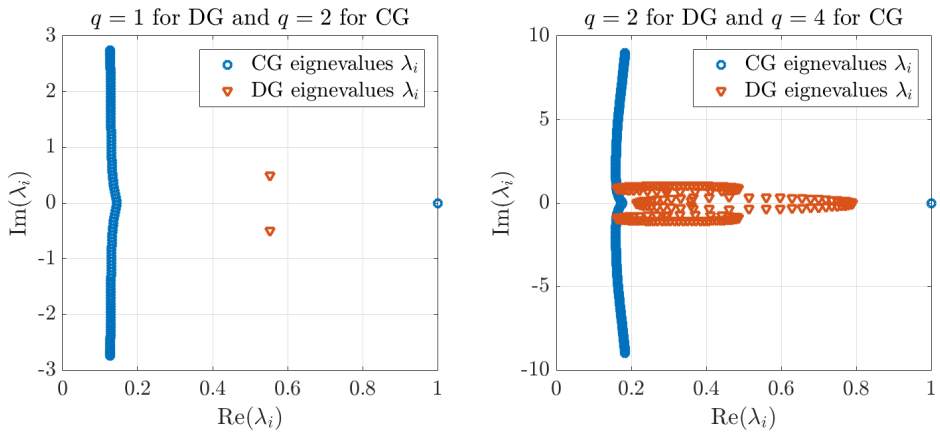

Figure 2.4. Eigenvalues of discrete operators for the CG the DG case, using the same parameters as in Figure 2.3.

Given the high convergence order of the DG discretization, its stability and its superior robustness when combined with a parallel iterative solver (see Figure 2.3), we will use this discretization in the space-time discretization of (2.1).

2.2 Space-time weak formulation

We now introduce the full space-time weak formulation of problem (2.1). For simplicity we initially consider $r(u) = 0$; the treatment of a non-linear reaction term will be included in Chapter 4. Let us consider a uniform partition in time $0 = t_0 < t_1 < \cdots < t_N = T$ with $\Delta t = T/N$. We define the m-th space-time slab $\mathcal{E}^m = [t_m, t_{m+1}] \times \Omega$ for $m = 0, \ldots, N-1$. Assuming the solution $u(t, \mathbf{x})$ to be sufficiently regular over $[0, \mathrm{T}] \times \Omega$, we multiply the PDE in (2.1) by a sufficiently regular test function $v(t, \mathbf{x})$ satisfying the same boundary conditions as $u(t, \mathbf{x})$ and we integrate over $\mathcal{E}^m$:

$$
\begin{aligned}
&\int_{\mathcal{E}^m} \big[\partial_t u(t,\mathbf{x}) - \nabla \cdot K \nabla u(t,\mathbf{x})\big] v(t,\mathbf{x}) \mathrm{d}t\mathrm{d}\mathbf{x} = \int_{\mathcal{E}^m} f(t,\mathbf{x}) v(t,\mathbf{x}) \mathrm{d}t\mathrm{d}\mathbf{x} \\
&\iff \int_\Omega \int_{t_m}^{t_{m+1}} \partial_t u(t,\mathbf{x}) v(t,\mathbf{x}) \mathrm{d}t\mathrm{d}\mathbf{x} - \int_{t_m}^{t_{m+1}} \int_\Omega \nabla \cdot K \nabla u(t,\mathbf{x}) v(t,\mathbf{x}) \mathrm{d}\mathbf{x}\mathrm{d}t \\
&\qquad = \int_{\mathcal{E}^m} f(t,\mathbf{x}) v(t,\mathbf{x}) \mathrm{d}t\mathrm{d}\mathbf{x} \\
&\iff \int_\Omega \Big[u(t,\mathbf{x}) v(t,\mathbf{x}) \Big|_{t_m}^{t_{m+1}} - \int_{t_m}^{t_{m+1}} u(t,\mathbf{x}) \partial_t v(t,\mathbf{x}) \mathrm{d}t \Big] \mathrm{d}\mathbf{x} \\
&\qquad - \int_{t_m}^{t_{m+1}} \Big[\underbrace{\int_{\partial\Omega} v(t,\mathbf{x}) K \nabla u(t,\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}) \mathrm{d}\sigma(\mathbf{x})}_{=0} - \int_\Omega [K \nabla u(t,\mathbf{x})] \cdot \nabla v(t,\mathbf{x}) \mathrm{d}\mathbf{x} \Big] \mathrm{d}t \\
&\qquad = \int_{\mathcal{E}^m} f(t,\mathbf{x}) v(t,\mathbf{x}) \mathrm{d}t\mathrm{d}\mathbf{x}.
\end{aligned}
$$

This means that, for every $m = 0, \ldots, N-1$ and every sufficiently regular test function v satisfying $v(t, \mathbf{x}) = 0$ for $(t, \mathbf{x}) \in (0, \mathrm{T}) \times \partial\Omega$, the solution u satisfies

$$
a_m(u, v) = F_m(v), \tag{2.17}
$$

where

$$a_m(u,v) = -\int_{\mathcal{E}^m} u(t,\mathbf{x})\partial_t v(t,\mathbf{x})\mathrm{d}t\mathrm{d}\mathbf{x} + \int_{\mathcal{E}^m} [K\nabla u(t,\mathbf{x})]\cdot\nabla v(t,\mathbf{x})\mathrm{d}t\mathrm{d}\mathbf{x} + \int_\Omega \left[u(t_{m+1}^-,\mathbf{x})v(t_{m+1},\mathbf{x}) - u(t_m^-,\mathbf{x})v(t_m,\mathbf{x})\right]\mathrm{d}\mathbf{x}, \tag{2.18}$$

$$F_m(v) = \int_{\mathcal{E}^m} f(t,\mathbf{x})v(t,\mathbf{x})\mathrm{d}t\mathrm{d}\mathbf{x}, \tag{2.19}$$

using the DG *upwind flux* and the notation from Section 2.1.

2.3 Space-time discretization

Let us consider an admissible partition $\mathcal{T}_h = \{T_1, ..., T_n\}$ of Ω into n elements and define h as the maximum length of sides in $\mathcal{T}_h$. We define the q-degree DG approximation space (as in (2.9)) and the p-degree C^k FE approximation space as follows, for all $q, p, n \in \mathbb{N}$ and $0 \le k \le p-1$:

$$\mathscr{W}_{N,[q]} = \left\{w : \ w|_{[t_m,t_{m+1}]} \in \mathbb{P}_q \text{ for all } m = 0,\dots,N-1\right\},$$
$$\mathscr{W}_{n,[p,k]} = \left\{w \in C^k(\Omega) : w|_T \in \mathbb{P}_p \text{ for all } T \in \mathcal{T}_h \text{ with } w(\mathbf{x}) = 0 \text{ if } \mathbf{x} \in \Gamma_D\right\},$$

with dimensions

$$\overline{N} = \dim(\mathscr{W}_{N,[q]}) = N(q+1),$$
$$\overline{n} = \dim(\mathscr{W}_{n,[p,k]}).$$

Remark 2.7. *The dimension of $\mathscr{W}_{n,[p,k]}$ is not known in advance but depends on the particular choice of basis. For example, in Section 3.3 we will consider B-splines on a uniform grid with Ω being a cuboid and an expression of $\overline{n}$ will be provided.*

Let $\{\phi_1,\dots,\phi_{\overline{N}}\}$ be a basis for $\mathscr{W}_{N,[q]}$, let $\{\varphi_1,\dots,\varphi_{\overline{n}}\}$ be a basis for $\mathscr{W}_{n,[p,k]}$. Finally we set a space-time approximation space with a tensor structure:

$$\mathscr{W} = \mathscr{W}_{N,[q]} \otimes \mathscr{W}_{n,[p,k]} = \mathrm{span}(\psi_j = \phi_{j_1} \otimes \varphi_{j_2} : \ \boldsymbol{j} = \mathbf{1},\dots,\boldsymbol{N}), \qquad \boldsymbol{N} = (\overline{N},\overline{n}),$$

referring to [Benedusi et al., 2018b] for the multi-index notation.

Remark 2.8. *In the literature the notation cG(p)dG(q) is often used when referring to this discretization technique with $k = 0$.*

We look for an approximation $u_h(t,\mathbf{x})$ of the solution $u(t,\mathbf{x})$ by solving the following discrete problem: find $u_h \in \mathscr{W}$ such that, for all $m = 0,\dots,N-1$ and all $v \in \mathscr{W}$,

$$a_m(u_h, v) = F_m(v), \tag{2.20}$$

where $a_m(u,v)$ and $F_m(v)$ are given by (2.18) and (2.19), respectively. Considering that $\{\psi_j : \boldsymbol{j} = \mathbf{1},\dots,\boldsymbol{N}\}$ is a basis for $\mathscr{W}$, we have $u_h = \sum_{j=1}^{N} u_j\psi_j$ for a unique vector $\mathbf{u} = [u_j]_{j=1}^{N}$ and, by linearity, the computation of u_h reduces to finding $\mathbf{u}$ such that, for all $m = 0,\dots,N-1$,

$$A_m\mathbf{u} = \mathbf{f}_m, \tag{2.21}$$

where

$$\mathbf{f}_m = [F_m(\psi_i)]_{i=1}^{N}, \tag{2.22}$$

$$A_m = \left[a_m(\psi_j,\psi_i)\right]_{i,j=1}^{N}. \tag{2.23}$$

2.4 Space-time matrix assembly

In time we use the Lagrangian nodal basis functions $\{\ell_0,\dots,\ell_q\}$ described in Section 2.1.3 for the reference element $[-1,1]$. In space for $\{\varphi_1,\dots,\varphi_{\bar{n}}\}$ we consider B-splines of degree p and smoothness C^k. In Chapter 3 we will provide a specific description of such a basis for a uniform rectangular grid over Ω. For every

$\boldsymbol{i},\boldsymbol{j}=\mathbf{1},\dots,\boldsymbol{N}$, the $(\boldsymbol{i},\boldsymbol{j})$ entry of the matrix A_m appearing in (2.23) is given by

$$\begin{aligned}
(A_m)_{\boldsymbol{ij}} &= a_m(\psi_{\boldsymbol{j}},\psi_{\boldsymbol{i}}) \\
&= -\int_{\mathcal{E}^m}\psi_{\boldsymbol{j}}(t,\mathbf{x})\partial_t\psi_{\boldsymbol{i}}(t,\mathbf{x})\mathrm{d}t\mathrm{d}\mathbf{x}+\int_{\mathcal{E}^m}[K\nabla\psi_{\boldsymbol{j}}(t,\mathbf{x})]\cdot\nabla\psi_{\boldsymbol{i}}(t,\mathbf{x})\mathrm{d}t\mathrm{d}\mathbf{x} \\
&\quad+\int_\Omega\left[\psi_{\boldsymbol{j}}(t_{m+1}^-,\mathbf{x})\psi_{\boldsymbol{i}}(t_{m+1}^-,\mathbf{x})-\psi_{\boldsymbol{j}}(t_m^-,\mathbf{x})\psi_{\boldsymbol{i}}(t_m^+,\mathbf{x})\right]\mathrm{d}\mathbf{x} \\
&= -\int_{t_m}^{t_{m+1}}\phi_{j_1}(t)\phi_{i_1}'(t)\mathrm{d}t\int_\Omega\varphi_{j_2}(\mathbf{x})\varphi_{i_2}(\mathbf{x})\mathrm{d}\mathbf{x} \\
&\quad+\int_{t_m}^{t_{m+1}}\phi_{j_1}(t)\phi_{i_1}(t)\mathrm{d}t\int_\Omega[K\nabla\varphi_{j_2}(\mathbf{x})]\cdot\nabla\varphi_{i_2}(\mathbf{x})\mathrm{d}\mathbf{x} \\
&\quad+\left[\phi_{j_1}(t_{m+1}^-)\phi_{i_1}(t_{m+1}^-)-\phi_{j_1}(t_m^-)\phi_{i_1}(t_m^+)\right]\int_\Omega\varphi_{j_2}(\mathbf{x})\varphi_{i_2}(\mathbf{x})\mathrm{d}\mathbf{x} \\
&= -\int_{-1}^{1}\ell_{j_1}(\tau)\ell_{i_1}'(\tau)\mathrm{d}\tau\int_\Omega\varphi_{j_2}(\mathbf{x})\varphi_{i_2}(\mathbf{x})\mathrm{d}\mathbf{x} \\
&\quad+\frac{\Delta t}{2}\int_{-1}^{1}\ell_{j_1}(\tau)\ell_{i_1}(\tau)\mathrm{d}\tau\int_\Omega[K\nabla\varphi_{j_2}(\mathbf{x})]\cdot\nabla\varphi_{i_2}(\mathbf{x})\mathrm{d}\mathbf{x} \\
&\quad+\left[\ell_{j_1}(1)\ell_{i_1}(1)-\ell_{j_1}(1)\ell_{i_1}(-1)\right]\int_\Omega\varphi_{j_2}(\mathbf{x})\varphi_{i_2}(\mathbf{x})\mathrm{d}\mathbf{x} \qquad (2.24)
\end{aligned}$$

for all $m=0,\dots,N-1$. We define the standard spatial mass and stiffness matrices $M_{n,[p,k]},K_{n,[p,k]}\in\mathbb{R}^{\overline{n}\times\overline{n}}$ as

$$M_{n,[p,k]}=\left[\int_\Omega\varphi_j(\mathbf{x})\varphi_i(\mathbf{x})\mathrm{d}\mathbf{x}\right]_{i,j=1}^{\overline{n}}, \qquad (2.25)$$

$$K_{n,[p,k]}=\left[\int_\Omega[K\nabla\varphi_j(\mathbf{x})]\cdot\nabla\varphi_i(\mathbf{x})\mathrm{d}\mathbf{x}\right]_{i,j=1}^{\overline{n}}. \qquad (2.26)$$

Finally, as in (2.12), we can rewrite (2.24) using definitions (2.13)–(2.15) and (2.25)–(2.26), for $m=0,\dots,N-1$, as

$$(K_{[q]}\otimes M_{n,[p,k]}+\frac{\Delta t}{2}M_{[q]}\otimes K_{n,[p,k]})\mathbf{u}_{m+1}=(J_{[q]}\otimes M_{n,[p,k]})\mathbf{u}_m+\mathbf{f}_m, \qquad (2.27)$$

with $\mathbf{u}_m, \mathbf{f}_m \in \mathbb{R}^{(q+1)\overline{n}}$ and $\mathbf{u} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_N]$ in (2.21). Similarly to the time problem (2.16), we can write (2.27) in the following monolithic form

$$\begin{bmatrix} A_n^{[q,p,k]} & & & \\ B_n^{[q,p,k]} & A_n^{[q,p,k]} & & \\ & \ddots & \ddots & \\ & & B_n^{[q,p,k]} & A_n^{[q,p,k]} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_N \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 - \mathbf{f}_0 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_N \end{bmatrix} \iff C_{N,n}^{[q,p,k]} \mathbf{u} = \mathbf{f}, \tag{2.28}$$

where $A_n^{[q,p,k]}, B_n^{[q,p,k]} \in \mathbb{R}^{(q+1)\overline{n} \times (q+1)\overline{n}}$ are given by

$$A_n^{[q,p,k]} = K_{[q]} \otimes M_{n,[p,k]} + \frac{\Delta t}{2} M_{[q]} \otimes K_{n,[p,k]}, \tag{2.29}$$

$$B_n^{[q,p,k]} = -J_{[q]} \otimes M_{n,[p,k]}, \tag{2.30}$$

and $C_{N,n}^{[q,p,k]} \in \mathbb{R}^{\overline{N}\overline{n} \times \overline{N}\overline{n}}$

$$C_{N,n}^{[q,p,k]} = \begin{bmatrix} A_n^{[q,p,k]} & & & \\ B_n^{[q,p,k]} & A_n^{[q,p,k]} & & \\ & \ddots & \ddots & \\ & & B_n^{[q,p,k]} & A_n^{[q,p,k]} \end{bmatrix}. \tag{2.31}$$

The initial condition is imposed through the term $\mathbf{f}_0 = (-J_{[q]} \otimes I_{\overline{n}}) \cdot [0, ..., 0, \mathbf{u}_0]^T$ where

$$\mathbf{u}_0 = \left[\int_\Omega u_0(\mathbf{x}) \varphi_i(\mathbf{x}) d\mathbf{x} \right]_{i=1}^{\overline{n}} \in \mathbb{R}^{\overline{n}}. \tag{2.32}$$

2.5 A priori and a posteriori discretization error analysis

We provide the most relevant error estimates for the space-time solution that we use to validate the numerical implementation. We consider problem (2.1) with $K = I_d$ and $r \equiv 0$. Additionally we require Ω to be a convex domain as the estimates we report are based on this assumption. We use the following notation: in $I_m = [t_m, t_{m+1}]$

$$\|u\|_{I_m} = \sup_{t \in I_m} \|u(t)\|, \qquad \|u\|_{k,I_m} = \sup_{t \in I_m} \|u(t)\|_k,$$

where $\|\cdot\|$ is the norm in $L^2(\Omega)$, $\|\cdot\|_k$ the one in $H^k(\Omega)$ and $u(t) = u(t, \cdot)$. Let us consider *a priori* error bounds, i.e. containing quantities depending on the regularity of the analytical solution u.

Theorem 2.9. *We have, for the solution of* (2.20)*, using linear finite elements in space, i.e.* $p = 1$ *and* $k = 0$*, for* $q = \{0, 1\}$ *the estimates at the time nodes* $M = 1, ..., N$

$$\|u_h(t_M) - u(t_M)\| \leq C L_M \max_{1 \leq m \leq M} (h^2 \|u\|_{2,I_m} + \Delta t \|u_t\|_{I_m}) \qquad \text{for } q = 0,$$
$$\|u_h(t_M) - u(t_M)\| \leq C L_M \max_{1 \leq m \leq M} (h^2 \|u\|_{2,I_m} + \Delta t^3 \|u_{tt}\|_{2,I_m}) \qquad \text{for } q = 1,$$

with $L_M = (\log(M))^{1/2} + 1$ *and* C *a positive constant.*

Proof. See Theorem 12.6–12.7 in Thomée [1984] □

Similarly Eriksson and Johnson [1995] proved an estimate for the error $u - u_h$ in $L^\infty([0, T]; L^2(\Omega))$ which is of order $q + 1$ globally in time and $2q + 1$ at the nodes, as we might expect from the discussion in Section 2.1.3. See Figure 2.5 for a numerical test to validate our implementation.

Later on, Makridakis and Babuska [1997] proved an estimate in more general form

$$\|u_h - u\|_{L^2([0,T]\times\Omega)} < C \max_{1 \leq m \leq N} (\|h^{p+1} u\|_{p+1,I_m}) + \max_{1 \leq m \leq N} (\|\Delta t^{q+1} u^{(q+1)}\|_{I_m}).$$

The result holds provided that the weak mesh condition $h^2 \leq \beta \Delta t$, for $\beta > 0$ small enough, holds. A similar estimate can be provided if the reaction term $r(u)$ is not neglected. We refer to Theorem 3.1 from Estep and Larsson [1993] for such an estimate where a dependency on the regularity of $r(u)$ and f is introduced.

A priori estimates provide bounds depending on the regularity of the exact solution u that, in general, is unknown. So they can be used to predict the accuracy of a numerical scheme. On the other hand *a posteriori* bounds depend only on the right hand side f and the computed solution. For this reason they can be used for the design of adaptive schemes. Introducing the interior edges $\{\gamma\}$ of the partition $\mathcal{T}_h$ and for $u_h \in \mathcal{W}_{n,[1,0]}$ let $[\partial u_h / \partial n]_\gamma$ be the jump in the normal derivative across γ; we define the norm

$$\|u_h\|_{2,h}^2 = \sum_\gamma |[\partial u_h / \partial n]_\gamma|^2.$$

Theorem 2.10. *We have, for the solution of* (2.20)*, using linear finite elements in space, i.e.* $p = 1$ *and* $k = 0$*, for* $q = 1$*, the estimates at the time node* $M = 1, ..., N$

$$\|u_h(t_M) - u(t_M)\| \leq C L_M \max_{m \leq M} ((h^2 + \Delta t)\|f\|_{I_m} + \\ + h^2 \|u_h(t_m)\|_{2,h} + \|u_h(t_m) - u_h(t_{m-1})\|).$$

Proof. See Theorem 12.9 in Thomée [1984] □

See Theorem 2.4 by Eriksson and Johnson [1991] for an analogous result for $q = 0$ and the thesis by Sabawi [2018] for a comprehensive study about error estimates for non-linear parabolic problems.

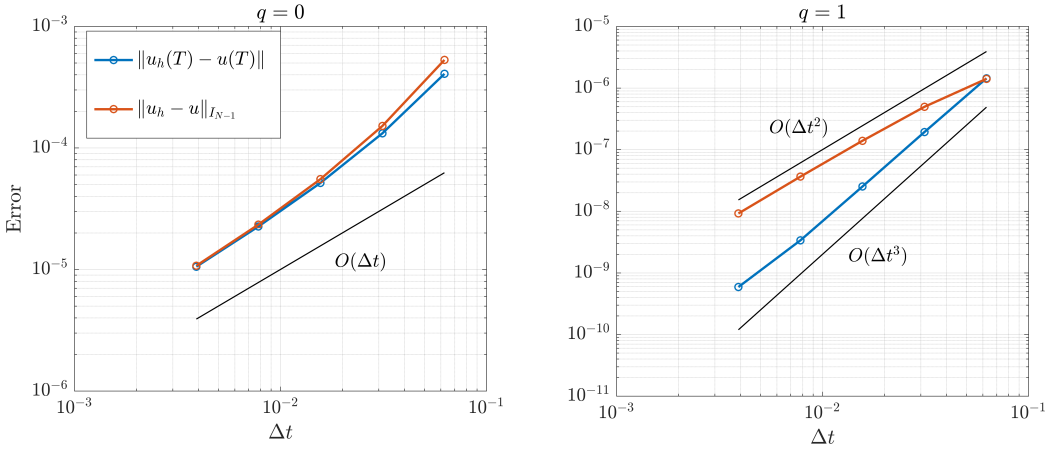

Figure 2.5. Convergence behavior of the numerical solution of problem (2.31) using $p = 1, k = 0, q = \{0, 1\}, T = 1, \Omega = [0, 1]$ and $n = 1500$ grid points in space. We used the initial solution $u_0 = \sin(\pi x)$ and the correspondent analytical solution $u = \sin(\pi x)e^{-\pi^2 t}$. Convergence orders support Theorem 2.9 as the reference slopes show. In particular we considered the $L^2(\Omega)$ error and the final time T at in the final temporal interval I_{N-1}.

Chapter 3

Spectral Tools

Suppose that a linear PDE is discretized by a numerical method and the computation of its numerical solution reduces to the solution of a linear system X_n of size n. What is often observed in practice is that X_n enjoys an asymptotic spectral distribution as $n \to \infty$, i.e., there exists a matrix-valued function $\mathbf{f}$ which describes the asymptotic distribution of the eigenvalues of X_n. We refer to $\mathbf{f}$ as the spectral symbol of the sequence $\{X_n\}_n$.

The spectral information carried by the symbol, is not only interesting from a theoretical viewpoint, but can also be used for practical purposes. For example, it is known that the convergence properties of mainstream iterative solvers, such as multigrid and preconditioned Krylov methods, strongly depend on the spectral features of the matrices to which they are applied. The symbol $\mathbf{f}$ can then be exploited to design efficient solvers of this kind for the matrix X_n, and to analyze/predict their performance. In this regard, we recall that noteworthy estimates on the superlinear convergence of the conjugate gradient method obtained by Beckermann and Kuijlaars [2001] are closely related to the asymptotic spectral distribution of the considered matrices. Furthermore, in the context of Galerkin and collocation isogeometric analysis (IgA) discretizations of elliptic PDEs, the symbol computed in a sequence of recent papers [Donatelli et al., 2016; Garoni, 2018; Garoni et al., 2014; **?**] was exploited in [Donatelli et al., 2015b,a, 2017] to devise and analyze optimal and robust multigrid solvers for IgA linear systems.

In Section 3.1 we present a few algebraic preliminary notions. In Section 3.2, a short description and properties of the spectral symbol are given; we refer to [Garoni and Serra-Capizzano, 2017, 2018] for a complete study. In Section 3.3 we derive the analytic form of the symbol associated with system (2.31), under the mild assumption that the time grid is not *too fine* w.r.t. the spatial one. In

Section 3.4 numerical experiments will be presented to validate the previous theoretical study. In Section 3.5 a study on the condition number of (2.31) is presented.

Sections 3.1–3.4 are covered and depth in [Benedusi et al., 2018b], where a rigorous and more general analysis is presented. We remark that in this chapter we considered the spatial domain Ω as d-dimensional cuboid discretized on a regular grid; this condition is necessary to apply the symbol theory, largely based on (block) Toeplitz matrices. Nevertheless, as discussed at the end of the chapter, the estimates we obtain can be useful also when Ω is not a cuboid.

3.1 Prelimaries

3.1.1 Matrix norms

For all $X \in \mathbb{C}^{m\times m}$ the eigenvalues and singular values of X are denoted by $\lambda_j(X)$, $j = 1,\dots,m$, and $\sigma_j(X)$, $j = 1,\dots,m$, respectively. The ∞-norm and the 2-norm (spectral norm) of both vectors and matrices are denoted by $\|\cdot\|_\infty$ and $\|\cdot\|$, respectively. We recall, e.g. from [Garoni and Serra-Capizzano, 2017, Section 2.4.1], that

$$\|X\| \le \sqrt{\|X\|_\infty \|X^T\|_\infty}, \qquad \forall X \in \mathbb{C}^{m\times m}. \tag{3.1}$$

For $X \in \mathbb{C}^{m\times m}$, let $\|X\|_1$ be the trace-norm (or Schatten 1-norm) of X, i.e., the sum of all the singular values of X. Since $\operatorname{rank}(X)$ is the number of nonzero singular values of X and $\|X\|$ is the maximal singular value of X, we have

$$\|X\|_1 \le \operatorname{rank}(X)\|X\| \le m\|X\|, \qquad \forall X \in \mathbb{C}^{m\times m}. \tag{3.2}$$

3.1.2 Tensor products

Tensor (Kronecker) products possess a lot of nice algebraic properties. One of them is the associativity, which allows one to omit parentheses in expressions like $X_1 \otimes X_2 \otimes \cdots \otimes X_d$. Another property is the bilinearity: for each matrix $X \in \mathbb{C}^{m_1\times m_2}$, the application $Y \mapsto X \otimes Y$ is linear on $\mathbb{C}^{\ell_1\times\ell_2}$ for all $\ell_1, \ell_2 \in \mathbb{N}$; and for each matrix $Y \in \mathbb{C}^{\ell_1\times\ell_2}$, the application $X \mapsto X \otimes Y$ is linear on $\mathbb{C}^{m_1\times m_2}$ for all $m_1, m_2 \in \mathbb{N}$. If X_1, X_2 can be multiplied and Y_1, Y_2 can be multiplied, then

$$(X_1 \otimes Y_1)(X_2 \otimes Y_2) = (X_1X_2) \otimes (Y_1Y_2). \tag{3.3}$$

For all matrices X, Y, we have $(X \otimes Y)^* = X^* \otimes Y^*$ and $(X \otimes Y)^T = X^T \otimes Y^T$. In particular, if X, Y are Hermitian (resp., symmetric) then $X \otimes Y$ is also Hermitian

(resp., symmetric). If $X \in \mathbb{C}^{m\times m}$ and $Y \in \mathbb{C}^{\ell\times\ell}$, the eigenvalues and singular values of $X \otimes Y$ are, respectively, $\{\lambda_i(X)\lambda_j(Y) : \ i = 1,\dots,m,\ j = 1,\dots,\ell\}$ and $\{\sigma_i(X)\sigma_j(Y) : \ i = 1,\dots,m,\ j = 1,\dots,\ell\}$; see, e.g., [Garoni and Serra-Capizzano, 2017, Exercise 2.5]. In particular, for all $X \in \mathbb{C}^{m\times m}$ and $Y \in \mathbb{C}^{\ell\times\ell}$, we have

$$\|X \otimes Y\| = \|X\|\,\|Y\|. \tag{3.4}$$

If $X_\ell \in \mathbb{C}^{m_\ell\times m_\ell}$ for $\ell = 1,\dots,d$, then

$$(X_1 \otimes X_2 \otimes \cdots \otimes X_d)_{\boldsymbol{ij}} = (X_1)_{i_1 j_1}(X_2)_{i_2 j_2}\cdots(X_d)_{i_d j_d}, \quad \boldsymbol{i},\boldsymbol{j} = \mathbf{1},\dots,\boldsymbol{m}, \tag{3.5}$$

where $\boldsymbol{m} = (m_1, m_2, \dots, m_d)$. For every $\boldsymbol{m} = (m_1, m_2) \in \mathbb{N}^2$ there exists a permutation matrix $\Pi_{\boldsymbol{m}}$ of size $m_1 m_2$ such that

$$X_2 \otimes X_1 = \Pi_{\boldsymbol{m}}(X_1 \otimes X_2)\Pi_{\boldsymbol{m}}^T \tag{3.6}$$

for all matrices $X_1 \in \mathbb{C}^{m_1\times m_1}$ and $X_2 \in \mathbb{C}^{m_2\times m_2}$; see, e.g., [Garoni et al., 2015a, Lemma 1].

3.2 The spectral symbol

We say that a matrix-valued function $\mathbf{f} : D \to \mathbb{C}^{s\times s}$, defined on a measurable set $D \subseteq \mathbb{R}^\ell$, is measurable (resp., is continuous, belongs to $L^p(D)$) if its components $f_{ij} : D \to \mathbb{C}$, $i,j = 1,\dots,s$, are measurable (resp., are continuous, belong to $L^p(D)$). Moreover, we say that $\mathbf{f}$ is Hermitian (resp., symmetric) if $\mathbf{f}(\mathbf{y})$ is Hermitian (resp., symmetric) for all $\mathbf{y} \in D$. We denote by μ_ℓ the Lebesgue measure in $\mathbb{R}^\ell$ and by $C_c(\mathbb{R})$ (resp., $C_c(\mathbb{C})$) the set of continuous complex-valued functions with compact support defined over $\mathbb{R}$ (resp., $\mathbb{C}$).

Definition 1. *Let $\{X_n\}_n$ be a sequence of matrices, with X_n of size d_n tending to infinity, and let $\mathbf{f} : D \to \mathbb{C}^{s\times s}$ be a measurable matrix-valued function defined on a set $D \subset \mathbb{R}^\ell$ with $0 < \mu_\ell(D) < \infty$. We say that $\{X_n\}_n$ has an asymptotic spectral distribution described by $\mathbf{f}$, and we write $\{X_n\}_n \sim_\lambda \mathbf{f}$, if*

$$\lim_{n\to\infty}\frac{1}{d_n}\sum_{j=1}^{d_n}F(\lambda_j(X_n)) = \frac{1}{\mu_\ell(D)}\int_D \frac{\sum_{i=1}^{s}F\left(\lambda_i(\mathbf{f}(\mathbf{y}))\right)}{s}\mathrm{d}\mathbf{y}, \qquad \forall\, F \in C_c(\mathbb{C}).$$

In this case, $\mathbf{f}$ is referred to as the spectral symbol of the sequence $\{X_n\}_n$.

Whenever we write a spectral distribution relation such as $\{X_n\}_n \sim_\lambda \mathbf{f}$, it is understood that $\{X_n\}_n$ and $\mathbf{f}$ are as in Definition 1.

Remark 3.1. *The informal meaning behind Definition* 1 *is the following: assuming that* $\mathbf{f}$ *possesses* s *Riemann-integrable eigenvalue functions* $\lambda_i(\mathbf{f}(\mathbf{y}))$, $i = 1, \dots, s$, *the eigenvalues of* X_n, *except possibly for* $o(d_n)$ *outliers, can be subdivided into* s *different subsets of approximately the same cardinality; and the eigenvalues belonging to the* i*th subset are approximately equal to the samples of the* i*th eigenvalue function* $\lambda_i(\mathbf{f}(\mathbf{y}))$ *over a uniform grid in the domain* D. *For instance, if* $\ell = 1$, $d_n = ns$, *and* $D = [a, b]$, *then, assuming we have no outliers, the eigenvalues of* X_n *are approximately equal to*

$$\lambda_i\Big(\mathbf{f}\Big(a + j\frac{b-a}{n}\Big)\Big), \quad j = 1, \dots, n, \quad i = 1, \dots, s,$$

for n *large enough; similarly, if* $\ell = 2$, $d_n = n^2 s$, *and* $D = [a_1, b_1] \times [a_2, b_2]$, *then, assuming we have no outliers, the eigenvalues of* X_n *are approximately equal to*

$$\lambda_i\Big(\mathbf{f}\Big(a_1 + j_1\frac{b_1-a_1}{n}, a_2 + j_2\frac{b_2-a_2}{n}\Big)\Big), \quad j_1, j_2 = 1, \dots, n, \quad i = 1, \dots, s,$$

for n *large enough; and so on for* $\ell \geq 3$.

Remark 3.2. *Let* $D = [a_1, b_1] \times \cdots \times [a_\ell, b_\ell] \subset \mathbb{R}^\ell$ *and let* $\mathbf{f} : D \to \mathbb{C}^{s\times s}$ *be a measurable function possessing* s *real-valued Riemann-integrable eigenvalue functions* $\lambda_i(\mathbf{f}(\mathbf{y}))$, $i = 1, \dots, s$. *Compute for each* $r \in \mathbb{N}$ *the uniform samples*

$$\lambda_i\Big(\mathbf{f}\Big(a_1 + j_1\,\frac{b_1-a_1}{r}, \dots, a_\ell + j_\ell\,\frac{b_\ell-a_\ell}{r}\Big)\Big), \quad j_1, \dots, j_\ell = 1, \dots, r, \quad i = 1, \dots, s,$$

sort them in non-decreasing order and put them in a vector $(\varsigma_1, \varsigma_2, \dots, \varsigma_{sr^\ell})$. *Let* $\kappa_r : [0, 1] \to \mathbb{R}$ *be the piecewise linear non-decreasing function that interpolates the samples* $(\varsigma_0 = \varsigma_1, \varsigma_1, \varsigma_2, \dots, \varsigma_{sr^\ell})$ *over the nodes* $(0, \frac{1}{sr^\ell}, \frac{2}{sr^\ell}, \dots, 1)$, *i.e.,*

$$\begin{cases} \kappa_r\Big(\dfrac{i}{sr^\ell}\Big) = \varsigma_i, \quad i = 0, \dots, sr^\ell, \\ \kappa_r \text{ linear on } \left[\dfrac{i}{sr^\ell}, \dfrac{i+1}{sr^\ell}\right] \text{for } i = 0, \dots, sr^\ell - 1. \end{cases}$$

Then κ_r *converges a.e. over* $[0, 1]$ *to some function* κ *as* $r \to \infty$, *and*

$$\int_0^1 F(\kappa(y))\mathrm{d}y = \frac{1}{\mu_\ell(D)}\int_D \frac{\sum_{i=1}^s F(\lambda_i(\mathbf{f}(\mathbf{y})))}{s}\mathrm{d}\mathbf{y}, \qquad \forall\, F \in C_c(\mathbb{C}). \tag{3.7}$$

This result can be proved by adapting the arguments used in [Garoni and Serra-Capizzano, 2017, *solution of Exercise* 3.1] and [Barbarino, 2017]. *The function*

κ is referred to as the canonical rearranged version of $\mathbf{f}$*. What is interesting about κ is that, by* (3.7)*, if $\{X_n\}_n \sim_\lambda \mathbf{f}$ then $\{X_n\}_n \sim_\lambda \kappa$, i.e., if $\mathbf{f}$ is a spectral symbol of $\{X_n\}_n$ then κ is a spectral symbol of $\{X_n\}_n$ as well. Moreover, κ is a univariate scalar function and hence it is much easier to handle than $\mathbf{f}$. For a visual comparison between $\mathbf{f}$ and κ see Figures 3.2 and 3.3.*

Two very useful tools for determining spectral distributions are the following; see [Garoni et al., 2015b, Theorem 3.3] for the first one and [Mazza et al., 2019, Theorem 4.3] for the second one.

Theorem 3.3. *Let $\{X_n\}_n, \{Y_n\}_n$ be sequences of matrices, with $X_n, Y_n \in \mathbb{C}^{d_n \times d_n}$ and d_n tending to infinity as $n \to \infty$, and assume the following.*

1. *Every X_n is Hermitian and $\{X_n\}_n \sim_\lambda \mathbf{f}$.*
2. *$\|X_n\|, \|Y_n\| \le C$ for all n, with C a constant independent of n.*
3. *$\|Y_n\|_1 = o(d_n)$ as $n \to \infty$.*

Then $\{X_n + Y_n\}_n \sim_\lambda \mathbf{f}$.

Theorem 3.4. *Let $\{X_n\}_n$ be a sequence of Hermitian matrices, with $X_n \in \mathbb{C}^{d_n \times d_n}$ and d_n tending to infinity as $n \to \infty$, and let $\{P_n\}_n$ be a sequence of matrices, with $P_n \in \mathbb{C}^{d_n \times \delta_n}$ such that $P_n^* P_n = I_{\delta_n}$ and $\delta_n \le d_n$ such that $\delta_n/d_n \to 1$ as $n \to \infty$. Then,*

$$\{X_n\}_n \sim_\lambda \mathbf{f} \quad \Longleftrightarrow \quad \{P_n^* X_n P_n\}_n \sim_\lambda \mathbf{f}.$$

Another result of interest herein is stated in the next lemma [Benedusi et al., 2018a, Lemma 2.6]. Throughout this section, for any $\boldsymbol{s} \in \mathbb{N}^d$ and any functions $\mathbf{f}_i : D_i \to \mathbb{C}^{s_i \times s_i}$, $i = 1, \dots, d$, the tensor-product function $\mathbf{f}_1 \otimes \cdots \otimes \mathbf{f}_d : D_1 \times \cdots \times D_d \to \mathbb{C}^{P(\boldsymbol{s}) \times P(\boldsymbol{s})}$ is defined as

$$(\mathbf{f}_1 \otimes \cdots \otimes \mathbf{f}_d)(\zeta_1, \dots, \zeta_d) = \mathbf{f}_1(\zeta_1) \otimes \cdots \otimes \mathbf{f}_d(\zeta_d), \qquad (\zeta_1, \dots, \zeta_d) \in D_1 \times \cdots \times D_d.$$

Lemma 3.5. *Let $\{X_n\}_n, \{Y_n\}_n$ be sequences of Hermitian matrices, with $X_n \in \mathbb{C}^{d_n \times d_n}$, $Y_n \in \mathbb{C}^{\delta_n \times \delta_n}$, and both d_n and δ_n tending to infinity as $n \to \infty$. Assume $\|X_n\|, \|Y_n\| \le C$ for all n and for some constant C independent of n. Let $\mathbf{f} : D \subseteq \mathbb{R}^\ell \to \mathbb{C}^{r \times r}$ and $\mathbf{g} : E \subseteq \mathbb{R}^\ell \to \mathbb{C}^{s \times s}$ be measurable Hermitian matrix-valued functions, with $0 < \mu_\ell(D) < \infty$ and $0 < \mu_\ell(E) < \infty$. Then,*

$$\{X_n\}_n \sim_\lambda \mathbf{f}, \quad \{Y_n\}_n \sim_\lambda \mathbf{g} \quad \Longrightarrow \quad \{X_n \otimes Y_n\}_n \sim_\lambda \mathbf{f} \otimes \mathbf{g}.$$

3.3 The space-time symbol

Let us consider equation (2.1) on the spatial domain $\Omega = (0,1)^d$ where Ω could be replaced by any rectangular domain without affecting the essence of this section. We partition the spatial domain using n elements per dimension and we define $h = 1/n$. Let $B_{1,[p,k]}, \dots, B_{n(p-k)+k+1,[p,k]}$ be the B-splines of degree p and smoothness C^k defined on the knot sequence

$$\Big\{\underbrace{0,\dots,0}_{p+1}, \underbrace{\frac{1}{n},\dots,\frac{1}{n}}_{p-k}, \underbrace{\frac{2}{n},\dots,\frac{2}{n}}_{p-k}, \dots, \underbrace{\frac{n-1}{n},\dots,\frac{n-1}{n}}_{p-k}, \underbrace{1,\dots,1}_{p+1}\Big\}.$$

We define the spatial basis $\{\varphi_1,\dots,\varphi_{\bar{n}}\} = \{\varphi_{\mathbf{1}},\dots,\varphi_{\boldsymbol{n(p-k)+k-1}}\}$ for $\mathscr{W}_{n,[p,k]}$ from Section 2.2 as tensor product of the B-splines in each dimension:

$$\begin{aligned}\varphi_{\boldsymbol{s}} &= B_{\boldsymbol{s}+\mathbf{1},[p,k]} \\ &= B_{s_1+1,[p,k]} \otimes \cdots \otimes B_{s_d+1,[p,k]}, \qquad \boldsymbol{s} = \mathbf{1},\dots,\boldsymbol{n(p-k)+k-1},\end{aligned}$$

where bold indices represent d-multi-indices, for example for $d = 3$ we have $\boldsymbol{n} = (n,n,n)$ and similarly for $\boldsymbol{p}$ and $\boldsymbol{k}$. For the formal definition of the B-splines, as well as their properties, we refer the reader to [Schumaker, 2007]. See Figure 3.1 for an example of B-splines graphs.

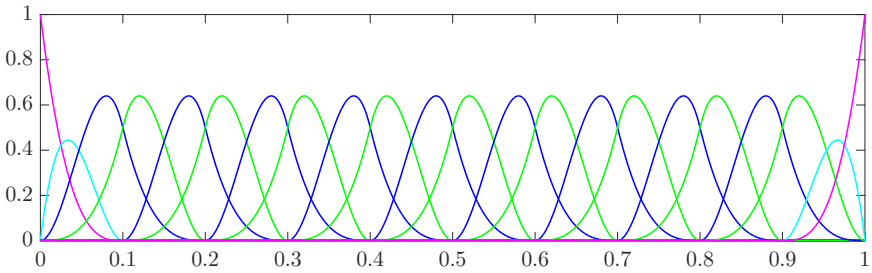

Figure 3.1. B-splines $B_{1,[p,k]},\dots,B_{n(p-k)+k+1,[p,k]}$ of degree $p = 3$ and smoothness $k = 1$ for $n = 10$.

In this setting the mass and stiffness matrices from (2.25)–(2.26) are given by:

$$M_{n,[p,k]} = \left[\int_{[0,1]^d} B_{\boldsymbol{j}+\mathbf{1},[p,k]}(\mathbf{x}) B_{\boldsymbol{i}+\mathbf{1},[p,k]}(\mathbf{x})\mathrm{d}\mathbf{x}\right]_{\boldsymbol{i},\boldsymbol{j}=\mathbf{1}}^{\boldsymbol{n(p-k)+k-1}}, \tag{3.8}$$

$$K_{n,[p,k]} = \left[\int_{[0,1]^d} \left[K(\mathbf{x})\nabla B_{\boldsymbol{j}+\mathbf{1},[p,k]}(\mathbf{x})\right]\cdot\nabla B_{\boldsymbol{i}+\mathbf{1},[p,k]}(\mathbf{x})\mathrm{d}\mathbf{x}\right]_{\boldsymbol{i},\boldsymbol{j}=\mathbf{1}}^{\boldsymbol{n(p-k)+k-1}}. \tag{3.9}$$

and the total number of degrees of freedom in space is

$$\overline{n} = \dim(\mathscr{W}_{n,[p,k]}) = (n(p-k)+k-1)^d.$$

We report a useful estimate regarding the spatial matrices (3.8)–(3.9) that is going to be used later on.

Lemma 3.6. *For $0 \leq k \leq p-1$, $K = I_d$ and $d = 1$ we have*

$$\|nM_{n,[p,k]}\|_\infty, \|n^{-1}K_{n,[p,k]}\|_\infty \leq C_p$$

for some constant C_p depending only on p.

Proof. See Lemma 4.3 from [Benedusi et al., 2018b]. □

We now report the main result from Benedusi et al. [2018b], i.e., the theorem defining the symbol correspondent to space-time matrices.

Theorem 3.7. *Let $q, p, k \geq 0$ be integers with $0 \leq k \leq p-1$. Let $C_{N,n}^{[q,p,k]}$ be the space-time matrix defined in* (2.31) *with spatial matrices from* (3.8) *and* (3.9)*. Suppose that the diffusion coefficient $K : (0,1)^d \to \mathbb{R}^{d\times d}$ is a symmetric matrix-valued function in $L^\infty((0,1)^d)$ and that the following condition is met:*

$$N = N(n) \text{ is such that } N \to \infty \text{ and } N/n^2 \to 0 \text{ as } n \to \infty.$$

Then, for the sequence of normalized space-time matrices $\{2Nn^{d-2}C_{N,n}^{[q,p,k]}\}_n$ we have the spectral distribution relation

$$\{2Nn^{d-2}C_{N,n}^{[q,p,k]}\}_n \sim_\lambda \mathbf{f}_{[q,p,k]},$$

where:

- *the spectral symbol $\mathbf{f}_{[q,p,k]} : [0,1]^d \times [-\pi,\pi]^d \to \mathbb{C}^{(q+1)(p-k)^d \times (q+1)(p-k)^d}$ is defined as*

$$\mathbf{f}_{[q,p,k]}(\mathbf{x},\boldsymbol{\theta}) = \mathbf{f}_{[p,k]}(\mathbf{x},\boldsymbol{\theta}) \otimes TM_{[q]}; \tag{3.10}$$

- *$\mathbf{f}_{[p,k]} : [0,1]^d \times [-\pi,\pi]^d \to \mathbb{C}^{(p-k)^d \times (p-k)^d}$ is defined as*

$$\mathbf{f}_{[p,k]}(\mathbf{x},\boldsymbol{\theta}) = \sum_{i,j=1}^{d} K_{ij}(\mathbf{x})(H_{[\boldsymbol{p},\boldsymbol{k}]})_{ij}(\boldsymbol{\theta}); \tag{3.11}$$

- *$H_{[p,k]}$ is a $d \times d$ block matrix whose (i,j) entry is a $(p-k)^d \times (p-k)^d$ block defined as in* [Benedusi et al., 2018b, *Eq.* (5.12)]*;*

- *T is the final time in* (2.1) *and $M_{[q]}$ is given in* (2.13).

Proof. From [Benedusi et al., 2018b] see Theorem 5.1 for a proof where $K = I_d$ and Theorem 5.2 for the general result. □

Let us give an informal description of the proof of Theorem 3.7 for $d = 1$. For $d > 1$ the proof is relying on the decomposition of (3.8) and (3.9) in tensor products of one dimensional operators, and, apart from this extra technicality, it is the same as in the case of $d = 1$. The proof is based on the following decomposition of $C_{N,n}^{[q,p,k]}$

$$2Nn^{-1}C_{N,n}^{[q,p,k]} = X_{N,n}^{[q,p,k]} + Y_{N,n}^{[q,p,k]},$$

with $X_{N,n}^{[q,p,k]}$ block diagonal and $Y_{N,n}^{[q,p,k]}$ block bi-diagonal:

$$X_{N,n}^{[q,p,k]} = 2Nn^{-1}\frac{\Delta t}{2} I_N \otimes M_{[q]} \otimes K_{n,[p,k]} = Tn^{-1} I_N \otimes M_{[q]} \otimes K_{n,[p,k]}, \tag{3.12}$$

$$Y_{N,n}^{[q,p,k]} = 2Nn^{-1}(I_N \otimes K_{[q]} \otimes M_{n,[p,k]} - L_N \otimes J_{[q]} \otimes M_{n,[p,k]}) \tag{3.13}$$

$$= 2Nn^{-1}(I_N \otimes K_{[q]} \otimes M_{n,[p,k]} + L_N \otimes B_n^{[q,p,k]}), \tag{3.14}$$

with L_N being a lower shift matrix of size N. By Lemma 3.6, property (3.4) and the equation $\|L_N\| = 1$, we have

$$\|X_{N,n}^{[q,p,k]}\| \le C, \quad \|Y_{N,n}^{[q,p,k]}\| \le CN/n^2,$$

where C denotes a generic positive constant independent of n and N. Since $N/n^2 \to 0$ by assumption, we have $\|Y_{N,n}^{[q,p,k]}\| \to 0$ as $n \to \infty$. Hence, by (3.2),

$$\|Y_{N,n}^{[q,p,k]}\|_1 \le \|Y_{N,n}^{[q,p,k]}\| \overline{n}\overline{N} = o(\overline{n}\overline{N}).$$

Since $X_{N,n}^{[q,p,k]}$ is symmetric, by Theorem 3.3 the thesis is proved if we show that

$$\{X_{N,n}^{[q,p,k]}\}_n \sim_\lambda \mathbf{f}_{[q,p,k]}. \tag{3.15}$$

By (3.6), there exists a permutation matrix $\Pi_{\overline{n},\overline{N}}$, depending only on $\overline{n}$ and $\overline{N}$, such that

$$\begin{aligned} X_{N,n}^{[q,p,k]} &= Tn^{-1} I_N \otimes M_{[q]} \otimes K_{n,[p,k]} \\ &= \Pi_{\overline{n},\overline{N}}\big(n^{-1} K_{n,[p,k]} \otimes I_N \otimes TM_{[q]}\big)\Pi_{\overline{n},\overline{N}}^T. \end{aligned}$$

Proving (3.15) is then equivalent to proving that

$$\{n^{-1} K_{n,[p,k]} \otimes I_N \otimes TM_{[q]}\}_n \sim_\lambda \mathbf{f}_{[q,p,k]} = \mathbf{f}_{[p,k]} \otimes TM_{[q]}. \tag{3.16}$$

By Lemma 3.5 the spectral distribution (3.16) is established as soon as we have proved that the symbol of the IgA stiffness matrices is given by

$$\{n^{-1}K_{n,[p,k]}\}_n \sim_\lambda \mathbf{f}_{[p,k]}. \tag{3.17}$$

See, for example, [Barbarino et al., 2020b,a] for a comprehensive study on this topic and in particular [Barbarino et al. [2020a], Theorem 6.8 and Remark 6.9]). The proof of Theorem 3.7, as well as the previous discussion, leads naturally to the following result, that will be essential for the design of a suitable block preconditioner for $C_{N,n}^{[q,p,k]}$.

Corollary 3.8. *Suppose the hypotheses of Theorem* 3.7 *are valid, and* $X_{N,n}^{[q,p,k]}$ *is defined as in* (3.12)*. Then,*

$$\{2Nn^{d-2}(I_N \otimes A_n^{[q,p,k]}\}_n \sim_\lambda \mathbf{f}_{[q,p,k]},$$
$$\{X_{N,n}^{[q,p,k]}\}_n \sim_\lambda \mathbf{f}_{[q,p,k]}.$$

Corollary 3.8 ensures that, when assumptions of Theorem 3.7 are met, the block diagonal matrix $I_N \otimes A_n^{[q,p,k]}$ is spectrally equivalent to the original operator. The same is true for $X_{N,n}^{[q,p,k]}$, the symmetric matrix containing just the spatial problem from $I_N \otimes A_n^{[q,p,k]}$.

3.4 Numerical experiments

In this section we validate Theorem 3.7 through numerical examples that illustrate the effectiveness of the spectral symbol $\mathbf{f}_{[q,p,k]}$ in describing the asymptotic spectral distribution of the space–time discretization matrix $C_{N,n}^{[q,p,k]}$. In Section 3.4.1 we focus on 1D examples. In Section 3.4.2 we present a 2D example. In Section 3.4.3 we give a hint on how to use the symbol in order to design an efficient solver for $C_{N,n}^{[q,p,k]}$. In all the examples of this section we consider the purely diffusive problem with $T = 1$ and homogeneous Dirichlet boundary condition in space. The diffusion coefficient $K(x)$ and the parameters N, n, q, p, k are specified in each example.

3.4.1 1D experiments

Since $d = 1$ and $T = 1$, the symbol of the sequence $\{2Nn^{-1}C_{N,n}^{[q,p,k]}\}_n$ under the assumptions of Theorem 3.7 is given by

$$\mathbf{f}_{[q,p,k]} : [0,1] \times [-\pi,\pi] \to \mathbb{C}^{(q+1)(p-k)\times(q+1)(p-k)},$$
$$\mathbf{f}_{[q,p,k]}(x,\theta) = K(x)\mathbf{f}_{[p,k]}(\theta) \otimes M_{[q]}, \tag{3.18}$$

where $\mathbf{f}_{[p,k]}(\theta)$ is defined in (3.11) and $M_{[q]}$ is defined in (2.13).

Remark 3.9. *Let $\lambda_1(\mathbf{f}_{[p,k]}(\theta)) \le \ldots \le \lambda_{p-k}(\mathbf{f}_{[p,k]}(\theta))$ be the eigenvalues of $\mathbf{f}_{[p,k]}(\theta)$ sorted in non-decreasing order. Since $\mathbf{f}_{[p,k]}(-\theta) = (\mathbf{f}_{[p,k]}(\theta))^T$ has the same eigenvalues as $\mathbf{f}_{[p,k]}(\theta)$, each eigenvalue function $\lambda_i(\mathbf{f}_{[p,k]}(\theta))$ is symmetric in θ, i.e., it is an even function. Therefore, also the eigenvalue functions of the symbol $\mathbf{f}_{[q,p,k]}(x,\theta)$ in* (3.18)*, namely $K(x)\lambda_i(\mathbf{f}_{[p,k]}(\theta))\lambda_j(M_{[q]})$, $i = 1,\ldots,p-k$, $j = 1,\ldots,q+1$, are symmetric in θ. Thus, the restriction*

$$\begin{aligned} &\mathbf{f}_{[q,p,k]} : [0,1]\times[0,\pi] \to \mathbb{C}^{(q+1)(p-k)\times(q+1)(p-k)}, \\ &\mathbf{f}_{[q,p,k]}(x,\theta) = K(x)\mathbf{f}_{[p,k]}(\theta)\otimes M_{[q]}, \end{aligned} \tag{3.19}$$

is again a spectral symbol for $\{2Nn^{-1}C_{N,n}^{[q,p,k]}\}_n$ according to Definition 1*. Furthermore, if $K(x) = 1$ then $\mathbf{f}_{[q,p,k]}(x,\theta)$ does not depend on the variable x and the restriction*

$$\begin{aligned} &\mathbf{f}_{[q,p,k]} : [0,\pi] \to \mathbb{C}^{(q+1)(p-k)\times(q+1)(p-k)}, \\ &\mathbf{f}_{[q,p,k]}(\theta) = \mathbf{f}_{[p,k]}(\theta)\otimes M_{[q]}, \end{aligned} \tag{3.20}$$

is again a spectral symbol for $\{2Nn^{-1}C_{N,n}^{[q,p,k]}\}_n$ according to Definition 1*. In all the examples of this section, when referring to the spectral symbol of the sequence $\{2Nn^{-1}C_{N,n}^{[q,p,k]}\}_n$, we mean either* (3.19) *or* (3.20)*, depending on whether $K(x)$ is non-constant or $K(x) = 1$.*

Example 3.10. *Let $K(x) = 1$ and take $q = 1$, $p = 1$, $k = 0$. In this basic case, a direct computation shows that the symbol* (3.20) *simplifies to*

$$\mathbf{f}_{[1,1,0]}(\theta) = (2-2\cos\theta)\begin{bmatrix} 3/2 & 0 \\ 0 & 1/2 \end{bmatrix}.$$

Its eigenvalue functions are given by

$$\lambda_1(\mathbf{f}_{[1,1,0]}(\theta)) = 1-\cos\theta, \qquad \lambda_2(\mathbf{f}_{[1,1,0]}(\theta)) = 3-3\cos\theta.$$

Figure 3.2 *shows the graphs of the eigenvalue functions over $[0,\pi]$ and the eigenvalues $\lambda_1,\ldots,\lambda_{2N(n-1)}$ of $2Nn^{-1}C_{N,n}^{[1,1,0]}$ for $N = 5$ and $n = 20$. The eigenvalues, which in this case are all real, are sorted so as to match the graphs of the eigenvalue functions and are represented by the dots placed at the points*

$$\begin{aligned} &(\theta_j,\lambda_j), && j = 1,\ldots,N(n-1), \\ &(\theta_j,\lambda_{j+N(n-1)}), && j = 1,\ldots,N(n-1), \end{aligned}$$

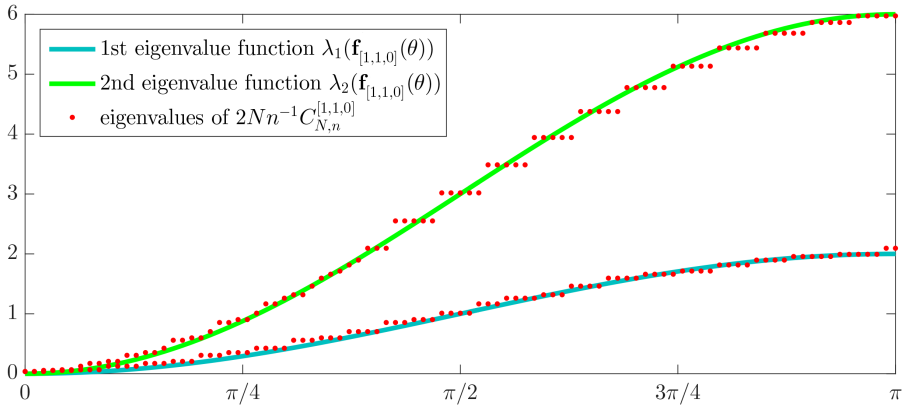

Figure 3.2. Example 3.10: Comparison between the spectrum of $2Nn^{-1}C_{N,n}^{[1,1,0]}$ and the symbol $\mathbf{f}_{[1,1,0]}(\theta)$ for $N=5$ and $n=20$.

where

$$\theta_j = \frac{(j-1)\pi}{N(n-1)-1}, \qquad j=1,\dots,N(n-1).$$

We clearly see from Figure 3.2 that, although N and n are not so large, the eigenvalues of $2Nn^{-1}C_{N,n}^{[1,1,0]}$ can be subdivided into 2 subsets of the same cardinality $N(n-1)$; and the eigenvalues belonging to the 1st (resp., 2nd) subset are approximately equal to the samples of the 1st (resp., 2nd) eigenvalue function over the uniform grid θ_j, $j=1,\dots,N(n-1)$. This agrees with the informal meaning of the spectral distribution $\{2Nn^{-1}C_{N,n}^{[1,1,0]}\}_n \sim_\lambda \mathbf{f}_{[1,1,0]}$ given in Remark 3.1. In particular, we observe no outliers in this case.

Example 3.11. *Let $K(x)=1$ and take $q=1$, $p=2$, $k=1$. In this case, the symbol (3.20) becomes*

$$\mathbf{f}_{[1,2,1]}(\theta) = \Big(1-\frac{2}{3}\cos\theta-\frac{1}{3}\cos(2\theta)\Big)\begin{bmatrix} 3/2 & 0 \\ 0 & 1/2 \end{bmatrix}.$$

Let κ be the canonical rearranged version of $\mathbf{f}_{[1,2,1]}(\theta)$ obtained as the limit of the piecewise linear functions κ_r, according to the construction of Remark 3.2. Then, by Remark 3.2, κ is again a spectral symbol for the sequence $\{2Nn^{-1}C_{N,n}^{[1,2,1]}\}_n$. Figure 3.3 shows the graph of κ and the real parts $\varrho_1,\dots,\varrho_{2Nn}$ of the eigenvalues of $2Nn^{-1}C_{N,n}^{[1,2,1]}$ for $N=35$ and $n=70$. The graph of κ has been obtained by plotting the graph of κ_r corresponding to a large value of r. The real parts of the eigenvalues have been sorted in non-decreasing order and placed at the points

$$(y_j,\varrho_j), \qquad y_j=\frac{j}{2Nn}, \qquad j=1,\dots,2Nn.$$

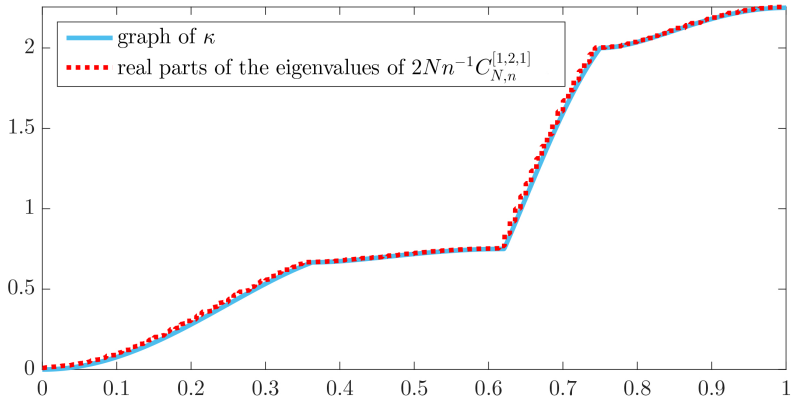

Figure 3.3. Comparison between the spectrum of $2Nn^{-1}C_{N,n}^{[1,2,1]}$ and the canonical rearranged version κ of the symbol $\mathbf{f}_{[1,2,1]}(\theta)$ for $N = 35$ and $n = 70$.

The imaginary parts of the eigenvalues are not shown in the figure because they are negligible (their maximum modulus is about 0.008*). We clearly see from the figure an excellent agreement between* κ *and the eigenvalues of* $2Nn^{-1}C_{N,n}^{[1,2,1]}$*. As illustrated in Figure* 3.4 *and Table* 3.1*, the agreement becomes perfect in the limit of mesh refinement* $2N = n \to \infty$*. Both Figure* 3.4 *and Table* 3.1 *show that* $\|\kappa(\mathbf{y}) - \boldsymbol{\varrho}\|_\infty \to 0$ *and* $\|\boldsymbol{\iota}\|_\infty \to 0$ *as* $2N = n \to \infty$*, where* $\kappa(\mathbf{y}) - \boldsymbol{\varrho} = (\kappa(y_1) - \varrho_1, \ldots, \kappa(y_{2Nn}) - \varrho_{2Nn})$ *is the vector of the errors and* $\boldsymbol{\iota} = (\iota_1, \ldots, \iota_{2Nn})$ *is the vector of the imaginary parts of the eigenvalues of* $2Nn^{-1}C_{N,n}^{[1,2,1]}$*.*

Table 3.1. Computation of the error $\|\kappa(\mathbf{y}) - \boldsymbol{\varrho}\|_\infty$ and the maximum modulus $\|\boldsymbol{\iota}\|_\infty$ of the imaginary parts of the eigenvalues of $2Nn^{-1}C_{N,n}^{[1,2,1]}$ for increasing values of $2N = n$.

$2N = n$	20	30	40	50	60	70
$\|\kappa(\mathbf{y}) - \boldsymbol{\varrho}\|_\infty$	0.2910	0.2010	0.1533	0.1239	0.1039	0.0895
$\|\boldsymbol{\iota}\|_\infty$	0.0208	0.0166	0.0133	0.0109	0.0093	0.0081

Example 3.12. *Let* $K(x) = 3 - \cos(10x)$ *and* $q = 2$*,* $p = 3$*,* $k = 2$*. The symbol* (3.19) *becomes*

$$\mathbf{f}_{[2,3,2]}(x,\theta) = K(x)\Big(\frac{2}{3} - \frac{1}{4}\cos\theta - \frac{2}{5}\cos(2\theta) - \frac{1}{60}\cos(3\theta)\Big)\times$$
$$\times \frac{1}{18}\begin{bmatrix} 16+\sqrt{6} & 0 & 0 \\ 0 & 16-\sqrt{6} & 0 \\ 0 & 0 & 4 \end{bmatrix}.$$

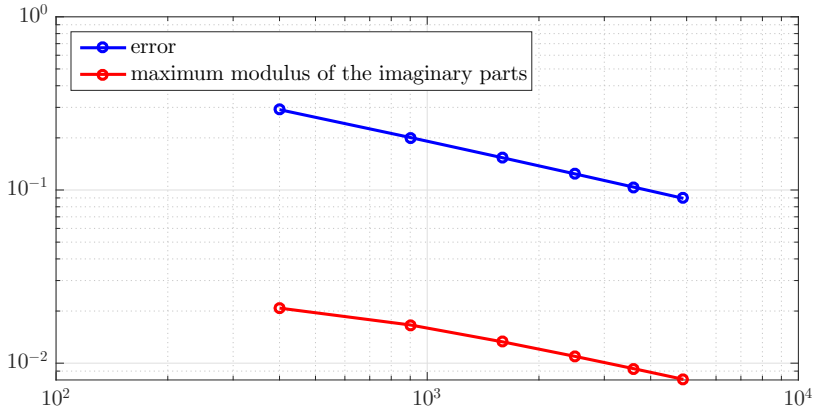

Figure 3.4. Graphs of the error $\|\kappa(\mathbf{y})-\boldsymbol{\varrho}\|_\infty$ and the maximum modulus $\|\boldsymbol{\iota}\|_\infty$ of the imaginary parts of the eigenvalues of $2Nn^{-1}C_{N,n}^{[1,2,1]}$ versus the matrix size $2Nn$ for increasing values of $2N = n$.

Figure 3.5 *shows the graph of the canonical rearranged version* κ *of* $\mathbf{f}_{[2,3,2]}(x,\theta)$ *and the real parts* $\varrho_1,\dots,\varrho_{3N(n+1)}$ *of the eigenvalues of* $2Nn^{-1}C_{N,n}^{[2,3,2]}$ *for* $N = n = 30$. *The real parts of the eigenvalues have been sorted in non-decreasing order and placed at the points*

$$(y_j,\varrho_j),\qquad y_j=\frac{j}{3N(n+1)},\qquad j=1,\dots,3N(n+1).$$

The imaginary parts of the eigenvalues are not shown in the figure because they are negligible (their maximum modulus is about 0.069*). We clearly see from the figure a good matching between* κ *and the eigenvalues of* $2Nn^{-1}C_{N,n}^{[2,3,2]}$. *In this case, however, we also note the appearance of a few outliers at the right end of the spectrum. Nevertheless, the total number of outliers is* 59*, which is negligible with respect to the matrix size* $3N(n+1) = 2790$. *Moreover, when passing from* $N = n = 30$ *to* $N = n = 40$*, the matrix size passes from* 2790 *to* 4920 *but the number of outliers only passes from* 59 *to* 79. *Further numerical experiments reveal that the number of outliers should be equal to* $2n-1$ *for all* N,n *such that* $N = n$*, and this is in agreement with Remark* 3.1*, according to which the number of outliers divided by the matrix size goes to* 0 *as the matrix size goes to to* ∞.

3.4.2 2D experiments

The next example provides a validation of Theorem 3.7 in the case $d = 2$.

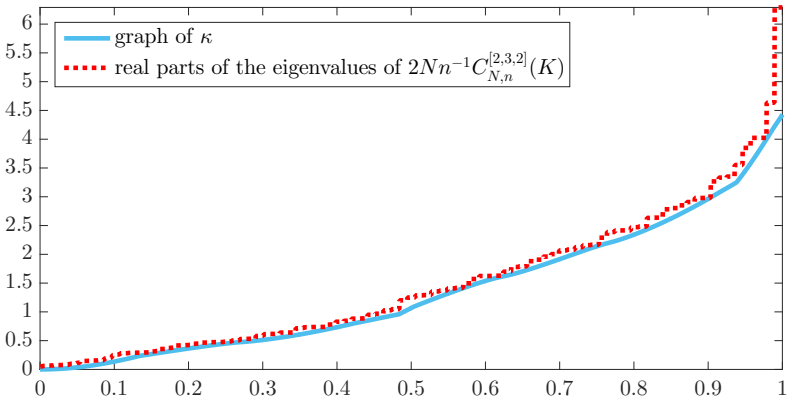

Figure 3.5. Comparison between the spectrum of $2Nn^{-1}C_{N,n}^{[2,3,2]}(K)$ and the canonical rearranged version κ of the symbol $\mathbf{f}_{[2,3,2]}(x,\theta)$ for $N=n=30$ and $K(x)=3-\cos(10x)$.

Example 3.13. *We consider anisotropic diffusion:*

$$K(x_1,x_2)=\begin{bmatrix}\cos(x_1)+x_2 & 0\\ 0 & x_1+\sin(x_2)\end{bmatrix}. \tag{3.21}$$

Let $q=2$, $p=1$ and $k=0$. The spectral symbol of the sequence $\{2NC_{N,n}^{[2,1,0]}\}_n$ under the assumptions of Theorem 3.7 is given by

$$\begin{aligned}&\mathbf{f}_{[2,1,0]}:[0,1]^2\times[0,\pi]^2\to\mathbb{C}^{3\times3}\\ &\mathbf{f}_{[2,1,0]}(x_1,x_2,\theta_1,\theta_2)=\Big[(\cos(x_1)+x_2)(2-2\cos(\theta_1))\Big(\frac{2}{3}+\frac{1}{3}\cos(\theta_2)\Big)\\ &\qquad+(x_1+\sin(x_2))\Big(\frac{2}{3}+\frac{1}{3}\cos(\theta_1)\Big)(2-2\cos(\theta_2))\Big]\times\\ &\qquad\times\frac{1}{18}\begin{bmatrix}16+\sqrt{6} & 0 & 0\\ 0 & 16-\sqrt{6} & 0\\ 0 & 0 & 4\end{bmatrix},\end{aligned} \tag{3.22}$$

where the restriction of the domain to $[0,1]^2\times[0,\pi]^2$ has the same motivation that we have seen in Remark 3.9 (the matrices $\mathbf{f}_{[2,1,0]}(x_1,x_2,\pm\theta_1,\pm\theta_2)$ have the same eigenvalues as $\mathbf{f}_{[2,1,0]}(x_1,x_2,\theta_1,\theta_2)$ for all $(\theta_1,\theta_2)\in[0,\pi]^2$). Figure 3.6 shows the graph of the canonical rearranged version κ of (3.22) and the real parts $\varrho_1,\dots,\varrho_{3N(n-1)^2}$ of the eigenvalues of $2NC_{N,n}^{[2,1,0]}$ for $N=5$ and $n=30$. The real parts of the eigenvalues have been sorted in non-decreasing order and placed at the points

$$(y_j,\varrho_j),\qquad y_j=\frac{j-1}{3N(n-1)^2-1},\qquad j=1,\dots,3N(n-1)^2.$$

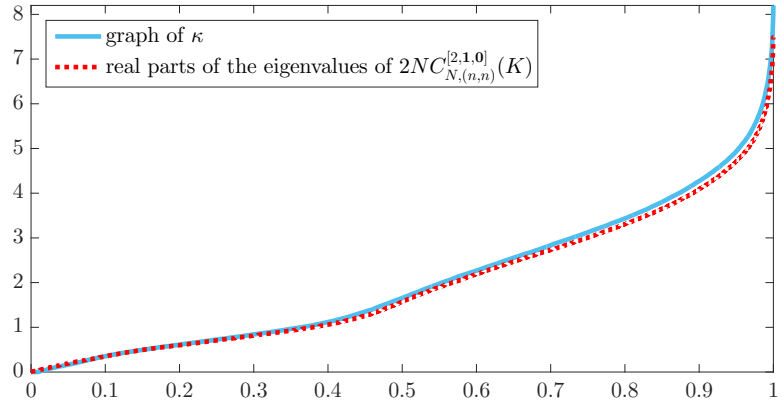

Figure 3.6. Comparison between the spectrum of $2NC^{[2,1,0]}_{N,n}$ and the canonical rearranged version κ of the symbol $\mathbf{f}_{[2,1,0]}(x_1, x_2, \theta_1, \theta_2)$ for $N = 5$, $n = 30$, and $K(x_1, x_2)$ as in (3.21).

The imaginary parts of the eigenvalues are not shown in the figure because they are negligible (their maximum modulus is about 0.011*). We clearly see from the figure an excellent agreement between* κ *and the eigenvalues of* $2NC^{[2,1,0]}_{N,n}$.

3.4.3 Preconditioning

In our final example, we show how to use the symbol in order to design an efficient block preconditioner for the space–time matrix $C^{[q,p,k]}_{N,n}$. The content of this section will be further developed in Section 5.2, where a detailed description of the preconditioned solver will be presented.

Example 3.14. *From Corollary 3.8 and the theory of (block) GLT sequences Garoni and Serra-Capizzano [2017, 2018]; Barbarino et al. [2020a] we expect that the sequence of preconditioned matrices*

$$(I_N \otimes A^{[q,p,k]}_{N,n})^{-1} C^{[q,p,k]}_{N,n},$$

as well as the sequence of preconditioned matrices

$$(X^{[q,p,k]}_{N,n})^{-1} (2N n^{d-2} C^{[q,p,k]}_{N,n}),$$

has an asymptotic spectral distribution described by the preconditioned symbol

$$(\mathbf{f}_{[q,p,k]})^{-1} \mathbf{f}_{[q,p,k]} = I.$$

Therefore, in view of the convergence properties of the GMRES method Saad [2003] *and the meaning of spectral symbol (see Remark* 3.1*), we may predict that the preconditioned GMRES (PGMRES) with preconditioner*

$$I_N \otimes A_{N,n}^{[q,p,k]} \quad \text{or with} \quad P_{N,n}^{[q,p,k]} = \frac{\Delta t}{2} I_N \otimes M_{[q]} \otimes K_{n,[p,k]}$$

for solving a linear system with coefficient matrix $C_{N,n}^{[q,p,k]}$ has an optimal convergence rate, i.e., the number of iterations for reaching a preassigned accuracy ε is independent of (or only weakly dependent on) the matrix size.

To illustrate this claim, consider the heat equation in two space dimensions $d = 2$, with forcing term $f(t,\mathbf{x}) = 1$, $K = I$ and final time $T = 1$. We focus on the linear system $C_{N,n}^{[q,p,k]}\mathbf{u} = \mathbf{f}$ arising from the space–time FE–DG discretization of this equation in the case where $q = 0$, $p = 2$, $k = 1$ and $N = n$. We solve this system, up to a precision $\varepsilon = 10^{-6}$, by means of the GMRES, the PGMRES with preconditioner $I_N \otimes A_{N,n}^{[q,p,k]}$, and the PGMRES with preconditioner $P_{N,n}^{[q,p,k]}$. The resulting number of iterations are collected in Table 3.2 *and graphically represented in Figure* 3.7*. We see that, while GMRES rapidly deteriorates with N, the convergence rates of the PGMRES methods show a much better dependence on N as the growth of the number of iterations is sublinear with respect to N and hence very mild with respect to the system size Nn^2. Moreover, in this case, the number of iterations are essentially the same for both the preconditioners $P_{N,n}^{[q,p,k]}$ and $I_N \otimes A_{N,n}^{[q,p,k]}$. Considering that $P_{N,n}^{[q,p,k]}$ is a pure tensor product whereas $I_N \otimes A_{N,n}^{[q,p,k]}$ is not (because $A_{N,n}^{[q,p,k]}$ is not), we suggest using $P_{N,n}^{[q,p,k]}$. Indeed, the solution of a linear system with coefficient matrix $P_{N,n}^{[q,p,k]}$ reduces to the solution of three linear systems with coefficient matrices I_N, $M_{[q]}$, and $K_{n,[p,k]}$, respectively. Clearly, the only difficult task is the solution of a linear system with coefficient matrix $K_{n,[p,k]}$. The details of such a solver will be presented in Section 5.2.*

Table 3.2. Number of iterations for solving, up to a precision of 10^{-6}, the linear system $C_{N,n}^{[q,p,k]}\mathbf{u} = \mathbf{f}$ arising from the space–time FE–DG discretization of the heat equation in the case where $d = 2$, $f(\mathbf{x},t) = 1$, $T = 1$, $q = 0$, $p = 2$, $k = 1$ and N = n.

$N = n$	10	20	30	40	50	60
system size Nn^2	10^3	20^3	30^3	40^3	50^3	60^3
GMRES	22	52	87	126	170	218
PGMRES with preconditioner $I_N \otimes A_{N,n}^{[q,p,k]}$	10	19	25	32	38	45
PGMRES with preconditioner $P_{N,n}^{[q,p,k]}$	11	19	25	32	38	45

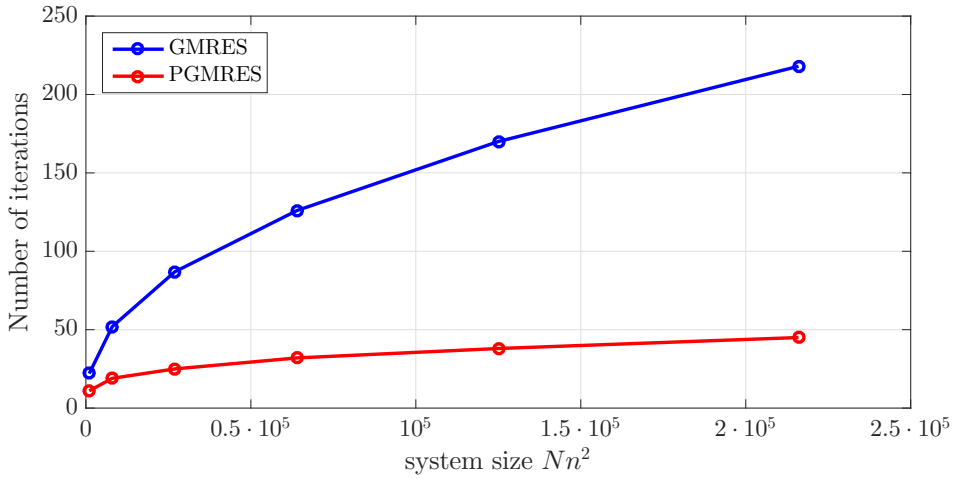

Figure 3.7. Number of iterations for solving, up to a precision of 10^{-6}, the same linear system as in Table 3.2 versus the system size Nn^2 for the same values of $N = n$ as in Table 3.2. The considered preconditioner for PGMRES is $P_{N,n}^{[q,p,k]}$.

3.5 On the condition number of space-time matrices

Let us define the discretization parameter (or CFL parameter) $\mu \in \mathbb{R}$

$$\mu = \frac{K\Delta t}{h^2}, \tag{3.23}$$

where, for simplicity, the diffusion coefficient K will be considered as a positive constant in this context. The parameter μ arises naturally from the discretization of the heat equation (cf. Remark 3.17) and is pivotal for the stability analysis. For example explicit Euler time stepping is stable for $\mu < 1/2$; see [Quarteroni et al., 2010] for a generic discussion on stability of θ-methods depending on μ. On the other hand in [Gander and Neumüller, 2016; Horton and Vandewalle, 1995; Franco et al., 2018] multilevel space-time solvers stability is evaluated w.r.t. μ. For this reason it makes sense to investigate the dependency of the condition number of the space-time matrix $\kappa(C_{N,n}^{[q,p,k]})$ on μ. Figures 3.8–3.9 show the condition number of both the space-time system and the time slab block $\kappa(A_n^{[q,p,k]})$, and the convergence behavior of various iterative solvers w.r.t. μ.

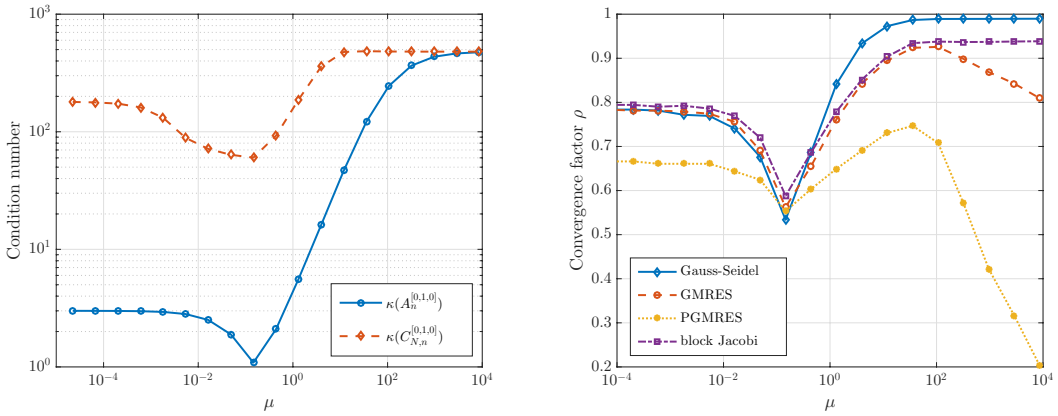

Figure 3.8. We consider the heat equation with $d = 1$, $K = 1$, $N = n = 30$, $p = 1$, $k = 0$, $q = 0$ and $\Delta t = \{3^{-13}, 3^{-12}, ..., 3^0\}$. Left: condition number of $C_{N,n}^{[q,p,k]}$ and the block $A_n^{[q,p,k]}$. Right: Convergence factor of various iterative solvers with relative $\mathsf{tol} = 10^{-9}$. We used sixty blocks for block Jacobi, used also for preconditioning GMRES. All the graphs in the two plots have minima close to $\mu = 0.1$.

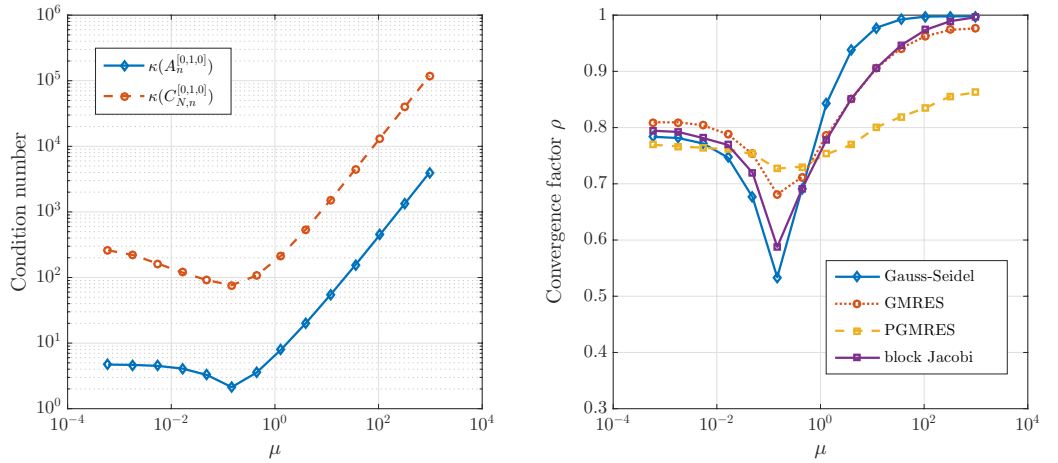

Figure 3.9. Same as Figure 3.8 with homogeneous Neumann boundary conditions instead of homogeneous Dirichlet.

Remark 3.15. *Figures 3.8–3.9 suggest that the condition number* $\kappa(A_n^{[q,p,k]})$ *is a good indicator for the behavior of* $\kappa(C_{N,n}^{[q,p,k]})$. *They seem to share the same* $\arg\min$,

that we identify with μ_{opt},i.e.:

$$\mu_{opt} = \arg\min_{\mu} \kappa(A_n^{[q,p,k]})(\mu) \simeq \arg\min_{\mu} \kappa(C_{N,n}^{[q,p,k]})(\mu).$$

In particular, from Figure 3.8, we can notice how the agreement between $\kappa(A_n^{[q,p,k]})$ and $\kappa(C_{N,n}^{[q,p,k]})$ is especially good for $\mu \to \infty$ as we expect from the assumptions of Theorem 3.7 (being $N/n^2 \to 0 \iff \mu \to \infty$) and Corollary 3.8; from a practical point of view, the case of $\mu \to \infty$ is the most relevant because of the definition of μ.

Remark 3.16. *Many approaches to solve $C_{N,n}^{[q,p,k]}$ rely on the application of block solvers to the diagonal blocks $A_n^{[q,p,k]}$; see, e.g., [McDonald and Wathen, 2016] and Section 3.4.3. For this reason the study of $\kappa(A_n^{[q,p,k]})$ is a relevant topic in itself.*

Remark 3.17. *In view of Lemma 3.6, if we define the following normalized matrices*

$$\widetilde{M}_{n,[p,k]} = nM_{n,[p,k]}, \qquad \widetilde{K}_{n,[p,k]} = n^{-1}K_{n,[p,k]},$$

considering that $\mu = K\Delta t \cdot n^2$, we can rewrite (2.29) as

$$A_n^{[q,p,k]} = \frac{1}{n}(K_{[q]} \otimes \widetilde{M}_{n,[p,k]} + \frac{\mu}{2} M_{[q]} \otimes \widetilde{K}_{n,[p,k]}). \tag{3.24}$$

For example, for $q = 0$ (i.e. implicit Euler) we get $K_{[0]} = 1$ and $M_{[0]} = 2$ and therefore

$$A_n^{[0,p,k]} = \frac{1}{n}(\widetilde{M}_{n,[p,k]} + \mu\widetilde{K}_{n,[p,k]}). \tag{3.25}$$

Then the conditioning of $A_n^{[0,p,k]}$ is going to depend on the interplay of the eigenvalues of the spatial mass and stiffness matrices trough the scaling factor μ. See Figure 3.10 for a comparison of the two spectra. Such idea can be applied also for a generic q, as it will be discussed at the end of the current section.

In some particular setting we can provide a more precise characterization of μ_{opt}. In the following theorem we give an analytical description of $\kappa(A_n^{[0,1,0]})(\mu)$. We refer to [Ekström et al., 2018] for the analytical representation of eigenvalues and eigenvectors of IgA operators that are going to be used in the proofs.

Theorem 3.18. *The condition number $\kappa(A_n^{[0,1,0]})(\mu)$ attains its minimum for $\mu_{opt} = \frac{1}{6}$ and $\kappa(A_n^{[0,1,0]})(\mu_{opt}) = 1$.*

Proof. We can describe precisely the eigenvalues of $K_{n,[1,0]}$ and $M_{n,[1,0]}$:

$$\lambda_j(K_{n,[1,0]}) = 2n(1-\cos(\theta_j)), \quad \lambda_j(M_{n,[1,0]}) = \frac{2+\cos(\theta_j)}{3n}, \tag{3.26}$$

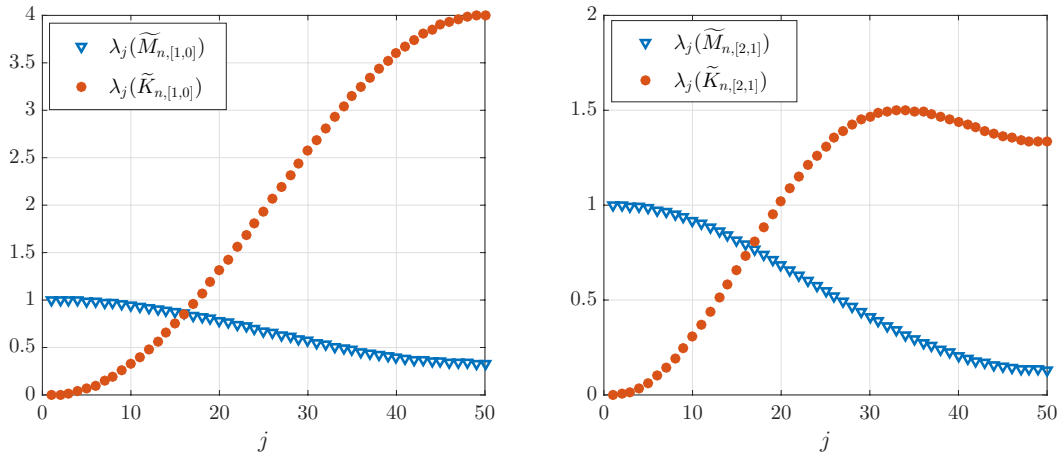

Figure 3.10. Eigenvalues of $\widetilde{M}_{n,[p,k]}$ and $\widetilde{K}_{n,[p,k]}$ for $d = 1$, $n = 50$, $p = \{1,2\}$ and $k = p-1$. For $p < 3$ the two matrices have the same eigenvectors and thus it makes sense to compare their eigenvalues.

for $\theta_j = \frac{j\pi}{n}$, with $j = 1, \dots, n-1$. Because $K_{n,[1,0]}$ and $M_{n,[1,0]}$ share the same eigenvectors, we can sum up the expressions in (3.26) according to (3.25) to get the eigenvalues of $A_n^{[0,1,0]}$:

$$\lambda_j(A_n^{[0,1,0]})(\mu) = \frac{1}{n}\left[\frac{2+\cos(\theta_j)}{3} + 2\mu(1-\cos(\theta_j))\right].$$

Being $A_n^{[0,1,0]}$ symmetric, we can compute an analytical expression of its condition number as

$$\kappa(A_n^{[0,1,0]})(\mu) = \frac{\max\limits_{j=1,\dots,n-1} \lambda_j(A_n^{[0,1,0]})(\mu)}{\min\limits_{j=1,\dots,n-1} \lambda_j(A_n^{[0,1,0]})(\mu)} = \frac{\max\limits_{j=1,\dots,n-1} \frac{2+\cos(\theta_j)}{3} + 2\mu(1-\cos(\theta_j))}{\min\limits_{j=1,\dots,n-1} \frac{2+\cos(\theta_j)}{3} + 2\mu(1-\cos(\theta_j))}. \tag{3.27}$$

Rearranging the min and max arguments, for any j we have

$$\frac{2+\cos(\theta_j)}{3} + 2\mu(1-\cos(\theta_j)) = \frac{2}{3} + 2\mu + \cos(\theta_j)\left(\frac{1}{3} - 2\mu\right),$$

showing that for $\mu_{\text{opt}} = \frac{1}{6}$ the dependency on θ_j is removed in both the numerator and the denominator of (3.27). Therefore

$$\kappa(A_n^{[0,1,0]})(\mu_{\text{opt}}) = \frac{\max\limits_{j=1,\dots,n-1} \frac{2}{3} + 2\mu_{\text{opt}}}{\min\limits_{j=1,\dots,n-1} \frac{2}{3} + 2\mu_{\text{opt}}} = \frac{\frac{2}{3} + 2\mu_{\text{opt}}}{\frac{2}{3} + 2\mu_{\text{opt}}} = 1. \tag{3.28}$$

Since $\kappa(A_n^{[0,1,0]})(\mu) \geq 1$ for all μ, μ_{opt} is a minimum of (3.27).

□

Remark 3.19. *Note that $A_n^{[0,1,0]}(\mu)$ becomes a multiple of the identity for $\mu = 1/6$:*

$$A_n^{[0,1,0]}(\mu) = \frac{1}{n}\begin{bmatrix} 4+2\mu & 1/6-\mu & & \\ 1/6-\mu & 4+2\mu & 1/6-\mu & \\ \ddots & \ddots & \ddots & \\ & 1/6-\mu & 4+2\mu & 1/6-\mu \\ & & 1/6-\mu & 4+2\mu \end{bmatrix}.$$

We can apply the same argument in the case of larger p. For example for $p = 2$ we get the following relation (again, we refer to [Ekström et al., 2018] for the formula of the eigenvalues of $K_{n,[2,1]}$ and $M_{n,[2,1]}$)

$$\begin{aligned}\lambda_j(A_n^{[0,2,1]}) &= \frac{1}{n}\left[\mu\left(1-\frac{2}{3}\cos(\theta_j)-\frac{1}{3}\cos(2\theta_j)\right)+\frac{11}{20}+\frac{13}{30}\cos(\theta_j)+\frac{1}{60}\cos(2\theta_j)\right] \\ &= \frac{1}{n}\left[\mu+\frac{11}{20}+\cos(\theta_j)\left(\frac{13}{30}-\frac{2}{3}\mu\right)+\cos(2\theta_j)\left(\frac{1}{60}-\frac{\mu}{3}\right)\right],\end{aligned}$$

for $\theta_j = \frac{j\pi}{\overline{n}}$, with $j = 1, \ldots, \overline{n}$. For $\mu = 13/20$ the $\cos(\theta_j)$ term vanishes and we obtain

$$\lambda_j(A_n^{[0,2,1]})\left(\frac{13}{20}\right) = \frac{1}{5n}(6-\cos(2\theta_j))$$

and

$$\kappa(A_n^{[0,2,1]})\left(\frac{13}{20}\right) = \frac{7}{5}. \tag{3.29}$$

Estimate (3.29) is not as sharp as the one from Theorem 3.18 but, in practice, it gives a reasonable result; we refer to Figure 3.11 for an experimental comparison.

Remark 3.20. *Both the estimates* (3.28) *and* (3.29) *are independent from the problem size n.*

Generic q

If we consider $q > 0$, with an algebraic manipulation of $A_n^{[q,p,k]}$, we can generalize the previous results. We consider the eigendecomposition $M_{n,[p,k]} = Q\Lambda_M Q^*$ and $K_{n,[p,k]} = Q\Lambda_K Q^*$. We assume that $M_{n,[p,k]}$ and $K_{n,[p,k]}$ have the same eigenvector

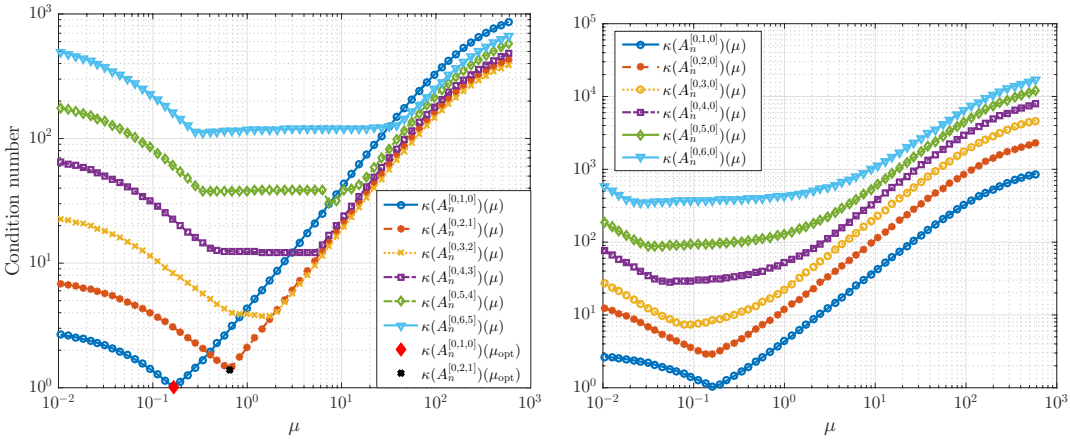

Figure 3.11. Condition numbers computed for $d = 1$, $n = 50$, $q = 0$ and various p and k. Left: IgA case, i.e. $k = p-1$; μ_{opt} from (3.28) and (3.29) are reported. Right: standard FE, i.e. $k = 0$.

matrix Q; this is true just for $p < 3$. Thus, we can write, from (2.29), using property (3.3),

$$\begin{aligned} A_n^{[q,p,k]} &= K_{[q]} \otimes Q\Lambda_M Q^* + \mu M_{[q]} \otimes Q\Lambda_K Q^* \\ &= (Q \otimes I_{q+1})(\Lambda_M \otimes K_{[q]})(Q \otimes I_{q+1})^* + \mu(Q \otimes I_{q+1})(\Lambda_K \otimes M_{[q]})(Q \otimes I_{q+1})^* \\ &= (Q \otimes I_{q+1})\left(\Lambda_M \otimes K_{[q]} + \mu\Lambda_K \otimes M_{[q]}\right)(Q \otimes I_{q+1})^*, \end{aligned}$$

then, for $i = 1, .., \overline{n}$, we can compute singular values of the non symmetric matrices $\Sigma_i \in \mathbb{R}^{(q+1)\times(q+1)}$

$$\Sigma_i := \lambda_i(M_{n,[p.k]})K_{[q]} + \mu\lambda_i(K_{n,[p.k]})M_{[q]}.$$

Finally, for $p < 3$ we get the general expression

$$\kappa(A_n^{[q,p,k]}) = \frac{\max\limits_{j=1,\dots,q+1} \max\limits_{i=1,\dots,\overline{n}} \sigma_j(\Sigma_i)}{\min\limits_{j=1,\dots,q+1} \min\limits_{i=1,\dots,\overline{n}} \sigma_j(\Sigma_i)}. \tag{3.30}$$

Equation (3.30) relies on the direct computation of the singular values of $\overline{n}$ small matrices (with size $q = 1$). See Figure 3.12 for an experimental confirm of the validity of equation (3.30).

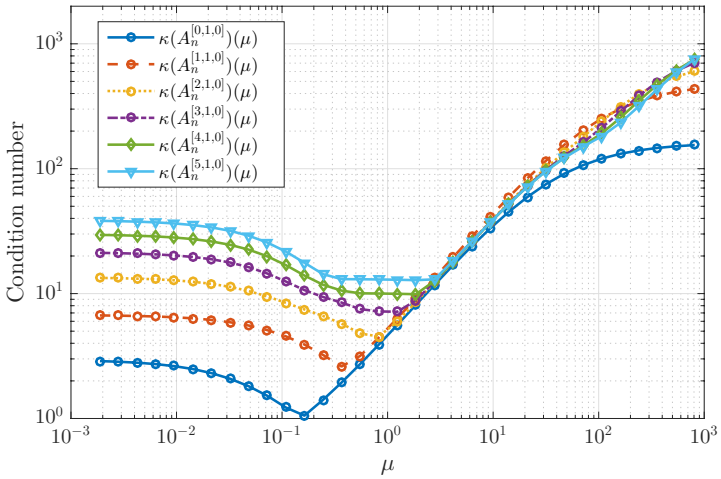

Figure 3.12. Condition number computed for $d = 1$, $n = 20$, $q = \{0, ..., 5\}$ and $p = 1$ and $k = 0$. Lines are obtained from equation (3.30) and markers with a direct computation of $\kappa(A_n^{[q,1,0]})$.

Remark 3.21. *The content of the current chapter relies on the assumption that Ω is a rectangular domain discretized by a regular grid. This choice gives rise to sufficiently regular matrix sequences (GLT sequences) for which it is possible to carry out the mathematical analyses, using the GLT theory. One could argue that such analyses are no longer useful in practice, where complex geometries are usually present. Even if sharp estimates given trough this chapter are not valid for arbitrary geometries, they can still give a hint on the general properties of operators arising from less trivial settings. In Figure 3.13 we show how Theorem 3.18, in practice, can still give an insight if a more complex geometry is used.*

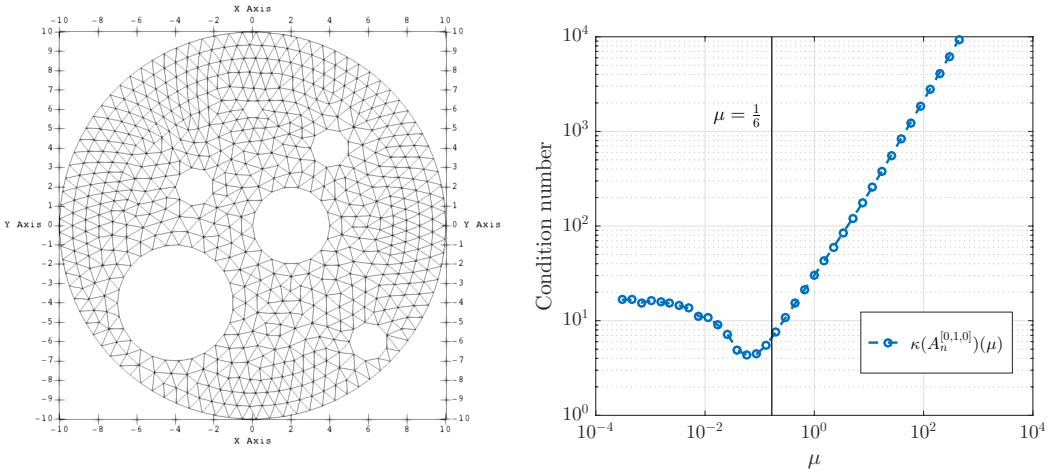

Figure 3.13. Non trivial geometry in 2D with the corresponding condition number $\kappa(A_n^{[0,1,0]})$ for various μ (i.e. Δt). The optimal μ_{opt} from Theorem 3.18 is also reported.

Chapter 4

Applications

In the previous chapters, we focused on a purely diffusive problem, which is often used to test and benchmark numerical methods and specifically parallel-in-time solution strategies in a simple setting. The results presented in Chapter 3 also rely on this setting where the reaction term in (2.1) is neglected. In this chapter, we describe two problems with applications in bio-medicine that will be part of the numerical experiments of Chapter 6. In particular, in Section 4.1, we consider a non-linear reaction-diffusion problem, often employed to model excitable media. In Section 4.2 we consider a diffusive problem with discontinuous coefficients, that appears when modeling the permeability through biological tissues.

4.1 Computational electrophysiology: the monodomain model

Computational electrophysiology is the branch of computational medicine that studies the modeling and simulation of the electrical properties of biological cells and tissues. One of the more relevant topics in this field of research is the simulation of the electrical activity in the human heart, heart failure being a major health problem worldwide. Obtaining fine measurements *in vivo* can be challenging; for this reason, numerical simulations may play a more and more important role in the understanding of heart diseases. The most complete model for describing the electrical activation of the cardiac tissue is the bidomain model that considers both extra and intra-cellular potentials (see for example Colli Franzone et al. [2006] for a detailed description). It consists of two non-linear reaction-diffusion equations coupled with a system of ordinary differential equations, containing up to 50 equations, describing the ionic currents in the cellular mem-

branes. Because of its structure, the bidomain model is computationally expensive.

Moreover, when discretizing the spatial domain, a fine discretization is required to capture the activation front effectively ($h \approx 10^{-4}$ m). With respect to time, the fast kinetics involved requires $\Delta t \approx 10^{-4}$ s. Because the cardiac muscle has size $\approx 10^{-1}$ m and a heart beat lasts for approximately 1 second, realistic 3D simulations of the heart can have more then 10^{10} degrees of freedom.

For this reason a simplified model is often used: the monodomain model, where there is no distinction between intra and extra-cellular potentials. This model approximates well the solution of the bidomain model and it is often used for the simulation of large parts of the heart. We refer the reader to Potse et al. [2006] for a comparison between the two models. The monodomain model is a particular instance of the PDE (2.1) coupled with an ODE and can be formulated as follows:

$$\begin{cases} \chi C_m \partial_t u - \nabla \cdot (K \nabla u) + \chi I_{ion}(u, w) = f & \text{in} \quad (0,T) \times \Omega, \\ \partial_t w = R(u, w) & \text{in} \quad (0,T) \times \Omega, \\ \mathbf{n} \cdot (K \nabla u) = 0 & \text{in} \quad (0,T) \times \partial\Omega, \\ u = u_0(\mathbf{x}), w = w_0 & \text{for} \quad t = 0, \end{cases} \tag{4.1}$$

where $u = u(\mathbf{x}, t) \in \mathbb{R}$ is the electric potential, $w = w(\mathbf{x}, t) \in \mathbb{R}$ an auxiliary coupling variable, $K = K(\mathbf{x})$ the conductivity tensor, $f = f(\mathbf{x}, t)$ a possible external current, χ the surface-to-volume ratio and C_m the membrane capacitance constant. The first equation in (4.1) is coupled with the ODE in the second line through the non-linear ionic current term I_{ion}. The last two equations provide boundary conditions in space and an initial condition in time. For I_{ion} we considered the Fitz-Hugh Nagumo (FHN) cubic model (cf. [Pezzuto et al., 2016])

$$\begin{aligned} &I_{ion}(u, w) = a(u - u_{\text{rest}})(u - u_{\text{thres}})(u - u_{\text{max}}) + C_w w(u - u_{\text{rest}}), \\ &R(u, w) = b(c(u - u_{\text{rest}}) - w), \end{aligned} \tag{4.2}$$

with parameters $a, b, c \in \mathbb{R}^+$ and the equilibrium states $u_{\text{rest}} \leq u_{\text{thres}} \leq u_{\text{max}}$, where u_{rest} and u_{max} are stable equilibrium states and u_{thres} is an unstable one. The informal meaning of the action of $I_{ion}(u, w)$ is the following: if $u(\mathbf{x}, t) > u_{\text{thres}}$ the tissue will be activated up to u_{max} in $\mathbf{x}$, while if $u(\mathbf{x}, t) < u_{\text{thres}}$ the potential u will decrease to u_{rest}. The second ODE in (4.1) is responsible for the depolarization of the activated tissue through the variable w; the speed of the depolarization can be controlled with $R(u, w)$ and C_w. For simplicity we consider $C_w = 0$, i.e. no depolarization occurs. In this setting $I_{ion}(u)$ depends just on u, as in (2.1). See Figure 4.1 for a 2D simulation of the solution of (4.1). It must be noticed that,

along with the FHN model, a large number of more rigorous ionic models are described in the literature, using up to 40 auxiliary variables (e.g. the Bernus ionic model or the Luo-Rudy models). We refer to [Keener and Sneyd, 1998] for an extensive overview on mathematical physiology.
In the monodomain model the interplay between the reaction term (controlled by the parameters $a, u_{\text{thres}}, u_{\text{rest}}, u_{\text{max}}$) and the diffusive one has to be tuned to produce a *wave-like* propagation of the electric front. See Figure 4.2 for a qualitative comparison of different parameter settings.
From Figures 4.1–4.2 it can be noticed how the gradient of the solution is different from zero in a small portion of the space-time domain. For this reason the discretization parameters Δt and h need to be small just during the upstroke of the action potential, i.e. in a small region of the space-time domain. Thus it makes sense to consider an adaptive approach in both space and time, as in [Cherry et al., 2000; Colli Franzone et al., 2006], and space-time discretizion, as in [Krause and Krause, 2016; Deuflhard et al., 2009]. Moreover, the large size of the discrete 3D problem requires to employ effective parallelization strategies to get reasonable time to solution, as in [Colli Franzone and Pavarino, 2004].

4.1.1 Discretization of the monodomain model

The discretization of (4.1) is essentially an extension of the one described in Section 2.3 for the heat equation. We can rearrange the terms in the first equation in (4.1) and obtain

$$\partial_t u - \nabla \cdot (\overline{K} \nabla u) = f - C_m^{-1} I_{\text{ion}}, \qquad \text{with} \quad \overline{K} = (\chi C_m)^{-1} K, \tag{4.3}$$

as in (2.1). The problem (2.28) is then modified to contain the discretization of the non-linear reaction term I_{ion}:

$$C_{N,n}^{[q,p,k]} \mathbf{u} = \mathbf{f} - \mathbf{r}(\mathbf{u}), \tag{4.4}$$

where $\mathbf{r}(\mathbf{u}) \in \mathbb{R}^{\overline{n}\overline{N}}$ can be constructed implicitly or explicitly; respectively:

IMPLICIT

$$\mathbf{r}_{\text{i}}(\mathbf{u}) = \left(I_N \otimes \frac{\Delta t}{2} M_{[q]} \otimes M_{n,[p,k]} \right) \mathbf{I}_{\text{ion}}(\mathbf{u}) \tag{4.5}$$

EXPLICIT

$$\mathbf{r}_{\text{e}}(\mathbf{u}) = \left(L_N \otimes \frac{\Delta t}{2} M_{[q]} \otimes M_{n,[p,k]} \right) \mathbf{I}_{\text{ion}}(\mathbf{u}) \tag{4.6}$$

with

$$\mathbf{I}_{\text{ion}}(\mathbf{u}) = C_m^{-1} \cdot [I_{\text{ion}}(u_1), I_{\text{ion}}(u_2), ..., I_{\text{ion}}(u_{\overline{n}\overline{N}})]^T \in \mathbb{R}^{\overline{n}\overline{N}},$$

and I_{ion} is given by (4.2) and the dependency on w is ignored because $C_w = 0$.

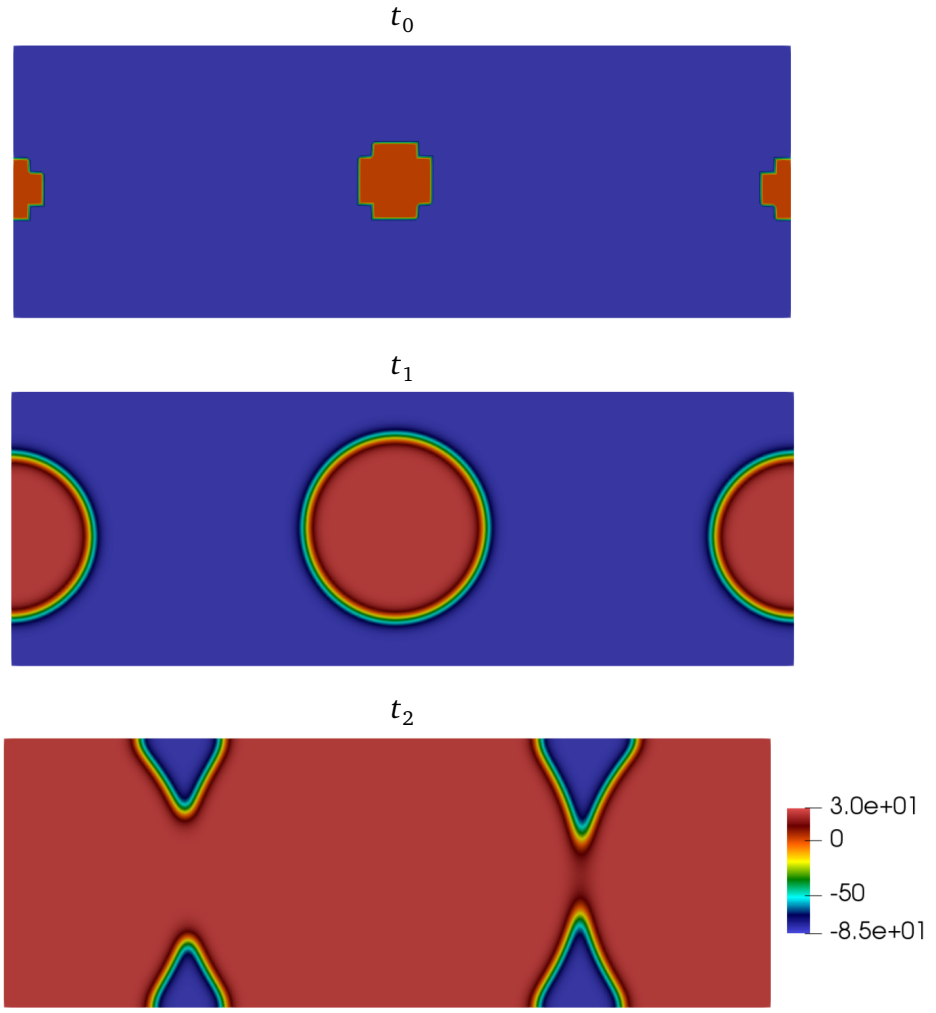

Figure 4.1. Qualitative evolution of the monodomain equation in a 2D domain for three subsequent time steps for the reference values $u_{\text{rest}} = -85$ mV, $u_{\text{thres}} = -57$ mV and $u_{\text{max}} = 30$ mV. The subsequent depolarization is not considered, i.e. $C_w = 0$.

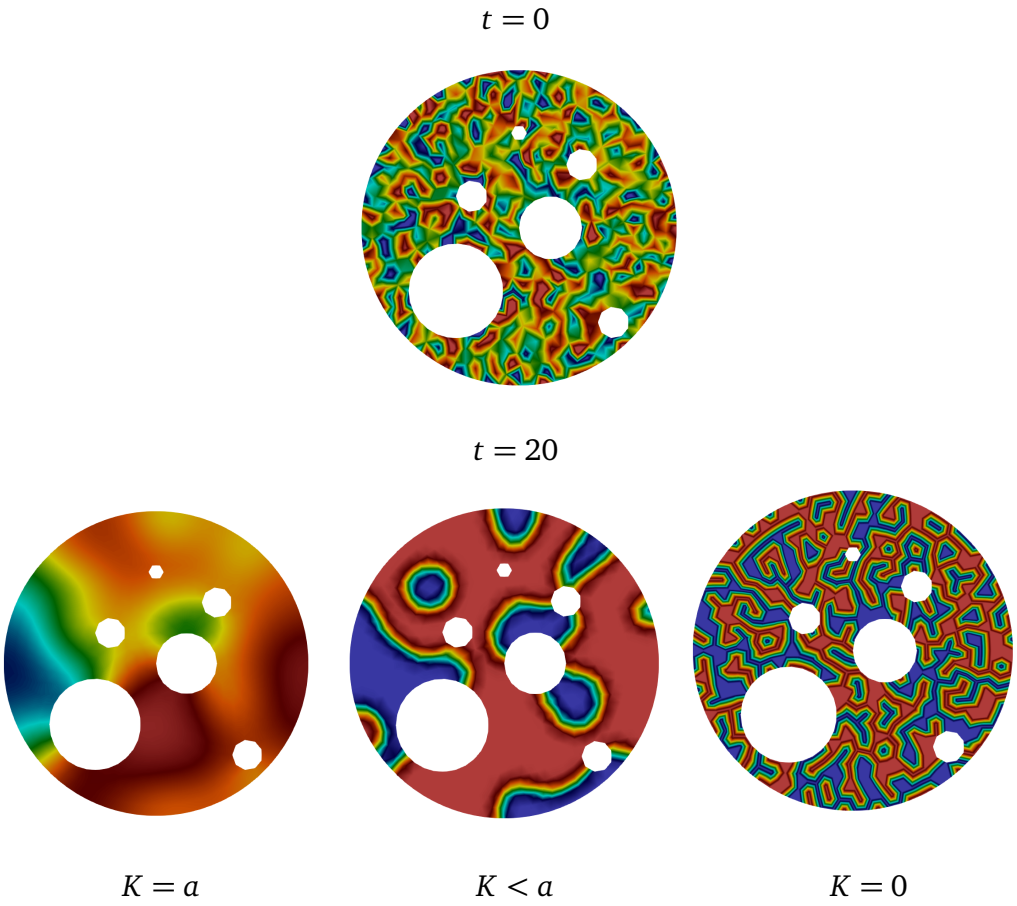

Figure 4.2. Qualitative example of the evolution of the monodomain equation in a circular 2D domain. Top: the initial solution $u_0 = \text{rand}\{[u_{\text{rest}}, u_{\text{max}}]\}$. Second line: the simulated solution after 20 ms for three different settings: from a diffusion dominated problem on the left to purely reactive on the right. (Mesh from Figure 3.13)

4.1.2 On the stability of the reaction term

Traditionally a semi-implicit approach is used for the time discretization of (4.1), i.e. the diffusion is treated implicitly and the reaction term explicitly; see, e.g., [Colli Franzone and Pavarino, 2004]. This technique can allow a certain freedom to choose Δt, even adaptively according to the heartbeat phase and the correspondent magnitude of the reaction term. For example, as the reaction term vanishes, larger Δt can be used.

Remark 4.1. *The presented formulation shows a certain limitation; the all at once*

solution of the space-time non-linear system (4.4) *requires a non linear solution method to be employed both for the implicit and explicit case.*

We are interested in the numerical stability of the discretization of the additional reactive term. For simplicity let us consider the simple cubic, purely reactive, ODE:

$$\begin{cases} u'(t) = a(u(t) - u^3(t)), & t \in (0, T], \\ u(0) = u_0. \end{cases} \tag{4.7}$$

corresponding to the ODE in (4.2) with $u_{\text{rest}} = -1, u_{\text{thres}} = 0, u_{\text{max}} = 1$ and $C_w = 0$. If (4.7) is solved explicitly we obtain the following time stepping scheme

$$u_{m+1} = u_m + \Delta t \cdot a(u_m - u_m^3) := g(u_m) \quad \text{for} \quad m = 0, ..., N-1. \tag{4.8}$$

By the Banach fixed-point theorem the previous iterative process converges to an equilibrium point $u^* \in \{u_{\text{rest}}, u_{\text{max}}\}$ if g is a contraction, i.e. if

$$|g'(u^*)| < 1 \quad \Leftrightarrow \quad |1 - 2a\Delta t| < 1 \quad \Leftrightarrow a\Delta t < 1. \tag{4.9}$$

On the other hand the implicit iteration

$$u_m - \Delta t \cdot a(u_m - u_m^3) = u_{m-1} \quad \text{for} \quad m = 0, ..., N-1, \tag{4.10}$$

in general is ill posed for $a\Delta t > 1$ because multiple roots are present and a *good* initial guess is necessary for the convergence of a non-linear solution method such as the Newton's method, cf. Figure 4.3 and Table 4.1.

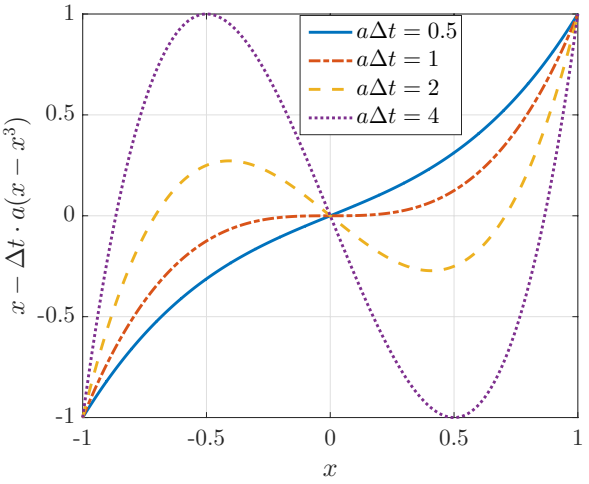

Figure 4.3. Graph of the non linear function (4.10) with multiples zeros for $a\Delta t > 1$.

In Table 4.1 we present the results of a convergence study for the iterative schemes (4.8) and (4.10).

$a\Delta t$	1/16	1/8	1/4	1/2	1	2
Explicit Euler	c	c	c	c	d	d
Implicit Euler, $u_{\text{in}} = 0$	5	6	7	8	27	d
Implicit Euler, $u_{\text{in}} = 0.5$	5	5	6	7	8	9

Table 4.1. Convergence of schemes (4.8)–(4.10) with $a = 1$, $T = 10$ and $u_0 = 0.2$ for increasing Δt. We use the symbol 'c' ('d') to denote the convergence (divergence) of the iterative scheme to the steady state $u_{\max} = 1$ at the final time T. For the implicit case we report the number of Newton iterations to converge for two initial guesses u_{in}. As expected from (4.9) the explicit iteration converges just for $a\Delta t < 1$.

In Chapter 6 we perform similar numerical tests for the full space-time system, coming to a similar conclusion: for large $a\Delta t$ both the implicit and explicit schemes are unreliable and, for the non-linear implicit iteration, a *good* initial guess must be provided.

4.2 Modeling diffusion through human skin

The second problem we consider is the dermal absorption of chemical substances through human skin. This process consists in the diffusion of a chemical from the outer layer of the skin (*Stratum Corneum*) to the internal ones. The study of skin transport is relevant in many applications, as drug development or to prevent the absorption of toxic chemicals (e.g. pesticides). Because in vitro experiments are limited by ethical and practical concerns, in silico studies can be an attractive alternative; for example see the reviews [Jepps et al., 2013; Naegel et al., 2013; Querleux, 2016].

Nevertheless, a three dimensional complete model of skin transport can be challenging in terms of computational resources and highly parallel strategies need to be employed. For a concurrency study using Parareal see the work by Kreienbuehl et al. [2015a].

The permeability of the skin depends largely on the structure of the Stratum Corneum where three mechanisms of diffusion have been proposed ([Hansen et al., 2008]):

- Through the corneocytes, i.e. the cells composing the SC.

- Through the lipids matrix between the corneocytes, faster w.r.t. the previous mechanism.
- Through the appendages (hair follicles, glands): usually insignificant due to the small area covered by the appendages.

We model this scenario considering a spatial domain $\Omega \subset \mathbb{R}^d$ composed of two disjoint subsets $\Omega_{\text{cor}}, \Omega_{\text{lip}} \subset \Omega$ and a diffusion PDE with jumping coefficients for $t \in [0, T]$:

$$\begin{cases} \partial_t u = \nabla \cdot K_{\text{p}}(\mathbf{x}) \nabla u, & \mathbf{x} \in \Omega, \\ u = c_0, & \mathbf{x} \in \partial\Omega_D \subset \partial\Omega, \\ \mathbf{n}_{\mathbf{x}} \cdot \nabla u = 0, & \mathbf{x} \in \partial\Omega / \partial\Omega_D. \end{cases} \tag{4.11}$$

In this context $u = u(\mathbf{x}, t)$ represents drug concentration and

$$K_{\text{p}}(\mathbf{x}) = \begin{cases} K_{\text{cor}} & \text{for} \quad \mathbf{x} \in \Omega_{\text{cor}}, \\ K_{\text{lip}} & \text{for} \quad \mathbf{x} \in \Omega_{\text{lip}}, \end{cases}$$

with typically $K_{\text{cor}} \ll K_{\text{lip}}$. We refer to Figure 4.4 for an example of a brick and mortar configuration for Ω.

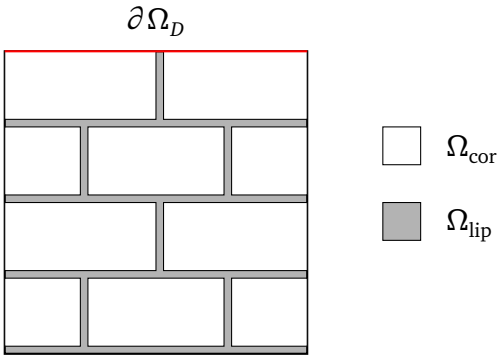

Figure 4.4. Example of a brick and mortar configuration of Ω. The concentration is fixed to c_0 at the red Dirichlet boundary.

Chapter 5

Solution Strategies

Some of parallel-in-time solvers are based on the coupling between coarse and fine time propagators. Ideally, a coarse, fast and sequential time stepper can provide initial conditions for a parallel fine time stepper along the whole time domain.

For this reason, most of the parallel-in-time algorithms are, or can be framed, in a multilevel-in-time setting; for example MGRIT, Parareal in [Gander et al., 2018b] or PFASST in [Bolten et al., 2017]. Moreover, when dealing with a space-time discretization, where time is somehow considered as an additional spatial dimension, it is natural to extend the same paradigm for the solving process and consider space-time multigrid type algorithms.

Specialized parallel solvers have been recently developed for the large linear systems arising from space-time discretizations. We mention in particular the space-time parallel multigrid proposed by Gander and Neumüller [2016], the parallel preconditioners for space-time isogeometric analysis proposed by Hofer et al. [2019] as well as the block preconditioned GMRES by McDonald and Wathen [2016].

In Section 5.1 we present a semi-geometric space-time multigrid that is well suited to deal with complex geometries. In Section 5.2 we focus on the design of a robust PGMRES that can be used as smoother. In Section 5.3 we describe the non-linear solver used in the monodomain context.

5.1 Space-time multigrid

Multigrid solvers are expected to exhibit an optimal convergence when applied to elliptic problems but they have proven to be efficient, with some precautions, also for space-time discretizations of parabolic problems. In particular, the dis-

cretization of the heat equation presents an anisotropy in the time direction and therefore introduces unsymmetric off-diagonal entries in the system $C_{N,n}^{[q,p,k]}$.

When dealing with anisotropic problems[1] standard multigrid convergence rate deteriorates (see Briggs et al. [2000]). Traditionally there are various ways to address this problem, incorporating the asymmetry in the particular choice of smoothers and/or coarsening strategies. Some viable options are:

- Semi-coarsening and standard relaxation
- Full coarsening and line relaxation
- Semi-coarsening and line relaxation.

We refer to [Falgout et al., 2017] for a scaling analysis using these different strategies. In [Horton and Vandewalle, 1995] it was studied how space-time multigrid convergence is not independent of the discretization parameter μ unless semi-coarsening is adopted; the experiments were combined with a local Fourier analysis (LFA) to predict multigrid convergence factors. We present a new perspective on the coarsening strategy based on the results of Section 3.5.

5.1.1 Coarsening strategy

Let us consider a hierarchy of L levels with corresponding discretization parameters $h_l, \Delta t_l$ and $\mu_l = K\Delta t_l/h_l^2$ for $l = 1, ..., L$ with $h_1 = h$, $\Delta t_1 = \Delta t$ and L corresponding to the coarsest level. When constructing the $l+1$ coarse space from the one of level l, we consider the following new discretization parameters:

$$h_{l+1} = C_x h_l, \qquad \Delta t_{l+1} = C_t \Delta t_l \quad \Longrightarrow \quad \mu_{l+1} = \frac{C_t}{C_x^2}\mu_l,$$

where $C_x, C_t \geq 1$ are the coarsening factors in time and space respectively. In the naive setting, i.e. full coarsening, $C_x = C_t = 2$ and μ is varying through the level hierarchy: $\mu_{l+1} = \mu_l/2$. Figures 5.1–5.2 illustrate how different coarsening strategies (i.e. choices of C_x and C_t) determine how μ_l varies in the $(h, \Delta t)$ plane.

[1] A standard example is the anisotropic Laplacian in 2D in the form $u_{xx} + \varepsilon u_{yy} = f(x, y)$ for $\varepsilon \in \mathbb{R}$.

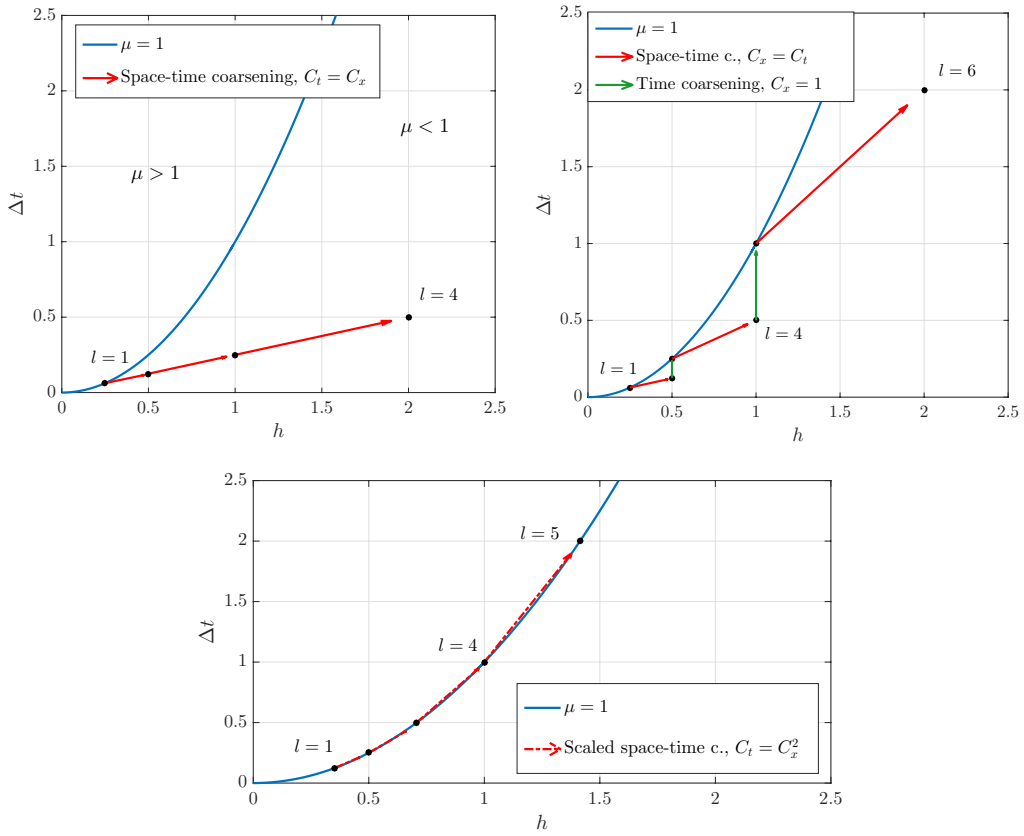

Figure 5.1. Variation of μ_l with $l = \{1, 2, 3, ...\}$ starting from $h_1 = 1/4$, $\Delta t_1 = 1/16$ and $K = 1$ i.e. $\mu_1 = 1$ using various coarsening strategies. For example, for full coarsening (i.e. space-time coarsening, top left), $\mu_1 > \mu_2 > ... > \mu_L$. On the other hand, if $C_t = C_x^2$ then $\mu_l = \mu_1$ for any level l.

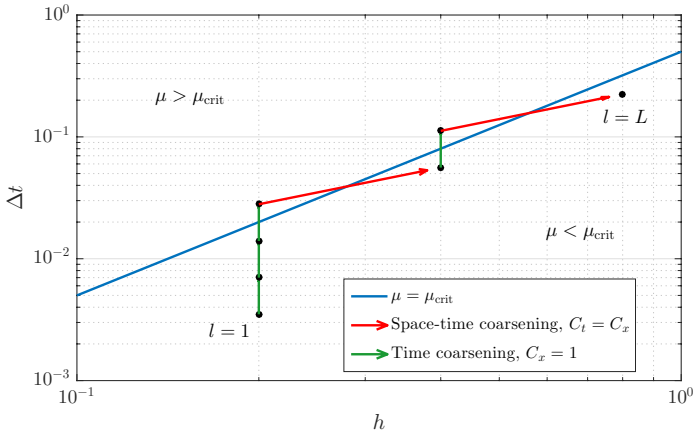

Figure 5.2. Adaptive coarsening strategy as in [Horton and Vandewalle, 1995] based on a priori choice of μ_{crit}: if $\mu_l < \mu_{\text{crit}}$ semi-coarsening in time is used for the $l+1$ level. Otherwise full coarsening is employed. In this case a loglog scale is used to represent a portion of the $(h, \Delta t)$ plane.

In Chapter 3 (Figure 3.8) we showed how the condition number of the space-time system $C_{N,n}^{[q,p,k]}$ increases and the convergence of standard smoothers degrades as μ deviates from a certain μ_{opt}. If auxiliary coarse problems have an increasingly worse conditioning we can expect the multigrid convergence to be not satisfactory. For this reason for large μ (resp. small μ) time coarsening (resp. space coarsening) should be avoided. We show this principle in Figure 5.3 where convergence factors of a two level multigrid along with coarse level conditioning are compared for different coarsening strategies.

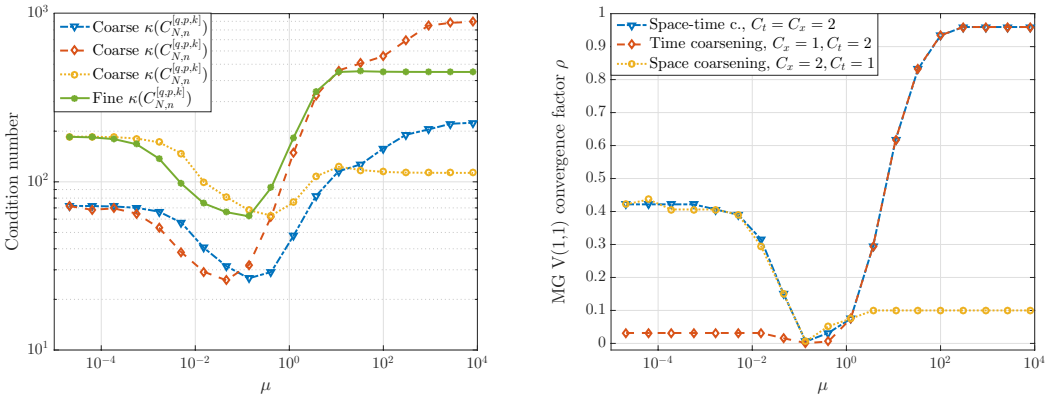

Figure 5.3. Right: convergence factor to solve $C_{N,n}^{[0,1,0]}$ in $\Omega = [0,1]$ with $K = 1$ and a random initial right hand side $\mathbf{f}_0$ using two level multigrid with one Gauss-Seidel iteration as smoother. We used $n = N = 30$ and $h = 1/30$ and varying $\Delta t \in \{3^{-16}, 3^{-15}, ..., 3^1, 3^2\}$. Left: the corresponding coarse problem conditioning is shown with the same type of line. Standard interpolation for 1D uniform grids is used in space and/or in time, as in (5.1) and (5.2).

We remark that the convergence factors in Figure 5.3 have the same behavior of previous analysis present in the literature [Franco et al., 2018] and suggest the coarsening strategy in Algorithm 1.

Algorithm 1: Adaptive coarsening strategy

Input: $\mu_{\text{opt}}, \mu_l, \delta \in \mathbb{R}^+$
if $|\mu - \mu_{opt}| \leq \delta$ **then**
  $C_x, C_t > 1$ `// space-time coarsening`
else if $\mu > \mu_{opt} + \delta$ **then**
  $C_x > 1, C_t = 1$ `// semi-coarsening in space`
else
  $C_x = 1, C_t > 1$ `// semi-coarsening in time`

5.1.2 Space-time transfer operators

Constructing space-time transfer operators, in the tensor product setting, is straightforward. Let us initially consider the time transfer operator $T_l^{l+1} \in \mathbb{R}^{N_{l+1} \times N_l}$ where $N_l = N/C_t^{l-1}$ is the number of time steps in level l. We employ the standard 1D interpolation to construct T_l^{l+1}; for example, for $C_t = 2$, we would get

$$T_l^{l+1} = \begin{bmatrix} 1 & & & & \\ 1/2 & 1/2 & & & \\ & 1 & & & \\ & 1/2 & 1/2 & & \\ & & \ddots & & \\ & & 1/2 & 1/2 & \\ & & & 1 & \end{bmatrix}. \tag{5.1}$$

Similarly we consider the spatial transfer operator $S_l^{l+1} \in \mathbb{R}^{\bar{n}_{l+1} \times \bar{n}_l}$. The construction of S_l^{l+1} requires a more detailed description, that will be presented in Section 5.1.3. Noticeably if $C_t = 1$ ($C_x = 1$) then $T_l^{l+1} = I_N$ ($S_l^{l+1} = I_{\bar{n}}$). Finally we construct the space-time transfer operator $I_l^{l+1} \in \mathbb{R}^{\bar{n}_{l+1} N_{l+1}(q+1) \times \bar{n}_l N_l (q+1)}$ as

$$I_l^{l+1} = T_l^{l+1} \otimes I_{q+1} \otimes S_l^{l+1}. \tag{5.2}$$

We can notice that, because of I_{q+1} in (5.2), the order in time q is constant along the multilevel hierarchy. We mention that it is possible to obtain a p-multigrid type algorithm replacing I_{q+1} by a suitable transfer.

5.1.3 A semi-geometric approach for space coarsening

When dealing with complex geometries and unstructured meshes in multiple dimensions, it can be challenging to generate coarse spaces. Moreover, geometric multigrid, in contrast to its algebraic counterpart, cannot be used as a *black-box* solver and interpolation operators must be provided by the users. We propose a semi-geometric approach to overcome these limitations. To build S_l^{l+1} in (5.2) we consider the bounding cuboid of the fine input mesh (in 2D or 3D). Then, a sequence of coarse meshes can be automatically constructed by refining uniformly the cuboid, we refer to Figure 5.4 for a visual example of this technique and Algorithm 2 for the description of the mesh creation.

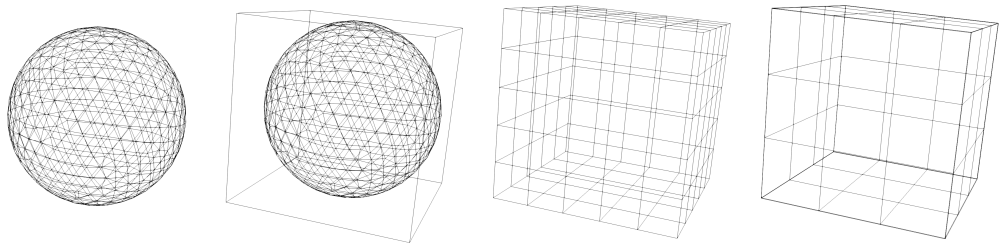

Figure 5.4. Illustration of the coarsening procedure in space for a three level multigrid. We consider the 3D mesh of a sphere, its bounding cube and two coarse meshes. This approach is rather flexible; an external user would just need to provide the spatial fine mesh and the type of coarsening to use, i.e. C_x.

Algorithm 2: Semi-geometric multigrid: coarse meshes generation

Input: Fine mesh $\mathcal{T}_1$ with size n, #levels L, coarsening factor C_x
Output: Coarse meshes $\{\mathcal{T}_2, \mathcal{T}_3, ..., \mathcal{T}_L\}$

```
// Compute x_1, x_2 ∈ ℝ^d defining the bounding box of T_1
```
$[\mathbf{x}_1, \mathbf{x}_2] = \text{boundingBox}(\mathcal{T}_1)$
```
// Compute scaling factors α
```
$\boldsymbol{d} = (\mathbf{x}_2 - \mathbf{x}_1)$
$\boldsymbol{\alpha} = d \cdot \boldsymbol{d} / \text{sum}(\boldsymbol{d})$
```
// Compute element per side n_e (always even)
```
$n_e = 2 \cdot \lfloor \sqrt[d]{n}/2 \rfloor$
```
// Construct coarse meshes
```
for $i = 2;\ i = L;\ i = i + 1$ **do**
  $n_e = n_e / C_x$
  $\mathcal{T}_i = \text{buildCuboid}(\mathbf{x}_1, \mathbf{x}_2, \lceil n_e \cdot \boldsymbol{\alpha} \rceil)$
end
```
// Where the function buildCuboid(x,y,n), with x,y,n ∈ ℝ^d, builds a mesh
   from the d−dimensional cuboid defined by the vertices x and y with n
   grid points per side.
```

We refer to Dickopf and Krause [2013] for a numerical study about multilevel methods on non-nested meshes.

Remark 5.1. *From an algorithmic point of view, coarsening a mesh is not a trivial task. For this reason, when creating a mesh hierarchy, it is convenient to start from a coarse mesh and proceed by refinement. This standard approach has two main limitations: firstly a coarse input mesh could be not available, if just the fine one*

is provided. Secondly, some accuracy in representing the fine geometry is lost when refining a coarse mesh. The semi-geometric approach, on the other hand, overcomes these issues and allows the user to choose C_x at run-time with no limitations.

This strategy requires a stable and efficient technique to transfer discrete fields between different spatial meshes, possibly not nested. We consider both classical interpolation and variational transfer operators: the L^2-projection. The L^2-projections are proven to be optimal and stable and in general superior to interpolation [Hesch and Betsch, 2006]. In particular, we employ a local approximation of the L^2-projection, which is constructed by exploiting the properties of the dual basis [Wohlmuth, 2000; Popp et al., 2012]. This local approximation allows us to construct the transfer operator explicitly in such a way that it can be applied by means of a simple sparse matrix-vector multiplication. In a parallel computing environment where meshes are arbitrarily distributed the assembly of the mass matrices related to L^2-projection is not trivial. The construction of the discrete L^2-projection requires us to detect and compute intersections between the elements of the coarse mesh and the elements of the finer mesh which might be stored in different memory address spaces (e.g., on a super-computing cluster). For this purpose we use the parallel tree-search algorithms and parallel assembly routines described in [Krause and Zulian, 2016].

Remark 5.2. *When the coarse mesh of the bounding box is created it might happen to get elements that are completely external to the initial unstructured finest mesh. This results in zero rows and columns in the transfer operator S_1^2 and subsequently in the coarse operator, assembled via Galerkin assembly. For this reason we set to one the diagonal entry of all the zero rows in the coarse operators.*

5.1.4 Domain decomposition and smoothing methods

We use block preconditioned GMRES, as a parallel smoother. We notice how other types of block smoothers, such as block Jacobi or Gauss-Seidel, are less reliable, in particular when dealing with the non-linear problem (2.1) and realistic parameters (cf. Section 6.4).

Remark 5.3. *In Section 2.1.3 we considered a specific ordering of the basis functions: time first. This corresponds to the order of tensor products in* (2.27) *and along the thesis. This choice is arbitrary, and somehow conventional, and corresponds to a particular ordering of the vector of unknowns* $\mathbf{u}$*. On the other hand, especially for high q, flipping the order will change the sparsity pattern of $A_n^{[q,p,k]}$ and $B_n^{[q,p,k]}$ and therefore the block preconditioner for $C_{N,n}^{[q,p,k]}$; see Figures 5.5–5.6 for a numerical example.*

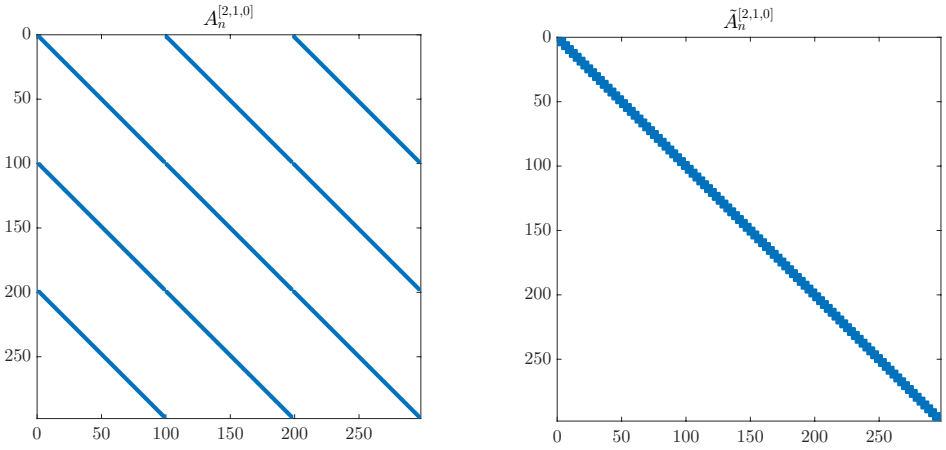

Figure 5.5. Left: sparsity pattern of the diagonal block $A_n^{[2,1,0]}$ for $\Omega = [0,1]$ and $n = 100$. Right: the tensor product order is reverted in $\tilde{A}_n^{[2,1,0]}$ i.e.

$$\tilde{A}_n^{[2,1,0]} = M_{n,[1,0]} \otimes K_{[2]} + \frac{\Delta t}{2} \otimes K_{n,[1,0]} \otimes M_{[2]}.$$

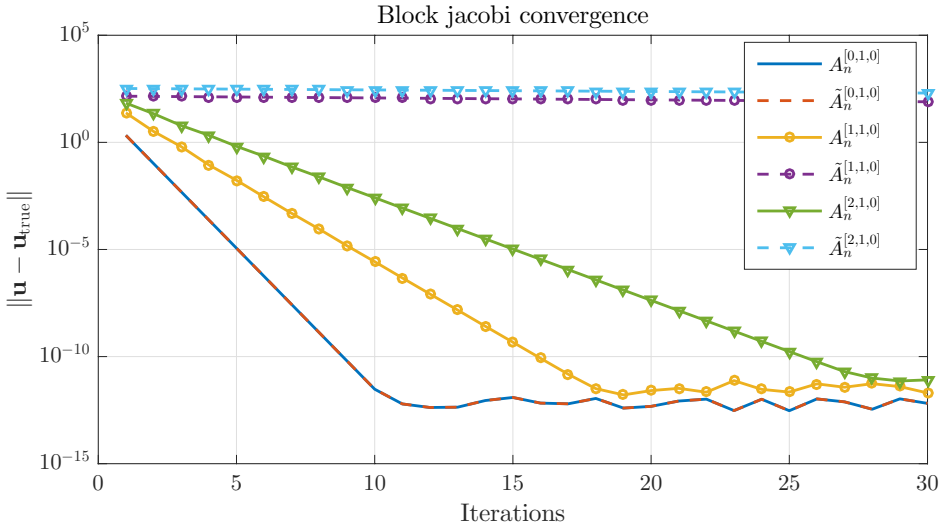

Figure 5.6. Block Jacobi iterations to solve system (2.28) with right hand side $\mathbf{f} = [\mathbf{1}, \mathbf{0}, ..., \mathbf{0}]^T$, $q = \{0,1,2\}$, $\Omega = [0,1]$, $n = 100$, $N = 20$, $T = 1$ and both orderings. We used $q+1$ blocks for time step, i.e. $\overline{N}$ total blocks. Let us recall that $A_n^{[0,p,k]} = \tilde{A}_n^{[0,p,k]}$.

Section 5.2 will be devoted to the description of a specialized preconditioner for

space-time discretizations in a tensor form.

5.2 Robust symbol based PGMRES

We present a preconditioned GMRES, based on the content of Chapter 3, that is efficient and robust w.r.t. the spatial parameters n, p and k. Let us consider the symmetric preconditioner

$$P_{N,n}^{[q,p,k]} = \frac{\Delta t}{2} I_N \otimes M_{[q]} \otimes K_{n,[p,k]}, \tag{5.3}$$

presented and discussed in Example 3.14. Let us remind the reader that, under Theorem 3.7 assumptions, we know that $P_{N,n}^{[q,p,k]}$ is spectrally equivalent to the original operator $C_{N,n}^{[q,p,k]}$ (cf. Corollary 3.8). Therefore, in view of the convergence properties of the GMRES method [Saad, 2003], we may expect that the resulting PGMRES for solving a linear system with coefficient matrix $C_{N,n}^{[q,p,k]}$ has an optimal convergence rate, i.e., the number of iterations for reaching a preassigned accuracy ε is independent of (or only weakly dependent on) the matrix size. As in Table 3.2, but more exhaustively, we show how this expectation is realized for various examples of problem (2.28). The resulting number of iterations are collected in Tables 5.1–5.2. We see from the tables that the GMRES solver rapidly deteriorates with increasing n, and it is not robust with respect to p, k. On the other hand, the convergence rate of the proposed PGMRES is robust with respect to all the spatial parameters n, p, k. However, as it is known, each PGMRES iteration requires solving a linear system with coefficient matrix given by the preconditioner $P_{N,n}^{[q,p,k]}$, and this is not required in a GMRES iteration. Thus, in order to prove that the proposed PGMRES is fast, we have to show that we are able to solve efficiently a linear system associated with $P_{N,n}^{[q,p,k]}$. This is the subject of Sections 5.2.1 and 5.2.2. It should be noted, however, that, while the discussion in Section 5.2.1 is general, in Section 5.2.2 we need the IgA assumption $k = p-1$.

5.2.1 Fast tensor solver for the preconditioner $P_{N,n}^{[q,p,k]}$

The main observation of this section is that, thanks to the tensor structure of $P_{N,n}^{[q,p,k]}$ (see (5.3)), the solution of a linear system with coefficient matrix $P_{N,n}^{[q,p,k]}$ reduces to the solution of three linear systems with coefficient matrices I_N, $M_{[q]}$, $K_{n,[p,k]}$. Indeed, using the canonical algorithm for tensor-product matrices to

Table 5.1. Number of iterations G$[p,k]$ and PG$[p,k]$ needed by, respectively, GMRES and PGMRES with preconditioner $P_{N,n}^{[q,p,k]}$, for solving the linear system (2.28), up to a precision $\varepsilon = 10^{-6}$, in the case where

$$K(x_1,x_2) = \begin{bmatrix} \cos(x_1)+x_2 & 0 \\ 0 & x_1+\sin(x_2) \end{bmatrix},$$

$f(t,\mathbf{x}) = 1$, $T = 1$, $q = 1$, $N = 20$ and $\Omega = [0,1]^d$. The number of DoFs is given by $\overline{N}\bar{n} = 40(n(p-k)+k-1)^2$.

n	G[1,0]	PG[1,0]	G[2,0]	PG[2,0]	G[2,1]	PG[2,1]	G[3,1]	PG[3,1]
20	147	38	231	38	94	38	165	38
40	302	38	469	38	188	38	337	38
60	459	38	707	38	285	38	509	38
80	617	38	945	38	381	38	682	38
100	775	38	1184	38	478	38	856	38
120	933	38	1424	38	575	38	1030	38
n	G[4,1]	PG[4,1]	G[4,2]	PG[4,2]	G[5,2]	PG[5,2]	G[5,3]	PG[5,3]
20	224	38	238	38	190	38	171	38
40	457	38	325	38	382	38	283	38
60	691	38	463	38	576	38	427	38
80	926	38	623	38	770	38	572	38
100	1161	38	782	38	964	38	718	38
120	1372	38	942	38	1158	38	864	38

Table 5.2. Number of iterations G$[p,k]$ and PG$[p,k]$ needed by, respectively, GMRES and PGMRES with preconditioner $P_{N,n}^{[q,p,k]}$, for solving the linear system (2.28), up to a precision $\varepsilon = 10^{-6}$, in the case where

$$K(x_1,x_2) = \begin{bmatrix} (2+\cos x_1)(1+x_2) & \cos(x_1+x_2)\sin(x_1+x_2) \\ \cos(x_1+x_2)\sin(x_1+x_2) & (2+\sin x_2)(1+x_1) \end{bmatrix},$$

$f(t,\mathbf{x}) = 1$, $T = 1$, $q = 2$, $N = 20$ and $\Omega = [0,1]^2$. The number of DoFs is given by $\overline{N}\bar{n} = 60(n(p-k)+k-1)^2$.

n	G[2,0]	PG[2,0]	G[2,1]	PG[2,1]	G[3,0]	PG[3,0]	G[3,2]	PG[3,2]
20	158	24	63	24	220	24	61	24
40	319	24	127	24	442	24	124	24
60	480	24	192	24	665	24	187	24
80	641	24	258	24	889	24	252	24
100	802	24	324	24	1112	24	317	24
120	963	24	390	24	1335	24	382	24
n	G[4,0]	PG[4,0]	G[4,3]	PG[4,3]	G[5,0]	PG[5,0]	G[5,4]	PG[5,4]
20	350	24	64	24	358	24	71	24
40	573	24	131	24	714	24	140	24
60	859	24	199	24	1070	24	213	24
80	1146	24	268	24	1426	24	287	24
100	1433	24	338	24	1782	24	361	24
120	1720	24	408	24	2138	24	437	24

solve the system $P_{N,n}^{[q,p,k]}\mathbf{x} = \mathbf{y}$, we obtain

$$\begin{aligned}
\mathbf{x} &= (P_{N,n}^{[q,p,k]})^{-1}\mathbf{y} \\
&= \Big(\frac{2}{\Delta t} I_N \otimes M_{[q]}^{-1} \otimes K_{n,[p,k]}^{-1}\Big)\mathbf{y} \\
&= (\widetilde{M}_{N,[q]} \otimes K_{n,[p,k]}^{-1})\mathbf{y} \\
&= (\widetilde{M}_{N,[q]} \otimes I_{\bar{n}})(I_{N(q+1)} \otimes K_{n,[p,k]}^{-1})\mathbf{y} \\
&= (\widetilde{M}_{N,[q]} \otimes I_{\bar{n}}) \begin{bmatrix} K_{n,[p,k]}^{-1} & & & \\ & K_{n,[p,k]}^{-1} & & \\ & & \ddots & \\ & & & K_{n,[p,k]}^{-1} \end{bmatrix} \mathbf{y} \qquad (5.4) \\
&= \mathrm{vec}(K_{n,[p,k]}^{-1} Y \widetilde{M}_{N,[q]}), \qquad (5.5)
\end{aligned}$$

where:

- $\widetilde{M}_{N,[q]} = \frac{2}{\Delta t} I_N \otimes M_{[q]}^{-1}$ can be pre-computed with a negligible cost, because $M_{[q]}$ is a small $(q+1)\times(q+1)$ matrix (if Gauss–Radau nodes are used, $M_{[q]}$ is also diagonal and hence $\widetilde{M}_{N,[q]}$ is diagonal as well);
- $\mathrm{vec}(X)$ is the column-wise form of X, that is the vector obtained by stacking the columns of X;
- Y is the $\bar{n} \times \overline{N}$ matrix such that $\mathrm{vec}(Y) = \mathbf{y}$.

It is then clear that the computation of the solution $\mathbf{x}$ reduces to solving the $\overline{N}$ linear systems $K_{n,[p,k]}\mathbf{x}_i = \mathbf{y}_i$, $i = 1,\dots,\overline{N}$, where $\mathbf{y}_i$ is the ith column of Y, and multiply the resulting matrix $K_{n,[p,k]}(K)^{-1}Y$ by $\widetilde{M}_{N,[q]}$. Note that the various $\mathbf{x}_i$ can be computed in parallel as the computation of $\mathbf{x}_i$ is independent of the computation of $\mathbf{x}_j$ whenever $i \neq j$. Depending on the implementation and the parallel setting, it can be advantageous to express $\mathbf{x}$ using $\mathrm{vec}(\cdot)$ as in (5.5) or tensor products as in (5.4). The next section is devoted to show how to solve efficiently a linear system associated with $K_{n,[p,k]}$ in the maximal smoothness case $k = p-1$, that is, the case corresponding to IgA. For notational simplicity, the matrix $K_{n,[p,p-1]}$ will be denoted by $K_{n,[p]}$, and similarly for $M_{n,[p]}$ and $C_{N,n}^{[q,p]}$.

5.2.2 Fast multigrid for the IgA Poisson problem

The convergence of standard multigrid methods for elliptic problems is not satisfactory for high order finite element discretizations, as shown, for example, in

[Gahalaut et al., 2013]. In case of IgA, the convergence rates become close to 1 as the order p increases. This issue can be understood in terms of the symbol of the discretized Laplacian $K_{n,[p]}$ that we denote, for $d = 1$, as f_p. In particular, as proved in [Donatelli et al., 2017], $f_p(\pi)$ converges exponentially to zero as $p \to \infty$, cf. Figure 5.7. This results in spurious small eigenvalues associated with high frequency eigenvectors that are not properly *attacked* by fine level smoothers. As in [Donatelli et al., 2015b], a possible solution to this complication is to precondition $K_{n,[p]}$ with the corresponding mass matrix $M_{n,[p]}$. This technique recovers the monotonicity w.r.t. θ of the resulting operator $M_{n,[p]}^{-1}K_{n,[p]}$, cf. Figure 5.8.

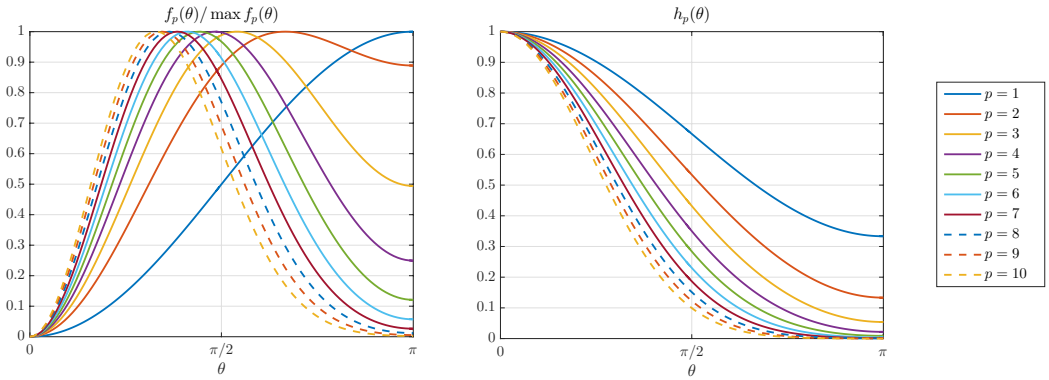

Figure 5.7. Symbols corresponding to $K_{n,[p]}$, i.e. $f_p(\theta)$ (left), and to $M_{n,[p]}$, i.e. $h_p(\theta)$ (right); $f_p(\theta)$ is normalized to attain his maximum in one. We refer to [Garoni et al., 2019] for the analytical expressions to compute f_p and h_p. Notice how, for $p > 1$, the monotonicity of f_p is lost. Moreover $f_p(\pi)$ and $h_p(\pi)$ both converge to zero with the same order, as $p \to \infty$.

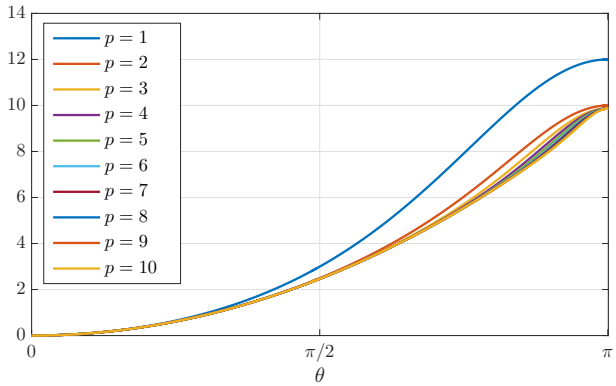

Figure 5.8. Symbol $e_p(\theta)$ associated to $M_{n,[p]}^{-1}K_{n,[p]}$. For $p \to \infty$, $e_p(\theta) \to \theta^2$.

For a comparative preliminary study we consider the following solver for $K_{n,[p]}^{-1}$:

- At the finest level, we perform ν post-smoothing steps by means of the PCG method. The proposed preconditioner used on the finest level is either $M_{n,[p]}$ or an incomplete LU (ILU) preconditioner.
- At all coarse levels, we use one post-smoothing iteration by the Gauss-Seidel method.
- At all levels, we use standard bisection for the interpolation and restriction operators, and we assemble coarse-grid operators through Galerkin assembly.

We remark that the proposed solver is a nested multigrid as the pre-smoothing is absent at each level. This choice, w.r.t. to other multigrid strategies, is motivated by timing performance.

In Tables 5.3 and 5.4, we compare the multigrid convergence using different smoothers at the finest level and varying p. We note how the multigrid performance with respect to p differs using the listed approaches. In particular, as expected, for standard Gauss-Seidel smoothing, the multigrid convergence quickly deteriorates when increasing p. We also remark how the number of smoothing steps ν has to be increased for high p to obtain reasonable convergence.

Table 5.3. Number of iterations needed by a 5-level cascade multigrid for solving the linear system $K_{n,[p]}\mathbf{y} = \mathbf{1}$ up to a precision $\varepsilon = 10^{-8}$, in the case where $d = 2$, $K(\mathbf{x}) = I_2$. We perform ν smoothing steps at the finest level and a single Gauss-Seidel iteration at coarse levels. The smoother at the finest level is explicitly indicated in the table. We use $n = 259 - p$, corresponding to a number of DoFs equal to $\overline{n} = 257^2$.

$p\,[\nu]$	2[3]	3[4]	4[4]	5[5]	6[6]	7[7]	8[8]
Gauss-Seidel smoother	5	5	5	22	114	384	2154
ILU($K_{n,[p]}$) PCG smoother	5	5	5	7	20	68	282
$M_{n,[p]}$ PCG smoother	6	6	7	7	7	8	9

Remark 5.4. *Clearly, it is worth preconditioning $K_{n,[p]}$ with $M_{n,[p]}$ only if the latter matrix can be solved efficiently. This is the case, for example, if $M_{n,[p]}$ possesses a pure tensor structure, due to the rectangular nature of the physical domain Ω. In practice, using as preconditioner the incomplete LU factorization of $K_{n,[p]}$, like in the second line of Tables 5.3 and 5.4, can be competitive in terms of run-time and can*

Table 5.4. Number of iterations needed by a 5-level cascade multigrid for solving the linear system $K_{n,[p]}\mathbf{y} = \mathbf{1}$ up to a precision $\varepsilon = 10^{-8}$, in the case where $d = 2$,

$$K(x_1, x_2) = \begin{bmatrix} (3+x_1)e^{x_2} & 0.1 \\ -0.1(x_1+x_2) & \cos x_1 + 2 \end{bmatrix}.$$

We perform ν smoothing steps at the finest level and a single Gauss-Seidel iteration at coarse levels. The smoother at the finest level is explicitly indicated in the table. We use $n = 259 - p$, corresponding to a number of DoFs equal to $\overline{n} = 257^2$. Experiments are run in parallel with 20 processors.

$p\,[\nu]$	2[3]	3[4]	4[4]	5[5]	6[6]	7[7]	8[8]
Gauss-Seidel smoother	8	6	15	31	120	395	2260
ILU($K_{n,[p]}$) PCG smoother	7	6	6	10	25	71	348
$M_{n,[p]}$ PCG smoother	8	9	13	13	15	17	19

be adopted when dealing with unstructured spatial grids, where the tensor structure of $M_{n,[p]}$ is no longer available.

5.2.3 PGMRES for the space-time problem: less is more

The solver suggested in the current section for a linear system with coefficient matrix $C_{N,n}^{[q,p]}$ is PGMRES with preconditioner $P_{N,n}^{[q,p]} := P_{N,n}^{[q,p,p-1]}$; the solution of a linear system associated with $P_{N,n}^{[q,p]}$, which is required at each PGMRES iteration, is performed via the tensor solver described in Section 5.2.1 coupled with a suitable multigrid method for the space stiffness matrix $K_{n,[p]}$.

Actually, as it becomes clear experimentally (cf. Section 6.2.2), *the PGMRES method converges faster if the linear system with coefficient matrix $P_{N,n}^{[q,p]}$occurring at each PGMRES iteration is not solved exactly*. More precisely, when applying to $K_{n,[p]}$ the multigrid method described in Section 5.2.2, it is enough to perform only a few multigrid iterations in order to achieve an excellent PGMRES run-time and, in fact, only one multigrid iteration is sufficient. Unexpectedly, it was discovered experimentally that the PGMRES run-time is further improved if we do not perform *any smoothing step at the finest level* in the few multigrid iterations applied to $K_{n,[p]}$ (that is, smoothing steps occur only at coarse levels) while maintaining the p-robustness (cf. Section 6.2). For this reason the argumentation of Section 5.2.2, about the most appropriate fine level smoother, becomes no longer relevant in this context.

In view of these experimental observations, we propose to solve a linear sys-

tem with coefficient matrix $C_{N,n}^{[q,p]}$ in the following way:

- apply to the given system the PGMRES algorithm with preconditioner $P_{N,n}^{[q,p]}$;
- apply to the linear system with coefficient matrix $P_{N,n}^{[q,p]}$ occurring at each PGMRES iteration the tensor solver described in Section 5.2.1;
- the tensor solver would require solving $\overline{N}$ linear systems with coefficient matrix $K_{n,[p]}$ as for equations (5.4) or (5.5); instead of solving exactly these systems, apply to each of them η multigrid iterations involving no smoothing steps at the finest level, a single Gauss-Seidel post-smoothing step at coarse levels, and standard bisection for the interpolation and restriction operators at all levels (following the Galerkin approach).

Note that this solver can be referred to as an inexact PGMRES method because, at each iteration, the linear system associated with the preconditioner is not solved exactly. As we shall see in the numerical tests of Section 6.2.2, the choice $\eta = 1$ yields the best performance of the solver.

5.3 Non-linear solver for the space-time monodomain equation

The discretized monodomain equation in (4.4) is non-linear. We linearize it by means of the Newton method; in particular, we solve $\mathbf{F}(\mathbf{u}) = 0$, with

$$\mathbf{F}(\mathbf{u}) = C_{N,n}^{[q,p,k]}\mathbf{u} + \mathbf{r}(\mathbf{u}) - \mathbf{f}, \tag{5.6}$$

using the linear solvers discussed in the current chapter to *invert* the Jacobian

$$\mathbf{JF}(\mathbf{u}) = C^{[q,p,k]} + \mathbf{Jr}(\mathbf{u}).$$

From (4.5) and (4.6) we have

$$\mathbf{Jr}_{\mathrm{i}}(\mathbf{u}) = \left(I_N \otimes \frac{\Delta t}{2} M_{[q]} \otimes M_{n,[p,k]}\right) \mathbf{JI}_{\mathbf{ion}}(\mathbf{u}),$$
$$\mathbf{Jr}_{\mathrm{e}}(\mathbf{u}) = \left(L_N \otimes \frac{\Delta t}{2} M_{[q]} \otimes M_{n,[p,k]}\right) \mathbf{JI}_{\mathbf{ion}}(\mathbf{u}),$$

for the implicit and explicit case respectively with

$$\mathbf{JI}_{\mathbf{ion}}(\mathbf{u}) = \begin{bmatrix} I'_{\mathrm{ion}}(u_1) & & & \\ & I'_{\mathrm{ion}}(u_2) & & \\ & & \ddots & \\ & & & I'_{\mathrm{ion}}(u_{\overline{nN}}) \end{bmatrix}.$$

Remark 5.5. *When dealing with nonlinear problems, Newton-MG is not the only option in terms of multilevel strategies. Using a non-linear multigrid, such as FAS, is also possible [Henson, 2002]. The latter strategy relies on a good representation of the non-linear operator on coarse spaces. When solving the monodomain equation, choosing a fine discretization in space and time is essential to capture the correct physical behavior, cf. Figure 5.9 . For this reason we can assume the Newton-MG approach to be preferable for this specific problem, as the Jacobian* **JF** *can also be evaluated with relative ease.*

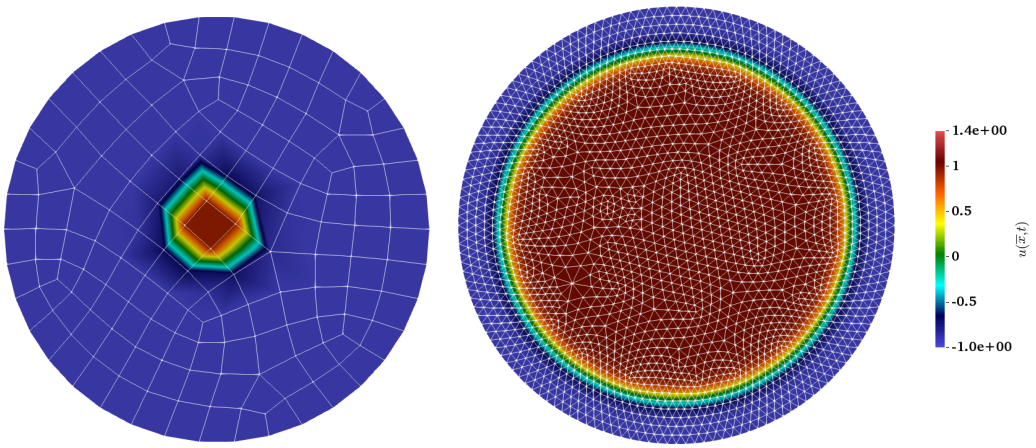

Figure 5.9. Solutions at the final time $T = 3$ to (5.6) using $p = 1, k = 0, q = 0, K = 0.01, N = 40$, the initial forcing factor $f(x,t) = 4\,\chi_{[0,t_1)} \cdot e^{-\|x\|^2/0.01}$ and various n: $h \simeq 0.15$ (left) and $h \simeq 0.05$ (right). The spatial discretization step h has to be small enough to capture the expected front propagation (right) that is totally absent in the coarse model (left).

Chapter 6

Numerical Experiments

In Section 6.1 we describe our implementation focusing on its main parallel functionalities.

Section 6.2 is dedicated to results concerning the symbol precondiotioned GMRES, compared to a standard ILU-PGMRES. Particular emphasis is placed on p-robustness and parallel scaling.

In Section 6.3 we present results concerning the semi-geometric space-time multigrid. We discuss how the convergence of the semi-geometric approach may differ from the standard geometric one, with various numerical tests with complex geometries. Moreover, we explore how coarsening strategies, anisotropies and jumping coefficients can affect the space-time multigrid convergence and scaling.

While Sections 6.2–6.3 are dedicated to the solution of the heat equation, Section 6.4 tackles the solution of the monodomain problem, focusing on the stability of the described non-linear space-time approach.

In Section 6.5 we compare the scaling of the proposed space-time multigrid method with the one of PFASST, a state-of-the-art competitor.

Parallel numerical experiments have been performed on the multicore partition of the supercomputer Piz Daint of the Swiss national supercomputing centre (CSCS)[1]. We remark that the used Piz Daint partition is composed of computer nodes with 36 cores each. This might result in a certain loss of parallel efficiency when going from 16 cores (likely assigned to a single node) to 64 cores, as node to node communication is introduced.

[1] `https://www.cscs.ch/computers/piz-daint/`

6.1 Implementation and parallelization

6.1.1 External libraries

For the numerics of this section, as well as throughout this thesis, we used the C++ frameworks PETSc [Balay et al., 2019a,b] and the embedded domain specific language Utopia[2] [Zulian et al., 2016] for the parallel linear algebra and the linear and non-linear solvers. We stress that, when multiple processors are used, a block Jacobi preconditioning is employed by default by PETSc. For the assembly of high order IgA finite elements, we used the PetIGA[3] package [Dalcin et al., 2016]. On the other hand, to deal with non trivial geometries for the applications of Chapter 4, Utopia wraps functions from LibMesh[4] [Kirk et al., 2006] for the finite element assembly.

6.1.2 Design and functionalities

The software for assembling and solving the space-time problem has been designed to be highly modular and flexible, such that any functionality is not depending on the other ones. Figure 6.1 gives a graphical representation of the code structure. Let us give a short description of its main functionalities:

- `Space_assembler` : as the name suggests, this is an interface responsible for the assembly of the spatial operators. Sub-classes can encode a specific spatial problem. Even if in this thesis we focus on the Laplacian assembler, an assembly routine from an elasticity problem were also tested. Spatial matrices are assembled using libMesh or PetIGA, depending on the context once a mesh file is provided.

- `Time_assembler`: encodes time operators. We focused on the DG(q) approximation but other time advancing schemes have been implemented (e.g. Crank-Nicolson).

- `Space-time_assembler`: core class where space-time operators are assembled through tensor products (with the method `kron()`) and distributed in memory (with the method `distribute()`).

- `Space-time_multigrid`: where the coarse spaces hierarchy is generated and space-time transfer operators are created.

[2] `https://bitbucket.org/zulianp/utopia`
[3] `https://bitbucket.org/dalcinl/petiga`
[4] `http://libmesh.github.io/index.html`

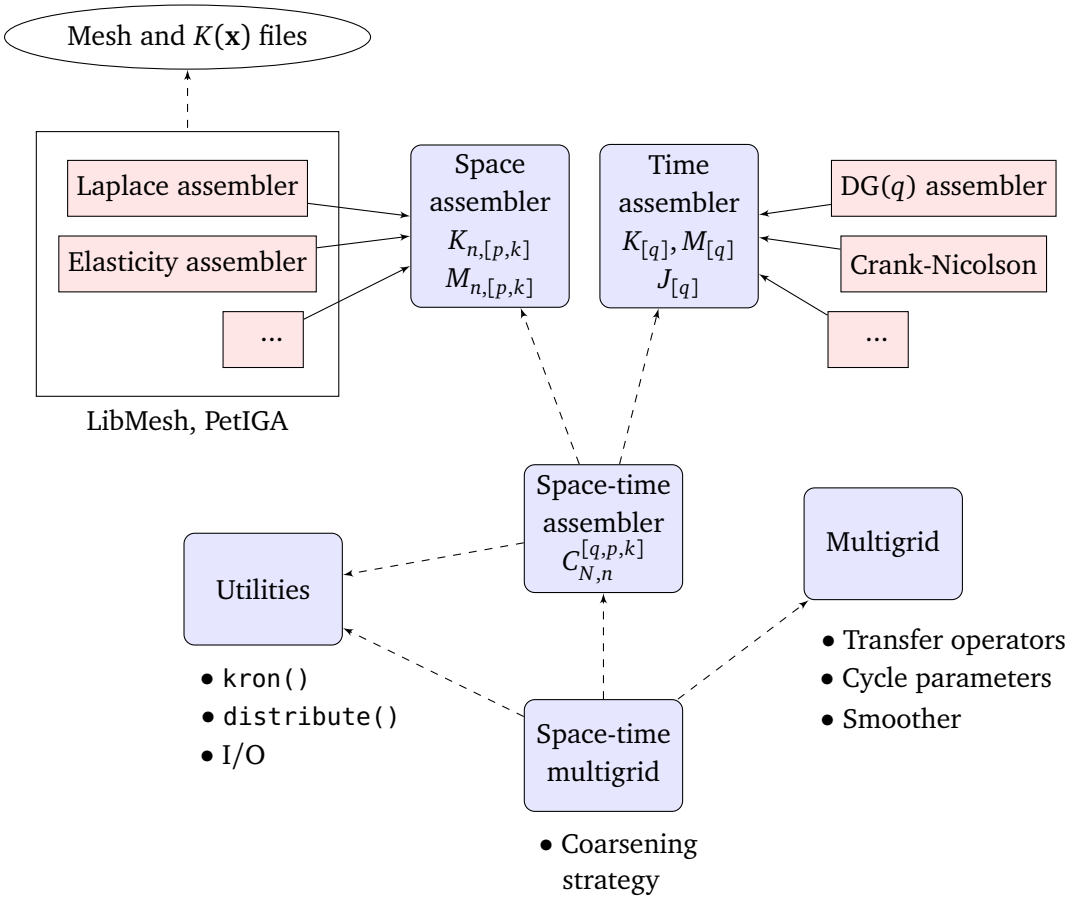

Figure 6.1. Class diagram of the software. Dashed arrows represent dependencies between classes and solid ones imply inheritance. Some of the operators are shown inside the box of the class where they are stored as attributes. Some methods or functionalities of classes are listed with bullets.

The method `kron()` implements the tensor product between two matrices in a distributed fashion. The tensor product is not a *local* operation, and portions of both input operators may need to be moved between processors; cf. Figure 6.2.

The function `distribute()` controls how the space-time matrices are distributed among multiple processors. If P is the number of processors used, the distribution of the space-time matrix $C_{N,n}^{[q,p,k]}$ is trivial only when $P = N$ and every processor gets exactly one time-slab block from the PETSc default distributed matrix assembly. If we include spatial parallelism, i.e. $P > N$, the subdivision of $C_{N,n}^{[q,p,k]}$ can not be left to chance. In particular we subdivide firstly $C_{N,n}^{[q,p,k]}$ in N blocks that are progressively partitioned. Figure 6.3 illustrates how this procedure generates suitable block preconditioners.

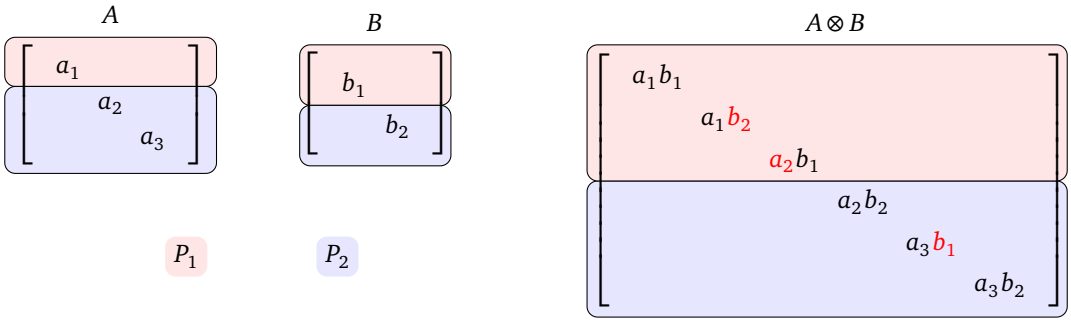

Figure 6.2. Row-wise distribution layout of the diagonal matrices A, B and their tensor product using two processors P_1 and P_2. In $A \otimes B$ red text labels quantities that need to be moved between processors.

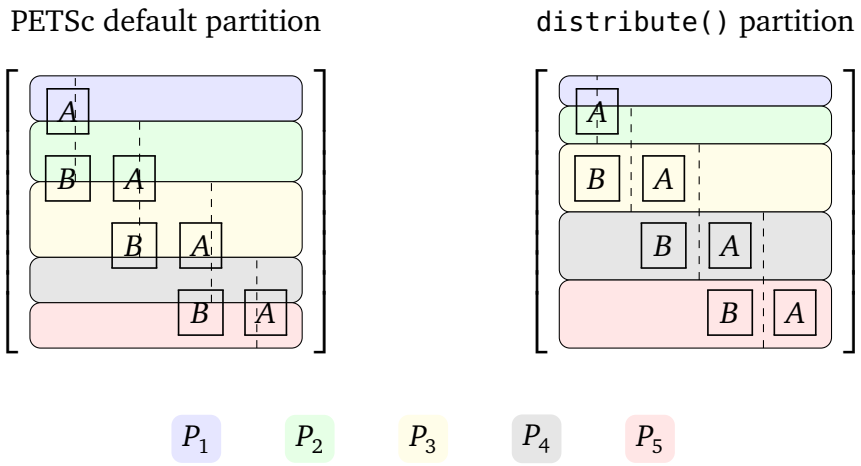

Figure 6.3. Partitions of the space-time matrix (2.31) using five processors and four time steps, i.e. $P = 5$ and $N = 4$. The PETSc default partition does not account for the structure of the space-time problem and produces block diagonal preconditioners that are not symmetric; diagonal blocks are delimited by dashed lines. Using `distribute()`the matrix $C_{N,n}^{[q,p,k]}$ is initially subdivided into N blocks that are progressively partitioned as P increases.

The PETSc library uses, as default, parallel preconditioning (such as block Jacobi) when multiple processors are used to solve a linear system. We refer to the PETSc website[5] for the list of the current implementations of linear solvers.

[5] `https://www.mcs.anl.gov/petsc/documentation/linearsolvertable.html`

6.2 Robust PGMRES for isogeometric analysis

In this section we present the results regarding the solver introduced in Section 5.2.3.

6.2.1 Experimental settings

In the numerics of this section, we solve the linear system $C_{N,n}^{[q,p]}$ from (2.31) arising from the choices $d = 2$, $f(t, \mathbf{x}) = 1$, $T = 1$. The values of $K(\mathbf{x})$, N, n, q, p are specified in each example. As in Chapter 3, we consider a uniform grid on $\Omega = [0, 1]^d$. For each solver considered herein, we use $\varepsilon = 10^{-8}$ as a tolerance and the PETSc default stopping criterion[6]. We use the zero vector as initial guess for the linear solver. Whenever we report the run-time of a solver, the time spent in I/O operations and matrix assembly (also for transfer operators) is ignored; run-times are always expressed in seconds. In all the tables below, the number of iterations needed by a given solver to converge within the tolerance $\varepsilon = 10^{-8}$ is reported in square brackets next to the corresponding run-time. Throughout this section, we use the following abbreviations for the solvers.

- $\boxed{\text{ILU}(C_{N,n}^{[q,p]})\ \text{PGMRES}}$

  Preconditioned GMRES with preconditioner given by an incomplete LU (ILU) factorization of the system matrix $C_{N,n}^{[q,p]}$.

- $\boxed{\text{MG}_{\eta,\nu}^{L}\ \text{PGMRES}}$

  The proposed solver, as described in Section 5.2.3, with η multigrid iterations applied to $K_{n,[p]}$. Each multigrid iteration involves ν Gauss-Seidel smoothing steps at the finest level (typically $\nu = 0$) and 1 Gauss-Seidel smoothing step at the coarse levels. The superscript L denotes the number of multigrid levels.

- $\boxed{\text{TMG}_{\eta,\nu}^{L}\ \text{PGMRES}}$

[6] PETSc, by default, uses the relative preconditioned residual in the stopping criterion: when solving the system $A\mathbf{u} = \mathbf{f}$ with preconditioner P, we stop iterating at the first vector $\mathbf{u}$ satisfying

$$\|P^{-1}(\mathbf{f} - A\mathbf{u})\| / \|P^{-1}\mathbf{f}\| < \varepsilon.$$

The same as "$\text{MG}^{L}_{\eta,\nu}$ PGMRES", with the only difference that the multigrid iterations are performed with the telescopic option, thus giving rise to the telescopic multigrid (TMG) [Douglas, 1996; May et al., 2016]. This technique consists in halving the number of processors used across the grid hierarchy: if N_f processors are used on the fine grid ($l = 1$), then we use $N_f/2^{l-1}$ processors on level l. This strategy turned out to be essential for the parallel multigrid performance, as shown in Section 6.2.3. In a naive setting the number of degrees of freedom per core decreases rapidly on coarse levels, resulting in a degradation of convergence on those levels.

6.2.2 Convergence study and timing

Tables 6.1–6.3 illustrate the performance of the proposed solver in terms of number of iterations and run-times. It is clear from the table that the solver is superior to the classical PGMRES with preconditioner given by the ILU factorization of the system matrix $C^{[q,p]}_{N,n}$. Moreover, the best performance of the solver is obtained when applying to $K_{n,[p]}$ a single multigrid iteration ($\eta = 1$) with no smoothing steps at the finest level ($\nu = 0$). It should also be noted that the solver is considerably robust with respect to the spline degree p as both number of iterations and run-time do not grow significantly with p.

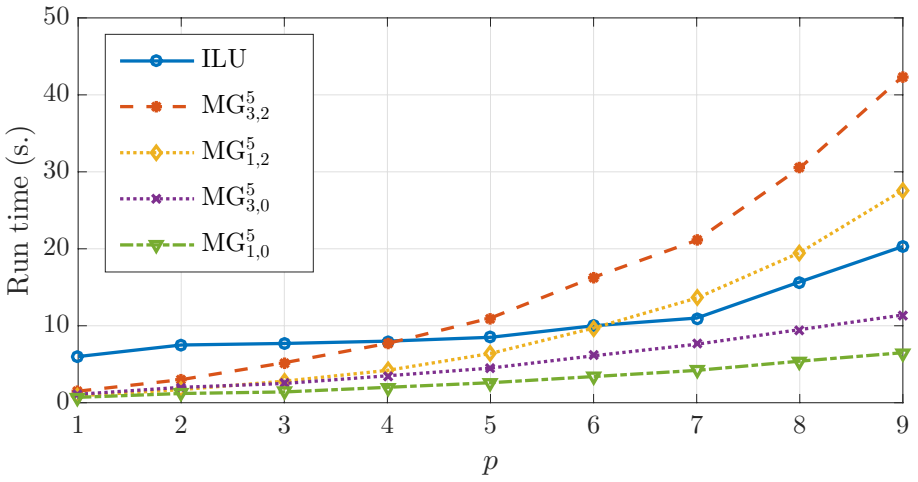

Figure 6.4. Graphical representation of the run-times reported in Table 6.1.

In particular, Tables 6.2–6.3 show how p-robustness and performance of the proposed solver are superior, w.r.t. the ILU-PGMRES, when a non trivial $K(\mathbf{x})$ is considered.

Table 6.1. PGMRES run-times and iterations (using 64 cores) to solve the linear system (2.31) up to a precision of 10^{-8}, according to the experimental setting described in Section 6.2.1. We used $K(\mathbf{x}) = I_2$, $q = 0$, $N = 32$ time steps and $n = 259 - p$. The total size of the space-time system (number of DoFs) is given by $32 \cdot 257^2$. A graphical representation of the run-times is given in Figure 6.4.

p	ILU($C_{N,n}^{[0,p]}$) PGM.	$MG_{3,2}^5$ PGM.	$MG_{1,2}^5$ PGM.	$MG_{3,0}^5$ PGM.	$MG_{1,0}^5$ PGM.
1	6.0 [545]	1.5 [33]	0.9 [33]	1.1 [33]	0.7 [33]
2	7.5 [332]	3.0 [33]	1.7 [33]	2.0 [33]	1.2 [33]
3	7.7 [249]	5.2 [33]	2.8 [33]	2.5 [33]	1.4 [33]
4	8.0 [211]	7.7 [33]	4.2 [34]	3.5 [33]	2.0 [33]
5	8.5 [182]	11.0 [34]	6.4 [37]	4.5 [33]	2.6 [33]
6	10.1 [164]	16.3 [36]	9.7 [41]	6.1 [33]	3.4 [33]
7	11.0 [147]	21.1 [38]	13.6 [44]	7.6 [33]	4.2 [33]
8	15.7 [166]	30.5 [43]	19.5 [52]	9.5 [33]	5.4 [33]
9	20.3 [174]	42.4 [50]	27.6 [61]	11.4 [33]	6.5 [33]

Table 6.2. PGMRES run-times and iterations (using 64 cores) to solve the linear system (2.31) up to a precision of 10^{-8}, according to the experimental setting described in Section 6.2.1. We used

$$K(x_1, x_2) = \begin{bmatrix} \cos(x_1) + x_2 & 0 \\ 0 & x_1 + \sin(x_2) \end{bmatrix},$$

$q = 1$, $N = 20$ time steps and $n = 131 - p$. The total size of the space-time system (number of DoFs) is given by $40 \cdot 129^2$.

p	1	2	3	4	5
ILU($C_{N,n}^{[q,p]}$) PGMRES	1.3 [442]	2.0 [282]	2.7 [210]	3.0 [178]	3.5 [153]
$MG_{1,0}^5$ PGMRES	0.4 [46]	0.7 [46]	0.9 [46]	1.2 [45]	1.6 [45]
p	6	7	8	9	
ILU($C_{N,n}^{[q,p]}$) PGMRES	4.0 [137]	5.0 [132]	6.9 [143]	10.4 [176]	
$MG_{1,0}^5$ PGMRES	2.1 [46]	2.7 [47]	3.3 [47]	4.2 [49]	

Table 6.3. PGMRES run-times and iterations (using 64 cores) to solve the linear system (2.31) up to a precision of 10^{-8}, according to the experimental setting described in Section 6.2.1. We used

$$K(x_1, x_2) = \begin{bmatrix} (2+\cos x_1)(1+x_2) & \cos(x_1+x_2)\sin(x_1+x_2) \\ \cos(x_1+x_2)\sin(x_1+x_2) & (2+\sin x_2)(1+x_1) \end{bmatrix},$$

$q = 0$, $N = 20$ time steps and $n = 259 - p$. The total size of the space-time system (number of DoFs) is given by $20 \cdot 257^2$.

p	1	2	3	4	5
ILU($C_{N,n}^{[q,p]}$) PGMRES	2.2 [441]	2.7 [256]	3.4 [189]	3.7 [161]	4.2 [141]
$MG_{1,0}^{5}$ PGMRES	0.2 [12]	0.4 [11]	0.4 [11]	0.6 [11]	0.7 [12]
p	6	7	8	9	
ILU($C_{N,n}^{[q,p]}$) PGMRES	4.9 [127]	5.7 [115]	9.5 [151]	41.2 [477]	
$MG_{1,0}^{5}$ PGMRES	0.9 [12]	1.2 [13]	2.4 [13]	1.9 [14]	

6.2.3 Strong and weak scaling

In the scaling experiments, besides the multigrid already considered above, we also use a TMG for performance reasons (cf. Section 6.2.1 for details and references about the TMG). From Table 6.4 and Figure 6.5 we see that the proposed solver, especially when using the TMG option, shows a nearly optimal strong scaling with respect to the number of cores. Table 6.5 and Figure 6.6 illustrate the weak scaling properties of the proposed solver, which possesses a remarkably superior parallel efficiency with respect to the standard ILU preconditioning in terms of iteration count and run-time. In fact, the efficiency of the proposed solver can be estimated to be about three times the one of the standard ILU-PGMRES.

Table 6.4. Strong scaling: PGMRES run-times and iterations to solve the linear system (2.31) up to a precision of 10^{-8}, according to the experimental setting described in Section 6.2.1. We used $K(\mathbf{x}) = I_2$, $q = 0$, $p = 3$, $N = 64$ time steps and $n = 384$. The total size of the space-time system (number of DoFs) is given by $64 \cdot 385^2$. Additionally we report, up to parallel saturation, run-times for a time-sequential forward solve, i.e. using the parallel $\text{TMG}^7_{1,0}$ PGMRES for the spatial problem with traditional time stepping with implicit Euler.

Cores	ILU($C^{[q,p]}_{N,n}$) PGMRES	$\text{MG}^7_{1,0}$ PGMRES	$\text{TMG}^7_{1,0}$ PGMRES	Forward solve
1	2591 [394]	304.2 [63]	304.2 [63]	25.0
2	1220 [403]	139.4 [63]	142.5 [63]	19.1
4	603 [394]	70.0 [63]	69.5 [63]	15.0
8	340.9 [396]	39.7 [63]	41.5 [63]	11.7
16	232.6 [396]	20.4 [63]	25.9 [63]	10.0
32	112.0 [397]	26.1 [63]	21.3 [63]	11.6
64	99.3 [399]	12.0 [63]	10.7 [63]	11.6
128	55.0 [479]	6.2 [64]	5.7 [63]	-
256	28.7 [497]	3.4 [64]	2.8 [62]	-
512	18.1 [531]	2.6 [64]	1.7 [62]	-
1024	11.5 [603]	4.7 [65]	1.0 [62]	-
2048	60.0 [2397]	16.9 [65]	0.85 [62]	-
4096	35.3 [2571]	65.0 [65]	0.83 [62]	-

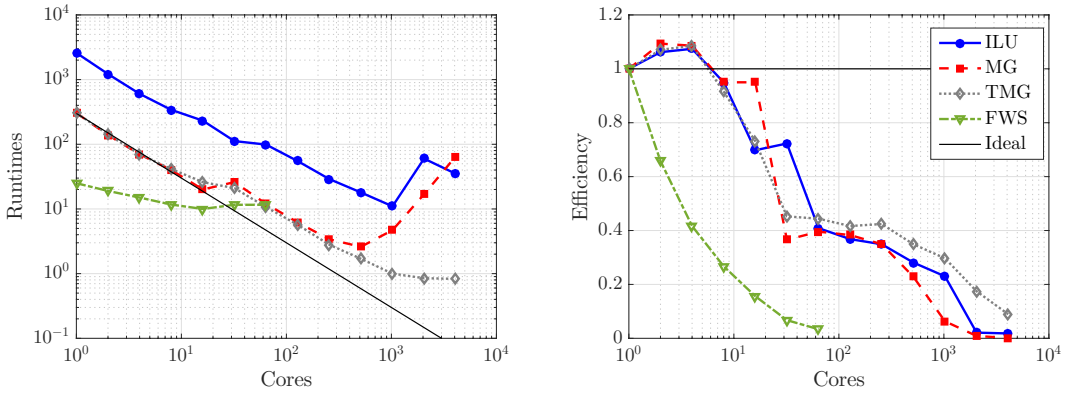

Figure 6.5. Graphical representation of the run-times reported in Table 6.4 and the corresponding efficiency, "FWS" standing for forward solve.

Remark 6.1. *Figure 6.5, similarly to the results in [Falgout et al., 2014], well illustrates how a sequential time stepping approach is advantageous when just a limited amount of parallel resources area available. In this particular experiment the space-time approach pays off when #Cores* $\gtrsim 100$ *and an additional factor of 10 in speedup can be achieved.*

Table 6.5. Space-time weak scaling: PGMRES iterations and run-time to solve the linear system (2.31) up to a precision of 10^{-8}, according to the experimental setting described in Section 6.2.1. We used $K(\mathbf{x}) = I_2$, $q = 0$, $p = 2$, and $(N, n) = (8, 65)$, $(16, 129)$, $(32, 256)$, $(64, 512)$. The ratio DoFs/Cores is constant in the table.

$[\text{Cores}, n, N, L]$	$[1, 65, 8, 4]$	$[8, 129, 16, 5]$	$[64, 257, 32, 6]$	$[512, 513, 64, 7]$
ILU($C_{N,n}^{[q,p]}$) PGM.	0.18 [47]	0.74 [113]	10.9 [332]	103.8 [1354]
$\text{TMG}_{1,0}^{L}$ PGMRES	0.08 [10]	0.21 [17]	0.90 [33]	1.97 [62]

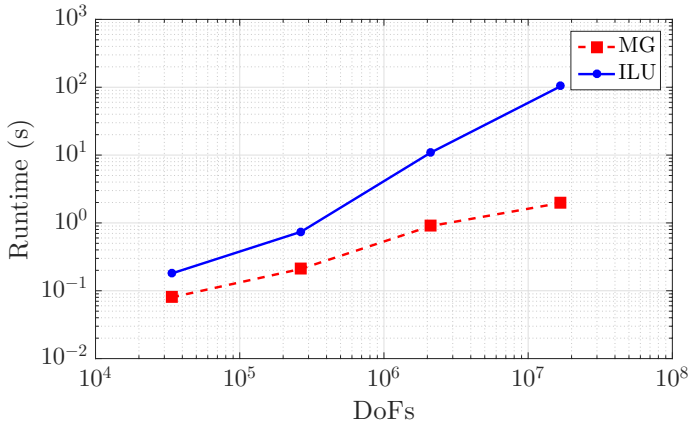

Figure 6.6. Graphical representation of the run-times reported in Table 6.5.

6.3 Semi-geometric space-time multigrid

In this section we perform several numerical experiments on the space-time multigrid described in Section 5.1. In all the examples we discretize Ω with linear finite elements, i.e. $p = 1$ and $k = 0$, and we consider homogeneous Neumann boundary conditions in space. Moreover, we use the following multigrid common settings: V-cycling with $\nu = 3$ pre/post smoothing steps of the ILU preconditioned GMRES and a tolerance of 10^{-9}. We remark that when multiple processors are used a block Jacobi preconditioner is used *on top* of the ILU one, using the decomposition described in Figure 6.3. The type of transfer used (L^2 projection or interpolation) to assemble coarse operators will be specified case by case: if nothing is specified L^2 projection will be used. We will use these settings also in Section 6.4 and 6.5.

6.3.1 On the semi-geometric approach for the spatial problem

As described in Section 5.1.3 we use a semi-geometric approach to generate coarse meshes of the spatial domain. Because of the novelty of this technique, we perform a specific numerical study on its convergence properties. In particular, can we observe the same convergence rates of geometric multigrid? We try to answer this question with 2D and 3D experiments.

Experimental settings

Let us consider the Poisson problem with a Gaussian forcing term:

$$\Delta u(\mathbf{x}) = 4 \cdot e^{-\|\mathbf{x}\|^2/2}, \quad \mathbf{x} \in \Omega. \tag{6.1}$$

The domain Ω will be specified in each example, where we compare the geometric approach with the semi-geometric one. For the latter a coarsening factor $C_x = 2$ is used. Run times (in seconds) and iterations in square brackets are collected in Tables 6.6-6.9. For practical reasons, for geometric multigrid, we consider hierarchies with maximum 3 levels, as we can not generate coarse meshes at run-time.

Table 6.6. Run-times and iterations to solve problem (6.1) using a geometric and a semi-geometric approach on the geometry reported in Figure 6.7. The type of transfer used is indicated in parentheses.

	$L = 2$	$L = 3$	$L = 4$
MG (L^2 proj.)	0.032 [5]	0.014 [5]	-
Semi-geometric MG (interp.)	0.026 [4]	0.014 [5]	0.009 [5]
Semi-geometric MG (L^2 proj.)	0.035 [4]	0.012 [4]	0.009 [4]

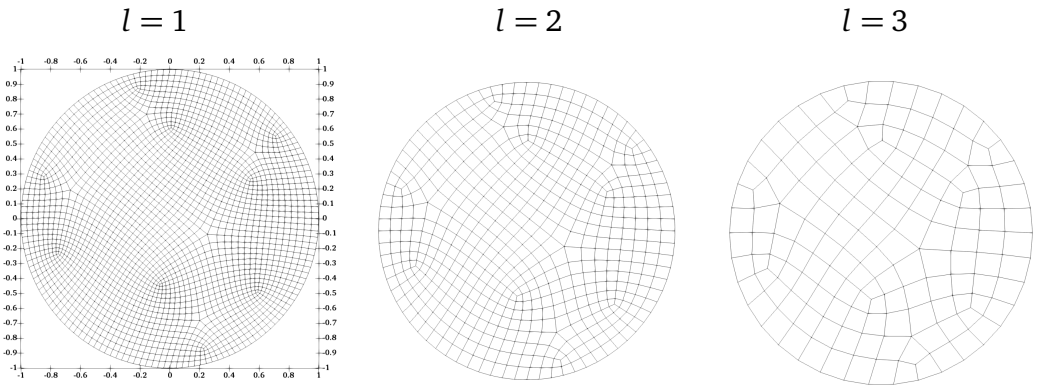

Figure 6.7. Fine mesh ($l = 1$) and corresponding coarse meshes ($l = 2, 3$) used for the geometric multigrid with #elements n (respectively): 2317, 560 and 160. For the semi-geometric multigrid case, nested uniform grids on the bounding square are used as coarse meshes, with #elements n: 576, 144 and 36.

Remark 6.2. *We observe an h-independent convergence rate for the proposed semi-geometric multigrid, using the geometry reported in Figure 6.7 . In particular, the multigrid converges in 4 iterations using 4,3 and 2 levels for the three meshes respectively, such that the coarsest problem has 36 elements in all three cases.*

Table 6.7. Run-times and iterations to solve problem (6.1) using a geometric and a semi-geometric approach on the geometry of Figure 6.8. Standard interpolation has been used.

	$L=2$	$L=3$	$L=4$
MG	0.1 [4]	0.04 [4]	-
Semi-geometric MG	0.05 [5]	0.04 [12]	0.03 [12]

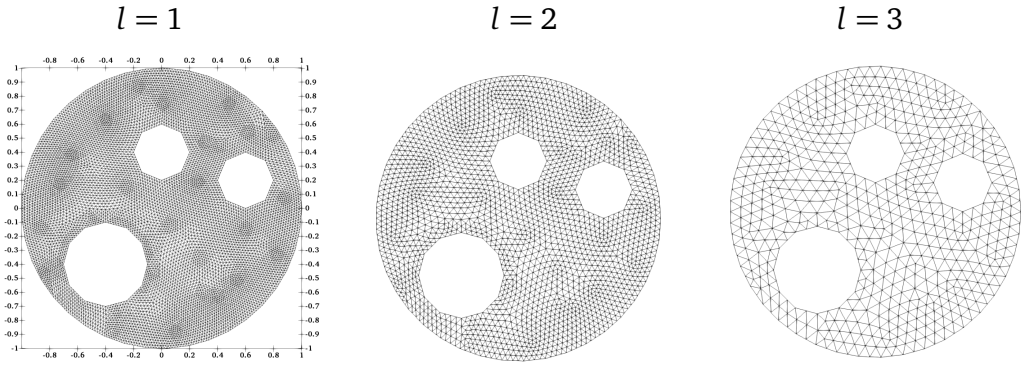

Figure 6.8. Fine mesh and corresponding coarse meshes with n (respectively)= 8498, 2192 and 581 used for the geometric multigrid. For the semi-geometric multigrid nested uniform grids on the bounding square are used as coarse meshes, with #elements: 961, 121 and 8.

Table 6.8. Run-times and iterations to solve the Poisson problem (6.1) using a geometric and a semi-geometric approach on the geometry of Figure 6.9. L^2 projection has been used to assemble transfer operators. As reference value, the ILU-PGMRES converges in 3.9 seconds with 621 iterations.

	$L=2$	$L=3$	$L=4$
MG	7.4 [5]	1.4 [5]	-
Semi-geometric MG	7.1 [5]	1.9 [10]	1.5 [10]

Table 6.9. Run-times and iterations to solve the Poisson problem (6.1) using a geometric and a semi-geometric approach on the geometry of Figure 6.10 on four cores. L^2 projection has been used to assemble transfer operators. As reference value, the ILU-PGMRES converges in 4.0 seconds and 555 iterations.

	$L=2$	$L=3$	$L=4$
MG	5.7 [5]	1.1 [5]	-
Semi-geometric MG	4.1 [4]	4.8 [53]	4.7 [53]

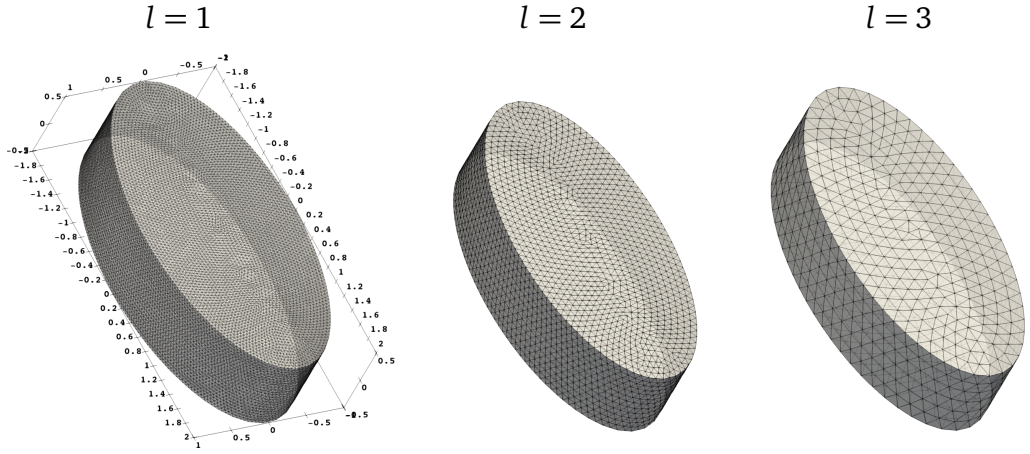

Figure 6.9. Fine mesh and corresponding coarse meshes with n (respectively): $138{\cdot}10^3$, 18163 and 2494 used for the geometric multigrid. For the semi-geometric multigrid uniform grids on the bounding cube are used as coarse meshes, with n: 17576, 2197 and 343

As a qualitative example, In Figure 6.11, we report the solution of problem (6.1) on a non convex disconnected domain Ω distributed between four processors (cf. Figure 6.12).

Remark 6.3. *The experiments in the current section suggest how the described semi-geometric approach is especially well suited when the problem geometry is well approximated by its bounding box. This is true, in particular, for $L > 2$, where we observe poor smoothers convergence on the coarse levels. Replacing the bounding box with a bounding polygon might address this issue, at the cost of a higher computational effort to build coarse spaces.*

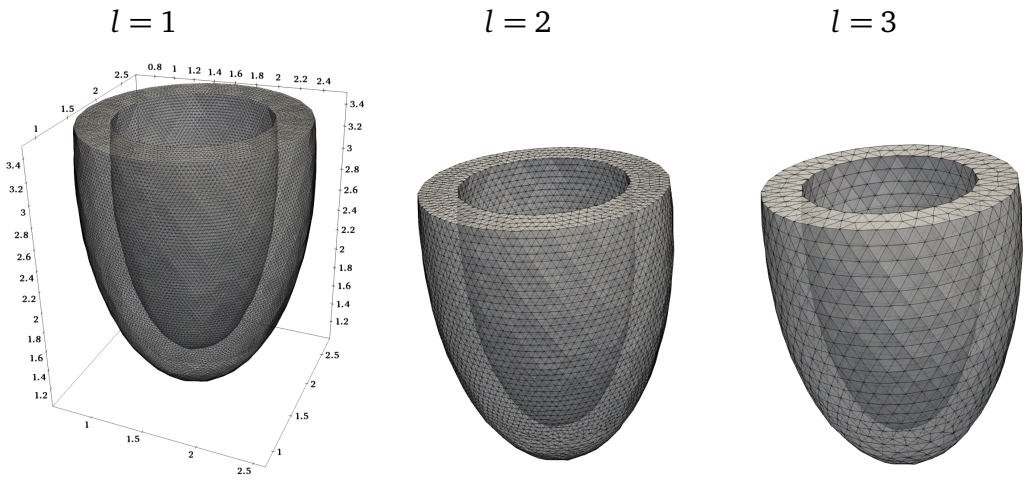

Figure 6.10. Fine mesh and corresponding coarse meshes with n (respectively): $162{\cdot}10^3$, $22{\cdot}10^3$ and 3154 used for the geometric multigrid. For the semi-geometric multigrid uniform grids on the bounding cube are used as coarse meshes, with n: $19{\cdot}10^3$, 2744 and 343.

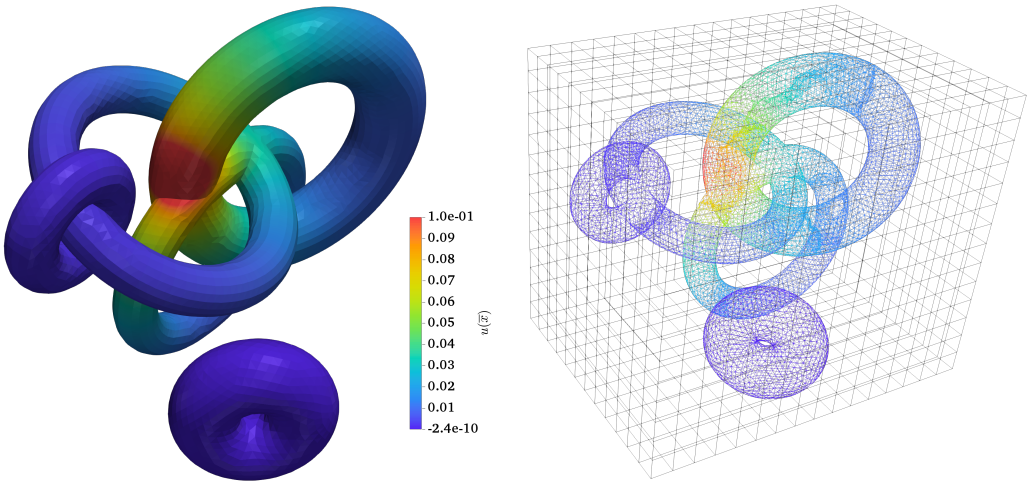

Figure 6.11. Left: solution of the problem (6.1) using a two level semi-geometric multigrid converged in 19 iterations. Right: the automatically generated coarse mesh.

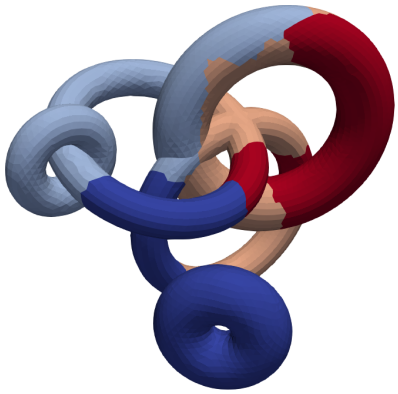

Figure 6.12. Non trivial geometry distributed among four processors.

6.3.2 Space-time multigrid convergence for the heat equation

We consider the diffusion problem (2.1) ($r = 0$) discretized in (2.28) and solved on four cores. We impose the initial condition through the forcing term

$$f(t, \mathbf{x}) = \chi_{\{0\}}(t) \cdot 4\, e^{\|\mathbf{x}\|^2/2}, \tag{6.2}$$

and set $u_0(\mathbf{x}) = 0$, i.e. $\mathbf{f} = [\mathbf{f}_1, \mathbf{0}, ..., \mathbf{0}]^T$ in (2.28). Additional problem/solver parameters will be specified in each example. We use the following notation:

- $\boxed{\mathrm{MG}^L[C_x, C_t]}$

  Semi-geometric multigrid with L levels and C_x, C_t coarsening factors.

As expected from the discussion of Section 5.1.1 we can observe how the convergence rate of the proposed space-time multigrid is determined by the coarsening strategy, i.e. C_x and C_t (cf. Tables 6.10–6.11 and Figure 6.13). In all the following tables, as sanity check, multigrid convergence is compared with the one of the ILU-PGMRES solver (ILU-PGM.).

Table 6.10. Run-times and iterations to solve (2.28) using the geometry of the finest mesh in Figure 6.8, $K = I_2$, $q = 0$ and $N = 129$ time steps with a 4 levels semi-geometric multigrid. We vary the type of coarsening strategy and the final time T (i.e. Δt and μ); given the particular geometry ($h \simeq 0.05$) we have $\mu = 3.75 \cdot T$. The L^2 projection has been used to assemble transfer operators.

	MG^4[2,2]	MG^4[2,1]	MG^4[4,1]	MG^4[$\sqrt{2}$,2]	MG^4[2,4]	ILU-PGM.
$T = 10^{-3}$	0.78 [2]	1.0 [2]	0.4 [2]	6.4 [2]	0.5 [2]	0.2 [19]
$T = 10^{-2}$	0.97 [3]	1.3 [3]	0.5 [3]	6.4 [3]	0.7 [3]	0.2 [16]
$T = 10^{-1}$	1.2 [4]	1.5 [4]	0.9 [5]	6.1 [4]	0.9 [4]	0.6 [49]
$T = 1$	1.6 [6]	1.8 [5]	0.9 [5]	7.0 [6]	2.3 [13]	7.3 [578]
$T = 10$	3.2 [18]	4.3 [15]	1.8 [11]	9.9 [13]	6.7 [36]	19.0 [1454]
$T = 10^2$	17.2 [79]	13.6 [51]	3.5 [36]	32.0 [63]	31.5 [184]	55 [4135]

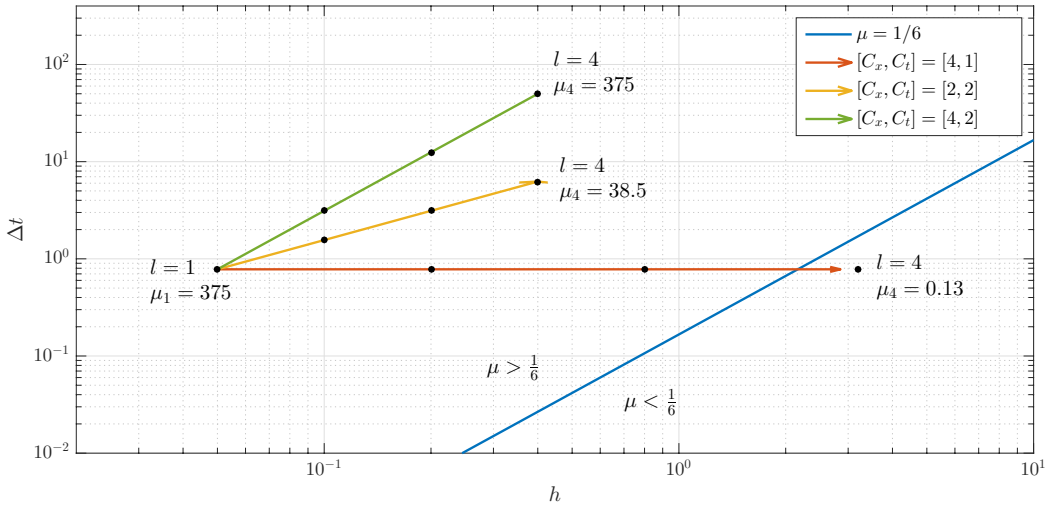

Figure 6.13. Representation of the space-time hierarchy on the $(h, \Delta t)$ plane for the setting of Table 6.10 and $T = 100$, corresponding to the last row of the table. Black dots represents μ values along the multigrid hierarchy (from $l = 1$ to $l = 4$) using different coarsening strategies. We can notice how, in this particular case, an aggressive spatial coarsening (red arrow) is expected to produce a better conditioned sequence of operators, in agreement with the number of iterations reported in Table 6.10.

Table 6.11. Run-times and iterations to solve (2.31) using the 3D geometry of the intermediate mesh in Figure 6.9, $K = I_2$, $q = 0$ and $T = 1$ with a 2 levels semi-geometric multigrid. We vary the type of coarsening strategy and the number of time steps N (i.e. Δt and μ); given the particular geometry ($h \simeq 0.086$) we have $\mu \simeq 135/N$. The L^2 projection has been used to assemble transfer operators.

	MG^2[2,2]	MG^2[2,1]	MG^2[4,1]	MG^2[4,2]	MG^2[2,4]	ILU-PGM.
$N = 5, \mu \simeq 32$	3.2 [14]	1.0 [5]	0.2 [6]	0.6 [14]	3.7 [26]	0.5 [129]
$N = 9, \mu \simeq 16$	4.1 [9]	1.5 [4]	0.5 [6]	0.7 [9]	6.0 [23]	1.5 [168]
$N = 17, \mu \simeq 8$	10.4 [13]	3.0 [4]	0.9 [6]	1.9 [13]	11.3 [21]	3.5 [205]
$N = 33, \mu \simeq 4$	15.8 [9]	7.4 [4]	1.9 [5]	3.0 [9]	30.2 [26]	12 [358]
$N = 65, \mu \simeq 2$	22.5 [5]	21.3 [4]	4.6 [5]	4.6 [6]	45.7 [15]	22.8 [328]
$N = 129, \mu \simeq 1$	80 [4]	73 [7]	13.7 [5]	10.6 [5]	72 [6]	43.9 [328]

Remark 6.4. *Table 6.11 illustrates how time coarsening (i.e. $C_t > 1$) is not effective, in terms of both iterations and run time, as μ increases. Moreover, especially for 3D problems, $n \gg N$ and space coarsening is more relevant to reduce coarse problem size.*

6.3.3 Anisotropic or jumping diffusion coefficient

Both applications presented in Chapter 4, in realistic settings, present non-smooth or anisotropic diffusion coefficients. In the electrophysiology application, for example, the fiber orientation plays an important role in the electrical activation of the cardiac tissue, see Figure 6.14 for a visual example[7].

On the other hand, the convergence of multigrid methods is known to deteriorate when diffusion coefficients exhibit anisotropy or large jumps [Hackbusch, 2013; Chan and Wan, 2000]. In the next examples we test the presented semi-geometric multigrid in such a scenario.

Example 6.5. *Let us consider the case of an anisotropic $K_{an}(\mathbf{x})$ on a cube, such that the diffusion is strong along one of the main diagonals. A precise description of $K_{an}(\mathbf{x})$ is reported in Appendix A. We solve* (2.28) *with $q = 1$, $T = 1$, $N = 65$ on uniform mesh with $n = 21^3$. Again, we use the forcing term* (6.2) *using semi-geometric multigrid with $L = 3$ and $C_x = C_t = 2$ (i.e. space-time coarsening) on four cores. We represent the solution in Figure 6.15, compared to the one where a constant diffusion is used. We observe convergence to the desired tolerance in 6*

[7] Source: `https://commons.wikimedia.org/wiki/File:2006_Heart_Musculature.jpg`

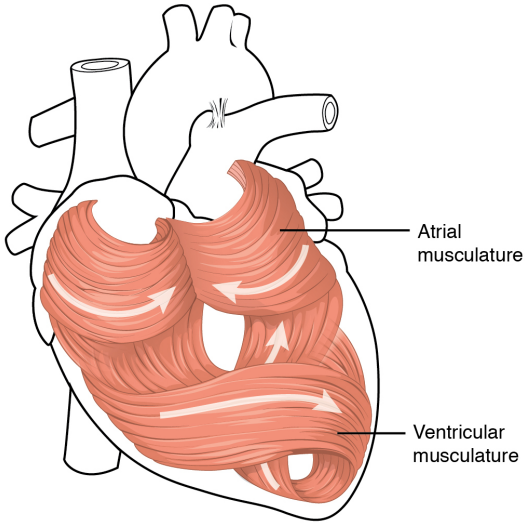

Figure 6.14. The potential front travels mainly along the direction of the cardiac fibers.

iterations, while, in the constant diffusion case, the solver converges in 5. We can conclude that, in this setting, there is no significant degradation in the multigrid performance.

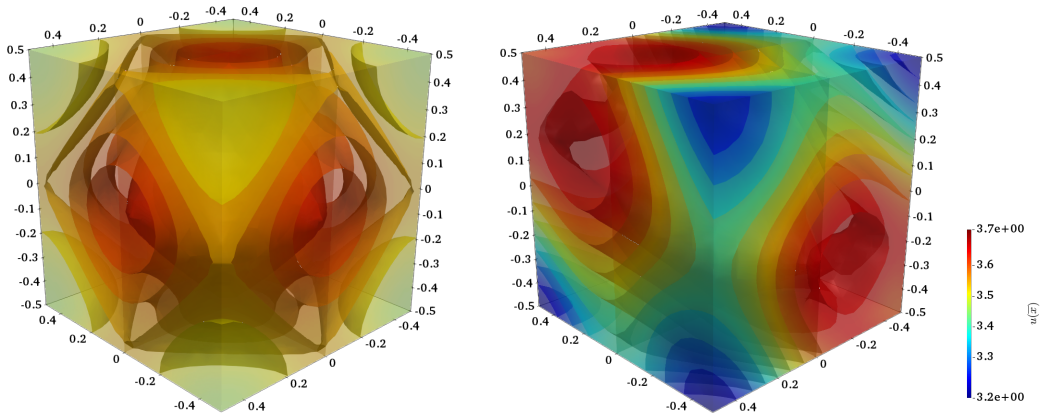

Figure 6.15. Comparison between the contours of the solution of Example 6.5 at $t = T/2$ for a constant diffusion $K(\mathbf{x}) = 1$ (left) and the anisotropic one (right).

Example 6.6. *Let us consider the case of a complex geometry coupled with a less regular anisotropy; in particular we used an anisotropic* $K_{an}(\mathbf{x})$ *which includes some*

spatially correlated noise (cf. Figure 6.16) and $n = 27 \cdot 10^3$, $h_{\max} = 1$. *A precise description of* $K_{an}(\mathbf{x})$ *is reported in Appendix A. We solve* (2.28) *with the Gaussian forcing term* (6.2) *and* $q = 0$, $T = 10^3$ *and* $N = 65$. *In Figure 6.17 we report the solution for* $t = T$, *compared to the one where a constant diffusion* $K_{const}(\mathbf{x}) = 2 \cdot 10^{-3}$ *is used. In the next table we compare iterations and run-times needed to converge by a two-level semi-geometric multigrid and by an ILU-PGMRES, both run on four processors.*

	MG^2[2,2]	MG^2[4,1]	MG^2[4,2]	ILU-PGMRES
$K_{\text{const}}(\mathbf{x})$	25.0 [26]	8.6 [37]	7.4 [39]	4.2 [169]
$K_{\text{an}}(\mathbf{x})$	523 [2017]	700+ [3000+]	600+ [3000+]	62.2 [2516]

In this case we can observe a severe degradation of performances for all the approaches.

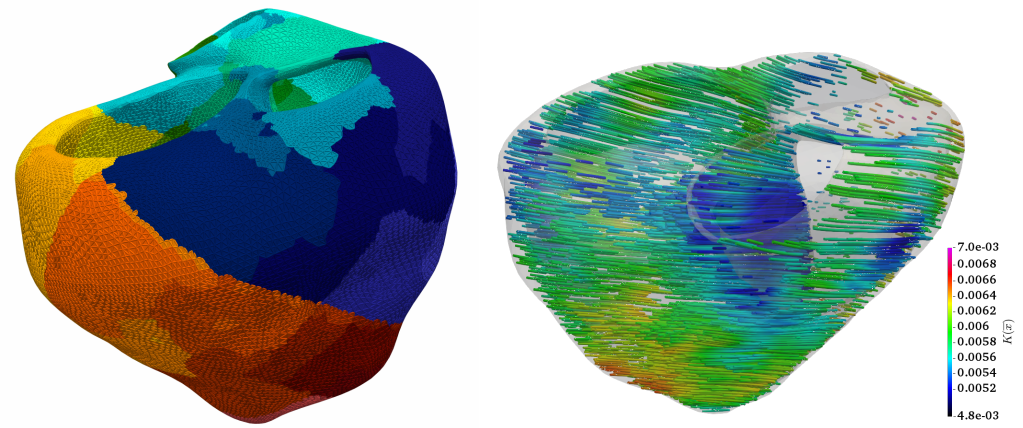

Figure 6.16. Left: heart geometry meshed with $n \simeq 5 \cdot 10^5$; different colors represent the mesh distribution among 32 processors. Right: visualization of the anisotropic tensor diffusion $K_{\text{an}}(\mathbf{x})$ from Appendix A (not physiological).

Example 6.7. *Let us consider, similarly to [Kreienbuehl et al., 2015b], the case of a jumping diffusion coefficient (cf. Section 4.2) defined by*

$$K_{\text{jump}}(\mathbf{x}) = \begin{cases} 1, & \text{if} \quad |\sin(8x_1)\sin(8x_2)| < 0.05, \\ 10^{-3}, & \text{else.} \end{cases} \tag{6.3}$$

We solve (2.28) *with the Gaussian forcing term* (6.2) *and* $q = 0$, $T = 1$ *and* $N = 65$ *on the disk geometry of Figure 6.7 discretized with* $n = 9406$. *In Figure 6.18 we*

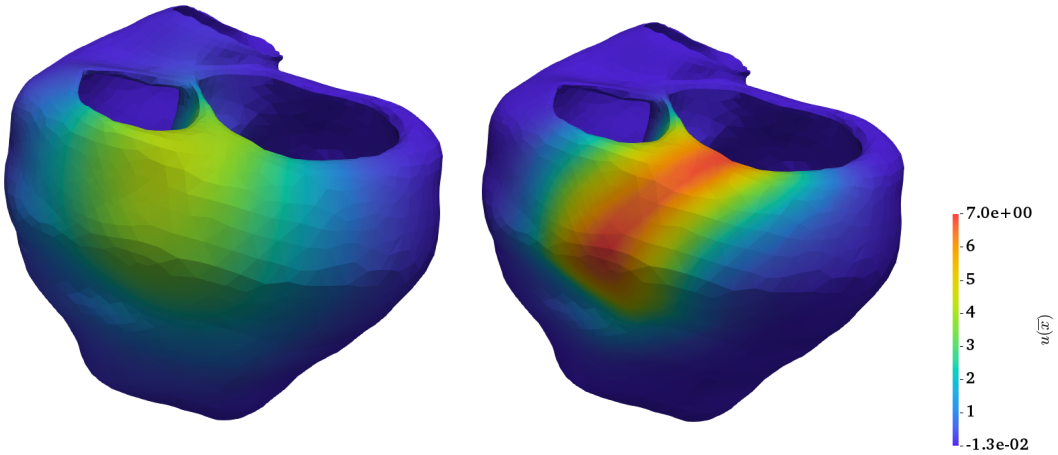

Figure 6.17. Solutions from Example 6.6 at final time T for a constant diffusion (left) versus an anisotropic one (right).

report the solution for $t = T$. In the next table we compare iterations and run-times needed to converge by four level semi-geometric multigrid and by an ILU-PGMRES, both run on four processors.

	MG^4[2,2]	MG^4[4,1]	MG^4[4,2]	ILU-PGMRES
$K(\mathbf{x}) = 1$	3.6 [13]	1.3 [7]	2.3 [13]	21.9 [820]
$K(\mathbf{x}) = 10^{-3}$	1.9 [6]	1.0 [5]	1.1 [6]	1.6 [17]
$K_{\text{jump}}(\mathbf{x})$	5.9 [22]	4.6 [26]	4.9 [28]	9.8 [340]

Example 6.8. *As in Example 6.7 we use the discontinuous K_{jump} from* (6.3)*. We solve* (2.28) *with the forcing term*

$$f(t, \mathbf{x}) = \chi_{\{0\}}(t) \cdot 10\, e^{(x_2-1)^2/0.1^2},$$

and $q = 0$, $T = \{1, 10\}$ and $N = 65$ on $\Omega = [-1, 1]^2$ discretized with a uniform grid with $n = 162^2$ elements ($h = 0.0125$). In Figure 6.19 we report the solution for $t = 5$. In the next table we compare iterations and run-times needed to converge by three level semi-geometric multigrid and by an ILU-PGMRES, both run on four processors. We can define

$$\mu(\mathbf{x}) = K(\mathbf{x})\Delta t/h^2 \simeq K(\mathbf{x}) \cdot 10^2 \cdot T.$$

	MG^3[2,2]	MG^3[4,1]	MG^3[4,2]	ILU-PGMRES
$K(\mathbf{x}) = 1, T = 1$	24.2 [15]	3.2 [5]	6.8 [14]	83.4 [1038]
$K_{\text{jump}}(\mathbf{x}), T = 1$	39.1 [17]	12.5 [23]	13.0 [27]	29.9 [367]
$K(\mathbf{x}) = 1, T = 10$	60.0 [67]	3.2 [5]	30.8 [62]	261[3072]
$K_{\text{jump}}(\mathbf{x}), T = 10$	49.6 [52]	13.4 [24]	35.6 [72]	73.2 [997]

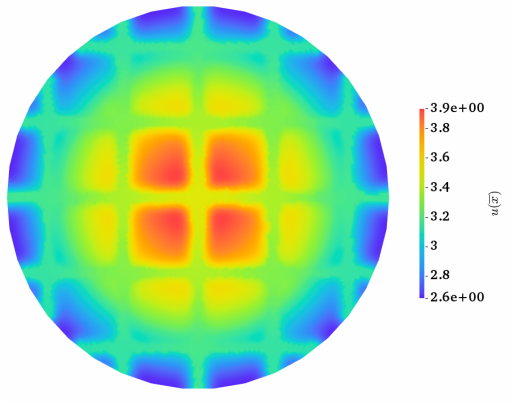

Figure 6.18. Solution $u(\mathbf{x}, T)$ from Example 6.7.

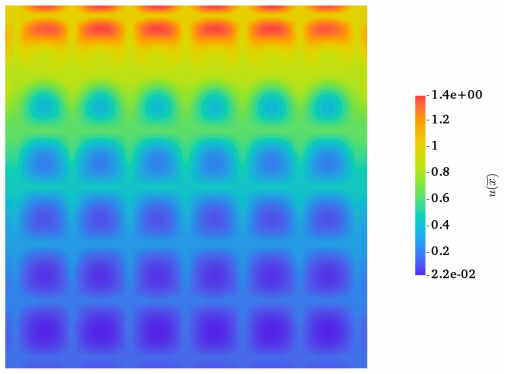

Figure 6.19. Solution $u(\mathbf{x}, 5)$ from Example 6.8.

6.3.4 Strong scaling and weak scaling in time

We present, in Examples 6.9–6.11, strong and weak scaling experiments. For technical reasons, in this section, the telescopic option for multigrid was not used, resulting in a certain loss in parallel efficiency[8].

Example 6.9. *Let us consider the solution of problem* (2.28) *on* $\Omega = [-0.5, 0.5]^3$ *uniformly discretized with* $n = 40^3$ *elements with the right hand side 6.2. Moreover let* $q = 0, T = 1, K(\mathbf{x}) = 0.1$ *and* $N = 65$ *time steps. We compare run-times and iterations using a space-time multigrid with* $L = 4, C_x = 3, C_t = 2$ *with an ILU-PGMRES and a forward solve (FWS). In the forward solve we use a spatial multigrid* $(L = 4, C_x = 2)$ *to solve the single time block,* $A_n^{[q,p,k]}\mathbf{u}_{t+1} = B_n^{[q,p,k]}\mathbf{u}_t$*, equivalent to a block Gauss-Seidel approach on* $C_{N,n}^{[q,p,k]}$*. Results are collected in Table 6.12 and illustrated in Figure 6.20.*

[8] In these examples, we used the multigrid implementation of Utopia; the telescopic option can be used only if the built-in PETSc multigrid is used.

Cores	$MG^4[3,2]$	ILU-PGMRES	Forward solve ($MG^4[2]$)
1	54.4 [4]	181.2 [234]	27.2 [3]
2	31.2 [5]	76.0 [195]	19.5 [3]
4	17.0 [5]	49.1 [244]	11.0 [3]
8	9.7 [6]	29.7 [267]	6.8 [3]
16	5.4 [6]	18.2 [270]	4.7 [3]
32	3.4 [6]	20.1 [362]	3.3 [3]
64	2.5 [6]	11.6 [438]	3.0 [3]
128	1.9 [6]	7.1 [426]	4.3 [3]
256	1.6 [6]	3.2 [448]	4.6 [3]
512	0.9 [6]	1.6 [406]	9.1 [3]
1024	1.1 [6]	1.0 [450]	-
2048	1.2 [6]	0.4 [456]	-

Table 6.12. Run-times and iterations from Example 6.9.

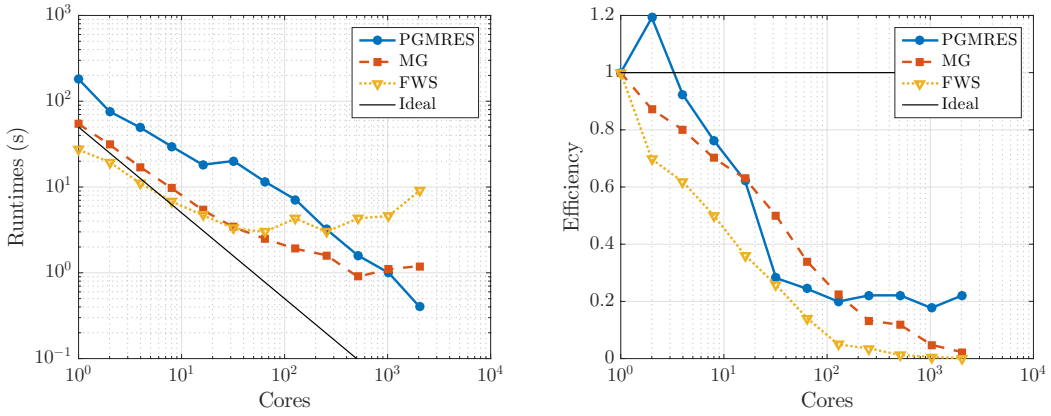

Figure 6.20. Results from Table 6.12 and corresponding efficiency.

In the next examples we test weak scaling in time, i.e. we increase N as the number of cores grows. We consider two scenarios: keeping T fixed or Δt fixed.

Example 6.10. *Let us consider the same problem setting described in Example 6.9 but using $N = \#\text{Cores}$. We compare, in Table 6.13, run-times and iterations needed to converge by a ILU-PGMRES and a space-time multigrid with $L = 5, C_x = 2$ and $C_t = \{1, 2, 4\}$. Run-times are illustrated in Figure 6.21.*

Cores $= N$	$MG^5[2,1]$	$MG^5[2,2]$	$MG^5[2,4]$	ILU-PGMRES
17	1.5 [3]	3.1 [7]	4.8 [21]	1.5 [94]
33	2.3 [4]	3.8 [6]	6.6 [19]	7.1 [263]
65	2.9 [6]	4.0 [6]	6.5 [18]	11.6 [429]
129	3.0 [6]	4.2 [6]	7.2 [20]	18.2 [669]
257	4.2 [9]	4.9 [7]	7.0 [18]	27 [987]
513	8.2 [16]	5.6 [9]	7.3 [16]	48 [1653]
1025	14.1 [34]	8.5 [14]	7.7 [19]	77 [2849]
2049	35.0 [62]	16.8 [29]	9.8 [19]	153 [5083]

Table 6.13. Run-times and iterations of Example 6.10; T is fixed to 1.

Example 6.11. *Let us consider the same problem setting described in Example 6.9 but using $N = \#Cores$ and $\Delta t = 0.015$ such that $T = N \cdot \Delta t$. We compare, in Table 6.14, run-times and iterations needed to converge by a ILU-PGMRES and a space-time multigrid with $L = 5, C_x = 2$ and $C_t = \{1, 2, 4\}$. Run-times are illustrated in Figure 6.22;*

Cores $= N$	$MG^5[2,1]$	$MG^5[2,2]$	$MG^5[2,4]$	ILU-PGMRES
17	1.6 [3]	2.9 [6]	3.8 [13]	0.6 [35]
33	2.0 [3]	4.0 [7]	5.5 [14]	5.9 [218]
65	2.4 [4]	4.0 [6]	6.5 [18]	10.7 [393]
129	3.0 [6]	4.1 [6]	8.5 [25]	18.3 [677]
257	3.1 [6]	4.3 [6]	11.9 [36]	27.5 [1019]
513	5.1 [8]	4.8 [7]	15.4 [48]	49 [1657]
1025	6.4 [10]	7.8 [9]	19.3 [97]	87 [2933]
2049	17.4 [11]	11.9 [14]	33.8 [101]	158 [5226]

Table 6.14. Run-times and iterations of Example 6.11; T is increasing with N but Δt (and μ) are fixed.

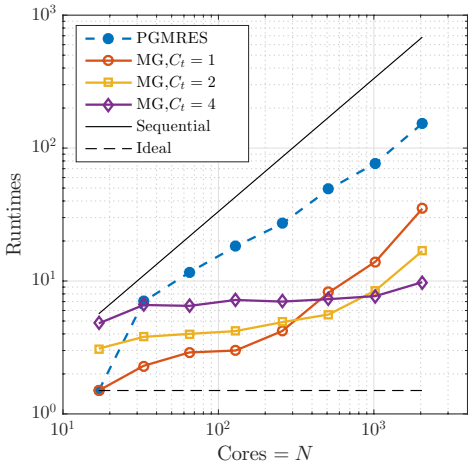

Figure 6.21. Table 6.13 run-times.

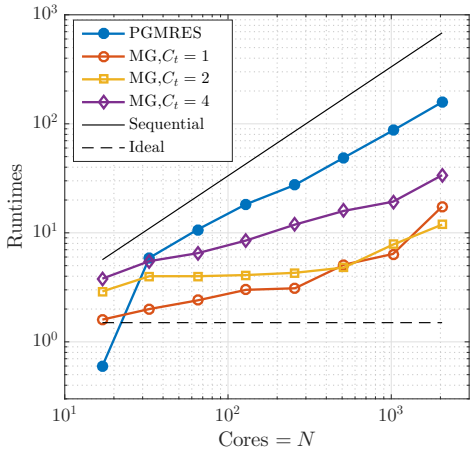

Figure 6.22. Table 6.14 run-times.

Remark 6.12. *Examples 6.10–6.11 show how time-coarsening, when possible, can be necessary to obtain near optimal weak scaling in time. Especially for large N time coarsening is essential to keep the computational cost of solving coarse problems limited.*

6.4 Monodomain equation

Let us consider the monodomain equation in (4.1), its discretization in (4.4) and its linearization discussed in Section 5.3. We made a distinction between an implicit or explicit treatment of the reaction term, $\mathbf{r}_\mathrm{i}$ or $\mathbf{r}_\mathrm{e}$. All the experiments in the current section concern *only the implicit approach*, the explicit one being disadvantageous, in all the cases considered, in terms of run-time and stability. If not specified, we set the monodomain parameters $C_m = \chi = 1$ in (4.1).

For performance reasons, we apply an adaptive strategy to set the linear solver tolerance in each Newton iteration. Given the non-linear residual $\mathbf{F}(\mathbf{u}^k)$ from (5.6), at the kth Newton iteration, we solve the Jacobian $\mathbf{JF}(\mathbf{u}^k)$ with tolerance

$$\mathrm{tol}_k = \|\mathbf{F}(\mathbf{u}^k)\| \cdot \min\left\{\sqrt{\|\mathbf{F}(\mathbf{u}^k)\|}, 0.5\right\},$$

resulting in an inexact Newton method [Eisenstat and Walker, 1996]. It is well known how the choice of the initial guess can strongly influence the convergence behavior of the Newton strategy (cf., for example, Table 4.1). We use the initial guess $\mathbf{u}^0 \equiv u_\mathrm{rest}$; this choice is motivated, along with the description of more sophisticated strategies in Appendix B.

In Examples 6.13–6.14 we explore how the non-linear solver convergence depends on the problem and discretization parameters. In Examples 6.15–6.17 we solve the monodomain equation in a realistic setting.

Example 6.13. *We solve* (4.4) *with* $N = 65, q = \{0,1\}, T = 2, K(\mathbf{x}) = 10^{-3}$ *on* $\Omega = [-1,1]^2$ *discretized with a uniform grid with* $n = 162^2$ *elements using the uniformly distributed random forcing term*

$$f(t,\mathbf{x}) = \chi_{\{0\}}(t) \cdot 4\, rand_{[-1,1]}(\mathbf{x}), \tag{6.4}$$

and the following FHN parameters in (4.2)*:* $a = \{0.1, 1, 10, 100\}, u_{rest} = -1, u_{thres} = 0, u_{max} = 1$ *and* $C_w = 0$*.We report the number of Newton iterations to reach a* 10^{-9} *tolerance using 4 cores.*

	$a = 0.1$	$a = 1$	$a = 10$	$a = 100$
$q = 0$	3	7	22	diverge
$q = 1$	3	7	22	diverge

In Figure 6.23 solutions for different choices of a are shown, in Figure 6.24 the corresponding Newton convergence is presented.

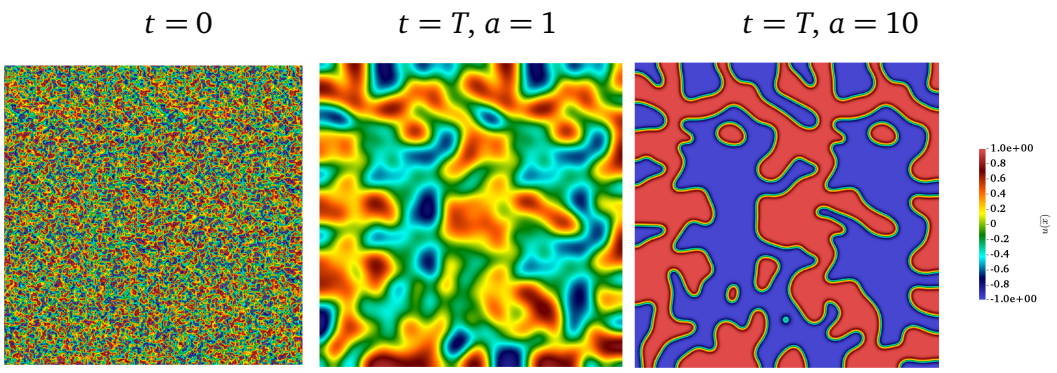

Figure 6.23. Solutions of the problem described in Example 6.13 for various t and a.

Example 6.14. *We solve* (4.4) *for various time parameters* N, q*,* T *and* $K(\mathbf{x}) = 10^{-3}$ *on* $\Omega = [-1,1]^2$ *discretized with a uniform grid with* $n = 162^2$ *elements using the uniformly distributed random forcing term in* (6.4) *and the following FHN parameters in* (4.2)*:* $a = 1, u_{rest} = -1, u_{thres} = 0, u_{max} = 1$ *and* $C_w = 0$*. We*

report the number of Newton iterations to reach a 10^{-9} tolerance using 4 cores in Tables 6.15, 6.16 and 6.17.

N	100	80	60	40	20	10	5	2
$q=0$	9	9	10	10	10	11	12	wrong sol.
$q=1$	-	-	9	9	9	9	10	13
$q=2$	-	-	-	9	9	9	9	12

Table 6.15. Newton iteration for Example 6.14 using $T=4$.

T	0.1	0.5	1	2	4	8	16	32
$q=0$	4	4	5	7	10	14	20	diverge
$q=1$	4	4	5	7	9	14	20	39
$q=2$	4	4	5	7	9	14	20	32

Table 6.16. Newton iterations for Example 6.14 using $N=20$ time steps.

T	1	2	4	8	16	32
$q=0$	6	7	10	14	20	29
$q=1$	5	7	9	14	20	-

Table 6.17. Newton iterations for Example 6.14 varying T and $N=T/5$ such that $\Delta t=0.2$ is fixed in all the entries.

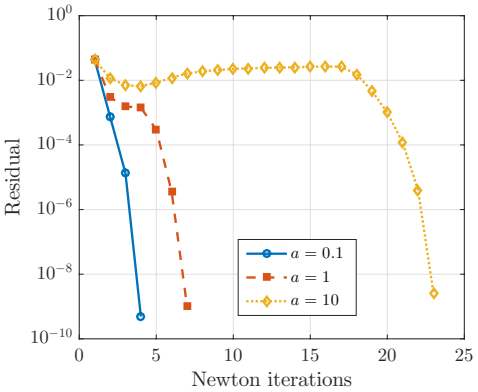

Figure 6.24. Newton convergence for Example 6.13 and $q=0$.

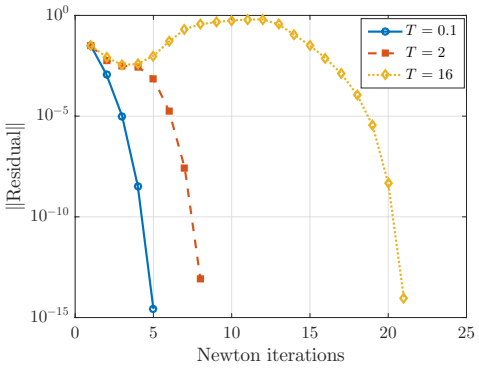

Figure 6.25. Newton convergence for Example 6.14 with $N=20$, $q=0$.

In the next examples we consider a more realistic setting and geometry. As discussed in [Pezzuto et al., 2016] and illustrated in Figure 5.9, to capture correctly the activation wave speed, we need to choose carefully the spatial discretization step, for example imposing $h \leq 1\,\text{mm}$. On the other hand, because the heart volume is approximately $100\,\text{mm}^3$, we would need a mesh with $n \simeq 10^6/h^3$ elements; for this reason it would be convenient to choose h as large as possible.

Example 6.15. *Let us solve the problem* (4.1) *using realistic parameters from [Pezzuto et al., 2016]:*

$$K(\mathbf{x}) = 0.3\,\text{ms}\cdot\text{mm}^{-1}, \quad C_m = 10^{-2}\mu\text{F}\cdot\text{mm}^{-1}, \quad \chi = 140\,\text{mm}^{-1},$$

$$u_{rest} = -85\,\text{mV}, \quad u_{thres} = -57\,\text{mV}, \quad u_{max} = 30\,\text{mV},$$

$$a = 1.4\cdot 10^{-5}\,\text{ms}\cdot\text{mm}^{-2}\text{mV}^{-2}$$

on a small portion of heart tissue $\Omega = [-10\,mm, 10\,mm]^3$ *discretized with* $n = 34^3$ *elements (i.e.* $h = 0.6\,mm$*),* $\Delta t = 0.05\,ms$*,* $q = 0$ *and the right hand side*

$$f(\mathbf{x}, t) = 4\,u_{max}\cdot\chi_{\{0\}}(t)\,e^{\|\mathbf{x}\|^2/2} - u_{rest}.$$

We report the number of Newton iterations and run-time to reach a 10^{-9} *tolerance using multiple cores and the ILU-PGMRES solver.*

#Cores	$N = 20$	$N = 40$	$N = 80$	$N = 160$	$N = 200$
4	12.6 [10]	33.0 [13]	86.7 [17]	289 [28]	-
16	4.6 [10]	11.9 [13]	29.0 [17]	93.3 [28]	141.7 [32]
64	2.9 [10]	12.3 [13]	29.5 [17]	82.4 [28]	120 [32]

As N *(and then* T*) increases more Newton iterations are required to converge, resulting in poor scaling and overall performance w.r.t. a time-sequential approach.*

Remark 6.16. *Let us consider that, using the parameters of Example 6.15, from* (4.3) *and* (3.23)*, we have*

$$\mu = \overline{K}(\mathbf{x})\Delta t/h^2 = K(\mathbf{x})(C_m\chi)^{-1}\Delta t/h^2 \simeq 3\cdot 10^{-2}.$$

Recalling the discussion in Section 5.1.1 as well as as the literature on space-time multigrid, for $\mu \ll 1$ *we can expect time semi-coarsening to be the only effective choice. However, time-coarsening, especially for a 3D problem, does not reduce the complexity of the coarse problems enough to make the multilevel approach convenient.*

Example 6.15 shows that, for practical purposes, the described approach is most likely unfeasible: the simulation of few heart beats would require $T \approx 10^3$ ms and thus $N \approx 2 \cdot 10^4$ time steps. For this reason we employ a hybrid approach, considering *M time blocks*, i.e. space-time problems on time intervals of length T, that we solve sequentially. When the ith time block is solved, with solution $\mathbf{u}_{[i]} = [\mathbf{u}_{[i],1}, ..., \mathbf{u}_{[i],N}]^T$, its final time solution $\mathbf{u}_{[i],N}$ is used as initial condition for the space-time problem on the next block:

$$\mathbf{u}_{[i+1],0} = \mathbf{u}_{[i],N}, \qquad \text{for} \quad i = 1, ..., M-1.$$

Moreover we can use $\mathbf{u}_{[i],N}$ as initial guess for the Newton iteration on the time block $i+1$, i.e.:

$$\mathbf{u}^0_{[i+1]} = I_N \otimes \mathbf{u}_{[i],N},$$

as illustrated in Figure 6.26.

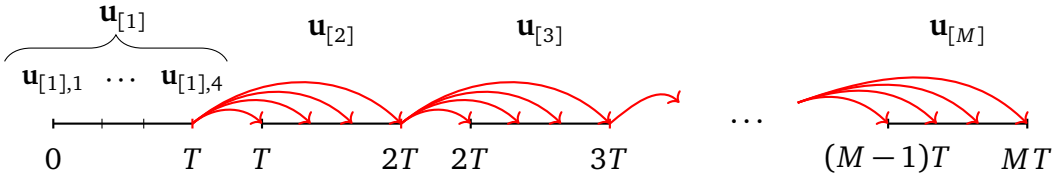

Figure 6.26. Representation of M time blocks with $N = 4$ time steps each. Red arrows show how an end-time solution is propagated forward in time, as Newton initial guess, on the following time blocks.

Example 6.17. *Let us consider the same monodomain parameters of Example 6.15 and the realistic geometry of the two ventricles from Figure 6.16 discretized with $n = 4.8 \cdot 10^5$ elements with $h = 1\,mm$. We solve (4.1) using various T, N and time blocks M, keeping constant the final time $M \cdot T = 200\,ms$ and $\Delta t = T/N = 1/32\,ms$, corresponding to a complete activation of the ventricles from the initial point-wise activation*

$$f(\mathbf{x}, t) = 10\, u_{max} \cdot \chi_{[0,t_1)}(t)\, e^{-\|\mathbf{x}\|^2/30} - u_{rest},$$

*cf. Figure 6.27. As N decreases the resulting strategy becomes more time-sequential: $N = 1$ corresponds to traditional time stepping. We report run-times (**in minutes**) using multiple cores in Table 6.18 using the ILU-PGMRES solver:*

#Cores	$N = 32$	$N = 1$
4	171	42.6
8	99.7	25.6
16	67.3	16.5
32	97.3	10.2
64	32.6	6.8
128	15.6	5.0
256	7.3	5.1
512	3.3	4.1
1024	1.6	4.6
2048	1.3	7.6
4096	1.0	14.6

Table 6.18. PGMRES run-times; $M = 6400/N$.

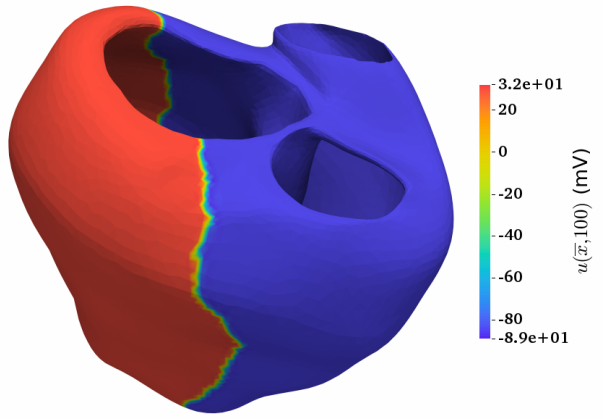

Figure 6.27. Solution of Example 6.17 at $t = 100\,\text{ms}$.

6.5 Comparison with PFASST

In the parallel-in-time literature (cf. Section 1.1) we can identify two main perspectives for multilevel approaches: Parareal and space-time multigrid (equivalent under certain restrictions [Gander et al., 2018b]). The PFASST algorithm [Emmett and Minion, 2012], was designed to overcome some limitations of Parareal. PFASST is based on the usage of high order SDC as iterative solver in time and can be seen as a p-multigrid algorithm; we refer to [Bolten et al., 2017] for a precise description of PFASST in the multigrid framework.

In Examples 6.18–6.23 we compare time-scaling of the space-time multigrid versus PFASST, using the Fortran library LibPFASST[9] that was extended to use PETSc data structures to make the comparison as fair as possible. For both approaches Gauss-Radau nodes are used to discretize time intervals (cf. Figure 2.1). We always compare schemes of the same order $(2q+1)$, for which the two algorithms provide the same solution, up to machine precision. For example, using PFASST on 3 levels, we use on each level, respectively, $q+1, q$ and $q-1$ Gauss-Radau nodes.

The tests are restricted to temporal parallelism, i.e. no spatial one is employed and we try to choose *optimal* solvers parameters in terms of run-time for both approaches. In PFASST, the spatial problem is solved with the ILU-PGMRES solver with default PETSc options and standard bisection is used to construct coarse problems in space ($C_x = 2$).

[9] `https://pfasst.lbl.gov/codes`

Example 6.18. *Strong scaling: let us consider the heat equation* (2.1) *on* $\Omega = [-100, 100]$ *discretized with* $n = 8192$ *elements and* $N = \{256, 1024\}$, $T = 1, q = 2, K(x) = 1$ *and the right hand side*

$$f(x,t) = 3\,\chi_{\{0\}}(t)\,e^{-x^2}.$$

We compare the run-times to solve (2.28) *via space-time multigrid (with parameters* $L = 6, C_t = C_x = 2$*), with the one using 3-level PFASST (PFA*^{3}*). Run-times (in seconds) are collected in Table 6.19 and illustrated in Figure 6.28.*

	$N = 256$		$N = 1024$	
#Cores	MG^6	PFA^3	MG^6	PFA^3
1	53.5	30.7	78	35.1
4	16.4	11.3	31.5	15.8
16	5.9	3.7	17.6	6.6
64	3.1	1.5	8.6	2.9
128	2.0	1.3	4.8	2.1
256	1.3	1.7	3.3	1.9
512	-	-	2.5	1.7
1024	-	-	2.3	2.0

Table 6.19. Run-times.

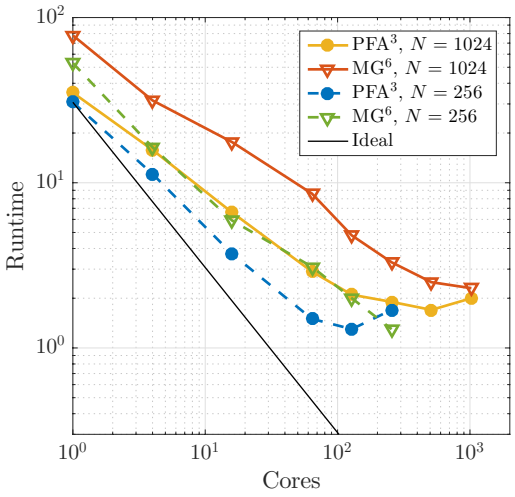

Figure 6.28. Run-times from Table 6.19.

Example 6.19. *Strong scaling: let us consider the heat equation* (2.1) *on* $\Omega = [0, 1]$ *with* $T = 0.1$*, discretized with* $N = n = 1025$*,* $q = \{0, \ldots, 4\}$ *and the initial forcing factor from*

$$f(x,t) = 2\,\chi_{\{0\}}(t)\,e^{-x^2/0.1^2}.$$

We show, in Tables 6.20–6.22, run-times and iterations of three multilevel strategies.

	MG^5[2,2]				
Cores	$q=1$	$q=2$	$q=3$	$q=4$	$q=5$
1	2.89 [6]	20.3 [18]	42.7 [19]	304 [88]	n.c.
4	0.86 [6]	5.5 [18]	11.2 [19]	79.5 [88]	n.c.
16	0.36 [8]	2.40 [21]	7.28 [35]	22.7 [68]	n.c.
64	0.17 [11]	1.14 [23]	3.33 [34]	8.70 [55]	n.c.
256	0.31 [37]	n.c.	n.c.	n.c.	n.c.
512	0.31 [25]	n.c.	n.c	n.c.	n.c.
1024	0.72 [42]	11.7 [489]	n.c	n.c.	n.c.

Table 6.20. Five levels space-time multigrid performance. We use the label "n.c." for "not converged".

	MG^7[2,1]				
Cores	$q=1$	$q=2$	$q=3$	$q=4$	$q=5$
1	1.46 [3]	4.39 [4]	8.22 [4]	14.8 [5]	51.9 [15]
4	0.38 [3]	1.14 [4]	2.50 [5]	3.93 [5]	13.4 [17]
16	0.18 [4]	0.52 [5]	0.98 [5]	1.77 [6]	2.30 [5]
64	0.07 [4]	0.26 [5]	0.48 [5]	0.89 [6]	1.39 [6]
256	0.06 [5]	0.12 [5]	0.21 [5]	0.33 [5]	0.58 [5]
512	0.07 [5]	0.13 [5]	0.21 [5]	0.35 [6]	0.54 [5]
1024	0.11 [5]	0.16 [5]	0.23 [5]	0.32 [5]	0.54 [5]

Table 6.21. Performance of a seven levels space-time multigrid, with no temporal coarsening.

	PFASST^3				
Cores	$q=1$	$q=2$	$q=3$	$q=4$	$q=5$
1	1.50 [1]	2.44 [1]	3.87 [2]	5.69 [3]	7.71 [3]
4	0.98 [3]	1.80 [5]	2.14 [5]	2.69 [6]	3.33 [6]
16	0.47 [4]	1.25 [12]	1.58 [13]	1.99 [14]	2.25 [14]
64	0.22 [6]	0.58 [19]	0.70 [21]	0.89 [24]	0.99 [24]
256	0.18 [7]	0.30 [26]	0.33 [28]	0.40 [33]	0.43 [32]
512	0.18 [8]	0.27 [29]	0.27 [32]	0.32 [37]	0.33 [37]
1024	0.21 [18]	0.28 [30]	0.28 [35]	0.30 [41]	0.31 [41]

Table 6.22. Three levels PFASST performance.

Remark 6.20. *In Example 6.19 we have $\mu \gg 1$ and, as we expect from the previous discussion, time coarsening is not beneficial for the convergence of space-time multigrid.*

Example 6.21. *Weak scaling in time: let us consider the same problem parameters as in Example 6.18 but with $N = \#\text{Cores}$ and the same solver settings. Run-times (in seconds) are collected in Table 6.23 and illustrated in Figure 6.29. In this experiment the run-times of PFASST increment significantly for $\#\text{Cores} \geq 512$. Run-times, on that range, must be taken with a grain of salt as they are not connected with an increase in PFASST iterations but are probably caused by a technical problem in time measures.*

#Cores$= N$	MG^6[2, 2]	PFA^3
17	0.9	0.6
33	1.2	0.7
65	1.6	1
129	1.4	0.7
257	1.5	0.9
513	1.9	1.7
1025	2.5	9
2049	3.2	-

Table 6.23. Run-times from Example 6.21 .

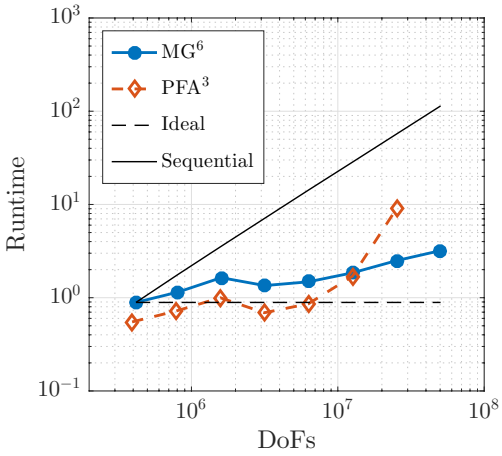

Figure 6.29. Run-times from Table 6.23. The number of degrees of freedom (DoFs) is given by DoFs $=$ Cores $\cdot\, n(q+1) \simeq N/4 \cdot 10^5$.

Example 6.22. *Weak time scaling in time: let us consider the same problem as in Example 6.19 but using $N = 4 \cdot$ Cores. We show, in Tables* **??***–6.25, run-times and iterations of PFASST and space-time multigrid.*

	$MG^7[2,1]$				
Cores	$q=1$	$q=2$	$q=3$	$q=4$	$q=5$
4	0.02 [6]	0.05 [9]	0.08 [9]	0.15 [9]	0.28 [12]
16	0.02 [5]	0.05 [6]	0.11 [7]	0.19 [8]	0.48 [13]
64	0.04 [5]	0.09 [6]	0.18 [7]	0.36 [8]	n.c.
256	0.08 [5]	0.16 [6]	0.28 [6]	0.45 [6]	n.c.
512	0.12 [5]	0.27 [6]	0.54 [7]	0.84 [7]	n.c.
1024	0.29 [5]	0.56 [7]	0.96 [7]	1.43 [7]	n.c.
2048	0.61 [6]	1.18 [7]	1.85 [7]	2.73 [7]	n.c.

Table 6.24. Performance of a seven levels space-time multigrid, with no temporal coarsening; "n.c" standing for not converged.

	$PFASST^3$				
Cores	$q=1$	$q=2$	$q=3$	$q=4$	$q=5$
4	0.03 [4]	0.06 [9]	0.07 [10]	0.09 [10]	0.10 [11]
16	0.04 [5]	0.10 [16]	0.15 [19]	0.17 [21]	0.20 [21]
64	0.07 [6]	0.16 [20]	0.20 [23]	0.25 [26]	0.28 [25]
256	0.17 [7]	0.27 [25]	0.33 [28]	0.39 [33]	0.42 [32]
512	0.29 [8]	0.43 [32]	0.50 [37]	0.60 [44]	0.65 [44]
1024	0.52 [11]	0.80 [57]	0.93 [68]	1.25 [97]	1.33 [94]
2048	1.02 [16]	n.c.	n.c.	n.c	n.c.

Table 6.25. Three levels PFASST performance; "n.c" standing for not converged.

Example 6.23. *Monodomain equation: let us consider problem* (4.1) *with the following settings:* $\Omega = [-100, 100], n = 8192, K(x) = 1, q = 1, T = 5, N = 256, a = 5, u_{\max} = 1, u_{\text{thres}} = 0, u_{\text{rest}} = -1$ *and the right hand side*

$$f(x,t) = \chi_{\{0\}} e^{-x^2/25},$$

corresponding to a narrow stimulus that is propagated on the entire domain at $t = T$*, i.e.* $u(x,T) \simeq u_{\max}$*. PFASST, similarly to standard time stepping techniques, gives the advantage to treat explicitly non-linear terms and thus avoiding the usage of a non-linear solver. In this example, for PFASST, we use an IM-EX strategy where we treat, as usual, the diffusion operator implicitly and the reaction one explicitly.*

Run-times are collected in Table 6.26 and illustrated in Figure 6.30. The two algorithms show the same scaling behavior but, because the space-time solver requires 35 Newton iterations to converge, it is approximately 35 times slower then PFASST.

#Cores	$MG^6[2,2]$	PFA^2
1	-	5.7
4	97	2.5
16	46	1.4
64	25	0.6
128	18	0.4
256	22	0.5

Table 6.26. Run-times from Example 6.23 .

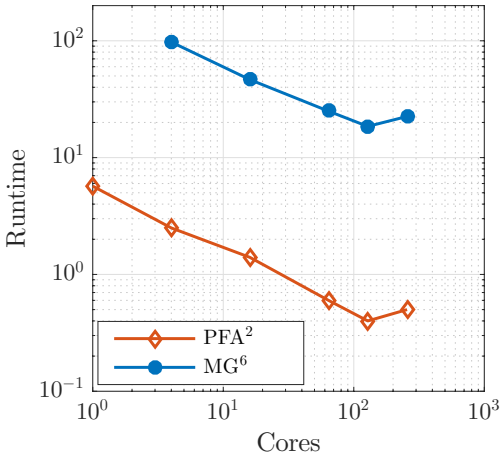

Figure 6.30. Run-times from Table 6.26.

The results of Examples 6.18–6.22 suggest that we can achieve similar strong and weak scaling with both approaches. Depending on the discretization settings it can be convenient to use one of the two approaches to minimize time-to-solution. For example, when a high order discretization in time is used, PFASST can be preferable. In fact, for low order methods, time coarsening within PFASST is limited. On the other hand, PFASST is more flexible when dealing with non-linear problems, as it allows explicit integration (or mixed IM-EX strategies) and might be preferable if an implicit treatment of the non-linear term is not necessary.

Chapter 7

Conclusions

In Chapter 2 we presented the space-time discretization of a parabolic problem using a tensor product finite element space between DG(q) basis in time and IgA(p,k) basis in space. Such approach can be considered generic has the DG discretization in time is equivalent to implicit time stepping (implicit Euler for $q=0$) and IgA boils down to *standard* finite element for $k=0$. In Section 2.1.3 we compared the properties of the two methods (cf. Figure 2.2) showing that the DG discretization is more suitable if block solvers are applied, w.r.t. a CG one (cf. Figure 2.3).

We then described the assembly and the parallel layout, of a large, not symmetric, sparse, space-time system ($C_{N,n}^{[q,p,k]}$), that can be distributed among multiple processors and solved in parallel, beyond the saturation of the parallelism in the space dimension (cf. Figure 6.5, and Examples 6.9 and 6.17).

This approach, in contrast to other parallel-in-time strategies, is mostly algebraic, in fact, it just requires to solve efficiently and in parallel a block linear system. For this purpose we analyzed and compared various strategies, combining multilevel techniques with PGMRES solvers.

In Chapter 3 we studied the spectral distribution of $C_{N,n}^{[q,p,k]}$, using the *symbol* theory, to construct a preconditioner that is spectrally equivalent to $C_{N,n}^{[q,p,k]}$, under some mild assumptions. Such preconditioner is symmetric and can be solved efficiently using a combination of a tensor solver and few multigrid iterations. In particular, simply skipping fine level smoothing can make the resulting MG-PGMRES robust in terms of the FE order p, considering that, for high p, standard multigrid fails (cf. Tables 6.1, 6.2 and 6.3).

The study of the spectral distribution of the space-time system $C_{N,n}^{[q,p,k]}$ can also explain how its conditioning depends on the underlying discretization, described by the parameter μ.

As pointed out in the specialized literature, the convergence of space-time multilevel methods is highly sensitive to μ. This dependency is usually explained using the Local Fourier Analysis (LFA) to describe the action of iteration matrices. We showed how this sensitivity is not restricted to space-time multigrid, but appears for all the iterative solvers that we tested (cf. Figures 3.8–3.9).

In particular, we always observe a sharp minimum in convergence rates w.r.t. μ that we motivated through the analysis of the interplay between the spectrum of spatial mass and stiffness matrices in $C_{N,n}^{[q,p,k]}$. For moderate p we were able to localize exactly such a minimum (cf. Theorem 3.18).

In Chapter 5 we introduced a *gray-box* space-time multigrid, that uses a semi-geometric approach to construct coarse spaces. The convergence and the parallel performance of such a solver is studied experimentally, through many numerical tests with not trivial settings, and theoretically, using the knowledge from the previous spectral analysis.

Regarding the space-time multigrid, we can conclude that:

- We observe how the semi-geometric multigrid convergence can degrade if the fine geometry is not well represented by a cuboid (cf. Table 6.9), and we comment how a bounding polygon might be more effective (cf. Remark 6.3).

- As previous literature suggests, a clever choice of space and time coarsening, depending on μ, is the key to obtain fast multigrid convergence. In practice, especially for 3D problems, solely space coarsening is still the most effective strategy in most of the cases; time coarsening can become essential for $\mu \ll 1$ (i.e. $K\Delta t \ll h^2$) and large N. We must consider that μ is not always the key parameter when it comes to coarsening. For example, if $n \gg N$, space coarsening is still fundamental for computational efficiency, whatever the value of μ is (cf. Remarks 6.4 and 6.16).

- Time coarsening becomes essential to ensure a nearly optimal weak scaling in time. In fact, without time coarsening, coarse problems become more and more expensive as the number of time steps N grows (cf. Example 6.11 and Remark 6.12).

- The space-time setting is convenient in terms of scalability, w.r.t. a traditional forward solve, if enough cores are invested. In the tests we presented time-parallelism is advantageous if #Cores ≥ 64 (cf. Figures 6.5, 6.20 and Remark 6.1).

Regarding the DG order q, using the space-time approach might be preferable for moderate q, since the size of the space-time matrix $C_{N,n}^{[q,p,k]}$ is proportional to $q+1$; for large q the size of $C_{N,n}^{[q,p,k]}$ may become prohibitive. Moreover, in the presented linear solvers, robustness w.r.t. q was not fully investigated and we can assume that it would be poor.

Finally we consider a non-linear diffusion-reaction problem from electrophysiology, for which an additional non-linear iterative process (an inexact Newton's method) has to be employed. Such problem shows some limitations of the presented space-time approach, especially w.r.t. to time discretization parameters (cf. Table 6.15 and 6.16). In particular it highlights that our space-time approach is, by construction, *fully implicit* so the flexibility to treat explicitly the reaction term, and avoid the Newton solver, is lost. We showed how such difficulties can be partially overcame using an hybrid strategy and renouncing to some time parallelism (cf. Example 6.17). However, if more elaborate models are used, with dozens of auxiliary variables, the space-time approach might be unfeasible, due too its high memory footprint. In this regard, a matrix free approach would be interesting to investigate.

We mention that, in some applications, such as uncertainty quantification, an initial guess for the Newton solver that is very close to the solution may be available, resulting in the immediate convergence of the Newton iteration [Bader et al., 2019].

Finally, for the first time in the literature, in Section 6.5 we compare experimentally PFASST and space-time multigrid and we observe the same time scaling (cf. Figures 6.28 and 6.29). Let us keep in mind that PFASST is ideal for high order discretization in time while the space-time approach is preferable for low order ones. For the monodomain problem, though, PFASST allows to treat the reaction term explicitly, resulting in a faster run-time (cf. Figure 6.30). On the other hand, in practice, a highly accurate solution of the monodomain is usually not necessary and low order methods are commonly preferred.

Appendix A

Anisotropic diffusion description

In Examples 6.5 and 6.6 the anisotropic diffusion $K_{\text{an}}(\mathbf{x})$ was mentioned without any further specifications. Let us described how these matrices are constructed.

As in [Harbrecht and Schmidlin, 2020], given a $\mathbb{R}^d$ valued vector field $V(\mathbf{x})$ and $a \in \mathbb{R}$ we have

$$K_{\text{an}}(\mathbf{x}) = aI_d + (\|V(\mathbf{x})\| - a)\frac{V(\mathbf{x})V(\mathbf{x})^T}{\|V(\mathbf{x})\|^2}. \tag{A.1}$$

With this form, $K_{\text{an}}(\mathbf{x})$ is designed to diffuse in the direction of $V(\mathbf{x})$ with intensity $\|V(\mathbf{x})\|$ and in the perpendicular direction with intensity a. In both the examples we used $a = 0.1$ and a random (spatially correlated) $V(\mathbf{x})$. In particular, $V(\mathbf{x}) = \mathbf{1} + \gamma(\mathbf{x})$ where γ is determined trough a truncated Karhunen–Loève expansion with correlation kernel $C(\mathbf{x}_1, \mathbf{x}_2)$ given by

$$C(\mathbf{x}_1, \mathbf{x}_2) = \begin{bmatrix} \frac{1}{4}e^{-r^2/2} & 0 & \frac{1}{10}e^{-r^2/10} \\ 0 & \frac{1}{4}e^{-r^2/2} & 0 \\ \frac{1}{10}e^{-r^2/2} & 0 & (1+\sqrt{3}r)e^{-\sqrt{3}r} \end{bmatrix}, \tag{A.2}$$

with $r = \|\mathbf{x}_1 - \mathbf{x}_2\|$.

Appendix B

Multi-fidelity approach: the Eikonal model

As pointed out in Section 4.1.2, the convergence behavior of the Newton solver is strongly influenced by its initial guess, that must be chosen as close as possible to the solution. We can use a simpler (and computationally cheaper) model to get an rough estimate of the solution of (4.1) to be used as initial guess for the non-linear iteration.

For this reason we introduce the eikonal model: a time independent, non-linear model describing the activation times $T_{\text{act}}(\mathbf{x})$ on a domain Ω, given a propagation velocity $v \in \mathbb{R}$ and a region $\Gamma \subset \Omega$ where an impulse is generated initially:

$$\begin{cases} v\sqrt{(\nabla T_{\text{act}})^T K(\mathbf{x}) \nabla T_{\text{act}}} + \varepsilon \Delta T_{\text{act}} = 1 & \text{in} \quad \Omega, \\ T_{\text{act}} = 0 & \text{in} \quad \Gamma, \end{cases} \tag{B.1}$$

where $\varepsilon > 0$ a small stabilization parameter. The eikonal equation is computationally cheaper to solve w.r.t. the monodomain equation but the fine details of the wave front are lost. Once the eikonal solution $T_{\text{act}}(\mathbf{x})$ is computed we can define its space-time counterpart $u^0(\mathbf{x}, t)$ as follows:

$$\begin{cases} u^0(\mathbf{x}, t) = u_{\max} & \text{if} \quad t \geq T_{\text{act}}(\mathbf{x}), \\ u^0(\mathbf{x}, t) = u_{\text{rest}} & \text{if} \quad t < T_{\text{act}}(\mathbf{x}), \end{cases} \tag{B.2}$$

to be used as initial guess. This methodology has been employed effectively, as illustrated in Figure B.1, as the number of Newton iterations to convergence is reduced w.r.t. constant initial guesses. On the other hand, once the monodomain setting is fixed, v and Γ are not precisely known a priori and reverse engineering might be necessary to find them. In fact, the relation between monodomain and eikonal parameters is not trivial; once the former change the latter need to be

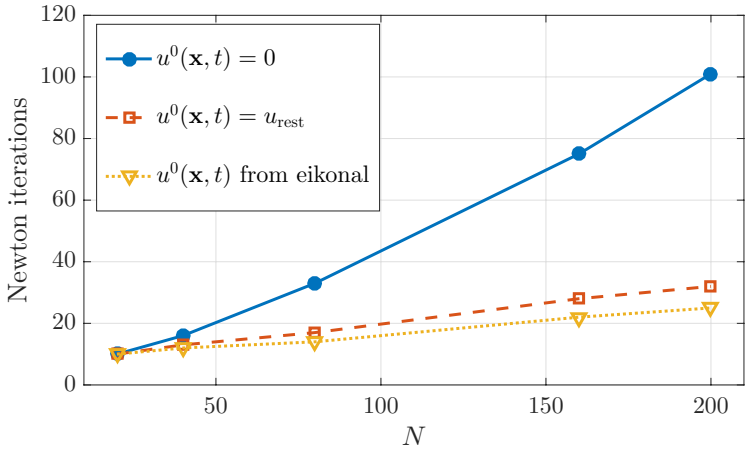

Figure B.1. Newton iterations to solve problem (4.1) with the same setting as in Example 6.15 with various initial guesses. The parameter ν and Γ in (B.1) have been chosen to reduce as much as possible the correspondent number of iterations. While the zero initial guess is significantly worse for convergence, using u_{rest} is acceptable.

recomputed accordingly. Because this process is hard to automatize we used, in the numerical experiments, the initial guess

$$u^0(\mathbf{x},t) \equiv u_{\text{rest}},$$

that still produces reasonable convergence.

Notation summary

We summarize the meaning of the most significant symbols used trough this thesis:

$T \in \mathbb{R}$: final time

$I_m = [t_m, t_{m+1}]$: time interval (or identity matrix, depending on the context)

$\Omega \in \mathbb{R}^d$: spatial domain

$n \in \mathbb{N}$: number of space elements

$\overline{n} \in \mathbb{N}$: total number of degrees of freedom in space

$N \in \mathbb{N}$: number of time steps

$\overline{N} = N(q+1) \in \mathbb{N}$: total number of degrees of freedom in time

$p \in \mathbb{N}$: polynomial order of the space basis

$k \in \mathbb{N}$: polynomial regularity of the space basis (C^k)

$q \in \mathbb{N}$: polynomial order of the time basis

$\mathscr{E}^m = [t_m, t_{m+1}] \times \Omega$, mth space-time slab

$K(\mathbf{x}) \in \mathbb{R}^{d\times d}$ or $\mathbb{R}$: diffusion coefficients

$I_n \in \mathbb{R}^{n\times n}$: identity matrix of size n

$L_n \in \mathbb{R}^{n\times n}$: lower shift identity matrix of size n

$\mu \in \mathbb{R}$: CFL discretization parameter, cf. equation (3.23)

$K_{n,[p,k]}, M_{n,[p,k]} \in \mathbb{R}^{\overline{n}\times\overline{n}}$: stiffness and mass matrices in space

$K_{[q]}, M_{[q]}, J_{[q]} \in \mathbb{R}^{(q+1)\times(q+1)}$: stiffness, mass and coupling matrices in time

$A_n^{[q,p,k]}, B_n^{[q,p,k]} \in \mathbb{R}^{\overline{n}(q+1)\times\overline{n}(q+1)}$: single time slab operators.

$C_{N,n}^{[q,p,k]} \in \mathbb{R}^{\overline{N}\overline{n}\times\overline{N}\overline{n}}$: space-time system

$P_{N,n}^{[q,p,k]} \in \mathbb{R}^{\overline{N}\overline{n}\times\overline{N}\overline{n}}$: space-time preconditioner

$\mathbf{f}_{[q,p,k]}$: space-time symbol in (3.10), with κ its rearranged version

$\chi, C_m, C_w, a, b, c, u_{\text{rest}}, u_{\text{thres}}, u_{\max} \in \mathbb{R}$: monodomain equation (4.1) parameters

$C_x, C_t \geq 1$: space and time coarsening factors

$L, \eta, \nu \in \mathbb{N}$: multigrid levels, iterations and smoothing steps respectively

$\mathbf{F}(\cdot) : \mathbb{R}^{\overline{N}\overline{n}} \to \mathbb{R}^{\overline{N}\overline{n}}$: non-linear residual

$M \in \mathbb{N}$: number of time blocks

Bibliography

Abedi, R., Petracovici, B. and Haber, R. [2006]. A space–time discontinuous Galerkin method for linearized elastodynamics with element-wise momentum balance, *Computer Methods in Applied Mechanics and Engineering* **195**(25): 3247 – 3273.

Abramowitz, M. and Stegun, I. A. (eds) [1992]. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, Dover Publications, Inc., New York. Reprint of the 1972 edition.

Adelaide, H., Bohatier, C. and Jourdan, F. [2002]. New results on mesh adaptation in space-time finite element method, *ASME 2002 Engineering Technology Conference on Energy*, American Society of Mechanical Engineers, pp. 963–972.

Adjerid, S., Devine, K. D., Flaherty, J. E. and Krivodonova, L. [2002]. A posteriori error estimation for discontinuous Galerkin solutions of hyperbolic problems, *Comput. Methods Appl. Mech. Engrg.* **191**(11-12): 1097–1112.
URL: *http://dx.doi.org/10.1016/S0045-7825(01)00318-8*

Andreev, R. [2012]. *Stability of space-time Petrov-Galerkin discretizations for parabolic evolution equations*, PhD thesis, ETH Zurich.

Aziz, A. K. and Monk, P. [1989]. Continuous finite elements in space and time for the heat equation, *Mathematics of Computation* **52**(186): 255–274.

Bader, S. B., Benedusi, P., Quaglino, A., Zulian, P. and Krause, R. [2019]. Space-time multilevel monte carlo methods and their application to cardiac electrophysiology, *arXiv preprint arXiv:1911.06066* .

Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W. D., Karpeyev, D., Kaushik, D., Knepley, M. G., May, D. A., McInnes, L. C., Mills, R. T., Munson, T., Rupp, K., Sanan, P., Smith, B. F., Zampini, S., Zhang, H. and Zhang, H. [2019a]. PETSc

users manual, *Technical Report ANL-95/11 - Revision 3.12*, Argonne National Laboratory.
URL: *https://www.mcs.anl.gov/petsc*

Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W. D., Karpeyev, D., Kaushik, D., Knepley, M. G., May, D. A., McInnes, L. C., Mills, R. T., Munson, T., Rupp, K., Sanan, P., Smith, B. F., Zampini, S., Zhang, H. and Zhang, H. [2019b]. PETSc Web page, `https://www.mcs.anl.gov/petsc`.
URL: *https://www.mcs.anl.gov/petsc*

Barbarino, G. [2017]. Diagonal matrix sequences and their spectral symbols, *arXiv preprint arXiv:1710.00810* .

Barbarino, G., Garoni, C. and Serra-Capizzano, S. [2020a]. Block generalized locally Toeplitz sequences: theory and applications in the multidimensional case, *Electronic Transactions on Numerical Analysis* **53**: 113–216.

Barbarino, G., Garoni, C. and Serra-Capizzano, S. [2020b]. Block generalized locally Toeplitz sequences: theory and applications in the unidimensional case, *Electronic Transactions on Numerical Analysis* **53**: 28–112.

Beckermann, B. and Kuijlaars, A. B. [2001]. Superlinear convergence of conjugate gradients, *SIAM Journal on Numerical Analysis* **39**(1): 300–329.

Benedusi, P., Garoni, C., Krause, R., Li, X. and Serra-Capizzano, S. [2018a]. Discontinuous Galerkin discretization of the heat equation in any dimension: the spectral symbol. Technical report 2018-002, Dept. of Information Technology, Uppsala University.
URL: *http://www.it.uu.se/research/publications/reports/2018-002/2018-002-nc.pdf*

Benedusi, P., Garoni, C., Krause, R., Li, X. and Serra-Capizzano, S. [2018b]. Space-time fe-dg discretization of the anisotropic diffusion equation in any dimension: The spectral symbol, *SIAM Journal on Matrix Analysis and Applications* **39**(3): 1383–1420.

Benedusi, P., Hupp, D., Arbenz, P. and Krause, R. [2016]. A parallel multigrid solver for time-periodic incompressible navier–stokes equations in 3d, *Numerical Mathematics and Advanced Applications ENUMATH 2015*, Springer, pp. 265–273.

Besier, M. and Rannacher, R. [2012]. Goal-oriented space–time adaptivity in the finite element Galerkin method for the computation of nonstationary incompressible flow, *International Journal for Numerical Methods in Fluids* **70**(9): 1139–1166.

Betsch, P. and Steinmann, P. [2000]. Conservation properties of a time fe method. part i: time-stepping schemes for n-body problems, *International Journal for Numerical Methods in Engineering* **49**(5): 599–638.

Betsch, P. and Steinmann, P. [2001]. Conservation properties of a time fe method-part ii: Time-stepping schemes for non-linear elastodynamics, *International Journal for Numerical Methods in Engineering* **50**(8): 1931–1955.

Bolten, M., Moser, D. and Speck, R. [2017]. A multigrid perspective on the parallel full approximation scheme in space and time, *Numerical Linear Algebra with Applications* **24**(6): e2110.

Briggs, W. L., Henson, V. E. and McCormick, S. F. [2000]. *A Multigrid Tutorial*, 2nd edn, SIAM, Philadelphia.

Burrage, K. [1997]. Parallel methods for ODEs, *Advances in Computational Mathematics* **7**: 1–3.
URL: *http://dx.doi.org/10.1023/A:1018997130884*

Chan, T. F. and Wan, W. [2000]. Robust multigrid methods for nonsmooth coefficient elliptic linear systems, *Journal of computational and applied mathematics* **123**(1-2): 323–352.

Cherry, E. M., Greenside, H. S. and Henriquez, C. S. [2000]. A space-time adaptive method for simulating complex cardiac dynamics, *Physical Review Letters* **84**(6): 1343.

Colli Franzone, P., Deuflhard, P., Erdmann, B., Lang, J. and Pavarino, L. [2006]. Adaptivity in space and time for reaction-diffusion systems in electrocardiology, *SIAM Journal on Scientific Computing* **28**(3): 942–962.

Colli Franzone, P. and Pavarino, L. F. [2004]. A parallel solver for reaction-diffusion systems in computational electrocardiology, *Mathematical models and methods in applied sciences* **14**(06): 883–911.

Dalcin, L., Collier, N., Vignal, P., Cortes, A. and Calo, V. [2016]. Petiga: A framework for high-performance isogeometric analysis, *Computer Methods in Applied Mechanics and Engineering* **308**: 151–181.

Delfour, M., Hager, W. and Trochu, F. [1981]. Discontinuous Galerkin methods for ordinary differential equations, *Mathematics of Computation* **36**(154): 455–473.

Deuflhard, P., Erdmann, B., Roitzsch, R. and Lines, G. T. [2009]. Adaptive finite element simulation of ventricular fibrillation dynamics, *Computing and visualization in science* **12**(5): 201–205.

Dickopf, T. and Krause, R. [2013]. Numerical study of the almost nested case in a multilevel method based on non-nested meshes, *Domain Decomposition Methods in Science and Engineering XX*, Springer, pp. 551–558.

Donatelli, M., Garoni, C., Manni, C., Serra-Capizzano, S. and Speleers, H. [2015a]. Robust and optimal multi-iterative techniques for IgA collocation linear systems, *Computer Methods in Applied Mechanics and Engineering* **284**: 1120–1146.

Donatelli, M., Garoni, C., Manni, C., Serra-Capizzano, S. and Speleers, H. [2015b]. Robust and optimal multi-iterative techniques for IgA Galerkin linear systems, *Computer Methods in Applied Mechanics and Engineering* **284**: 230–264.

Donatelli, M., Garoni, C., Manni, C., Serra-Capizzano, S. and Speleers, H. [2016]. Spectral analysis and spectral symbol of matrices in isogeometric collocation methods, *Mathematics of Computation* **85**(300): 1639–1680.

Donatelli, M., Garoni, C., Manni, C., Serra-Capizzano, S. and Speleers, H. [2017]. Symbol-based multigrid methods for Galerkin b-spline isogeometric analysis, *SIAM Journal on Numerical Analysis* **55**(1): 31–62.

Douglas, C. C. [1996]. A review of numerous parallel multigrid methods, *Applications on Advanced Architecture Computers*, SIAM, pp. 187–202.

Dutt, A., L., G. and V., R. [2000]. Spectral deferred correction methods for ordinary differential equations, *BIT Numerical Mathematics* **40**(2): 241–266.

Eisenstat, S. C. and Walker, H. F. [1996]. Choosing the forcing terms in an inexact newton method, *SIAM Journal on Scientific Computing* **17**(1): 16–32.

Ekström, S.-E., Furci, I., Garoni, C., Manni, C., Serra-Capizzano, S. and Speleers, H. [2018]. Are the eigenvalues of the b-spline isogeometric analysis approximation of $-\Delta u = \lambda u$ known in almost closed form?, *Numerical Linear Algebra with Applications* **25**(5): e2198.

Emmett, M. and Minion, M. L. [2012]. Toward an efficient parallel in time method for partial differential equations, *Communications in Applied Mathematics and Computational Science* **7**: 105–132.
URL: *http://dx.doi.org/10.2140/camcos.2012.7.105*

Eriksson, K., Estep, D., Hansbo, P. and Johnson, C. [1996]. *Computational differential equations*, Vol. 1, Cambridge University Press.

Eriksson, K. and Johnson, C. [1991]. Adaptive finite element methods for parabolic problems i: A linear model problem, *SIAM Journal on Numerical Analysis* **28**(1): 43–77.

Eriksson, K. and Johnson, C. [1995]. Adaptive finite element methods for parabolic problems ii: Optimal error estimates in $L_\infty L_2$ and $L_\infty L_\infty$, *SIAM Journal on Numerical Analysis* **32**(3): 706–740.

Eriksson, K., Johnson, C. and Thomée, V. [1985]. Time discretization of parabolic problems by the discontinuous Galerkin method, *ESAIM: Mathematical Modelling and Numerical Analysis* **19**(4): 611–643.

Estep, D. and Larsson, S. [1993]. The discontinuous Galerkin method for semilinear parabolic problems, *ESAIM: Mathematical Modelling and Numerical Analysis* **27**(1): 35–54.

Falgout, R. D., Friedhoff, S., Kolev, T. V., MacLachlan, S. P. and Schroder, J. B. [2014]. Parallel time integration with multigrid, *SIAM Journal on Scientific Computing* **36**: C635–C661.
URL: *http://dx.doi.org/10.1137/130944230*

Falgout, R. D., Friedhoff, S., Kolev, T. V., MacLachlan, S. P., Schroder, J. B. and Vandewalle, S. [2017]. Multigrid methods with space–time concurrency, *Computing and Visualization in Science* **18**(4-5): 123–143.

Feistauer, M., Kučera, V., Najzar, K. and Prokopová, J. [2011]. Analysis of space–time discontinuous Galerkin method for nonlinear convection–diffusion problems, *Numerische Mathematik* **117**(2): 251–288.

Franco, S. R., Gaspar, F. J., Pinto, M. A. V. and Rodrigo, C. [2018]. Multigrid method based on a space-time approach with standard coarsening for parabolic problems, *Applied Mathematics and Computation* **317**: 25–34.

Gahalaut, K. P., Kraus, J. K. and Tomar, S. K. [2013]. Multigrid methods for isogeometric discretization, *Computer methods in applied mechanics and engineering* **253**: 413–425.

Gander, M. J. [2015]. 50 years of time parallel time integration, *Multiple Shooting and Time Domain Decomposition*, Springer.
URL: *http://dx.doi.org/10.1007/978-3-319-23321-5*$_3$

Gander, M. J. and Güttel, S. [2013]. PARAEXP: A parallel integrator for linear initial-value problems, *SIAM Journal on Scientific Computing* **35**(2): C123–C142.
URL: *http://dx.doi.org/10.1137/110856137*

Gander, M. J. and Halpern, L. [2013]. Techniques for locally adaptive time stepping developed over the last two decades, *Domain decomposition methods in science and engineering XX*, Springer, pp. 377–385.

Gander, M. J., Kwok, F. and Zhang, H. [2018a]. Multigrid interpretations of the parareal algorithm leading to an overlapping variant and mgrit, *Computing and Visualization in Science* .
URL: *https://doi.org/10.1007/s00791-018-0297-y*

Gander, M. J., Kwok, F. and Zhang, H. [2018b]. Multigrid interpretations of the parareal algorithm leading to an overlapping variant and mgrit, *Computing and Visualization in Science* **19**(3-4): 59–74.

Gander, M. J. and Neumüller, M. [2016]. Analysis of a new space-time parallel multigrid algorithm for parabolic problems, *SIAM Journal on Scientific Computing* **38**(4): A2173–A2208.

Garoni, C. [2018]. Spectral distribution of pde discretization matrices from isogeometric analysis: the case of l1 coefficients and non-regular geometry, *J. Spectral Theory* **8**: 297–313.

Garoni, C., Manni, C., Pelosi, F., Serra-Capizzano, S. and Speleers, H. [2014]. On the spectrum of stiffness matrices arising from isogeometric analysis, *Numerische Mathematik* **127**(4): 751–799.

Garoni, C. and Serra-Capizzano, S. [2017]. *Generalized Locally Toeplitz Sequences: Theory and Applications, Vol I*, Springer.

Garoni, C. and Serra-Capizzano, S. [2018]. *Generalized Locally Toeplitz Sequences: Theory and Applications, Vol II*, Springer.

Garoni, C., Serra-Capizzano, S. and Sesana, D. [2015a]. Spectral analysis and spectral symbol of d-variate $\mathbb{Q}_p$ Lagrangian fem stiffness matrices, *SIAM Journal on Matrix Analysis and Applications* **36**(3): 1100–1128.

Garoni, C., Serra-Capizzano, S. and Sesana, D. [2015b]. Tools for determining the asymptotic spectral distribution of non-Hermitian perturbations of Hermitian matrix-sequences and applications, *Integral equations and operator theory* **81**(2): 213–225.

Garoni, C., Speleers, H., Ekström, S.-E., Reali, A., Serra-Capizzano, S. and Hughes, T. J. [2019]. Symbol-based analysis of finite element and isogeometric b-spline discretizations of eigenvalue problems: exposition and review, *Archives of Computational Methods in Engineering* **26**(5): 1639–1690.

Hackbusch, W. [1984]. Parabolic multi-grid methods, *Computing Methods in Applied Sciences and Engineering, VI* pp. 189–197.
URL: *http://dl.acm.org/citation.cfm?id=4673.4714*

Hackbusch, W. [2013]. *Multi-grid methods and applications*, Vol. 4, Springer Science & Business Media.

Hansen, S., Henning, A., Naegel, A., Heisig, M., Wittum, G., Neumann, D., Kostka, K.-H., Zbytovska, J., Lehr, C.-M. and Schaefer, U. F. [2008]. In-silico model of skin penetration based on experimentally determined input parameters. part i: Experimental determination of partition and diffusion coefficients, *European Journal of Pharmaceutics and Biopharmaceutics* **68**(2): 352–367.

Harbrecht, H. and Schmidlin, M. [2020]. Multilevel methods for uncertainty quantification of elliptic pdes with random anisotropic diffusion, *Stochastics and Partial Differential Equations: Analysis and Computations* **8**(1): 54–81.

Henson, V. E. [2002]. Multigrid methods for nonlinear problems: an overview, *Technical report*, Lawrence Livermore National Lab., CA (US).

Hesch, C. and Betsch, P. [2006]. A comparison of computational methods for large deformation contact problems of flexible bodies, *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* **86**(10): 818–827.

Hofer, C., Langer, U., Neumüller, M. and Schneckenleitner, R. [2019]. Parallel and robust preconditioning for space-time isogeometric analysis of parabolic evolution problems, *SIAM Journal on Scientific Computing* **41**(3): A1793–A1821.

Horton, G. and Knirsch, R. [1992]. A time-parallel multigrid-extrapolation method for parabolic partial differential equations, *Parallel Computing* **18**(1): 21–29.
URL: *http://dx.doi.org/10.1016/0167-8191(92)90108-J*

Horton, G. and Vandewalle, S. [1995]. A Space-Time Multigrid Method for Parabolic Partial Differential Equations, *SIAM Journal on Scientific Computing* **16**(4): 848–864.
URL: *http://dx.doi.org/10.1137/0916050*

Hulbert, G. M. and Hughes, T. J. [1990]. Space-time finite element methods for second-order hyperbolic equations, *Computer methods in applied mechanics and engineering* **84**(3): 327–348.

Hulme, B. L. [1972a]. Discrete Galerkin and related one-step methods for ordinary differential equations, *Mathematics of Computation* **26**(120): 881–891.

Hulme, B. L. [1972b]. One-step piecewise polynomial Galerkin methods for initial value problems, *Mathematics of Computation* **26**(118): 415–426.

Hussain, S., Schieweck, F. and Turek, S. [2011]. Higher order Galerkin time discretizations and fast multigrid solvers for the heat equation, *Journal of Numerical Mathematics* **19**(1): 41–61.

J. Nivergelt [1964]. Parallel methods for integrating ordinary differential equations, *Comm. ACM* **4**: 731–733.

Jamet, P. [1978]. Galerkin-type approximations which are discontinuous in time for parabolic equations in a variable domain, *SIAM Journal on Numerical Analysis* **15**(5): 912–928.

Jepps, O. G., Dancik, Y., Anissimov, Y. G. and Roberts, M. S. [2013]. Modeling the human skin barrier–towards a better understanding of dermal absorption, *Advanced drug delivery reviews* **65**(2): 152–168.

Keener, J. P. and Sneyd, J. [1998]. *Mathematical physiology*, Vol. 1, Springer.

Kirk, B. S., Peterson, J. W., Stogner, R. H. and Carey, G. F. [2006]. `libMesh`: A C++ library for parallel adaptive mesh refinement/coarsening simulations, *Engineering with Computers* **22**(3–4): 237–254. `http://dx.doi.org/10.1007/s00366-006-0049-3`.

Klaij, C., van der Vegt, J. and van der Ven, H. [2006]. Space-time discontinuous Galerkin method for the compressible Navier-Stokes equations, *Journal of Computational Physics* **217**(2): 589 – 611.

Krause, D. [2013]. *Scalable space-time adaptive simulation tools for computational electrocardiology*, PhD thesis, Università della Svizzera italiana.

Krause, D. and Krause, R. [2016]. Enabling local time stepping in the parallel implicit solution of reaction-diffusion equations via space-time finite elements on shallow tree meshes, *Applied Mathematics and Computation* **277**: 164–179.

Krause, R. and Zulian, P. [2016]. A parallel approach to the variational transfer of discrete fields between arbitrarily distributed unstructured finite element meshes, *SIAM Journal on Scientific Computing* **38**(3): C307–C333.

Kreienbuehl, A., Naegel, A., Ruprecht, D., Speck, R., Wittum, G. and Krause, R. [2015a]. Numerical simulation of skin transport using parareal, *Computing and visualization in science* **17**(2): 99–108.

Kreienbuehl, A., Naegel, A., Ruprecht, D., Speck, R., Wittum, G. and Krause, R. [2015b]. Numerical simulation of skin transport using Parareal, *Computing and Visualization in Science* **17**: 99–108.

Ladyzhenskaia, O. A., Solonnikov, V. A. and Ural'ceva, N. N. [1968]. *Linear and quasi-linear equations of parabolic type*, Vol. 23, American Mathematical Soc.

Langer, U., Matculevich, S. and Repin, S. [2016]. A posteriori error estimates for space-time IgA approximations to parabolic initial boundary value problems, *arXiv preprint arXiv:1612.08998* .

Langer, U., Moore, S. E. and Neumüller, M. [2016]. Space–time isogeometric analysis of parabolic evolution problems, *Computer methods in applied mechanics and engineering* **306**: 342–363.

Lasaint, P. and Raviart, P. [1974]. On a finite element method for solving the neutron transport equation, *Mathematical Aspects of Finite Elements in Partial Differential Equations, Proceedings of a Symposium Conducted by the Mathematics Research Center, the University of Wisconsin-Madison, Madison, WI, USA*, pp. 1–3.

Li, X. and Wiberg, N.-E. [1998]. Implementation and adaptivity of a space-time finite element method for structural dynamics, *Computer Methods in Applied Mechanics and Engineering* **156**(1-4): 211–229.

Lions, J.-L., Maday, Y. and Turinici, G. [2001]. A "parareal" in time discretization of PDE's, *Comptes Rendus de l'Acadèmie des Sciences – Series I – Mathematics* **332**: 661–668.
URL: *http://dx.doi.org/10.1016/S0764-4442(00)01793-6*

Makridakis, C. G. and Babuska, I. [1997]. On the stability of the discontinuous Galerkin method for the heat equation, *SIAM journal on numerical analysis* **34**(1): 389–401.

May, D. A., Sanan, P., Rupp, K., Knepley, M. G. and Smith, B. F. [2016]. Extreme-scale multigrid components within petsc, *Proceedings of the Platform for Advanced Scientific Computing Conference*, pp. 1–12.

Mazza, M., Ratnani, A. and Serra-Capizzano, S. [2019]. Spectral analysis and spectral symbol for the 2d curl-curl (stabilized) operator with applications to the related iterative solutions, *Mathematics of Computation* **88**(317): 1155–1188.

McDonald, E. [2016]. *All-at-once solution of time-dependent PDE problems*, PhD thesis, University of Oxford.

McDonald, E. and Wathen, A. [2016]. A simple proposal for parallel computation over time of an evolutionary process with implicit time stepping, *Numerical Mathematics and Advanced Applications ENUMATH 2015*, Springer, pp. 285–293.

Miller, S. T. and Haber, R. B. [2008]. A spacetime discontinuous Galerkin method for hyperbolic heat conduction, *Computer Methods in Applied Mechanics and Engineering* **198**(2): 194–209.

Naegel, A., Heisig, M. and Wittum, G. [2013]. Detailed modeling of skin penetration—an overview, *Advanced drug delivery reviews* **65**(2): 191–207.

Neumüller, M. [2013]. Space-time methods: fast solvers and applications.

Pezzuto, S., Hake, J. and Sundnes, J. [2016]. Space-discretization error analysis and stabilization schemes for conduction velocity in cardiac electrophysiology, *International journal for numerical methods in biomedical engineering* **32**(10): e02762.

Popp, A., Wohlmuth, B. I., Gee, M. W. and Wall, W. A. [2012]. Dual quadratic mortar finite element methods for 3d finite deformation contact, *SIAM Journal on Scientific Computing* **34**(4): B421–B446.

Potse, M., Dubé, B., Richer, J., Vinet, A. and Gulrajani, R. M. [2006]. A comparison of monodomain and bidomain reaction-diffusion models for action potential propagation in the human heart, *IEEE Transactions on Biomedical Engineering* **53**(12): 2425–2435.

Quarteroni, A., Sacco, R. and Saleri, F. [2010]. *Matematica numerica*, Springer Science & Business Media.

Querleux, B. [2016]. *Computational Biophysics of the Skin*, Jenny Stanford Publishing.

Richter, T., Springer, A. and Vexler, B. [2013]. Efficient numerical realization of discontinuous Galerkin methods for temporal discretization of parabolic problems, *Numerische Mathematik* **124**(1): 151–182.

Saad, Y. [2003]. *Iterative methods for sparse linear systems*, Vol. 82, siam.

Sabawi, M. A. M. [2018]. *Discontinuous Galerkin timestepping for nonlinear parabolic problems*, PhD thesis, Department of Mathematics.

Sachetto Oliveira, R., Martins Rocha, B., Burgarelli, D., Meira Jr, W., Constantinides, C. and Weber dos Santos, R. [2018]. Performance evaluation of gpu parallelization, space-time adaptive algorithms, and their combination for simulating cardiac electrophysiology, *International journal for numerical methods in biomedical engineering* **34**(2): e2913.

Schieweck, F. [2010]. A-stable discontinuous Galerkin–petrov time discretization of higher order, *Journal of Numerical Mathematics* **18**(1): 25–57.

Schmich, M. and Vexler, B. [2008]. Adaptivity with dynamic meshes for space-time finite element discretizations of parabolic equations, *SIAM Journal on Scientific Computing* **30**(1): 369–393.

Schötzau, D. and Wihler, T. P. [2010]. A posteriori error estimation for hp-version time-stepping methods for parabolic partial differential equations, *Numerische Mathematik* **115**(3): 475–509.

Schumaker, L. [2007]. *Spline functions: basic theory*, Cambridge University Press.

Shakib, F. and Hughes, T. J. [1991]. A new finite element formulation for computational fluid dynamics: Ix. fourier analysis of space-time Galerkin/least-squares algorithms, *Computer Methods in Applied Mechanics and Engineering* **87**(1): 35–58.

Shu, C.-W. [2014]. *Discontinuous Galerkin Method for Time-Dependent Problems: Survey and Recent Developments*, Springer International Publishing, pp. 25–62.

Smears, I. [2016]. Robust and efficient preconditioners for the discontinuous Galerkin time-stepping method, *IMA Journal of Numerical Analysis* **37**(4): 1961–1985.

Sommeijer, B. [1993]. Parallel-iterated Runge-Kutta methods for stiff ordinary differential equations, *Journal of Computational and Applied Mathematics* **45**(1): 151–168.
URL: *http://dx.doi.org/10.1016/0377-0427(93)90271-C*

Speck, R. and Ruprecht, D. [2016]. Toward fault-tolerant parallel-in-time integration with PFASST, *Parallel Computing* .
URL: *http://dx.doi.org/10.1016/j.parco.2016.12.001*

Steinbach, O. [2015]. Space-time finite element methods for parabolic problems, *Computational Methods in Applied Mathematics* **15**(4): 551–566.

Sudirham, J. J. J. [2005]. Space-time discontinuous Galerkin methods for convection-diffusion problems-application to wet-chemical etching.

Sudirham, J., van der Vegt, J. and van Damme, R. [2006]. Space-time discontinuous Galerkin method for advection-diffusion problems on time-dependent domains, *Applied Numerical Mathematics* **56**(12): 1491 – 1518.

Tezduyar, T. E., Sathe, S., Keedy, R. and Stein, K. [2006]. Space–time finite element techniques for computation of fluid-structure interactions, *Computer methods in applied mechanics and engineering* **195**(17-18): 2002–2027.

Thomée, V. [1984]. *Galerkin finite element methods for parabolic problems*, Vol. 1054, Springer.

Wohlmuth, B. I. [2000]. A mortar finite element method using dual spaces for the lagrange multiplier, *SIAM journal on numerical analysis* **38**(3): 989–1012.

Zhao, S. and Wei, G.-W. [2014]. A unified discontinuous Galerkin framework for time integration, *Mathematical methods in the applied sciences* **37**(7): 1042–1071.

Zulian, P., Kopaničáková, A., Nestola, M. C. G., Fink, A., Fadel, N., Rigazzi, A., Magri, V., Schneider, T., Botter, E., Mankau, J. and Krause, R. [2016]. Utopia: A

C++ embedded domain specific language for scientific computing. Git repository, https://bitbucket.org/zulianp/utopia.
URL: *https://bitbucket.org/zulianp/utopia*