NUMERICAL METHODS FOR BOUNDARY VALUE PROBLEMS WITH APPLICATIONS TO BIOENGINEERING

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF
MATERIAL SCIENCE AND ENGINEERING
OF THE UNIVERSITY OF IOANNINA
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Drosos Kourounis
November 2008

Αρ. εισ.:
1009/30-3-2009

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

____

(Antonios Charalambopoulos) Principal Advisor

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

____

(Dimitrios I. Fotiadis) Principal Co-Advisor

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

____

(Daniel Loghin) Principal Co-Advisor

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

____

(Klaus Gärtner)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

____

(Vasilios Kalpakidis)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Petros Maragos)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Michael Saunders)

Approved for the University Committee on Graduate Studies.

Abstract

Bioengineering is a broad-based engineering discipline that applies engineering principles and design to challenges in human health and medicine. It includes research and analysis of the mechanics of living organisms and the application of engineering principles to biological systems. This research and analysis can be carried out on multiple levels, from the molecular, wherein biomaterials such as collagen and elastin are considered, all the way up to the tissue and organ level. Applications that benefit from bioengineering include medical devices, diagnostic equipment, biocompatible materials, and many others.

Precise modeling of biomechanical phenomena is based on principles and laws of continuum mechanics. There is a vast literature on the subject, including the books by Fung [62, 63, 64] and Kojic et al. [91] and the references therein. One is immediately overwhelmed by the plethora of mathematical equations accompanying the description of every biological process. Circulation, motion, growth, living tissues, cardiovascular mechanics, electrocardiography (ECG), electroencephalography (EEG), magnetoencephalography (MEG), ultrasonography and source localization are all described by a specific boundary value problem (BVP) involving a system of coupled or uncoupled partial differential equations (PDEs). For instance, under most circumstances, blood is modeled as an incompressible Newtonian fluid and the blood flow is governed by the incompressible Navier-Stokes equations [60, 61]. At tissue level the arterial walls, muscles and other soft tissues such as tendon, ligament and cartilage can all be modeled as continuous media. In the investigation of bone healing through guided ultrasound waves [126, 127], bones are modeled as transversely isotropic elastic materials and the elastic wave propagation problem is governed by the Navier equation.

The underlying BVPs for modeling biomedical systems involve linearly or nonlinearly coupled systems of PDEs in quite non-trivial geometries. Much effort has been devoted to the understanding of similar phenomena in one space dimension, where the underlying equations can be solved analytically in a first attempt to investigate and monitor physical processes of this type. However, one space dimension inevitably hides evolution phenomena that are by nature three-dimensional, and the one-dimensional models fail in some rather simple cases.

The development and validation of such models in any case relies heavily on experimental investigations, which nowadays are increasingly coupled to computer modeling. Computer modeling, in turn, employs sophisticated numerical methods for solving the underlying boundary value problems. The accurate and efficient treatment of such BVPs in realistic three-dimensional geometries is a computationally very demanding process and a highly active area of research.

Traditionally, BVPs can be broadly classified as either exterior or interior, and numerical methods specialize to either case. In the first category, one usually finds scattering processes and wave propagation phenomena. The second category includes phenomena that evolve in a fixed bounded domain. Both problem types model several processes encountered in bioengineering and both are considered in this work. Our contributions are presented in the following three chapters.

Chapter 1: Sonography Sonography is widely known as an ultrasound-based diagnostic imaging technique used to visualize muscles and internal organs, their size, structures and possible pathologies or lesions. There are a plethora of diagnostic and therapeutic applications in medicine using sound waves propagating at specific frequencies. The choice of the underlying frequencies of the generated acoustic waves depends on the application. For example, in ultrasound imaging there is a trade-off between spatial resolution of the image and imaging depth: lower frequencies produce less resolution but penetrate deeper into the body.

Despite the widespread use of ultrasonography as an efficient and "safe" imaging technique, there are several weaknesses of ultrasonic imaging as well as risks and side-effects [84, 88, 89]. Sonographic devices have trouble penetrating bones. They perform very poorly when there is gas between the transducer and the organ of interest because of the extreme differences in acoustic impedance. Even in the absence of bone or air the depth of penetration of ultrasound is limited, making it difficult to image structures deep in the body. These weaknesses are a direct consequence of the high frequencies of ultrasound waves. Waves of lower frequencies would potentially not suffer from such problems. This fact strongly encourages the investigation of their suitability as a possible replacement of ultrasound waves in imaging applications.

A first step in this direction is the quest for devising mathematical methods for the efficient treatment of the direct acoustic scattering problem. This is an exterior BVP governed by the Helmholtz equation. We employ an analytical method for the investigation of the direct scattering problem using a prolate spheroid, which can model several organs or structures of the human body. The efficiency of analytical approaches employed for this purpose is strongly influenced by physical and geometrical characteristics of the scatterers involved, as well as the frequency of the incident field. Our contribution in this work is three-fold.

First, a novel basis of Helmholtz outward-radiating eigensolutions for the underlying spheroidal geometry is introduced by employing the Vekua transformation, which maps the kernel of the Laplace operator to the kernel of the Helmholtz operator. The scattered field is then represented as an infinite expansion of this complete set, which is inevitably truncated to allow numerical schemes to be developed.

Second, the coefficients of the truncated expansion are provided by the solution of linear systems constructed to minimize the L^2 norm of the error of the boundary condition satisfaction, introduced by truncation, on the scatterer's surface.

Finally, a study of the condition of the matrices involved in the aforementioned linear systems revealed the need for computation in arbitrary precision. Appropriate multiple-precision software was developed, allowing a thorough convergence study and investigation of the sensitivity of the solution with respect to parameters related to the scattering problem and the adopted numerical scheme. Up to 180 digits of precision were needed to achieve reliable solutions. Our results are accompanied by visualizations of far-field patterns, clarifying the preferred scattering directions for a wide range of frequencies and eccentricities of the spheroidal scatterer.

Chapter 2: Convection-Diffusion-Reaction Coupled convection-diffusion-reaction PDE systems model a wide variety of biomedical and physical applications. For instance, mass transport of oxygen in arteries and transport of the low density lipoprotein (LDL), well known as an atherogenic molecule, are governed by convection-diffusion PDE systems [91]. The modeling of thrombosis by continuum-based methods involves nonlinearly coupled convection-diffusion-reaction systems [91]. Gas flow in the airway tree (respiration system) is also modeled by convection-diffusion equations [64]. Tumor growth modeling involves nonlinear reaction-diffusion systems [92, 93]. Finally, EEG source localization in the modeling of focal epilepsy is described by the Poisson equation [8, 9, 10, 124]. Industrial applications include semiconductors, fuel cells, and many others.

The numerical treatment of such problems must overcome two main obstacles. This chapter focuses on the first obstacle: the quality of results. Most standard numerical schemes fail when applied to convection-dominated PDEs. For instance, the standard Finite Element Bubnov-Galerkin method obtains solutions that suffer from instabilities, which appear as nonphysical ripples contaminating the solution domain. Stabilization approaches are needed to eliminate numerical ripples where possible in order to achieve high quality solutions.

Classical approaches such as streamline-upwind Petrov-Galerkin (SUPG), Galerkin least-squares (GLS), continuous interior penalty (CIP), and least-squares finite-element methods, are all designed to provide stable solutions. However, they are incapable of removing numerical ripples completely, especially in areas where the solution exhibits steep gradients. Thus, quantities positive by nature (such as energy, concentrations, densities etc.) inevitably become negative and the corresponding results leave much to be desired. An approach that totally eliminates numerical oscillations and provides positivity-preserving results for transient problems is the class of Flux-Corrected Transport methods (FCT).

Our second chapter presents a new generalization of the FCT methodology to implicit finite element discretizations. It preserves all the desired properties (superlinear convergence, positivity) of the original scheme introduced in [94, 99] and refined in [97, 98], and at the same time speeds up the whole process by using a single sweep of the multidimensional FCT limiter at the first outer iteration. It thereby avoids the computation of an intermediate low-order solution, which was essential for the original algorithm. Finally, the suggested scheme favors the application of the discrete Newton approach employed for the solution of the nonlinear systems, as it simplifies considerably the assembly of the related discrete Jacobians.

Chapter 3: Domain Decomposition The second obstacle we encounter is common to every PDE system when a numerical solution method is employed. The obstacle is especially great for the Finite Element method. Discretization of the PDE system in that case leads to sparse algebraic linear systems involving millions of equations and unknowns. Especially in three-dimensional domains, the solution of the linear systems typically constitutes 90–95% of the total running time of the computer simulation. Efficient solution of such linear systems is still considered an open problem.

Motivated by the widespread availability of multi-processing computing systems, we are forced to consider solution methods specifically designed for parallel processing. A large category of such methods is the class of Domain Decomposition methods, which we investigate in our third chapter.

The usefulness of any domain decomposition method rests on the ability to solve a problem that is posed on the internal boundary introduced by the decomposition and involves a pseudo-differential operator: the Steklov-Poincaré operator. To this end, a great number of iterative approaches have been suggested in the literature; classical algorithms include Dirichlet-Neumann, Neumann-Neumann, FETI methods, Schwarz methods, together with two-level and overlapping variants.

An alternative that has not been considered to date but can be shown to be competitive is based on a well known property of the discrete Steklov-Poincaré operator: it is norm-equivalent to a Sobolev norm-matrix of index 1/2. Our contribution builds on recent results of Arioli and

Loghin, which provide explicit finite-element representations of fractional Sobolev norms. These discrete representations can be written in terms of generalised eigenvalue problems defined on the interface associated with the domain decomposition under consideration.

Our work focuses on a general class of scalar elliptic partial differential equations in both two and three dimensions. Given the non-sparse representation of fractional Sobolev norms, special algorithms for computing the action of these implicitly defined norm-matrices are required in order to maintain the scalability and efficiency in a domain decomposition context. We provide a description and analysis of optimal algorithms based on approximation of fractional Sobolev norms via sparse algorithmic approaches such as the Lanczos algorithm. In particular, our analysis indicates independence of the size of the problem. This prediction is tested on a range of discretizations of elliptic problems in both two and three dimensions. The performance of the resulting iterative solvers surpasses existing methodologies.

For example, the finite element method was applied to a scalar Poisson problem with Dirichlet boundary conditions on the geometry of the human brain. The mesh consisted of 164 million tetrahedra and 26 million nodes, and was partitioned into 2048 subdomains. The linear system of order 26 million was solved on a single processor (1.9 GHz) in around 30 minutes (with the residual norm reduced by a factor of 10^{-6}). With our preconditioner, the Flexible GMRES solver required only 24 iterations.

The scalability of the parallel implementation of our preconditioning approach is extensively benchmarked on SMP multiprocessing platforms under several different operating systems (Linux, Sun Solaris, IBM AIX). With 16 CPUs, speedups of around 13 were achieved for the factorization phase and about 9 for the solution phase. The speedups were found to depend on the operating system and the vendor-supplied BLAS implementation.

Acknowledgments

First and foremost I would like to thank my father, because without him this thesis would not have been possible. I would also like to express my gratitude towards my supervisor Antonios Charalambopoulos for introducing me to the field of Mathematical Modeling, for familiarizing me with the "hardcore" mathematical language, and for standing by me as a friend as well.

I am also indebted to professor Stefan Turek and his associates for introducing me, during a summer school, to the Finite Element method and the realm of Fluid Dynamics. Much appreciation and gratitude goes especially to Dr Dmitri Kuzmin for his long-term support and excellent collaboration.

I owe a lot to my friend and colleague Dr Leonidas N. Gergidis for his constant moral and scientific support as well as the great collaboration we enjoyed the last 5 years.

I am also extremely grateful to Dr Klaus Gärtner for his long-term guidance, endless support and fruitful discussions, for providing computer resources and the PARDISO direct sparse solver, which I have been using continuously during the last 5 years.

Many thanks go to Dr Hang Si for his eager support and collaboration as well as his excellent tetrahedral mesh generator TetGen, without which a large part of this thesis would not have been possible. I would also like to thank Professor Jonathan Richard Shewchuk, whose mesh generator Triangle was my starting point and an invaluable tool throughout my research.

I owe very many thanks to Dr Daniel Loghin and Dr Mario Arioli for their interest in my research, their invitation to the $H^{1/2}$ Domain Decomposition project, their patience, guidance, inspiring and endless discussions.

I have no words for expressing my gratitude to Professor Michael Saunders with whom I met last year. Not only he did an amazing job as a reader watching over the slightest detail, but he also helped tremendously with the typesetting of this thesis. I also thank him for his constant support, encouragement and belief in me, but mostly because he assisted so much in opening a new leaf in my life.

I would like to dedicate this dissertation to my family and to all those who stood by me in this rather long journey. Especially to those who will continue to stand by me until the end of my days.

Contents

List of Tables

List of Figures

Chapter 1

Sensitivity of the acoustic scattering problem in prolate spheroidal geometry with respect to wavenumber and shape

1.1 Introduction

The investigation of the scattering problem in spheroidal geometry has attracted scientific interest for several decades. This is due to the simplicity and the important property of the spheroidal system to fit quite accurately in several geometrical configurations lacking symmetry in one only Cartesian direction. Thus, it simulates successfully a large variety of inclusions or inhomogeneities participating in scattering processes. A lot of effort has been devoted especially in the realm of time harmonic acoustics to studying the direct scattering problem by spheroids. The adopted methodology depends crucially on the frequency range under consideration.

In the resonance region, the most popular approaches employ either separation of variables or T-matrix methods [73, 75, 158, 164]. In the vast majority of these approaches, the well known spheroidal wave functions dominate, which emerge via spectral analysis of the Helmholtz equation in spheroidal coordinates. These functions are constructed via a necessary intermediate numerical scheme, which becomes cumbersome and extremely complicated for spheroids with large focal distances and small semi axes ratio. In addition, the spheroidal wave functions are defined via infinite expansions in terms of basis functions of separable form, and the convergence becomes poor in the upper limit of the resonance region and even worse in the high frequency realm. Particularly for high frequencies, the only practical choice is to resort to asymptotic methods (e.g., ray tracing). These methods, on the other hand, are not error-controllable because they solve an approximate model instead of the original equations (the eikonal equation instead of the Helmholtz equation itself). The adoption of resonance techniques in the high frequency regime, in cases where this is realizable, increases severely the computational resources. For instance, the Mie theory for acoustic scattering by a sphere of radius a, predicts that the number of summation terms is proportional to $k^2a + c(ka)^{1/3} + b^2$ (k is the wavenumber and c, b suitable constants) as the frequency becomes higher [163].

It is well known that there are several numerical difficulties inherent in the solution of the scattering problem using analytical methods, strongly dependent on the geometrical features of the scatterer and the imposed frequency (or equivalently the induced wave number). Apart from the intrinsic numerical difficulty in the determination of the spheroidal wave functions, the standard methodology of expanding the scattered field in terms of the aforementioned basis leads to extremely ill-conditioned matrices in the linear systems arising from the boundary condition satisfaction. It is widely recognized [12, 24] that the 64-bit and 80-bit IEEE floating point arithmetic formats currently provided and utilized in most computer systems are inadequate for the inversion of ill-conditioned matrices of this type [149].

In our previous work [66], we introduced a new theoretical setting inspired by a novel concept [33] in which the Vekua transformation was adopted [159, 160, 161], in order to construct Helmholtz equation solutions by transforming appropriately the well known spheroidal harmonic functions. This setting avoids the standard spheroidal wave functions, thus removing all the intrinsic numerical deficiencies and truncation errors. In the present work, we extend the numerical investigation of the acoustic scattering problem by spheroidal scatterers (introduced in [66]) towards the high frequency regime.

In Section 1.2 we present the necessary theoretical outcome, taken from [66]. We also provide additional theoretical arguments and justifications regarding the completeness of the constructed solution set. The scattered field emanating from the interaction of a plane acoustic wave with an impenetrable soft spheroidal scatterer is represented as an expansion in terms of the elements of the Vekua basis. This infinite expansion is truncated and forced to satisfy the boundary condition on the scatterer's surface. For the determination of the expansion coefficients we follow the L^2 error norm minimization methodology [66], which has been proven very robust and reliable in the low-frequency regime even for very elongated spheroidal bodies. The numerical investigation of the system, the solution of which furnishes the truncated expansion coefficients and its conditioning, is presented in Section 1.3. There we also expose an extensive convergence analysis of our approach, in terms of truncation level, geometry and wavenumber. Additionally we provide the global dependence of the L^2 norm of the error as a function of the aforementioned three parameters. The cornerstone of the numerical implementation is arbitrary-precision software facilities [12, 13, 167], an indispensable tool for several scientific areas such as Experimental Mathematics, Climate Modeling [24], Computational Geometry [141] among many others, which in our case allows the solution of the encountered linear systems. Finally the main outcome of the direct scattering problem, i.e., the far-field pattern, is constructed and appropriately visualized for indicative wave numbers ranging from low frequencies to the dawn of high-frequency asymptotics.

1.2 Theoretical formulation

In [159, 161] one can find a very interesting one-to-one transformation connecting the solutions of Laplace and Helmholtz equations with regular behavior near the origin, for arbitrary space dimensions. In three dimensions, the transformation pair becomes

$$u(\mathbf{r}) = u_0(\mathbf{r}) - \frac{kr}{2}\int_0^1 u_0(t\mathbf{r}) J_1(kr\sqrt{1-t})\frac{\sqrt{t}}{\sqrt{1-t}}dt \tag{1.1}$$

and

$$u_0(\mathbf{r}) = u(\mathbf{r}) + \frac{kr}{2}\int_0^1 u(t\mathbf{r}) I_1(kr\sqrt{t(1-t)})\frac{dt}{\sqrt{1-t}}, \tag{1.2}$$

where $r = |\mathbf{r}| = (x_1{}^2 + x_2{}^2 + x_3{}^2)^{\frac{1}{2}}$, $u \in ker(\Delta + k^2)$, $u_0 \in ker\,\Delta$, and J_1, I_1 stand for the Bessel and the modified Bessel functions of order 1 respectively. The wave number k and the radial frequency ω of the scattering process are interrelated via the basic relation $k = \omega/c$, where c stands for the velocity of the acoustical waves. The transformation presented above clearly concerns regular, near the origin, functions and consequently refers to solutions of Laplace and Helmholtz equations in interior domains. We summarize briefly the main arguments needed to describe the construction of the novel set of Helmholtz equation solutions.

1.2.1 Construction of a new set of Helmholtz eigensolutions

Let us introduce the prolate spheroidal coordinate system, described by the spheroidal coordinates (μ, θ, ϕ). The prolate spheroidal coordinates are interrelated with the Cartesian ones via

$$x = \frac{\alpha}{2}\sinh\mu\,\sin\theta\,\cos\phi, \qquad \mu \in [0,\infty), \tag{1.3}$$

$$y = \frac{\alpha}{2}\sinh\mu\,\sin\theta\,\sin\phi, \qquad \theta \in [0,\pi], \tag{1.4}$$

$$z = \frac{\alpha}{2}\cosh\mu\,\cos\theta, \qquad \phi \in [0,2\pi), \tag{1.5}$$

where α stands for the focal distance. The complete set of eigensolutions of Laplace operator, which are regular harmonic functions at the origin, expressed in prolate spheroidal coordinates is given by

$$u_{nm}^{0}(\mathbf{r}) = P_n^m(\cosh\mu)P_n^m(\cos\theta)e^{im\phi}, \quad n = 0, 1, 2, \ldots, \ |m| \leq n, \tag{1.6}$$

where P_n^m are the Legendre functions. We substitute every member of (1.6) in (1.1) aiming at the construction of the corresponding eigensolution of Helmholtz equation. Then every harmonic function u_{nm}^0 gives rise to the dynamic eigensolution

$$u_{nm}(\mathbf{r}) = P_n^m(\cosh\mu)P_n^m(\cos\theta)e^{im\phi} - \frac{kr}{2}\int_0^1 P_n^m(\cosh\mu')P_n^m(\cos\theta')e^{im\phi'} J_1(kr\sqrt{1-t})\sqrt{\frac{t}{1-t}}dt, \tag{1.7}$$

where (μ', θ', ϕ') represent the spheroidal coordinates of $\mathbf{r}' = t\mathbf{r}$, $(0 \leq t \leq 1)$. Extended manipulations of the involved special functions in the integrand [33] transform $u_{nm}(\mathbf{r})$ to the representation

$$u_{nm}(\mathbf{r}) = \sum_{p=0}^{[\frac{n}{2}]}\sum_{l=0}^{[\frac{n-2p}{2}]} B_{n,m,p,l}\Gamma(n-2p+\frac{3}{2})\frac{J_{(n-2p+\frac{1}{2})}(kr)}{(\frac{kr}{2})^{n-2p+1/2}}P_{n-2p-2l}^m(\cosh\mu) P_{n-2p-2l}^m(\cos\theta)e^{im\phi}, \tag{1.8}$$

where

$$B_{n,m,p,l} = \begin{cases} \frac{(-1)^p(n+m)!(n-2p-2l-m)!(2n-2p)!}{(n-m)!(n-2p-2l+m)!p!}\frac{(n-2p-l)!(2n-4p-4l+1)}{(n-p)!l!2^{2p+2l}(2n-4p-2l+1)!} & |m| \leq n-2p-2l \\ 0 & |m| > n-2p-2l. \end{cases} \tag{1.9}$$

Representation (1.8) would have been an elegant formula for $u_{nm}(\mathbf{r})$ if the radial coordinate r were not encountered therein. Actually, the radial component r involves both spheroidal coordinates (μ, θ) and prevents $u_{nm}(\mathbf{r})$ from being expressible in terms of spheroidal coordinates in a fully separable manner. As described in detail in [33], expanding the Bessel function $J_{n-2p+1/2}$ as a power series of its argument and using functional properties of Legendre functions products, we obtain the following quasi-separate representation of (1.8):

$$u_{nm}(\mathbf{r}) = \sum_{q=0}^{\infty}\sum_{j=0}^{q}\sum_{p=0}^{[\frac{n}{2}]}\sum_{l=0}^{[\frac{n-2p}{2}]}\sum_{i=-j}^{j} A(q,j,n,m,p,l,i,c)(\sinh\mu)^{2q-2j}$$
$$P^m_{n-2p-2l}(\cosh\mu)P^m_{n-2p-2l+2i}(\cos\theta)e^{im\phi}. \qquad (1.10)$$

In (1.10), $c = ka/2$ and $A(q,j,n,m,p,l,i,c)$ is a specific function of its arguments [33]. However, expression (1.8) is a very useful "hybrid" form of $u_{nm}(\mathbf{r})$ as it consists of a finite superposition of products. Each one of those products is built by a harmonic separable part and a wave term, incorporating the wave number k appropriately accompanied by the radial distance r. This expression is the convenient one for constructing the exterior eigensolutions of Helmholtz equation. We consider separately the functions generated by splitting $e^{im\phi}$ into its real and imaginary parts and invoke the spherical Bessel functions instead of the cylindrical ones. Then

$$\left\{\begin{array}{c} u^c_{nm}(\mathbf{r}) \\ u^s_{nm}(\mathbf{r}) \end{array}\right\} = \sum_{p=0}^{[\frac{n}{2}]}\sum_{l=0}^{[\frac{n-2p}{2}]} B_{n,m,p,l}\Gamma(n-2p+\frac{3}{2})\frac{2}{\sqrt{\pi}}\frac{j_{(n-2p)}(kr)}{(\frac{kr}{2})^{n-2p}}$$
$$P^m_{n-2p-2l}(\cosh\mu)P^m_{n-2p-2l}(\cos\theta)\left\{\begin{array}{c} \cos(m\phi) \\ \sin(m\phi) \end{array}\right\}. \qquad (1.11)$$

We now replace in the formulae above the Bessel function with the corresponding outwards radiating Hankel one, thus constructing the functions $\hat{u}^c_{nm}$ and $\hat{u}^s_{nm}$, which are irregular in the vicinity of $r = 0$ and constitute candidates for being outgoing waves obeying to the Helmholtz equation. We treat only $\hat{u}^c_{nm}$ for simplicity, obtaining

$$\hat{u}^c_{nm}(\mathbf{r}) = \sum_{p=0}^{[\frac{n}{2}]}\sum_{l=0}^{[\frac{n-2p}{2}]} B_{n,m,p,l}\Gamma(n-2p+\frac{3}{2})\frac{2}{\sqrt{\pi}}\frac{h^{(1)}_{n-2p}(kr)}{(\frac{kr}{2})^{n-2p}}$$
$$P^m_{n-2p-2l}(\cosh\mu)P^m_{n-2p-2l}(\cos\theta)\cos(m\phi). \qquad (1.12)$$

We consider now Rayleigh's formula [15]

$$h^{(1)}_n(z) = -i(-1)^n z^n(\frac{1}{z}\frac{d}{dz})^n(\frac{e^{iz}}{z}), \qquad (1.13)$$

which suggests that the function

$$f^c_{nm}(k;\mathbf{r}) = k^{2n+1}(\Delta + k^2)\hat{u}^c_{nm}(\mathbf{r}), \qquad (1.14)$$

depending only on k (with $\mathbf{r} \neq 0$ kept fixed) is analytic in the upper half of the complex plane. The real part $v^c_{nm}(k;\mathbf{r})$ of $f^c_{nm}(k;\mathbf{r})$ is a harmonic function in the domain where it is defined. On the real axis ($Im(k) = 0$), we have

$$v^c_{nm}(k;\mathbf{r})|_{k\in R} = Re\{k^{2n+1}(\Delta + k^2)\hat{u}^c_{nm}(\mathbf{r})\}|_{k\in R} = k^{2n+1}(\Delta + k^2)u^c_{nm}(\mathbf{r}) = 0. \qquad (1.15)$$

On the large semicircle of the upper half-plane of radius R, the harmonic function $v^c_{nm}(k;\mathbf{r})$ takes values decaying to zero for $R\to\infty$. This is proven by considering formula (1.13) and examining the asymptotic behavior of the crucial term e^{iz} for $z = Re^{i\gamma}r$, where $\arg\gamma \in (0,\pi)$.

Thus, the harmonic function $v^c_{nm}(k;\mathbf{r})$ vanishes in the upper half-plane and clearly so does the analytic function f^c_{nm}. Then for real k, $(\Delta + k^2)\hat{u}^c_{nm}(\mathbf{r}) = 0$ provided $\hat{u}^c_{nm}$ belongs to $ker(\Delta + k^2)$. Similarly we treat $\hat{u}^s_{nm}(\mathbf{r})$ and finally obtain that all $\hat{u}_{nm}(\mathbf{r})$ constitute eigensolutions of the Helmholtz equation. The radiating character of $\hat{u}_{nm}(\mathbf{r})$ is due to the asymptotic behavior of the Hankel function.

1.2.2 Completeness of the new set

We proceed with the completeness of the constructed solution set, as our purpose is to expand our arbitrary Helmholtz equation solution in terms of this basis set. We remark first that $\hat{u}_{nm} = u_{nm} + i\tilde{u}_{nm}$, where $\tilde{u}_{nm}$ is constructed with the Bessel function of first kind in (1.8) replaced by the corresponding Neumann function. Let u be an arbitrary radiation solution of the Helmholtz equation. Its regular part $Regu$ can be expanded in terms of the complete set u_{nm} (in the space of regular Helmholtz equation solutions) as

$$Regu = \sum_{n,m}\gamma_{nm}u_{nm}. \qquad (1.16)$$

We define

$$w = \sum_{n,m}\gamma_{nm}\hat{u}_{nm} = Regu + i\sum_{n,m}\gamma_{nm}\tilde{u}_{nm}. \qquad (1.17)$$

The function $u - w$ disposes zero regular part and satisfies the Helmholtz equation and the well known Sommerfeld radiation condition,

$$\frac{\partial u^{sc}(\mathbf{r})}{\partial r} - iku^{sc}(\mathbf{r}) = O(\frac{1}{r^2}),\ r\to\infty. \qquad (1.18)$$

Then,

$$u - w = \frac{e^{ikr}}{r} g(\theta, \phi) + O(\frac{1}{r^2}),\ r \to \infty, \tag{1.19}$$

where $g(\theta, \phi)$ the scattering amplitude of the radiating field. Note that

$$Reg(u - w) = i\frac{sin(kr)}{r} g(\theta, \phi) + O(\frac{1}{r^2}) \tag{1.20}$$

and then $g(\theta, \phi)$ vanishes. Invoking Relich's lemma leads to the result that $u - w$ vanishes in the whole exterior domain. Thus $u = \sum_{n,m} \gamma_{nm} \hat{u}_{nm}$.

1.2.3 Direct scattering problem

The above analysis settles the background for developing the corresponding scattering problem. We consider a prolate spheroidal acoustically impenetrable (soft) scatterer occupying a specific region in R^3 defined by the scatterers's surface S, which is represented by the spheroidal surface

$$\mu = \mu_0. \tag{1.21}$$

The exterior region of the scatterer is denoted by D and is characterized by the range $\mu > \mu_0$, $0 \leq \theta \leq \pi$, $0 \leq \phi < 2\pi$ of spheroidal coordinates. The scatterer is illuminated by a time harmonic incident accoustic plane-wave of radial frequency ω. Suppressing the time dependence $e^{-i\omega t}$ in all the physical quantities of the scattering process, we represent the incident field by the time-reduced plane wave

$$u^{inc}(\mathbf{r}) = e^{i\mathbf{k}\cdot\mathbf{r}}, \mathbf{r} \in D, \tag{1.22}$$

where $\mathbf{k} = k\hat{\mathbf{k}}$, k is the wavenumber and $\hat{\mathbf{k}}$ is the direction of the incident field. The presence of the scatterer in the medium where the wave propagates gives rise to a secondary acoustic field, the scattered one denoted by u^{sc}, which satisfies exactly as the incident wave the Helmholtz equation

$$\Delta u^{sc}(\mathbf{r}) + k^2 u^{sc}(\mathbf{r}) = 0,\ \mathbf{r} \in D. \tag{1.23}$$

This field emanates from the scatterer and radiates to infinity, satisfying uniformly over all directions the Sommerfeld radiation condition (1.18). The total field $u(\mathbf{r}) = u^{inc}(\mathbf{r}) + u^{sc}(\mathbf{r})$ defined in $\overline{D} = D \cup S$ obeys on the scatterer's surface a specific type of boundary condition, depending on the special nature of the scatterer. We focus on the soft scatterer case, implying that

$$u(\mathbf{r}) = u^{inc}(\mathbf{r}) + u^{sc}(\mathbf{r}) = 0, \ \mathbf{r} \in S. \tag{1.24}$$

The methodology suggested here is based on exploiting the eigensolutions constructed in the previous section. More precisely, these eigensolutions are produced via the Vekua transformation of the complete set of the spheroidal harmonic separable solutions. From all these transformed fundamental solutions, we select the set of outgoing radiating fields, because only these functions satisfy radiation condition (1.18). We then expand the unknown scattered field in terms of the aforementioned radiating basic solutions to obtain

$$u^{sc}(\mathbf{r}) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} A_{nm} \hat{u}_{nm}(\mathbf{r}), \ \mathbf{r} \in D, \tag{1.25}$$

where the coefficients A_{nm} are to be determined.

1.2.4 Far field

The representation (1.25) can be exploited to provide the far-field pattern, which determines the behavior of the scattered field far from the scatterer and usually constitutes the measured quantity in direct scattering. What is necessary is to investigate the asymptotic behavior (for $\mathbf{r} \rightarrow \infty$) of the eigensolutions $\hat{u}_{nm}(\mathbf{r})$. For $\mathbf{r} \gg 1$, we apply an extended but straightforward asymptotic analysis of the special functions involved in the "hybrid" definition formula of $\hat{u}_{nm}(\mathbf{r})$, namely

$$\hat{u}_{nm}(\mathbf{r}) = \sum_{p=0}^{[\frac{n}{2}]} \sum_{l=0}^{[\frac{n-2p}{2}]} B_{n,m,p,l} \Gamma(n - 2p + \frac{3}{2}) \frac{H^{(1)}_{(n-2p+\frac{1}{2})}(kr)}{(\frac{kr}{2})^{n-2p+1/2}}$$
$$P^m_{n-2p-2l}(\cosh \mu) P^m_{n-2p-2l}(\cos \theta) e^{im\phi}. \tag{1.26}$$

The scattered field obtains the well known form

$$u^{sc}(\mathbf{r}) = \frac{e^{ikr}}{kr} f_\infty(\theta, \phi) + O(\frac{1}{r^2}), r \rightarrow \infty, \tag{1.27}$$

where the far-field pattern $f_\infty(\theta,\phi)$ is given by

$$f_\infty(\theta,\phi) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}\frac{2}{\sqrt{\pi}}e^{-i\frac{\pi}{4}}A_{nm}\sum_{p=0}^{[\frac{n}{2}]}B_{n,m,p,0}\Gamma(n-2p+\frac{3}{2})e^{-i\frac{1}{2}(n-2p+\frac{1}{2})\pi}$$
$$(\frac{4}{k\alpha})^{n-2p}\frac{2^{2p-n}[2(n-2p)]!}{(n-2p)!(n-2p-m)!}P^{m}_{n-2p}(\cos\theta)e^{(im\phi)}. \qquad (1.28)$$

Satisfaction of the boundary condition (1.24) leads to the determination of the expansion coefficients in (1.25) and hence to the solution of the direct scattering problem.

1.3 Numerical investigation and implementation

The numerical implementation of our approach involved both C++ and Mathematica software development, to allow consistency checks. Both implementations relied heavily on arbitrary precision, without which the range of the frequency of the incident wave, the geometry and the truncation level of the series that could be handled would be severely restricted. The need for arbitrary precision motivated the use of general computing environments like Mathematica [167], which incorporates arbitrary precision in a very natural way. The need for performance and convenience motivated the development of C++ software. Arbitrary-precision arithmetic in C++ was provided by the ARPREC library [13]. The implementations of special functions in C++ was based on [125, 170] with necessary modifications and tuning to the working precision. This involves recalculation of all the usual parameters and mathematical constants entering the definition of special functions to the desired precision and appropriate modifications of the source code of the special functions [125, 167, 170] in order to be evaluated to the required precision. The results obtained by both C++ and Mathematica implementations agreed to all but the last two or three decimal digits in any desired precision.

1.3.1 Assembly and solvability of the system

The most computationally demanding part of our approach is the assembly of the linear system whose solution provides the expansion coefficients. This is because the L^2 norm minimization approach needs expensive 2D quadratures, and further, all entries of the matrix and right-hand side involve special function evaluations that must converge to the working precision. Moreover, the aforementioned systems involve extremely ill-conditioned matrices with condition numbers ranging from 10^{10} to 10^{160} and more, as demonstrated in the sequel.

The solution of the linear systems was provided by Singular Value Decomposition (SVD) [68, 125], which allows direct calculation of the condition number of the matrices involved. As the most computationally intensive part of our approach was the assembly of the linear system and not its solution, in contrast to classical numerical methods, computational overhead associated with the specific choice of SVD over the classical LU decomposition was negligible.

In the scattering process under investigation we consider a spheroidal scatterer with large semi axis whose reduced length is kept constant and equal to one, while the small semi axis varies suitably, giving rise to several aspect ratios. The excitation of the scattering mechanism has been accomplished with a plane wave corresponding to the wavenumber propagation vector $\mathbf{k} = k\hat{\mathbf{k}}$, where $\hat{\mathbf{k}} = (\hat{\mathbf{x}}+\hat{\mathbf{y}}+\hat{\mathbf{z}})/\sqrt{3}$ and $k \in \{0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4\}\sqrt{3}$ (all in reduced units). We have applied an incident wave field of this form in order to handle a rather generic excitation case. This increases the complexity of the numerical calculations involved, compared to the one coordinate axis oriented stimulation, but reveals the reliability and robustness of the adopted methodology in the general propagation case.

We should point out that the dimensionless product of the incident wavenumber k and the characteristic dimension of the prolate spheroid, taken as the large semi axis C, is being expanded beyond low frequency $kC \ll 1$ or resonance region $kC \simeq 1$. The range for the dimensionless product is taken from $kC = 0.5\sqrt{3}$ to $kC = 4\sqrt{3}$ in all our "computer" scattering "experiments".

1.3.2 Condition number

In Fig. 1.1 (left) we plot the condition number of matrices arising in the adopted methodology, as a function of the truncation level of the series N at various wavenumbers for a semi axes ratio equal to 0.6. It is evident that the condition number is established mainly by the truncation level of the series, while the role of the wavenumber of the incident field is almost negligible especially for values beyond $k = 0.5\sqrt{3}$. The plot clearly reveals an exponential growth of the condition number with increasing N. On the contrary the dependence of the condition number on the wavenumber k is hardly noticeable as the wavenumber increases from $k = 1\sqrt{3}$ up to $k = 4\sqrt{3}$.

In Fig. 1.1 (right) we plot the condition number of the matrices for a truncation level of the series $N = 16$ as a function of the wavenumber k for several aspect-ratios of the scatterer, ranging from $a_R = 0.6$ to $a_R = 0.9$. For the low wavenumber region defined by $k \in [0.5\sqrt{3}, 1.5\sqrt{3}]$ there is an exponential decay with two different slopes in logarithmic representation for $k \in [0.5\sqrt{3}, 1\sqrt{3}]$ and from $k \in [1\sqrt{3}, 1.5\sqrt{3}]$. Surprisingly a plateau value for the condition number is being established for all the aspect ratios and the wavenumbers beyond $k = 1.5\sqrt{3}$. The condition of the system strongly depends on the truncation of the series and not the geometry under investigation or the wavenumber of the incident wave.

Due to the fact that those linear systems are highly ill-conditioned, the solution obtained by one back substitution is extremely inaccurate relative to the working precision, as we can see in

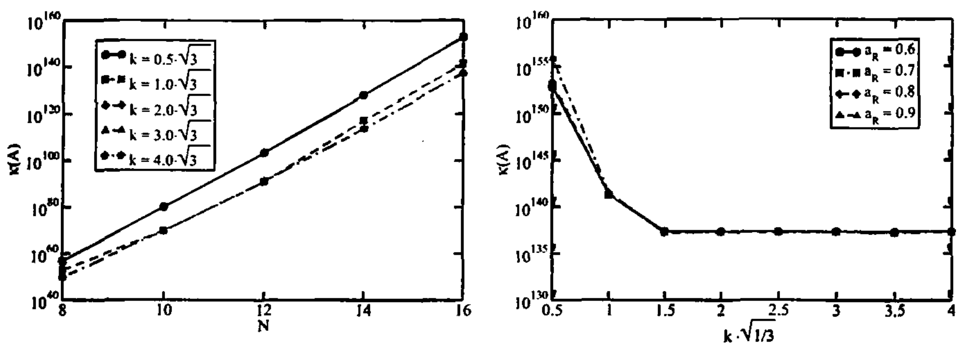


Figure 1.1: Condition number as a function of N for several k's at $a_R = 0.6$ (left) and as a function of the wavenumber k for several a_R's and $N = 16$ (right).

Table 1.1. To adjust the solution to our working precision we used iterative refinement [68, 125]. Error and residual bounds involved in the iterative refinement procedure were adjusted to the working precision. Below, the desired precision was set to 170 decimal digits and the update $\|\delta x\|_2$ and residual $\|r\|_2$ Euclidean norm tolerances were adjusted to 10^{-170} and 10^{-172} respectively. As a

Table 1.1: Update and residual norms during iterative refinement.

Iteration	$\|\delta x\|_2$	$\|r\|_2$
1	1.213932e+14	7.060036e-60
2	5.556136e-18	9.824089e-127
3	7.734852e-85	3.159688e-133
4	2.130082e-164	3.159688e-133
5	1.831668e-166	3.159688e-133

rule of thumb the working precision should be tuned [149] to $\log_{10} \kappa(A)$. It is evident that in the first step of iterative refinement the norm of the solution update is still too large and five steps of iterative refinement need to be performed in order to reduce the Euclidean norm of the error to our working precision. The computational cost of iterative refinement procedure is negligible compared to that of the matrix factorization. This suggests that iterative refinement is a cheap way of obtaining highly accurate results when necessary.

1.3.3 Convergence analysis

Our convergence study focuses on the treatment of boundary condition satisfaction (1.24). Due to the well-posedness of the direct scattering problem, convergence of the numerical solution to the solution of the exact scattering problem is guaranteed, provided the error related to the boundary condition satisfaction approaches zero. In what follows we present an extensive study of the convergence in L^2 norm of the so-constructed error function defined as

$$\begin{aligned}\|\epsilon_N\|_{L^2(S)} &= \left(\int_S |\epsilon_N(\theta,\phi)|^2 \, \mathrm{d}S\right)^{\frac{1}{2}},\\ \epsilon_N(\theta,\phi) &= (u_N^{sc}(\mathbf{r}) + u^{inc}(\mathbf{r}))|_{\mathbf{r}\in S}\\ &= \sum_{n=0}^{N}\sum_{m=-n}^{n} A_{nm}\hat{u}_{nm}(\mu_0,\theta,\phi) + e^{i\mathbf{k}\cdot\mathbf{r}(\mu_0,\theta,\phi)}. \end{aligned} \tag{1.29}$$

Computation of the integrals related to the L^2 norm was performed using Gauss-Legendre quadrature [125]. The number of required quadrature points was adjusted so that the L^2 norm of the error converged to the fifth significant digit. The way several parameters of our physical problem, like the aspect ratio of the prolate spheroidal scatterer and the frequency of the incident field, influence convergence is exposed in the plots that follow. The convergence study is supplemented with 3D plots of the distribution of the real and imaginary parts of the error on the scatterer's surface.

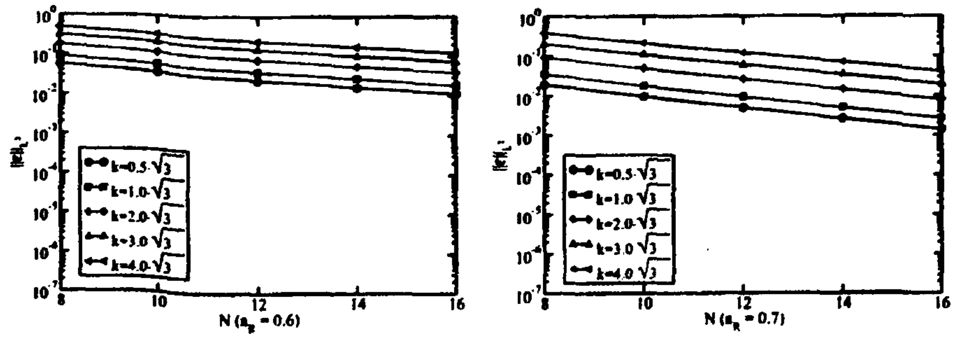


Figure 1.2: $\|\epsilon\|_{L^2(S)}$ as a function of the truncation level N with $a_R = 0.6, 0.7$ and $k = 0.5\sqrt{3}, \sqrt{3}, 2\sqrt{3}, 3\sqrt{3}, 4\sqrt{3}$.

Figs. 1.2 and 1.3 plot the dependence of the $\|\epsilon_N\|_{L^2(S)}$ as a function of the truncation level of the series N for several wavenumbers ($k = 0.5\sqrt{3}, \sqrt{3}, 2\sqrt{3}, 3\sqrt{3}, 4\sqrt{3}$) of the incident wave, for each of the scatterers ($a_R = 0.6, 0.7, 0.8, 0.9$) under consideration. The scale on the y axis is

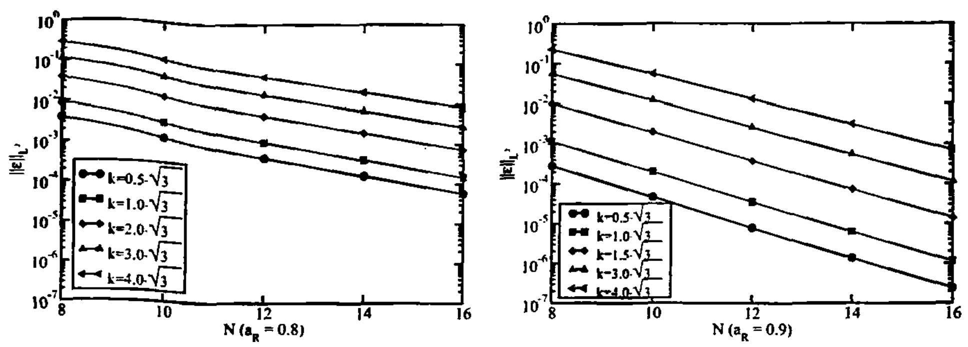


Figure 1.3: $\|\epsilon\|_{L^2(S)}$ as a function of the truncation level N with $a_R = 0.8, 0.9$ and $k = 0.5\sqrt{3}, \sqrt{3}, 2\sqrt{3}, 3\sqrt{3}, 4\sqrt{3}$.

logarithmic.

Several interesting properties regarding the convergence of our approach emerge from the plots. First of all, we observe that in all the cases, the convergence is clearly exponential. For each scatterer, the error increases with increasing wavenumber (consequently frequency), but the convergence rate remains the same independently of the frequency. This is a very nice property that suggests that incident waves of high frequencies can be sufficiently handled by the current approach, provided the convergence rate for the specific scatterer is sufficiently high. We observe however, that the convergence rate is strongly influenced by the aspect ratio of the spheroidal scatterer. The more elongated our spheroid becomes, the slower the series convergences. In Fig. 1.4 we plot $\|\epsilon_N\|_{L^2(S)}$ in logarithmic scale on the y axis, but now as a function of the wavenumber k for the aspect ratios under consideration: $a_R \in \{0.6, 0.7, 0.8, 0.9\}$ at $N = 16$. This plot reveals two important properties. As before, convergence rate clearly deteriorates with smaller aspect ratios. The error exponentially increases with increasing wavenumber k.

The error dependence on the truncation level of the series N and on the aspect ratio a_R, as depicted in Figs. 1.2, 1.3 and Fig. 1.4, allows a straightforward regression analysis, which will ideally express in closed form the L^2 norm of the error as a function of the three parameters involved. These are the truncation level N, the aspect ratio a_R and the wavenumber k. This is a three-step process, described below.

We begin by expressing the dependence of the logarithm of the L^2 norm of the error, which is clearly linear with respect to the truncation level of the series N, as

$$\ln(\|\epsilon\|_{L^2}(N, a_R, k)) = C(a_R, k)N + D(a_R, k), \tag{1.30}$$

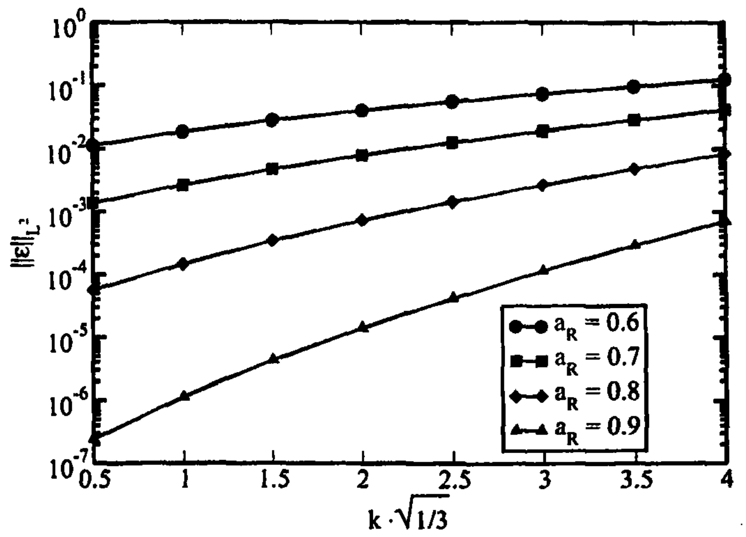


Figure 1.4: $||\epsilon||_{L^2}$ as a function of wavenumber for various aspect ratios at truncation level 16.

where $C(a_R, k)$, $D(a_R, k)$ are functions depending on the aspect ratio of the scatterer a_R and the wavenumber k. The values of $C(a_R, k)$, $D(a_R, k)$ for different aspect ratios and for indicative wavenumbers k obtained by regression analysis with sufficiently high correlation (0.999) are summarized in Table 1.2.

Having calculated the slope-regression coefficient $C(a_R, k)$ and the intercept-regression constant $D(a_R, k)$ we proceed with the investigation of the dependence of slope and intercept as functions of the wavenumber k.

The dependence of $C(a_R, k)$, as can be seen in Fig. 1.5 (left), is linear with respect to the wavenumber k and can be expressed as

$$C(a_R, k) = c_1(a_R)k + c_2(a_R), \qquad (1.31)$$

where c_1, c_2, depend only on the aspect ratio a_R. The values for $c_1(a_R), c_2(a_R)$ can be found in Table 1.3. We proceed by describing the dependence of $c_1(a_R), c_2(a_R)$ on the aspect ratio a_R by the following formulas:

$$c_i(a_R) = c_{i0} + c_{i1}a_R + c_{i2}a_R^2 \quad (i = 1, 2). \qquad (1.32)$$

Table 1.2: $C(a_R, k)$ and $D(a_R, k)$

a_R	k	$C(a_R, k)$	$D(a_R, k)$
0.60	$0.5\sqrt{3}$	-0.2023	-1.2552
	$1\sqrt{3}$	-0.1981	-0.8257
	$2\sqrt{3}$	-0.1920	-0.1828
	$3\sqrt{3}$	-0.1856	0.3331
	$4\sqrt{3}$	-0.1757	0.7056
0.70	$0.5\sqrt{3}$	-0.3299	-1.3274
	$1\sqrt{3}$	-0.3234	-0.7779
	$2\sqrt{3}$	-0.3096	0.0955
	$3\sqrt{3}$	-0.2951	0.7540
	$4\sqrt{3}$	-0.2809	1.2801
0.80	$0.5\sqrt{3}$	-0.5296	-1.3128
	$1\sqrt{3}$	-0.5191	-0.5413
	$2\sqrt{3}$	-0.4961	0.6951
	$3\sqrt{3}$	-0.4696	1.5875
	$4\sqrt{3}$	-0.4435	2.2968
0.90	$0.5\sqrt{3}$	-0.8822	-1.1351
	$1\sqrt{3}$	-0.8658	0.1568
	$2\sqrt{3}$	-0.8241	2.0133
	$3\sqrt{3}$	-0.7745	3.3340
	$4\sqrt{3}$	-0.7218	4.3189

Table 1.3: $c_1(a_R)$ and $c_2(a_R)$

a_R	$c_1(a_R)$	$c_2(a_R)$
0.6	0.0042	-0.2060
0.7	0.0081	-0.3373
0.8	0.0143	-0.5435
0.9	0.0266	-0.9104

Again, the regression analysis suggested

$$\begin{aligned} c_1(a_R) &= 0.0742 - 0.2429a_R + 0.2108a_R^2 \\ c_2(a_R) &= -2.0016 + 6.5219a_R - 5.8943a_R^2. \end{aligned} \tag{1.33}$$

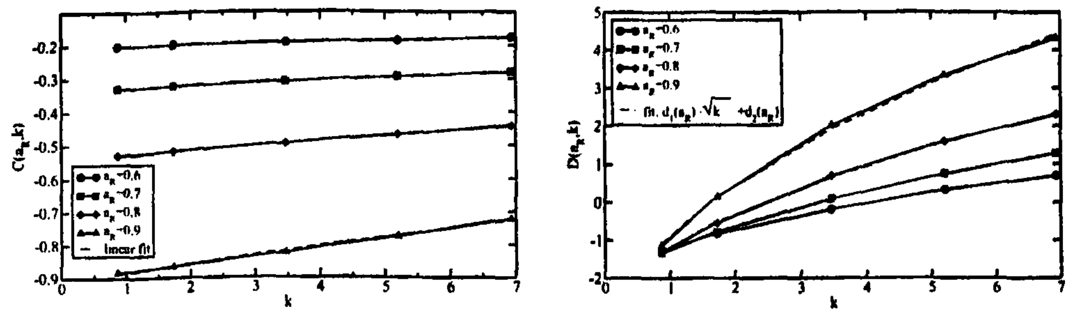


Figure 1.5: $C(a_R, k)$ (left) and $D(a_R, k)$ (right) as a function of wavenumber k for various aspect ratios a_R with the linear fit.

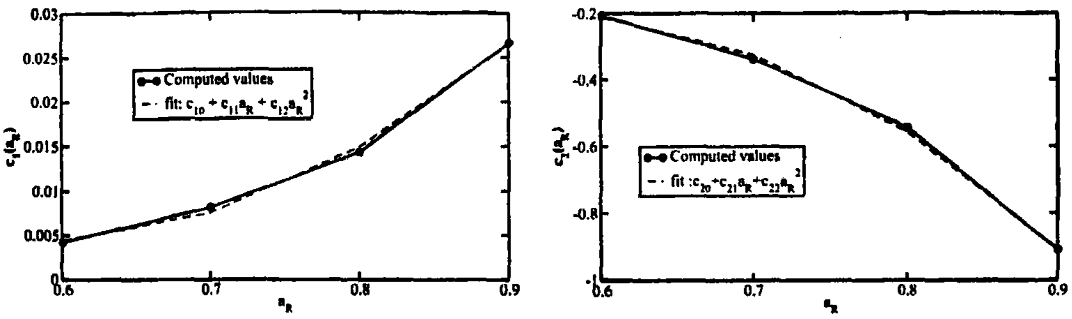


Figure 1.6: $c_1(a_R)$ (left) and $c_2(a_R)$ (right) in conjunction with the fitting function.

In Figs. 1.6 (left, right) we plot the adjusted curves with the computed values for $c_1(a_R), c_2(a_R)$.

We have followed a similar three-step procedure for the decomposition of $D(a_R, k)$, venturing to find the best fitting functions. The intercept $D(a_R, k)$ plotted in Fig. 1.5 (right) was expressed as a function of wavenumber k by the following formula:

$$D(a_R, k) = d_1(a_R)\sqrt{k} + d_2(a_R). \tag{1.34}$$

The values of the non-linear curve fit are presented in Table 1.4. We express the dependence of d_1, d_2 on a_R by

$$d_i(a_R) = d_{i0} + d_{i1}a_R + d_{i2}a_R^2, \quad i = 1, 2. \tag{1.35}$$

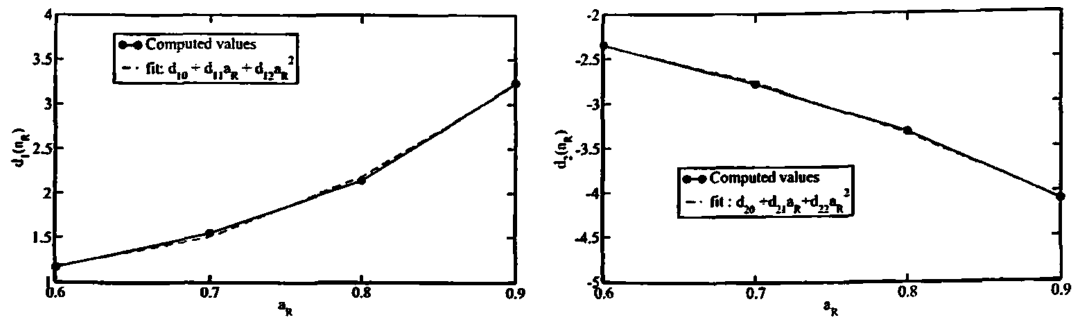


Figure 1.7: $d_1(a_R)$ left and $d_2(a_R)$ right (black) and fitting functions (red dashed).

Table 1.4: Values of $d_1(a_R)$, $d_2(a_R)$

a_R	$d_1(a_R)$	$d_2(a_R)$
0.6	1.1645	-2.3456
0.7	1.5464	-2.7846
0.8	2.1437	-3.3221
0.9	3.2320	-4.0927

The regression analysis suggested

$$\begin{aligned} d_1(a_R) &= 6.6344 - 19.6893a_R + 17.6595a_R^2 \\ d_2(a_R) &= -3.3617 + 6.656a_R - 8.2898a_R^2 \end{aligned} \tag{1.36}$$

Figures 1.7 (left, right) plot $d_1(a_R)$ and $d_2(a_R)$. Summarizing, nonlinear regression analysis suggested the following dependence of $\|\epsilon\|_{L^2}$:

$$\begin{aligned} \|\epsilon\|_{L^2} &= e^{p(N,a_R,k)}, \\ p(N,a_R,k) &= \left((0.0742 - 0.2429a_R + 0.2108a_R^2)k \right. \\ &\quad \left. - 2.0016 + 6.5219a_R - 5.8943a_R^2\right) N \\ &\quad + (6.6344 - 19.6893a_R + 17.6595a_R^2)\sqrt{k} \\ &\quad - 3.3617 + 6.656a_R - 8.2898a_R^2. \end{aligned} \tag{1.37}$$

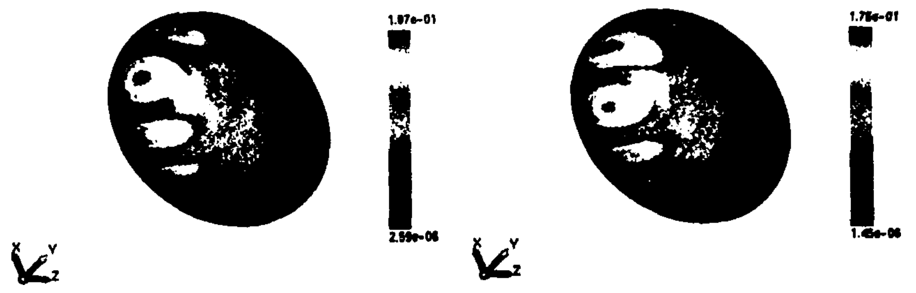


Figure 1.8: $Re(\epsilon)$ (left) and $Im(\epsilon)$ (right) at $a_R = 0.6$ and $k = 4\sqrt{3}$.

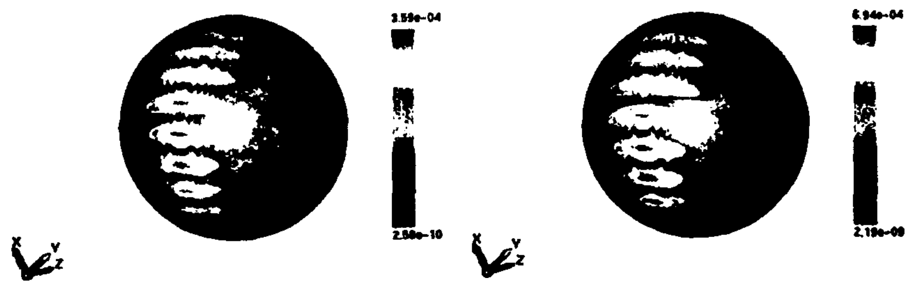


Figure 1.9: $Re(\epsilon)$ (left) and $Im(\epsilon)$ (right) at $a_R = 0.9$ and $k = 4\sqrt{3}$.

Additionally we have calculated the L^∞ norm defined by

$$\|\epsilon_N\|_{L^\infty(S)} = \operatorname{ess\,sup}_S |\epsilon_N| = \max_S |\epsilon_N|. \tag{1.38}$$

In Figs. 1.8, 1.9 (left, right) we plot both the real and imaginary parts of the error ϵ_N on the surface of the scatterer for $N = 16$, wavenumber $k = 4\sqrt{3}$ and aspect ratios $a_R = 0.6$ and $a_R = 0.9$ respectively. Those 3D plots provide a detailed description of the error distribution. We can see that the error attains it maximum value in both cases on the poles of the spheroidal scatterer, where the curvature is usually higher. Finally in Table 1.5, we provide indicative values for different aspect ratios for the real and imaginary part of the L^∞ norm, as well as for the L^2 norm of the error.

1.3.4 The far-field pattern

The far-field pattern constitutes the basic outcome of the analysis of the direct scattering problem. It is determined by Eq. (1.28), by substituting the calculated expansion coefficients provided by the preceding numerical process. In Figs.1.10–1.17 we visualize both the real and imaginary parts of the far-field pattern for the two extreme geometrical configurations $a_R = 0.6$, $a_R = 0.9$ and for wavenumbers starting from $k = 1.5\sqrt{3}$ because directivity becomes prominent from this specific value of the wavenumber. We have selected both real and imaginary parts, instead of the magnitude, to distinguish between the two different contributions. Color bars give the quantitative description of the scattered field. The direction of the incident field is shown packed together with the axes system. Both the far-field pattern and the scatterer are described on the same coordinate system and are rendered together to clarify the directionality of the scattered wave relative to the geometry of the scatterer.

Figs. 1.10–1.13 show the far-field pattern obtained by the interaction of our incident field with the prolate spheroid of semi axes ratio $a_R = 0.6$. Usually for very low frequencies (wavenumber $k \approx 0.1$) the far-field pattern exhibits an almost spherical shape, due to the fact that the incident field is not able (wavelength sensitivity) to follow the shape and curvature of the scatterer's surface. We observed this behavior by solving the forward scattering problem for wavenumbers around $k \approx 0.1$ (low frequency region). For this reason we begin in Figs. 1.10 (left, right) with wavenumber $k = 1.5\sqrt{3}$, where we distinguish a single main lobe. In Figs. 1.11 (left, right) ($k = 2\sqrt{3}$), two individual secondary lobes emerge and grow in size as wavenumber increases. This procedure leads to exaggerated twin lobes for $k = 4\sqrt{3}$ depicted in Fig. 1.13, indicating the redistribution of the scattering energy in particular favored directions. Directivity of the scattered field is observed for the imaginary part as well at different directions, suggesting that the total scattered field displays a rich pattern of preferred scattering directions.

The case of semi axes ratio $a_R = 0.9$ is presented in Figs. 1.14–1.17. Starting with wavenumber $k = 1.5\sqrt{3}$ we see as before, one main lobe both for the real and imaginary part of the far-field pattern. As the frequency increases, redistribution of energy occurs towards several favored directions, which is more than those we observed in the previous case. This is clearly expected because for $a_R = 0.9$ the prolate spheroid is hardly distinguishable from a sphere ($a_R = 1.0$), which is not true for ($a_R = 0.6$), and generally more symmetries in geometrical configurations result in energy redistribution to more preferable directions.

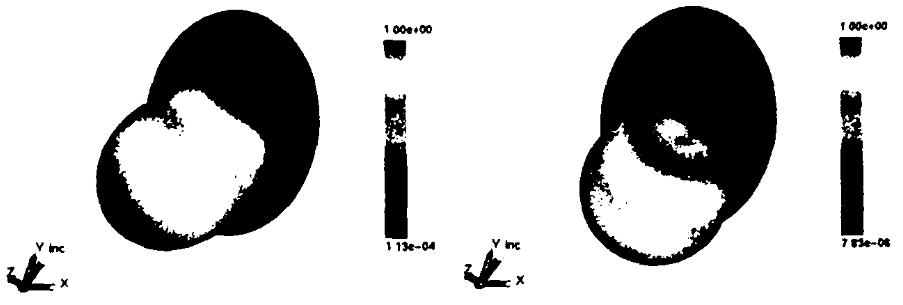


Figure 1.10: Far-field pattern Re left and Im right for $a_R = 0.6$ at $k = 1.5\sqrt{3}$.

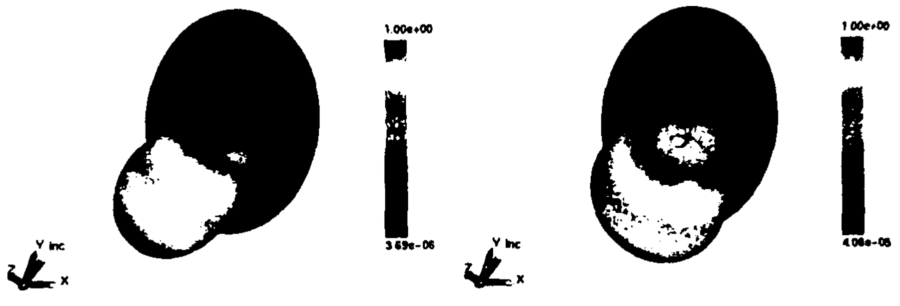


Figure 1.11: Far-field pattern Re left and Im right for $a_R = 0.6$ at $k = 2\sqrt{3}$.

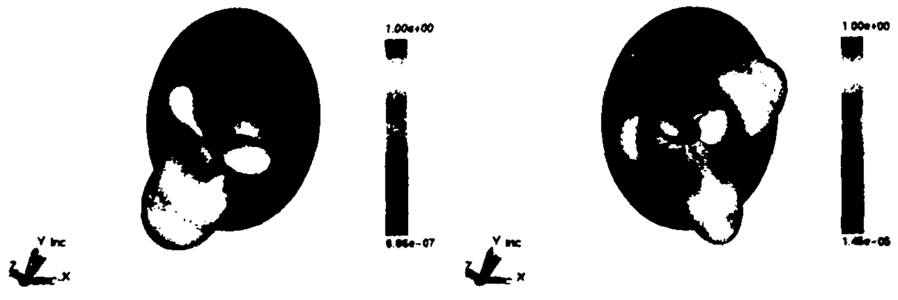


Figure 1.12: Far-field pattern Re left and Im right for $a_R = 0.6$ at $k = 3\sqrt{3}$.

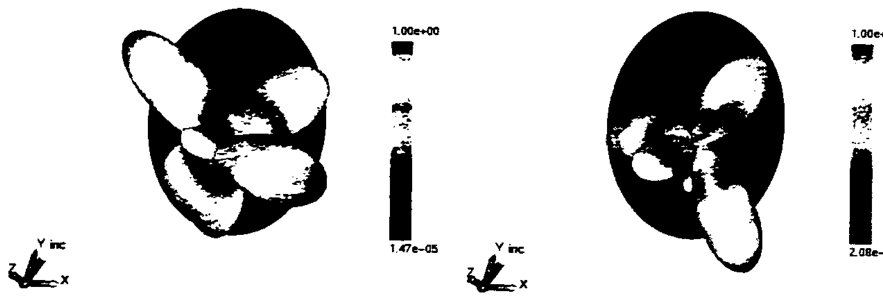


Figure 1.13: Far-field pattern Re (left) and Im (right) for $a_R = 0.6$ at $k = 4\sqrt{3}$.

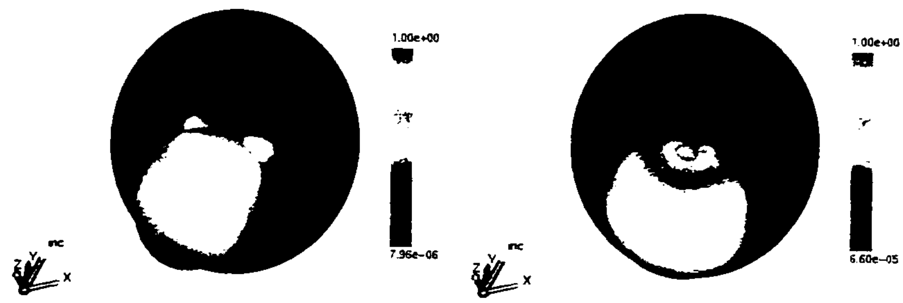


Figure 1.14: Far-field pattern Re (left) and Im (right) for $a_R = 0.9$ at $k = 1.5\sqrt{3}$.

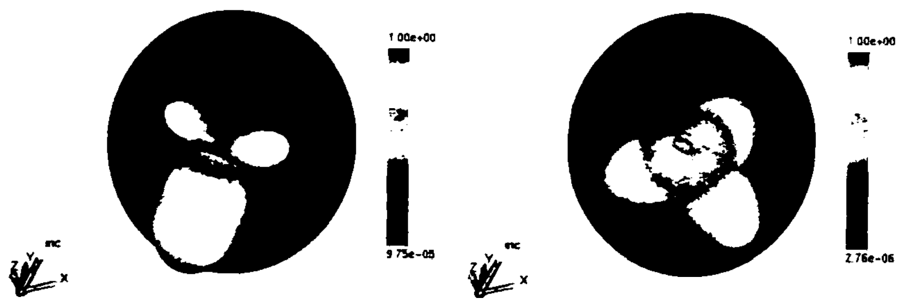


Figure 1.15: Far-field pattern Re (left) and Im (right) for $a_R = 0.9$ at $k = 2\sqrt{3}$.

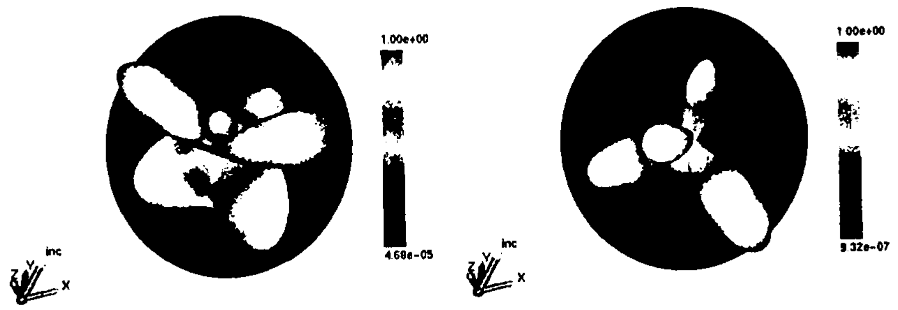


Figure 1.16: Far-field pattern Re (left) and Im (right) for $a_R = 0.9$ at $k = 3\sqrt{3}$.

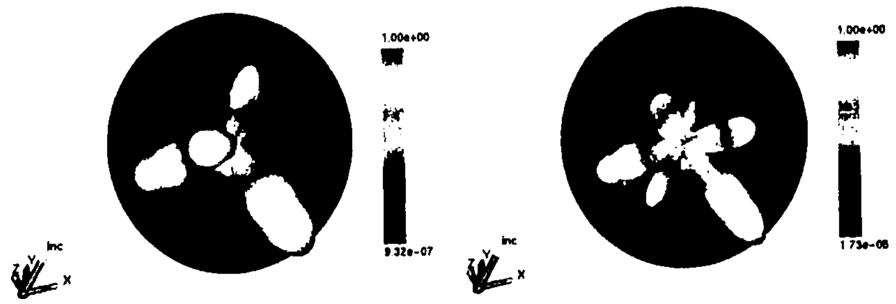


Figure 1.17: Far-field pattern Re (left) and Im (right) for $a_R = 0.9$ at $k = 4\sqrt{3}$.

Table 1.5: Error norms L^∞ and L^2 for truncation level N=16 as functions of the aspect ratio a_R and the dimensionless product kC of wave number and characteristic dimension of the scatterer. The latter being the large prolate spheroidal semi axis $C = 1$.

a_R	kC	$\|Re\{\epsilon\}\|_{L^\infty}$	$\|Im\{\epsilon\}\|_{L^\infty}$	$\|\epsilon\|_{L^2}$
0.60	$0.5\sqrt{3}$	1.240e-2	2.636e-2	1.146e-2
	$1.0\sqrt{3}$	2.730e-2	3.755e-2	1.864e-2
	$1.5\sqrt{3}$	5.352e-2	4.109e-2	2.773e-2
	$2.0\sqrt{3}$	7.019e-2	6.129e-2	3.966e-2
	$2.5\sqrt{3}$	8.042e-2	9.740e-2	5.510e-2
	$3.0\sqrt{3}$	1.031e-1	1.267e-1	7.393e-2
	$3.5\sqrt{3}$	1.472e-1	1.483e-1	9.637e-2
	$4.0\sqrt{3}$	1.865e-1	1.749e-1	1.234e-1
0.70	$0.5\sqrt{3}$	1.256e-3	3.018e-3	1.386e-3
	$1.0\sqrt{3}$	4.022e-3	4.615e-3	2.656e-3
	$1.5\sqrt{3}$	7.508e-3	6.585e-3	4.718e-3
	$2.0\sqrt{3}$	1.076e-2	1.264e-2	7.917e-3
	$2.5\sqrt{3}$	1.516e-2	1.995e-2	1.263e-2
	$3.0\sqrt{3}$	2.477e-2	2.806e-2	1.923e-2
	$3.5\sqrt{3}$	3.599e-2	3.836e-2	2.841e-2
	$4.0\sqrt{3}$	4.884e-2	5.549e-2	4.083e-2
0.80	$0.5\sqrt{3}$	4.647e-5	1.118e-4	5.766e-5
	$1.0\sqrt{3}$	1.881e-4	2.176e-4	1.470e-4
	$1.5\sqrt{3}$	3.810e-4	4.946e-4	3.418e-4
	$2.0\sqrt{3}$	7.020e-4	1.033e-3	7.257e-4
	$2.5\sqrt{3}$	1.394e-3	1.895e-3	1.433e-3
	$3.0\sqrt{3}$	2.522e-3	3.340e-3	2.686e-3
	$3.5\sqrt{3}$	4.364e-3	5.815e-3	4.819e-3
	$4.0\sqrt{3}$	7.291e-3	9.734e-3	8.285e-3
0.90	$0.5\sqrt{3}$	1.847e-7	3.900e-7	2.440e-7
	$1.0\sqrt{3}$	8.354e-7	1.519e-6	1.147e-6
	$1.5\sqrt{3}$	2.599e-6	5.411e-6	4.280e-6
	$2.0\sqrt{3}$	8.332e-6	1.617e-5	1.409e-5
	$2.5\sqrt{3}$	2.317e-5	4.680e-5	4.184e-5
	$3.0\sqrt{3}$	6.031e-5	1.188e-4	1.149e-4
	$3.5\sqrt{3}$	1.520e-4	3.018e-4	2.936e-4
	$4.0\sqrt{3}$	3.589e-4	6.941e-4	7.069e-4

Results and discussion

The sensitivity of the solution of the direct acoustic scattering problem in prolate spheroidal geometry with respect to the wavenumber and shape of the scatterer was investigated. We verified that one of the main issues is that we have to deal with extremely ill-conditioned linear systems. The systems cannot be solved with conventional 80-bit IEEE floating-point arithmetic formats and integration of arbitrary-precision software facilities is the only choice available. Iterative refinement can assist, when arbitrary-precision is integrated, in keeping the number of decimal digits required as low as possible, thereby improving the performance. The suggested analytical method, in conjunction with L^2 norm minimization of the error in satisfying the boundary condition on the surface of the scatterer, proved to be a very robust technique that could accurately handle a wide range of elongated prolate spheroidal bodies and quite high frequencies (or equivalently the induced wavenumber) of the incident wave. Our convergence study revealed, however, that for extremely elongated spheroidal bodies the convergence rate decreases. In contrast, the frequency (wavenumber) of the incident field does not affect the convergence rate but only increases the errors exponentially as it grows. This suggests that a solution to our problem for any frequency can be obtained, as long as the aspect ratio of the spheroid is such that it allows a relatively high convergence rate. The geometry of the scatterer proved to be the most crucial parameter affecting the convergence.

Chapter 2

Implicit FEM-FCT algorithms and discrete Newton methods for transient convection problems

2.1 Introduction

The advent of nonlinear high-resolution schemes for convection-dominated flows traces its origins to the *flux-corrected transport* (FCT) methodology introduced in the early 1970s by Boris and Book [23]. The fully multidimensional generalization proposed by Zalesak [168] has formed a very general framework for the design of FCT algorithms by representing them as a blend of linear high- and low-order approximations. Unlike other limiting techniques, which are typically based on geometric design criteria, flux correction of FCT type is readily applicable to finite element discretizations on unstructured meshes [110, 111]. A comprehensive summary of the state of the art can be found in [35, 96, 110, 169].

The design philosophy behind modern front-capturing methods involves a set of physical or mathematical constraints to be imposed on the discrete solution so as to prevent the formation of spurious undershoots and overshoots in the vicinity of steep gradients. To this end, the following algorithmic components are to be specified [96, 169]:

- a high-order approximation, which may fail to possess the desired properties;
- a low-order approximation, which does enjoy these properties but is less accurate;
- a way to decompose the difference between the above into a sum of skew-symmetric internodal fluxes that can be manipulated without violating mass conservation;

- a cost-effective mechanism for adjusting these antidiffusive fluxes in an adaptive fashion so that the imposed constraints are satisfied for a given solution.

Classical FCT algorithms are based on an explicit correction of the low-order solution whose local extrema serve as the upper/lower bounds for the sum of limited antidiffusive fluxes. In the case of an implicit time discretization, which gives rise to a nonlinear algebraic system, the same strategy can be used to secure the positivity of the right-hand side, whereas the left-hand side is required to satisfy the *M-matrix* property. A nonsingular discrete operator A with nonpositive off-diagonal entries $a_{ij} \leq 0$, $\forall j \neq i$ is called an M-matrix if all the coefficients of its inverse are nonnegative. Consequently, for an M-matrix, $Ax \geq 0$ implies that $x \geq 0$.

The rationale for the development of implicit FCT algorithms stems from the fact that the underlying linear discretizations must be stable. In particular, the use of an unstable high-order method may give rise to nonlinear instabilities that manifest themselves in significant distortions of the solution profiles as an aftermath of aggressive flux limiting. In the finite element context, a proper amount of streamline diffusion can be used to stabilize an explicit Galerkin scheme. However, the evaluation of extra terms increases the cost of matrix assembly and the time step must satisfy a restrictive 'CFL' condition. On the other hand, unconditionally stable implicit methods can be employed at large time steps (unless iterative solvers fail to converge or the positivity criterion is violated) and there is no need for any extra stabilization. Moreover, the overhead cost is insignificant, since the use of a consistent mass matrix leads to a sequence of linear systems even in the fully explicit case.

The generalized FEM-FCT methodology introduced in [94, 99] and refined in [97, 98] is applicable to implicit time discretizations, but the cost of iterative flux correction is rather high if the sum of limited antidiffusive fluxes and the nodal correction factors need to be updated in each outer iteration. In addition, the nonlinear convergence rates leave a lot to be desired in many cases. The use of 'frozen' correction factors computed at the beginning of the time step by the standard Zalesak limiter alleviates the convergence problems but the linearized scheme can no longer guarantee positivity. The semi-implicit limiting strategy to be described below makes it possible to overcome this problem and enforce the positivity constraint at a cost comparable to that of explicit flux correction. The resulting FEM-FCT algorithm is to be recommended for strongly time-dependent problems discretized in time by the Crank-Nicolson scheme. The design of general-purpose flux limiters that are more expensive but do not suffer from a loss of accuracy at large time steps is addressed in [95].

In this chapter, we compare a new semi-implicit FCT scheme to its semi-explicit prototype and focus on the iterative solution of the resulting nonlinear algebraic systems. As an alternative to the fixed-point defect correction scheme, which tends to converge rather slowly, a discrete Newton method tailored to the peculiarities of FEM-FCT schemes is developed. The sparse Jacobian matrix is approximated to second-order accuracy by means of divided differences and assembled edge-by-edge. The semi-implicit nature of the new FCT limiter makes the Jacobian assembly particularly efficient because the sparsity pattern of the underlying matrices is preserved. A detailed numerical study illustrates the potential of flux-corrected Galerkin schemes combined with discrete Newton methods for the treatment of nonlinearities.

2.2 Algebraic flux correction

In this chapter, we adopt an algebraic approach to the design of high-resolution schemes that consists of imposing certain mathematical constraints on discrete operators in order to achieve some favorable matrix properties. A handy algebraic criterion, which represents a multidimensional generalization of Harten's TVD theorem, was introduced by Jameson [81, 82], who proved that a semi-discrete scheme of the general form

$$\frac{\mathrm{d}u}{\mathrm{d}t} = Cu, \quad \text{with} \quad c_{ij} \geq 0 \; \forall j \neq i \quad \text{and} \quad \sum_j c_{ij} = 0, \tag{2.1}$$

is *local extremum diminishing* (LED). After the discretization in time by a two-level scheme, such methods remain positivity-preserving (PP) provided that each solution update $u^n \to u^{n+1}$ or the converged steady-state solution $u^{n+1} = u^n$ satisfies an algebraic system of the form

$$Au^{n+1} = Bu^n + f, \tag{2.2}$$

where A is an M-matrix, whereas B and f have no negative entries. Under these conditions, the positivity of the old solution carries over to the new one [96, 98]:

$$u^n \geq 0 \quad \Rightarrow \quad u^{n+1} = A^{-1}(Bu^n + f) \geq 0. \tag{2.3}$$

If the underlying spatial discretization is LED, then the off-diagonal coefficients of both matrices have the right sign, while the positivity condition $b_{ii} \geq 0$ for the diagonal entries of B yields a readily computable upper bound for admissible time steps [96]. In what follows, we discretize (2.1) in time using the standard θ-scheme, which yields $A = I - \theta\Delta tC$ and $B = I + (1-\theta)\Delta tC$. The resulting CFL-like condition for the time step Δt reads [96]

$$1 + \Delta t(1-\theta)\min_i c_{ii}^n \geq 0 \qquad \text{for} \quad 0 \leq \theta < 1. \tag{2.4}$$

The discretization is unconditionally positivity-preserving if a fully implicit time-stepping scheme

($\theta = 1$) is adopted. Of course, the above algebraic constraints are not the necessary but merely sufficient conditions for a numerical scheme to be local extremum diminishing and/or positivity preserving. In the linear case, they turn out to be far too restrictive. According to the well known Godunov theorem, linear schemes satisfying these criteria are doomed to be (at most) first-order accurate. On the other hand, a high-order discretization that fails to satisfy the imposed constraints unconditionally can be adjusted so that it admits an equivalent representation of the form (2.1) and/or (2.2), where the matrix entries may depend on the unknown solution. This idea makes it possible to construct a variety of nonlinear high-resolution schemes based on the so-called *algebraic flux correction* (AFC) paradigm. A detailed overview of this methodology is given in the survey article [96]. The design of flux limiters for finite element discretizations with a consistent mass matrix is addressed in [95].

To keep the presentation self-contained, we will follow the road map displayed in Fig. 2.1 and explain the meaning of all discrete operators in the next three sections. Roughly speaking, a high-order Galerkin discretization is to be represented in the generic form (2.2), where the matrices A and B do satisfy the above-mentioned positivity constraint. In order to guarantee that the vector f poses no hazard to positivity either, it is to be replaced by its limited counterpart f^* such that the right-hand side remains nonnegative for $u^n \geq 0$. This modification is mass-conserving provided that both f and f^* can be decomposed into skew-symmetric internodal fluxes as defined below. A family of implicit FEM-FCT schemes based on this algebraic approach was proposed in [94, 99] and combined with an iterative limiting strategy in [98]. In section 2.5.2 we present an alternative generalization of Zalesak's limiter that offers some extra advantages. The new approach to flux correction of FCT type is also based on the positivity constraint (2.2) but enforces it in another way so that the costly computation of nodal correction factors is performed just once per time step. The positivity of the resulting semi-implicit FCT algorithm is proven in section 2.5.3.

Solution strategies for the nonlinear algebraic system to be solved in each time step are presented in section 2.5.4. In particular, a discrete Newton method is proposed as a promising alternative to the standard defect correction approach. A suitable approximation to the Jacobian matrix is constructed using divided differences. In the framework of the semi-implicit FCT algorithm, this can be accomplished in a very efficient way because the correction factors are computed just once per time step in the first outer iteration. In contrast, the use of a semi-explicit FEM-FCT scheme results in an extended sparsity pattern for the Jacobian operator. The same side-effect is observed in the context of high-resolution schemes of TVD type [114].

1. Semi-discrete high-order scheme (Galerkin FEM)

$$M_C \frac{\mathrm{d}u}{\mathrm{d}t} = Ku \qquad \text{such that} \quad \forall j \neq i : \; k_{ij} < 0$$

2. Semi-discrete low-order scheme $L = K + D$

$$M_L \frac{\mathrm{d}u}{\mathrm{d}t} = Lu \qquad \text{such that} \quad l_{ij} \geq 0, \; \forall j \neq i$$

3. Nonlinear FEM-FCT algorithm $Au^{n+1} = Bu^n + f^*$,

where $A = M_L - \theta \Delta t L, \quad B = M_L + (1-\theta)\Delta t L$

Figure 2.1: Roadmap of matrix manipulations.

2.3 Semi-discrete high order scheme

As a standard model problem, consider the time-dependent continuity equation for a scalar quantity u transported by the velocity field $\mathbf{v}$, which is assumed to be known:

$$\frac{\partial u}{\partial t} + \nabla \cdot (\mathbf{v}u) = 0. \tag{2.5}$$

Let the discretization in space be performed by a (Galerkin) finite element method, which yields a DAE system for the vector of time-dependent nodal values:

$$M_C \frac{\mathrm{d}u}{\mathrm{d}t} = Ku, \tag{2.6}$$

where $M_C = \{m_{ij}\}$ denotes the consistent mass matrix and $K = \{k_{ij}\}$ is the discrete transport operator. The latter may contain some streamline diffusion used for stabilization purposes and/or to achieve better phase accuracy, e.g., in the framework of Taylor-Galerkin methods. Its skew-symmetric part $\frac{1}{2}(K - K^T)$ provides a consistent discretization of $\mathbf{v} \cdot \nabla$, whereas the symmetric part $\frac{1}{2}(K + K^T) - \text{diag}\{K\}$ represents a discrete (anti-)diffusion operator.

2.4 Semi-discrete low order scheme

In the case of linear discretizations, the algebraic constraints (2.1) and (2.2) can be readily enforced by means of 'discrete upwinding' as proposed in [94, 99]. For a semi-discrete finite element scheme of the form (2.6), the required matrix manipulations are as follows:

- replace the consistent mass matrix M_C by its lumped counterpart $M_L = \mathrm{diag}\{m_i\}$;
- render the operator K local extremum diminishing by adding an artificial diffusion operator $D = \{d_{ij}\}$ so as to eliminate all negative off-diagonal coefficients.

This straightforward 'postprocessing' transforms (2.6) into its linear LED counterpart

$$M_L \frac{\mathrm{d}u}{\mathrm{d}t} = Lu, \qquad L = K + D, \tag{2.7}$$

where D is supposed to be a symmetric matrix with zero row and column sums. For each pair of nonzero off-diagonal coefficients k_{ij} and k_{ji} of the high-order operator K, the optimal choice of the artificial diffusion coefficient d_{ij} reads [96, 99]

$$d_{ij} = \max\{-k_{ij}, 0, -k_{ji}\} = d_{ji}. \tag{2.8}$$

Alternatively, one can apply discrete upwinding to the skew-symmetric part $\frac{1}{2}(K - K^T)$ of the original transport operator K, which corresponds to

$$d_{ij} = \frac{|k_{ij} - k_{ji}|}{2} - \frac{k_{ij} + k_{ji}}{2} = d_{ji}. \tag{2.9}$$

In either case, the off-diagonal coefficients of the low-order operator $l_{ij} := k_{ij} + d_{ij}$ are nonnegative, as required by the LED criterion (2.1). Because of the zero row sum property of the artificial diffusion operator D, the diagonal coefficients of L are given by

$$l_{ii} := k_{ii} - \sum_{j \neq i} d_{ij}. \tag{2.10}$$

The semi-discretized equation for the nodal value $u_i(t)$ can be represented as

$$m_i \frac{\mathrm{d}u_i}{\mathrm{d}t} = \sum_{j \neq i} l_{ij}(u_j - u_i) + u_i \sum_j l_{ij}, \tag{2.11}$$

where $m_i = \sum_j m_{ij} > 0$ and $l_{ij} \geq 0$, $\forall i \neq j$. The last term in the above expression represents a discrete counterpart of $-u\nabla \cdot \mathbf{v}$, which is responsible for a physical growth of local extrema [96]. Recall that the operator D has zero row sums so that $u_i \sum_j l_{ij} = u_i \sum_j k_{ij}$ in (2.11). In the semi-discrete case, this term is harmless because (cf. [83])

$$u_i(t) = 0, \quad u_j(t) \geq 0, \quad \forall j \neq i \quad \Rightarrow \quad \frac{\mathrm{d}u_i}{\mathrm{d}t} \geq 0, \tag{2.12}$$

which proves that the low-order scheme (2.7) is positivity-preserving. For the fully discrete system to inherit this property, the time step should be chosen in accordance with the CFL-like condition (2.4) unless the backward Euler time-stepping ($\theta = 1$) is employed.

We would like to mention here that we have no proof for the convergence of the low order scheme to the solution of our original PDE. It does not result from some kind of consistent discretization of the PDE and contains artificial diffusion which inevitably reduces the approximation order especially in the vicinity of steep gradients. For finite element approximation we do not have currently any other alternative. A promising alternative is the one suggested by Gärtner [65] for Finite Volume schemes, provided that the techniques and ideas presented in there could be transferred to finite element discretizations.

2.5 Nonlinear FEM-FCT algorithm

The high-order system (2.6) discretized in time by a standard θ-scheme

$$[M_C - \theta\Delta t K]u^{n+1} = [M_C + (1-\theta)\Delta t K]u^n \tag{2.13}$$

admits an equivalent representation in the form (2.2) amenable to flux correction:

$$[M_L - \theta\Delta t L]u^{n+1} = [M_L + (1-\theta)\Delta t L]u^n + f(u^{n+1}, u^n). \tag{2.14}$$

The last term in the right-hand side is assembled from skew-symmetric internodal fluxes f_{ij}, which can be associated with the edges of the sparsity graph [96]:

$$f_i = \sum_{j \neq i} f_{ij}, \qquad \text{where} \quad f_{ji} = -f_{ij}. \tag{2.15}$$

Specifically, these *raw antidiffusive fluxes*, which offset the discretization error induced by mass lumping and discrete upwinding, are given by the formula [96, 98]

$$f_{ij} = [m_{ij} + \theta\Delta t d_{ij}^{n+1}]\,(u_i^{n+1} - u_j^{n+1}) - [m_{ij} - (1-\theta)\Delta t d_{ij}^{n}]\,(u_i^n - u_j^n). \tag{2.16}$$

Interestingly enough, the contribution of the consistent mass matrix consists of a truly antidiffusive implicit part and a diffusive explicit part, which has a strong damping effect. In fact, explicit mass diffusion of the form $(M_C - M_L)u^n$ has been used to construct the 'monotone' low-order method in the framework of explicit FEM-FCT algorithms [111].

In the case of an implicit time discretization ($0 < \theta \leq 1$), the nonlinearities inherent in the governing equation and/or the employed high-resolution scheme call for the use of an iterative solution strategy. Let successive approximations to the solution u^{n+1} at the new time level $t^{n+1} = t^n + \Delta t$ be computed step-by-step in the framework of a fixed-point iteration

$$u^{(m+1)} = u^{(m)} + [C^{(m)}]^{-1} r^{(m)}, \quad m = 0, 1, 2, \dots, \quad u^{(0)} = u^n, \tag{2.17}$$

where $C^{(m)}$ denotes a suitable 'preconditioner' (to be defined below) that should be easy to invert. The corresponding residual vector of the m-th outer iteration is given by

$$r^{(m)} = b^{(m)} - Au^{(m)}. \tag{2.18}$$

Here, A represents the 'monotone' evolution operator for the underlying low-order scheme

$$A = M_L - \theta \Delta t L, \qquad L = K + D, \tag{2.19}$$

which enjoys the M-matrix property because the off-diagonal entries of L are nonnegative by construction. The right-hand side $b^{(m)}$, which needs to be updated in each outer iteration, consists of a low-order part augmented by limited antidiffusion [96]:

$$b^{(m)} = Bu^n + f^*(u^{(m)}, u^n), \qquad B = M_L + (1 - \theta)\Delta t L. \tag{2.20}$$

In order to prevent the formation of nonphysical undershoots and overshoots, the raw antidiffusive fluxes f_{ij} should be multiplied by suitable correction factors so that

$$f_i^* = \sum_{j \neq i} \alpha_{ij} f_{ij}, \qquad \text{where} \qquad 0 \leq \alpha_{ij} \leq 1. \tag{2.21}$$

This adjustment transforms (2.14) into a nonlinear combination of the low-order scheme ($\alpha_{ij} \equiv 0$) and the original high-order one ($\alpha_{ij} \equiv 1$). The task of the flux limiter is to determine an optimal value of each correction factor α_{ij} individually so as to remove as much artificial diffusion as possible without violating the positivity constraint introduced in section 2.2.

In a practical implementation, the 'inversion' of the operator $C^{(m)}$ is also performed by a suitable iteration procedure for solving the sequence of linear subproblems

$$C^{(m)} \Delta u^{(m+1)} = r^{(m)}, \qquad m = 0, 1, 2, \dots. \tag{2.22}$$

After a certain number of inner iterations, the increment $\Delta u^{(m+1)}$ is applied to the last iterate

$$u^{(m+1)} = u^{(m)} + \Delta u^{(m+1)}. \tag{2.23}$$

A natural choice for the operator $C^{(m)}$, is the monotone low-order operator (2.19) so that the iteration procedure (2.17) yields the standard fixed-point defect correction scheme

$$Au^{(m+1)} = b^{(m)}, \qquad m = 0, 1, 2, \ldots. \tag{2.24}$$

Ideally, $C^{(m)}$ should be a good approximation to the Jacobian matrix $J^{(m)}$ with coefficients

$$J_{ij}^{(m)} = -\left.\frac{\partial r_i}{\partial u_j}\right|_{u=u^{(m)}} \tag{2.25}$$

evaluated at the last iterate $u^{(m)}$. It is well known that the convergence behavior of *Newton's method*, which corresponds to (2.17) with $C^{(m)} = J^{(m)}$, is quite sensitive to the initial guess $u^{(0)}$. Because linear subproblems (2.22) are solved by an iterative technique, the resulting algorithm is categorized as an inexact Newton method [42]. A simple inexact scheme is based on the following convergence criterion in each linear iteration:

$$\|J^{(m)}\Delta u^{(m+1)} - r^{(m)}\| \leq \eta \|r^{(m)}\|, \tag{2.26}$$

where the so-called forcing term $\eta \in [0, 1)$ can be chosen adaptively [55]. Furthermore, some globalization strategy may be required to enhance the robustness of Newton's method. For a detailed description of such techniques, which are mainly designed to guarantee a sufficient decrease of the nonlinear residual (2.18), the interested reader is referred to the literature, e.g., [87]. In the case of time-dependent problems, globalization is less critical because the solution from the last time step may serve as a good initial guess.

Linear subproblems (2.22) can be solved using a Krylov subspace method such as BiCGSTAB or GMRES combined with preconditioners of ILU type. Because of the M-matrix property of the evolution operator (2.19), its incomplete LU factorization unconditionally exists and is unique [113]. Hence, it is advisable to use A as preconditioner for the Krylov solver even if the Jacobian matrix (2.25) is adopted in the outer iteration procedure.

2.5.1 Semi-explicit FCT limiter

The first implicit FCT algorithm for finite element discretizations on unstructured meshes [94, 99] was based on the following limiting strategy, which was eventually superseded by further extensions proposed in a series of subsequent publications [97, 98].

1. Compute the high-order solution to (2.14) in an iterative way by solving (2.17) using the total amount of raw antidiffusion ($\alpha_{ij} \equiv 1$) to assemble the term f^*.
2. Evaluate the contribution of the consistent mass matrix to the raw antidiffusive fluxes (2.16) using the converged high-order solution as a substitute for u^{n+1}.
3. Solve the explicit subproblem $M_L\tilde{u} = Bu^n$ for the positivity-preserving intermediate solution $\tilde{u}$, which represents an explicit low-order approximation to $u(t^{n+1-\theta})$.
4. Invoke Zalesak's multidimensional FCT limiter to determine the correction factors α_{ij} so as to secure the positivity of the right-hand side as explained below.
5. Compute the final solution by solving the linear system $Au^{n+1} = b$, where

$$b_i = m_i\tilde{u}_i + \sum_{j\neq i} f_{ij}^*, \qquad f_{ij}^* = \alpha_{ij} f_{ij}. \tag{2.27}$$

In the fully explicit case ($\theta = 0$), we have $A = M_L$ so that $u^{n+1} = M_L^{-1}b$ can be computed explicitly from (2.24), and the classical FEM-FCT algorithm of Löhner et al. [110, 111] is recovered. The crux of the above generalization lies in the special choice of the operator A, which guarantees that the positivity of the right-hand side is preserved, whence

$$\tilde{u} \geq 0 \quad \Rightarrow \quad b \geq 0 \quad \Rightarrow \quad u^{n+1} = A^{-1}b \geq 0. \tag{2.28}$$

The flux correction process starts with an optional 'prelimiting' of the raw antidiffusive fluxes f_{ij}. It consists of cancelling the 'wrong' ones, which tend to flatten the intermediate solution and create numerical artifacts. The required adjustment is given by [95]

$$f_{ij}' := \max\{0, p_{ij}\}(\tilde{u}_i - \tilde{u}_j), \qquad p_{ij} = f_{ij}/(\tilde{u}_i - \tilde{u}_j). \tag{2.29}$$

The remaining fluxes are truly antidiffusive and need to be limited. The upper and lower bounds to be imposed on the net antidiffusive flux depend on the local extrema

$$\tilde{u}_i^{\max} = \max_{j\in S_i} \tilde{u}_j, \qquad \tilde{u}_i^{\min} = \min_{j\in S_i} \tilde{u}_j, \tag{2.30}$$

where $S_i = \{j \mid m_{ij} \neq 0\}$ denotes the set of nodes that share an element with node i.

In the worst case, all antidiffusive fluxes into node i have the same sign. Hence, it is worthwhile to treat the positive and negative ones separately, as proposed by Zalesak [168].

1. Evaluate the sums of all positive and negative antidiffusive fluxes into node i:

$$P_i^+ = \sum_{j\neq i} \max\{0, f'_{ij}\}, \qquad P_i^- = \sum_{j\neq i} \min\{0, f'_{ij}\}. \tag{2.31}$$

2. Compute the distance to a local maximum/minimum of the low-order solution:

$$Q_i^+ = \tilde{u}_i^{\max} - \tilde{u}_i, \qquad Q_i^- = \tilde{u}_i^{\min} - \tilde{u}_i. \tag{2.32}$$

3. Calculate the nodal correction factors, which prevent overshoots/undershoots:

$$R_i^+ = \min\{1, m_i Q_i^+/P_i^+\}, \qquad R_i^- = \min\{1, m_i Q_i^-/P_i^-\}. \tag{2.33}$$

4. Check the sign of f'_{ij} and apply $R_i^\pm$ or $R_j^\mp$, whichever is smaller, so that

$$\alpha_{ij} = \begin{cases} \min\{R_i^+, R_j^-\}, & \text{if } f'_{ij} > 0, \\ \min\{R_i^-, R_j^+\}, & \text{otherwise.} \end{cases} \tag{2.34}$$

This symmetric limiting strategy guarantees that the corrected right-hand side (2.27) satisfies the constraint $\tilde{u}_i^{\min} \leq b_i/m_i \leq \tilde{u}_i^{\max}$. Because the low-order operator A was designed to be an M-matrix, the resulting scheme proves positivity-preserving [96, 99].

It is worth mentioning that the constituents of the sums $P_i^\pm$ vary with Δt, while the corresponding upper/lower bounds $Q_i^\pm$ are fixed. Consequently, the correction factors α_{ij} produced by Zalesak's limiter depend on the underlying time step. This peculiarity of FCT methods turns out to be a blessing and a curse at the same time. On the one hand, a larger portion of the raw antidiffusive flux f_{ij} may be retained as the time step is refined. On the other hand, the accuracy of FCT algorithms deteriorates as Δt increases, because the positivity constraint (2.2) becomes too restrictive. The iterative limiting strategy proposed in [98] alleviates this problem to some extent by adjusting the correction factors α_{ij} in each outer iteration so as to recycle the rejected antidiffusion step-by-step. However, the cost of iterative flux correction is rather high, and severe convergence problems may occur. Therefore, other limiting techniques such as the general-purpose (GP) flux limiter introduced in [95] are to be preferred if the solution is expected to reach a steady state in the long run.

2.5.2 Semi-implicit FCT limiter

For truly time-dependent problems, the use of moderately small time steps is dictated by accuracy considerations so that flux limiting of FCT type is appropriate. In this case, the underlying time-stepping method should provide (unconditional) stability and be at least second-order accurate in order to capture the evolutionary details. For this reason, we favor an implicit time discretization of Crank-Nicolson type ($\theta = 1/2$) and mention the strongly A-stable fractional-step θ-scheme [152] as a promising alternative.

The semi-explicit limiting strategy presented in the previous section can be classified as an algorithm of predictor-corrector type because the implicit part of the raw antidiffusive flux (2.16) is evaluated using the converged high-order solution in place of u^{n+1}. This handy linearization, which can be traced back to the classical FEM-FCT procedure [111], makes it possible to perform flux correction in a very efficient way, as Zalesak's limiter is invoked just once per time step. However, a lot of CPU time needs to be invested in the iterative solution of the ill-conditioned high-order system and the convergence may even fail if the time step is too large. Moreover, the final solution fails to satisfy the nonlinear algebraic system (2.17) upon substitution. On the other hand, an update of the auxiliary quantities $P_i^\pm$, $Q_i^\pm$, and $R_i^\pm$ in each outer iteration would trigger the cost of flux limiting and compromise the benefits of implicit time-stepping. In order to circumvent this problem, let us introduce a semi-implicit FCT algorithm that can be implemented as follows:

- At the first outer iteration ($m = 1$), compute a set of antidiffusive fluxes $\bar{f}_{ij}$ that provide an explicit estimate for the admissible magnitude of $f_{ij}^* = \alpha_{ij} f_{ij}$.

  1. Initialize all auxiliary arrays by zeros: $P_i^\pm \equiv 0$, $Q_i^\pm \equiv 0$, $R_i^\pm \equiv 0$.

  2. Compute the positivity-preserving intermediate solution of low order:

  $$\tilde{u} = u^n + (1-\theta)\Delta t M_L^{-1} L u^n. \tag{2.35}$$

  3. For each pair of neighboring nodes i and j, evaluate the raw antidiffusive flux

  $$f_{ij}^n = \Delta t d_{ij}^n (u_i^n - u_j^n) \tag{2.36}$$

  and add its contribution to the sums of positive/negative edge contributions:

  $$P_i^\pm := P_i^\pm + \substack{\max \\ \min}\{0, f_{ij}^n\}, \qquad P_j^\pm := P_j^\pm + \substack{\max \\ \min}\{0, -f_{ij}^n\}. \tag{2.37}$$

4. Update the maximum/minimum admissible increments for both nodes:

$$Q_i^\pm := \begin{matrix}\max\\ \min\end{matrix} \{Q_i^\pm, \tilde{u}_j - \tilde{u}_i\}, \qquad Q_j^\pm := \begin{matrix}\max\\ \min\end{matrix} \left\{Q_j^\pm, \tilde{u}_i - \tilde{u}_j\right\}. \tag{2.38}$$

5. Relax the constraint $R_i^\pm \leq 1$ for the nodal correction factors and compute

$$R_i^\pm := m_i Q_i^\pm / P_i^\pm. \tag{2.39}$$

6. Multiply the raw antidiffusive fluxes f_{ij}^n by the minimum of $R_i^\pm$ and $R_j^\mp$:

$$\bar{f}_{ij} = \begin{cases} \min\{R_i^+, R_j^-\} f_{ij}^n, & \text{if } f_{ij}^n > 0, \\ \min\{R_i^-, R_j^+\} f_{ij}^n, & \text{otherwise.} \end{cases} \tag{2.40}$$

- At each outer iteration ($m = 1, 2, \dots$), assemble f^* and substitute it into (2.20).

1. Update the target flux (2.16) using the solution from the previous iteration:

$$\begin{aligned} f_{ij} &= [m_{ij} + \theta \Delta t d_{ij}^{(m)}](u_i^{(m)} - u_j^{(m)}) \\ &- [m_{ij} - (1-\theta)\Delta t d_{ij}^n]\,(u_i^n - u_j^n). \end{aligned} \tag{2.41}$$

2. Constrain each flux f_{ij} so that its magnitude is bounded by that of $\bar{f}_{ij}$:

$$f_{ij}^* = \begin{cases} \min\{f_{ij}, \max\{0, \bar{f}_{ij}\}\}, & \text{if } f_{ij} > 0, \\ \max\{f_{ij}, \min\{0, \bar{f}_{ij}\}\}, & \text{otherwise.} \end{cases} \tag{2.42}$$

3. Insert the limited antidiffusive fluxes f_{ij}^* into the right-hand side (2.20):

$$b_i^{(m)} := b_i^{(m)} + f_{ij}^*, \qquad b_j^{(m)} := b_j^{(m)} - f_{ij}^*. \tag{2.43}$$

Because f_{ij}^n is not the real target flux but merely an explicit predictor used to estimate the maximum amount of admissible antidiffusion, the multipliers $R_i^\pm$ are redefined so that the ratio $\bar{f}_{ij}/f_{ij}^n$ may exceed unity. However, the effective correction factors $\alpha_{ij} := f_{ij}^*/f_{ij}$ are bounded by 0 and 1, as required for consistency.

Instead of computing the optimal upper/lower bounds (2.32) for a given time step, it is also possible to use some reasonable fixed bounds and adjust the time step if this is necessary to satisfy a CFL-like condition (as in the case of TVD methods). For instance, the auxiliary quantities $Q_i^\pm$ can be computed using u^n instead of $\tilde{u}$:

$$Q_i^+ = \max_{j\in S_i} u_j^n - u_i^n, \qquad Q_i^- = \min_{j\in S_i} u_j^n - u_i^n. \tag{2.44}$$

The corresponding nodal correction factors $R_i^\pm$ should be redefined as [95]

$$R_i^\pm = (m_i - m_{ii})Q_i^\pm / P_i^\pm, \tag{2.45}$$

where $m_i - m_{ii} = \sum_{j\neq i} m_{ij}$ is the difference between the diagonal entries of the consistent and lumped mass matrices. This modification eliminates the need for evaluation of the intermediate solution $\tilde{u}$ in (2.35) and leads to a single-step FCT algorithm.

For a given time step, the multipliers (2.45) will typically be smaller than those defined by (2.39). However, in either case the denominator $P_i^\pm$ is proportional to Δt. Therefore, the difference between the effective correction factors α_{ij} will shrink and eventually vanish as the time step is refined. As long as Δt is sufficiently small, the accuracy of both FCT techniques depends solely on the choice of the underlying high-order scheme.

2.5.3 Positivity proof

The positivity proof for the semi-implicit FCT algorithm (2.35)–(2.43) follows that for the classical Zalesak limiter; see [96, 99]. In the nontrivial case $f_i^* \neq 0$, the i-th component of the right-hand side (2.20) admits the following representation:

$$b_i^* = m_i\tilde{u}_i + f_i^* = (m_i - \alpha_i)\tilde{u}_i + \alpha_i\tilde{u}_k, \tag{2.46}$$

where the coefficient $\alpha_i = f_i^*/(\tilde{u}_k - \tilde{u}_i)$ is defined in terms of the local extremum:

$$\tilde{u}_k = \begin{cases} \tilde{u}_i^{\max}, & \text{if } f_i^* > 0, \\ \tilde{u}_i^{\min}, & \text{if } f_i^* < 0. \end{cases} \tag{2.47}$$

This definition implies that $f_i^* = \alpha_i Q_i^\pm$, where $\alpha_i > 0$. By virtue of (2.46), the sign of the intermediate solution $\tilde{u}$ is preserved if the inequality $m_i - \alpha_i \geq 0$ holds.

In the case $f_i^* < 0$, the antidiffusive correction to node i is bounded from below by

$$m_iQ_i^- \leq R_i^-P_i^- \leq \sum_{j\neq i}\min\{0, f_{ij}\} \leq f_i^* = \alpha_iQ_i^-. \tag{2.48}$$

Likewise, a strictly positive antidiffusive correction $f_i^* > 0$ is bounded from above by

$$\alpha_i Q_i^+ = f_i^* \leq \sum_{j \neq i} \max\{0, \bar{f}_{ij}\} \leq R_i^+ P_i^+ \leq m_i Q_i^+. \tag{2.49}$$

It follows that $0 \leq \alpha_i \leq m_i$, which proves that $b_i^* \geq 0$ provided $\tilde{u}_i \geq 0$ and $\tilde{u}_k \geq 0$.

In light of the above, the semi-implicit FCT limiter is positivity-preserving as long as the diagonal coefficients of the matrix B as defined in (2.20) are nonnegative. The corresponding CFL-like condition (2.4) for the maximum admissible time step reads

$$(1-\theta)\Delta t \leq \min_i |m_i / l_{ii}|. \tag{2.50}$$

The positivity of the single-step algorithm based on the slack bounds (2.44)–(2.45) can be proven in a similar way using the following representation of the right-hand side:

$$b_i^* = (m_i - \alpha_i) u_i^n + \alpha_i u_k^n + (1-\theta)\Delta t \sum_j l_{ij} u_j^n. \tag{2.51}$$

In this case, the limited antidiffusive correction to node i can be estimated as follows:

$$(m_i - m_{ii}) Q_i^- \leq f_i^* \leq (m_i - m_{ii}) Q_i^+, \tag{2.52}$$

so that $m_i - \alpha_i \geq m_{ii}$. Thus, the right-hand side given by (2.51) preserves the sign of u^n if the time step satisfies the positivity constraint for all diagonal coefficients:

$$(1-\theta)\Delta t \leq \min_i |m_{ii} / l_{ii}|. \tag{2.53}$$

Under the above conditions, the M-matrix property of the low-order operator (2.19) is sufficient to guarantee that *each* solution update is positivity-preserving if the fixed-point iteration (2.17) is preconditioned by $C^{(m)} = A, \ \forall m$. On the other hand, only the fully converged solution is certain to remain positive if Newton's method ($C^{(m)} = J^{(m)}$) is employed.

2.5.4 Approximation of jacobians

For the practical application of Newton's method, it remains to devise an algorithm for the construction of the Jacobian matrix (2.25). For simplicity, superscript m will be omitted unless indicated otherwise. In what follows, differentiation is to be performed with respect to u, whereas u^n is regarded as a given constant. The nonlinear residual (2.18) depends on the 'monotone' operator A and on the right-hand side $b^{(m)}$ given by relations (2.19) and (2.20), respectively. Hence, it is advisable to split the Jacobian operator $J = \bar{J} + J^*$ into its 'upwind' part $\bar{J} = \{\bar{J}_{ij}\}$ and the contribution of the antidiffusive correction $J^* = \{J^*_{ij}\}$.

Formally, the upwind Jacobian is given by $\bar{J} = A'u + A$, which reduces to the M-matrix $A = M_L - \theta\Delta t L$ for linear model problems of the form (2.5). On the other hand, its derivative A' does not vanish if the original transport operator $K(u)$ and hence its local extremum diminishing counterpart $L(u) = K(u) + D(u)$ depend on the unknown solution. Since the artificial diffusion operator $D(u)$ was derived on the semi-discrete level by means of matrix manipulations, no analytical expression is available for $A'(u)$. As a remedy, the Jacobian matrix is approximated by means of divided differences. To this end, let $f : \mathbb{R}^n \to \mathbb{R}$ denote a generic function for which the central divided difference operator is defined as follows:

$$\mathscr{D}_k[f(u)] := \frac{f(u + \sigma e_k) - f(u - \sigma e_k)}{2\sigma}. \tag{2.54}$$

In the above equation, e_k denotes the k-th unit vector. The optimal choice of the perturbation parameter $0 < \sigma \ll 1$ requires some knowledge of the sensitivity of the function f and has been addressed by many authors. Following a strategy proposed by Nielsen et al. [116] it can be chosen proportional to the square root of the machine precision ϵ. The formula

$$\sigma = [\epsilon(1 + \|u\|)]^{\frac{1}{3}} \tag{2.55}$$

suggested in [123] takes into account the norm of the solution and is claimed to reduce the noise arising from the numerical evaluation of the function f. Some alternative choices are given in a survey paper on Jacobian-free Newton-Krylov methods by Knoll and Keyes [90].

The central divided-difference operator introduced in (2.54) yields a second-order accurate approximation for each entry of the upwind Jacobian $\bar{J}$:

$$\bar{J}_{ik} = \mathscr{D}_k[(Au)_i] + \mathcal{O}(\sigma^2). \tag{2.56}$$

Since $A = M_L - \theta\Delta t L$, the so-defined coefficient $\bar{J}_{ik}$ can be cast into the form [114]

$$\bar{J}_{ik} = \delta_{ik} m_i - \theta\Delta t\left(\bar{l}_{ik} + \sum_j \mathscr{D}_k[l_{ij}]u_j\right), \tag{2.57}$$

where $\delta_{ik} \in \{0, 1\}$ denotes the standard Kronecker delta symbol and the auxiliary quantity $\bar{l}_{ik}$ stands for the average of the perturbed evolution coefficients resulting from discrete upwinding:

$$\bar{l}_{ik} = \frac{l_{ik}(u + \sigma e_k) + l_{ik}(u - \sigma e_k)}{2}. \tag{2.58}$$

It is also possible to neglect averaging in (2.57) and replace the averaged transport operator $\bar{L}$ by the standard low-order term L. If the problem at hand is linear, the upwind Jacobian reduces to $\bar{J} = M_L - \theta\Delta t L$ because all divided differences vanish. It is worth mentioning that the decomposition into individual edge contributions yields an efficient assembly procedure for the Jacobian that can be considered as a viable alternative to the element-by-element procedure traditionally employed in the finite element community [31].

Interestingly enough, the operator $\bar{J}$ exhibits the same sparsity pattern as the finite element matrix, that is, $\bar{J}_{ij} \neq 0$ implies $m_{ij} \neq 0$. As soon as algebraic flux correction comes into play, this amenable property may be lost. For upwind-biased discretization schemes of TVD type, the phenomenon of matrix fill-in engendered by the flux limiter has been analyzed in section 4.2 of publication [114]. Because of conceptional similarities within the family of AFC schemes, essentially the same analysis remains valid for symmetric flux limiters [95, 96]. As we are about to see, the semi-implicit FCT algorithm presented in section 2.5.2 is free of this drawback and hence particularly suitable for the application of Newton's method.

To highlight this advantage of the semi-implicit approach, let us revisit the general approach to evaluating of the antidiffusive contribution J^* to the Jacobian operator. To this end, let the solution vector u be perturbed at some node, say k, and compute the compensating antidiffusive fluxes (2.21) based on the solution vectors $u + \sigma e_k$ and $u - \sigma e_k$ following the semi-explicit limiting algorithm presented in section 2.5.1. By construction, the nodal correction factors $R_i^\pm$ defined in (2.33) may be affected by this perturbation for all i from the set S_k of nodes that share an element with node k. Consequently, the final correction factors α_{ij} defined in (2.34) may have different values if at least one of the nodes i and j belongs to the set S_k. To put it in a nutshell, the impact of 'jiggling' the solution value at one particular node usually propagates along two edges ij by virtue of the correction factors α_{ij}, which are recalculated at each iteration. Figure 2.2 illustrates the difference between the sparsity pattern of the finite element matrix and that of the approximate Jacobian operator constructed as explained above.

As demonstrated in [114], the connectivity graph of the Jacobian is known *a priori* and can be directly constructed from that of the stiffness matrix by means of symbolic matrix multiplication. Because of the matrix fill-in, the amount of memory required to store the much denser Jacobian increases considerably and so does the computational cost of linear algebra operations such as matrix-vector multiplication, construction of an ILU decomposition, etc.

Interestingly, it turns out that the antidiffusive Jacobian operator J^* for the semi-implicit FCT algorithm presented in section 2.5.2 inherits the sparsity pattern of the finite element matrix. A closer look at equations (2.35)–(2.40) reveals that the initialization of the antidiffusive fluxes $\tilde{f}$ in the first outer iteration ($m = 1$) does not rely on the dependent variable $u^{(m)}$. Hence, the explicit

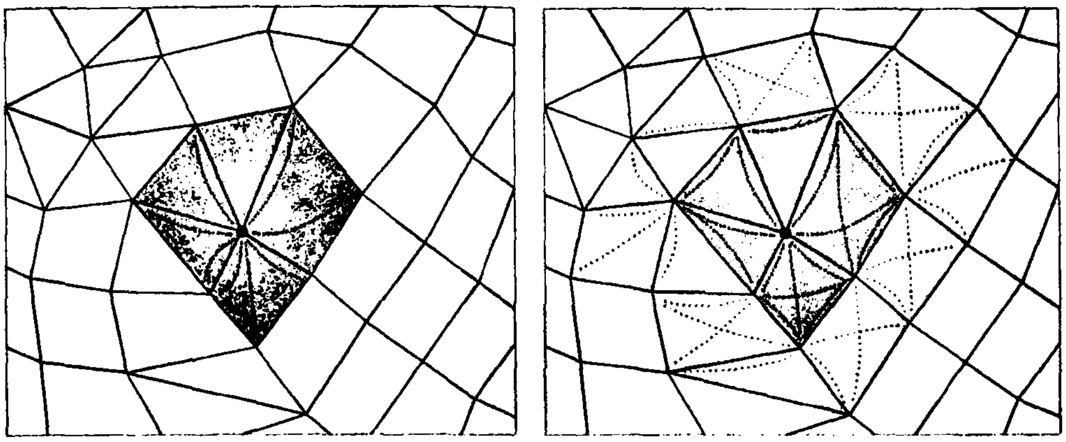


Figure 2.2: Sparsity pattern: Finite element matrix *vs.* Jacobian operator.

estimate for the admissible antidiffusion, which is already computed in the residual assembly, can be also adopted for the construction of the Jacobian. In essence, the target fluxes (2.16) that need to be updated at each iteration are constrained such that their magnitude is bounded by that of $\tilde{f}$. Let us emphasize the fact that the correction factors α_{ij} are only defined implicitly and hence will not entail a widening of the matrix stencil. As a result, the Jacobian part J^* can be assembled in a rather efficient way.

As a rule of thumb, the k-th column of J^* can be assembled by performing the algorithmic steps (2.41) and (2.42) based on the perturbed solution vectors $u + \sigma e_k$ and $u - \sigma e_k$ to evaluate the corresponding fluxes $f^{\overset{*}{+}} = f^*(u + \sigma e_k)$ and $f^{\overset{*}{-}} = f^*(u - \sigma e_k)$ and scale their difference by 2σ. However, this approach is quite expensive because it does not exploit the sparsity of the Jacobian, which is inherited from the global evolution operator A. In order to circumvent this problem, let us introduce an efficient algorithm for assembling the operator J^* for the semi-implicit FCT limiter in an edge-based fashion.

- At each outer iteration ($m = 1, 2, \dots$), initialize J^* to zero and rebuild it in a loop over edges ij. This process involves the following steps to be performed for $k = i$ and $k = j$:

  1. Evaluate the explicit antidiffusive contribution and the solution difference:

$$f_{ij}^n = [m_{ij} - (1-\theta)\Delta t d_{ij}^n](u_i^n - u_j^n), \qquad \Delta u_{ij}^{(m)} = u_i^{(m)} - u_j^{(m)}. \tag{2.59}$$

  2. Compute auxiliary coefficients using the perturbed solution $u^{(m)} \pm \sigma e_k$:

$$z_{ij,k}^+ = m_{ij} + \theta\Delta t d_{ij}(u^{(m)} + \sigma e_k), \qquad z_{ij,k}^- = m_{ij} + \theta\Delta t d_{ij}(u^{(m)} - \sigma e_k). \tag{2.60}$$

3. Update the perturbed target fluxes (2.16) depending on the index k:

$$f^{+}_{ij,k} = \begin{cases} z^{+}_{ij,k}[\Delta u^{(m)}_{ij} + \sigma] + f^{n}_{ij} & \text{if } k = i, \\ z^{+}_{ij,k}[\Delta u^{(m)}_{ij} - \sigma] + f^{n}_{ij} & \text{otherwise,} \end{cases} \tag{2.61}$$

$$f^{-}_{ij,k} = \begin{cases} z^{-}_{ij}[\Delta u^{(m)}_{ij} - \sigma] + f^{n}_{ij} & \text{if } k = i, \\ z^{-}_{ij}[\Delta u^{(m)}_{ij} + \sigma] + f^{n}_{ij} & \text{otherwise.} \end{cases} \tag{2.62}$$

4. Constrain each flux $f^{\pm}_{ij,k}$ so that its magnitude is bounded by that of $\tilde{f}_{ij}$:

$$f^{+*}_{ij,k} = \begin{cases} \min\{f^{+}_{ij,k}, \max\{0, \tilde{f}_{ij}\}\}, & \text{if } f^{+}_{ij} > 0, \\ \max\{f^{+}_{ij,k}, \min\{0, \tilde{f}_{ij}\}\}, & \text{otherwise,} \end{cases} \tag{2.63}$$

$$f^{-*}_{ij,k} = \begin{cases} \min\{f^{-}_{ij,k}, \max\{0, \tilde{f}_{ij}\}\}, & \text{if } f^{-}_{ij} > 0, \\ \max\{f^{-}_{ij,k}, \min\{0, \tilde{f}_{ij}\}\}, & \text{otherwise.} \end{cases} \tag{2.64}$$

5. Compute the divided difference and insert it into the k-th column of the Jacobian:

$$f^{*}_{ij,k} = \frac{1}{2\sigma}\left(f^{+*}_{ij,k} - f^{-*}_{ij,k}\right), \qquad J^{*}_{ik} := J^{*}_{ik} - f^{*}_{ij,k}, \quad J^{*}_{jk} := J^{*}_{jk} + f^{*}_{ij,k}. \tag{2.65}$$

The last three steps call for further explanation. Following expression (2.41), the implicit contribution of the target fluxes (2.61)–(2.62) is multiplied by the perturbed solution difference $(u \pm \sigma e_k)_i - (u \pm \sigma e_k)_j$, where k equals i or j. This yields four possible combinations $u_i \pm \sigma\delta_{ik} - u_j \mp \sigma\delta_{jk}$ for $j \neq i$ that need to be multiplied by the coefficients $z^{\pm}_{ij,k}$. The magnitude of the raw antidiffusive fluxes $f^{\pm}_{ij,k}$ is bounded by that of the explicit estimate $\tilde{f}_{ij}$. In the last step, the central difference of the limited fluxes is inserted into the k-th column of the Jacobian matrix. Following step (2.43) of the original algorithm, node j receives the same flux as node i but with opposite sign. Note that the antidiffusive fluxes are now applied to the left-hand side (2.65) so that the signs for nodes i and j are reversed.

The above algorithm is applicable to linear and nonlinear transport operators alike. It is worth mentioning that in the linear case the artificial diffusion coefficient d_{ij} does not depend on the solution so that the auxiliary quantity $z_{ij} \equiv m_{ij} + \theta\Delta t d_{ij}$ is not affected by solution variations. Moreover, the perturbed fluxes exhibit the following symmetry property:

$$f^{+}_{ij,i} = z_{ij}[\Delta u^{(m)}_{ij} + \sigma] + f^{n}_{ij} = f^{-}_{ij,j}, \tag{2.66}$$

$$f^{-}_{ij,i} = z_{ij}[\Delta u^{(m)}_{ij} - \sigma] + f^{n}_{ij} = f^{+}_{ij,j}. \tag{2.67}$$

As a consequence, the final flux that is inserted into the i-th column is also applied to column number j but with opposite sign. That is, the following skew-symmetry holds:

$$f^*_{ij,i} = \frac{1}{2\sigma}\left(f^{+*}_{ij,i} - f^{-*}_{ij,i}\right) = -f^*_{ij,j}. \tag{2.68}$$

Hence, it suffices to compute the divided difference (2.65) only for one index, say $k = i$, and update the four coefficients of the Jacobian simultaneously according to

$$\begin{aligned} J^*_{ii} &:= J^*_{ii} - f^*_{ij,i}, \qquad & J^*_{ji} &:= J^*_{ji} + f^*_{ij,i}, \\ J^*_{jj} &:= J^*_{jj} - f^*_{ij,i}, \qquad & J^*_{ij} &:= J^*_{ij} + f^*_{ij,i}. \end{aligned} \tag{2.69}$$

Roughly speaking, the calculation of the operator J^* is approximately twice as expensive as augmenting the right-hand side (2.20) by the antidiffusive fluxes making use of the semi-explicit FCT algorithm (2.41)–(2.43). As we are about to see, this extra cost clearly pays off in terms of total efficiency when it comes to time accurate simulation of transient flows. Remarkably, this improvement is already observed if the evolution operator $A = M_L - \theta\Delta t L$ is constant and can be assembled once and for all at the beginning of the simulation so that the standard defect correction approach (2.24) does not require further matrix evaluations. The benefits of Newton's method become even more significant if the preconditioner (2.19) needs to be updated in each outer iteration because of a nonlinear governing equation or a linear but time-dependent velocity field $\mathbf{v} = \mathbf{v}(\mathbf{x}, t)$, so that the costs for assembling the operators J and J^* may be neglected.

2.5.5 Convergence

A remark is in order regarding the convergence behavior of the fixed-point iteration (2.17). The converged solution u^{n+1} is supposed to satisfy a nonlinear algebraic system of the form

$$A^* u^{n+1} = B u^n, \tag{2.70}$$

where A^* is the nonlinear FCT operator, which includes some built-in antidiffusion:

$$A^* u^{n+1} := A u^{n+1} - f^*. \tag{2.71}$$

Clearly, the rate of convergence will depend on $\|A^* - C\|$, that is, the approximation property of the preconditioner C. On the one hand, the operator A as defined in (2.19) is linear and easy to 'invert' because it is an M-matrix. On the other hand, it represents a rather poor approximation to the original Galerkin operator $M_C - \theta\Delta t K$, which is recovered in the limit $\alpha_{ij} \to 1$. As a result, the convergence of a highly accurate FCT algorithm based on the standard defect correction approach is likely to slow down as the high-order solution is approached.

In light of the above, the lumped-mass version, which is obtained by setting $m_{ij} = 0$ in the definition of the raw antidiffusive flux, converges much faster than the one based on the consistent target flux (2.16). However, mass lumping may have a devastating effect on the accuracy of a time-dependent solution, as demonstrated by the numerical study performed in the next section. At the same time, the high phase accuracy provided by the consistent mass matrix comes at the cost of slower convergence, because the 'monotone' preconditioner A is based on M_L rather than M_C. The original high-order system (2.13) that corresponds to $\alpha_{ij} \equiv 1$ is particularly difficult to solve, even though it is linear (see below). Moreover, the number of outer iterations tends to increase as the mesh is refined.

In general, there is a trade-off between the accuracy of the numerical solution and convergence of the fixed-point iteration (2.24). Any modification of the flux limiter that makes it possible to accept more antidiffusion has an adverse effect on the nonlinear convergence rates. Conversely, more diffusive schemes converge much better but their accuracy leaves a lot to be desired. To overcome this shortcoming, the use of the discrete Newton method is advisable. The number of outer iterations required to drive the residual to some prescribed tolerance is drastically reduced and becomes largely independent of the grid refinement level. However, one should keep in mind that the Jacobian matrix (2.25) does not possess the M-matrix property so that intermediate solutions are not necessarily positivity-preserving.

2.6 Numerical examples

In order to evaluate the performance of the new algorithm, we apply it to several time-dependent benchmark problems discretized using the standard Galerkin method and the second-order accurate Crank-Nicolson time-stepping. After flux limiting, the order of approximation (in space and time) may vary depending on the local smoothness of the solution. The goal of this numerical study is to examine the accuracy of the resulting high-resolution scheme as well as the convergence behavior of the fixed-point iteration (2.17) and the implications of mass lumping. To this end, the semi-implicit FCT method (2.35)–(2.43) is compared with its semi-explicit prototype (2.29)–(2.34) and to the standard Galerkin discretization. Moreover, the standard defect correction scheme (2.24) and the discrete Newton approach are compared with respect to their nonlinear convergence rates as well as computational efficiency, that is, the total CPU time required to solve the nonlinear algebraic system (2.14) up to a prescribed tolerance.

2.6.1 Convection skew to the mesh

In order to study the convergence behavior of the semi-implicit and semi-explicit FEM-FCT algorithms as compared to that of the underlying Galerkin scheme, let us solve equation (2.5) with $\mathbf{v} = (1, 1)$ so that the initial profile is translated along the diagonal of the computational domain $\Omega = (0, 1) \times (0, 1)$. The numerical study is to be performed for two different initial configurations centered at the reference point $(x_0, y_0) = (0.3, 0.3)$.

TP1 The first test problem corresponds to the discontinuous initial condition

$$u(x,y,0) = \begin{cases} 1 & \text{if } \max\{|x-x_0|, |y-y_0|\} \le 0.1, \\ 0 & \text{otherwise.} \end{cases} \tag{2.72}$$

TP2 The second test problem deals with translation of a smooth function defined as

$$u(x,y,0) = \frac{1}{4}[1+\cos(10\pi(x-x_0))][1+\cos(10\pi(y-y_0))] \tag{2.73}$$

within the circle $\sqrt{(x-x_0)^2+(y-y_0)^2} \le 0.1$ and equal to zero elsewhere.

Figures 2.3–2.4 display the approximate solutions at time $t = 0.5$ computed using $\Delta t = 10^{-3}$ on a quadrilateral mesh consisting of 128×128 bilinear elements. The upper diagrams were produced by the consistent-mass semi-implicit FCT algorithm, which yields nonoscillatory solutions bounded by 0 and 1. The underlying high-order scheme remains stable but gives rise to nonphysical undershoots and overshoots, as seen in the lower diagrams.

In either case, the numerical solution was computed in an iterative way using the fixed-point defect correction scheme (2.17) preconditioned by the low-order operator (2.19). The stopping criterion was based on the Euclidean norm of the residual vector

$$r = Au^{n+1} - Bu^n - f^*, \qquad \|r\| = \sqrt{r^T r}, \tag{2.74}$$

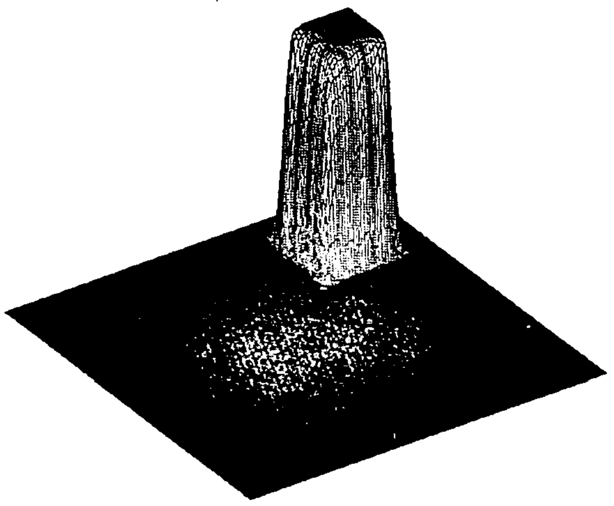
which was required to satisfy the inequality $\|r\| \le 10^{-4}$. The difference between the exact solution u and its finite element approximation u_h was measured in the L_1-norm,

$$\|u-u_h\|_1 = \int_\Omega |u-u_h|\,\mathrm{d}x \approx \sum_i m_i |u(x_i,y_i) - u_i|, \tag{2.75}$$

as well as in the L_2-norm defined by the following formula:

$$\|u-u_h\|_2^2 = \int_\Omega |u-u_h|^2\,\mathrm{d}x \approx \sum_i m_i |u(x_i,y_i) - u_i|^2, \tag{2.76}$$

FEM-FCT



Galerkin scheme

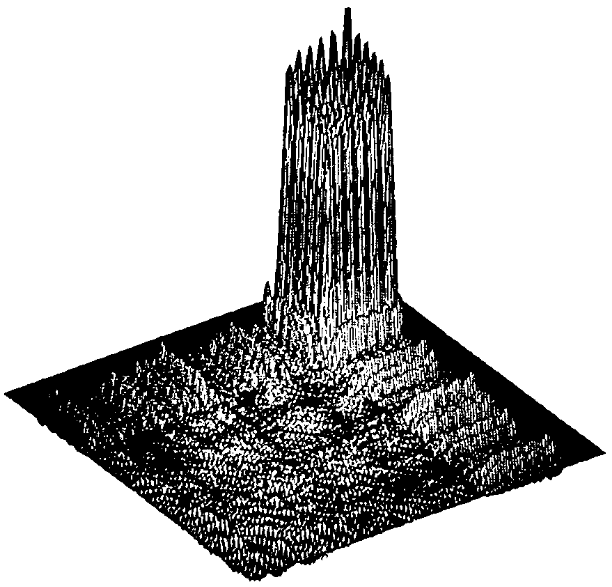


Figure 2.3: Convection skew to the mesh TP1: 128 × 128 Q_1−elements, $t = 0.5$.

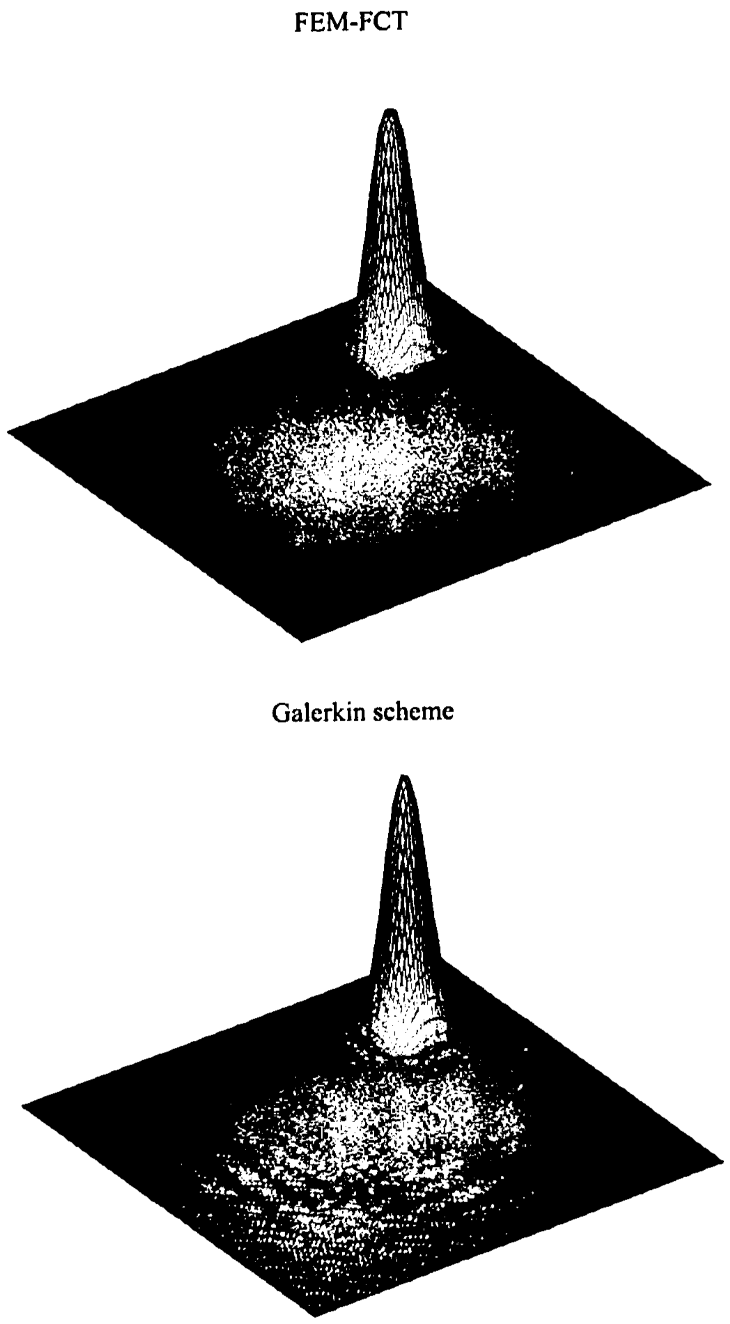


Figure 2.4: Convection skew to the mesh TP2: 128 × 128 Q_1−elements, $t = 0.5$.

where $m_i = \int_\Omega \varphi_i \, dx$ are the diagonal coefficients of the row-sum lumped mass matrix. Furthermore, the global minimum $u_{\min} = \min_i u_i$ and maximum $u_{\max} = \max_i u_i$ of the discrete solution u_h were compared to their analytical values 0 and 1.

Tables 2.1 and 2.2 illustrate the convergence behavior of the iterative flux/defect correction scheme as applied to the test problems TP1 and TP2 on three successively refined meshes. The first three columns in each table display the refinement level NLEV, the number of vertices/nodes NVT, and the total number of outer iterations NDC required to compute the numerical solution at $t = 0.5$. The different performance of the six algorithms under consideration supports the arguments

standard defect correction

NLEV	NVT	NDC	$\|u - u_h\|_1$	$\|u - u_h\|_2$	$u_{\min}$	$u_{\max}$
semi-implicit FCT / consistent mass matrix						
6	4,225	2,500	1.1737e-2	6.2176e-2	0.0	1.0
7	16,641	2,461	7.3688e-3	4.8577e-2	0.0	1.0
8	66,049	2,489	4.7039e-3	3.8715e-2	0.0	1.0
semi-implicit FCT / lumped mass matrix						
6	4,225	751	1.9356e-2	8.4294e-2	0.0	0.9988
7	16,641	1,000	1.2402e-2	6.5356e-2	0.0	1.0000
8	66,049	1,014	7.8511e-3	5.1182e-2	0.0	1.0000
Galerkin scheme / consistent mass matrix						
6	4,225	4,666	3.6283e-2	7.4952e-2	-0.2557	1.4505
7	16,641	7,379	2.7340e-2	5.8124e-2	-0.2743	1.3797
8	66,049	13,852	2.3000e-2	5.2536e-2	-0.4437	1.4080
Galerkin scheme / lumped mass matrix						
6	4,225	1,000	6.5181e-2	1.3073e-1	-0.4022	1.5608
7	16,641	1,423	4.7055e-2	9.8663e-2	-0.4340	1.5732
8	66,049	1,500	3.5126e-2	8.0298e-2	-0.3713	1.5628
semi-explicit FCT / consistent mass matrix						
6	4,225	3,190	9.3328e-3	5.4115e-2	0.0	1.0
7	16,641	3,220	5.4794e-3	4.1218e-2	0.0	1.0
8	66,049	3,590	3.3680e-3	3.2369e-2	0.0	1.0
semi-explicit FCT / lumped mass matrix						
6	4,225	1,500	1.9098e-2	8.3498e-2	0.0	0.9989
7	16,641	1,501	1.2422e-2	6.5348e-2	0.0	1.0000
8	66,049	1,540	7.8662e-3	5.1167e-2	0.0	1.0000

Table 2.1: Convection skew to the mesh: Convergence behavior for TP1.

presented in section 2.5.5. In particular, it can readily be seen that the use of the consistent mass matrix results in a much better accuracy but the convergence slows down, whereas the lumped-mass version is less accurate but much more efficient. If the difference $\|u^{n+1} - u^n\|$ is large, mass antidiffusion affects the convergence rates even stronger than the convective part of the antidiffusive flux. Since the latter is proportional to Δt, the mass lumping error plays a dominant role at small time steps such that $A \approx M_L$. On the other hand, the linear convergence rates improve since the condition number of A decreases and its diagonal dominance is enhanced as the time step is refined.

standard defect correction

NLEV	NVT	NDC	$\|u - u_h\|_1$	$\|u - u_h\|_2$	$u_{\min}$	$u_{\max}$
semi-implicit FCT / consistent mass matrix						
6	4,225	2,486	1.4799e-3	9.2813e-3	0.0	0.8562
7	16,641	1,833	4.3436e-4	2.7820e-3	0.0	0.9418
8	66,049	2,867	1.7887e-4	1.2032e-3	0.0	0.9740
semi-implicit FCT / lumped mass matrix						
6	4,225	1,000	4.2704e-3	2.7827e-2	0.0	0.7308
7	16,641	1,000	1.7834e-3	1.1294e-2	0.0	0.9218
8	66,049	736	7.6982e-4	4.6142e-3	0.0	0.9612
Galerkin scheme / consistent mass matrix						
6	4,225	2,500	1.3961e-3	2.6234e-3	-0.0158	0.9890
7	16,641	6,437	1.8892e-3	3.9001e-3	-0.0480	0.9925
8	66,049	13,700	2.3237e-3	8.1553e-3	-0.1363	1.0012
Galerkin scheme / lumped mass matrix						
6	4,225	1,000	1.0904e-2	4.2409e-2	-0.1911	0.8809
7	16,641	1,000	3.4837e-3	1.4234e-2	-0.0811	1.0098
8	66,049	1,000	1.3092e-3	4.3179e-3	-0.0322	1.0046
semi-explicit FCT / consistent mass matrix						
6	4,225	2,651	1.0770e-3	7.6799e-3	0.0	0.8555
7	16,641	2,328	2.8414e-4	2.1692e-3	0.0	0.9471
8	66,049	3,434	1.3188e-4	9.8597e-4	0.0	0.9775
semi-explicit FCT / lumped mass matrix						
6	4,225	1,500	4.2671e-3	2.7760e-2	0.0	0.7296
7	16,641	1,500	1.7751e-3	1.1237e-2	0.0	0.9211
8	66,049	1,500	6.4767e-4	3.8591e-3	0.0	0.9653

Table 2.2: Convection skew to the mesh: Convergence behavior for TP2.

Note that the consistent-mass Galerkin scheme faces severe convergence problems and the error may even increase in the course of mesh refinement (see Table 2.2). By contrast, the results computed by the semi-implicit FCT algorithm exhibit monotone grid convergence as well as some improvement of the convergence rates. Even the consistent-mass version converges slowly but surely to a nonoscillatory time-accurate solution. For large time steps, the single-step implementation based on (2.44)–(2.45) would be more diffusive and converge faster. However, for time steps as small as the one employed in this section, it would be just as accurate and converge at the same rate as the algorithm (2.35)–(2.43). The values of $u_{\max}$ in Table 2.2 reveal that flux correction may lead to undesirable 'peak clipping', which is a well known phenomenon discussed, e.g., in [96, 168]. On the other hand, the associated high-order solution is corrupted by undershoots and overshoots that are particularly large for discontinuous initial data (Table 2.1) and less pronounced for the smooth cosine hill (Table 2.2). These nonphysical oscillations result in a dramatic loss of accuracy and slow/no convergence, so that the results are inferior to those produced by the semi-implicit FCT algorithm using the same settings.

It is not unusual that semi-explicit flux correction (2.29)–(2.34) as applied at the end of each time step to the converged high-order predictor requires less outer iterations than the underlying Galerkin scheme (see Tables 2.1 and 2.2). However, the residual of the flux-corrected solution can no longer be controlled and the total number of defect correction steps is considerably greater than that for the semi-implicit FCT limiter, whereas the accuracy of the resulting solutions is comparable. Of course, the linear system (2.13) could be solved in one step (without resorting to defect correction) but this straightforward approach would inevitably lead to a severe deterioration of the linear convergence rates. Indeed, the high-order operator $M_C - \theta\Delta t K$ is much harder to 'invert' than the preconditioner A, which enjoys the M-matrix property. In many cases, the high-order solution may prove prohibitively expensive or even impossible to compute in such a brute-force way, unless a direct solver is employed. Hence, even linear high-order systems of the form (2.14) call for the use of iterative defect correction.

In order to obtain a better insight into the error reduction rate, Figure 2.5 displays the L_1-errors (top) and L_2-errors (bottom) of all six methods for both benchmark configurations. For each discretization, the solid line denotes the consistent mass matrix whereas the 'lumped' version is indicated by dashed lines. Obviously, the rate of convergence is the same for the implicit (circular markers) and explicit (square markers) FCT algorithm whereby the norm of the error is slightly smaller for the latter one if the consistent mass matrix is adopted. Interestingly enough, both FCT algorithms produce nearly the same results if mass lumping is performed. On the other hand, the solution produced by the high-order Galerkin scheme denoted by triangular markers is less accurate, which manifests itself in greater error norms. Moreover, it suffers from severe convergence

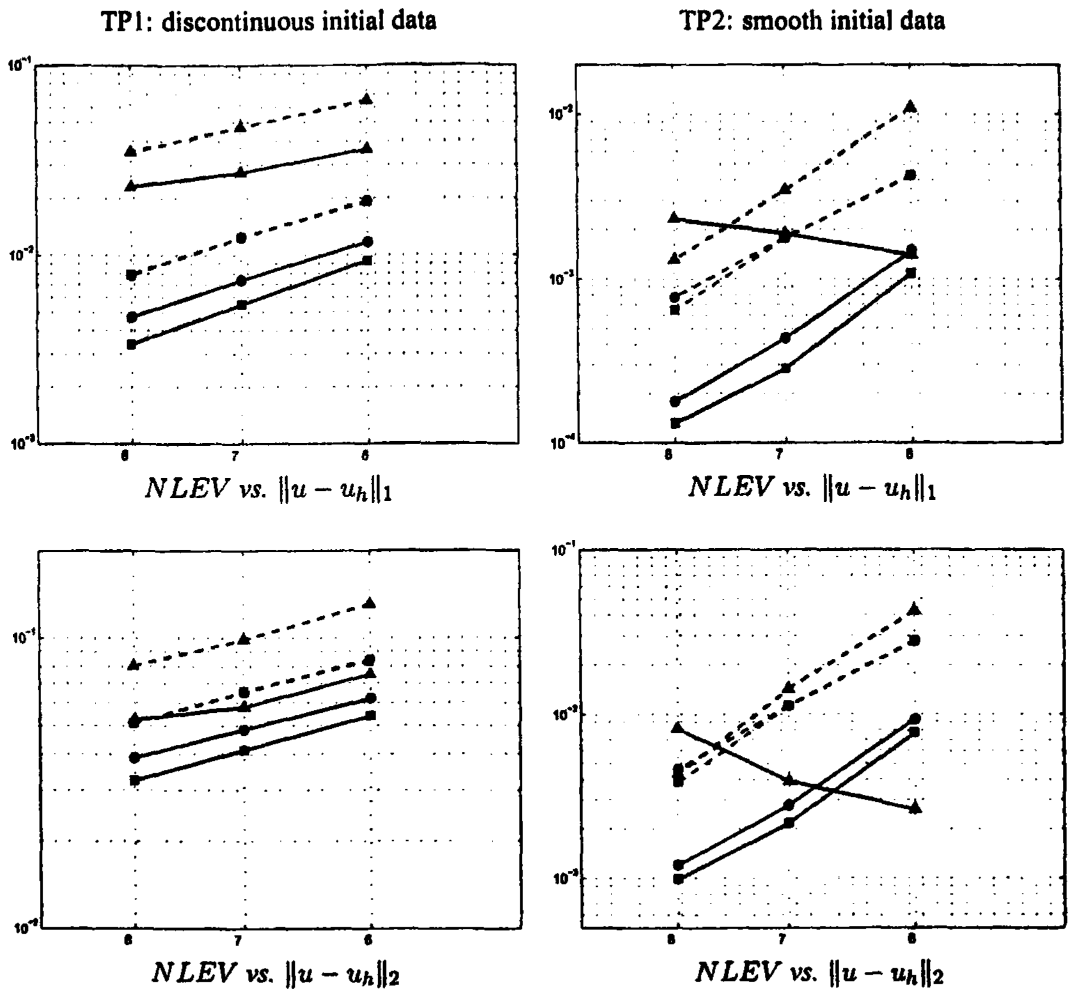


Figure 2.5: Convection skew to the mesh: error reduction.

problems if the consistent mass matrix is adopted and fails completely for the second test problem if the mesh is successively refined.

The marginally better accuracy of the semi-explicit FEM-FCT scheme as compared to its semi-implicit counterpart can be attributed to the better phase characteristics of the high-order Galerkin scheme employed at the predictor step. On the other hand, the involved splitting error may become pronounced in other settings, especially as the time step is increased. Moreover, the linear and/or nonlinear convergence rates leave a lot to be desired so that the semi-implicit approach combined with the discrete Newton method is preferable in many cases.

2.6.2 Swirling flow

Let us proceed to another two-dimensional benchmark problem proposed by LeVeque [103]. It deals with a swirling deformation of initial data by the incompressible velocity field given by

$$v_x = \sin^2(\pi x)\sin(2\pi y)g(t), \quad v_y = -\sin^2(\pi y)\sin(2\pi x)g(t).$$

The initial condition to be prescribed is a discontinuous function of the spatial coordinates that equals unity within a circular sector of $\pi/2$ radians and zero elsewhere:

$$u(x,y,0) = \begin{cases} 1 & \text{if } (x-1)^2 + (y-1)^2 < 0.8, \\ 0 & \text{otherwise.} \end{cases}$$

<u>TP3</u> For the first test problem, let us employ a constant velocity profile that corresponds to

$$g(t) \equiv 1.$$

<u>TP4</u> For the second test problem, we adopt a more 'agile' velocity field and let

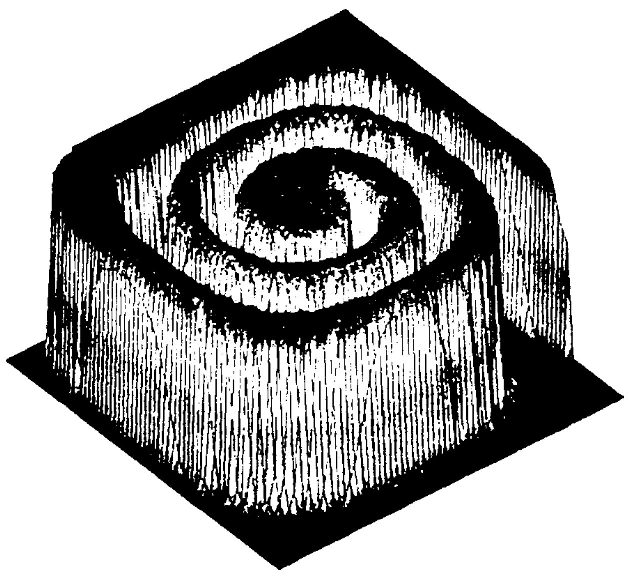
$$g(t) = \cos(\pi t/T), \qquad 0 \leq t \leq T.$$

For both benchmark configurations, the mass distribution assumes a complex spiral shape in the course of deformation. Figures 2.6–2.7 display the numerical solutions calculated by the semi-implicit FCT algorithm (2.35)–(2.43) with consistent mass matrix using the time step $\Delta t = 10^{-3}$.

Recall that for TP3, the low-order evolution operator A remains constant and can be assembled once and for all at the beginning of the simulation. The numerical results at time $t = 2.5$ are computed on a uniform mesh of 128×128 bilinear finite elements and depicted in Figure 2.6 (top). The use of a piecewise-linear finite element approximation on a triangular mesh with the same number of nodes yields virtually the same solution; see Figure 2.6 (bottom). For the difference between the underlying triangulations to be visible, both profiles were output on a coarser mesh consisting of 4,225 vertices. In either case, the resolution of discontinuities is seen to be remarkably crisp. These results compare well to those presented in [96] using algebraic flux correction schemes of TVD type.

On the other hand, the velocity vector is strongly time-dependent for benchmark TP4. After the startup, the flow gradually slows down and reverses at $t = T/2$ such that the initial profile is recovered as exact solution at the final time $t = T$; that is, $u(x,y,T) = u(x,y,0)$. The value $T = 1.5$ is used, which corresponds to performing $1,500$ time steps of size $\Delta t = 10^{-3}$. The

128 × 128 Q_1−elements



128 × 128 P_1−elements

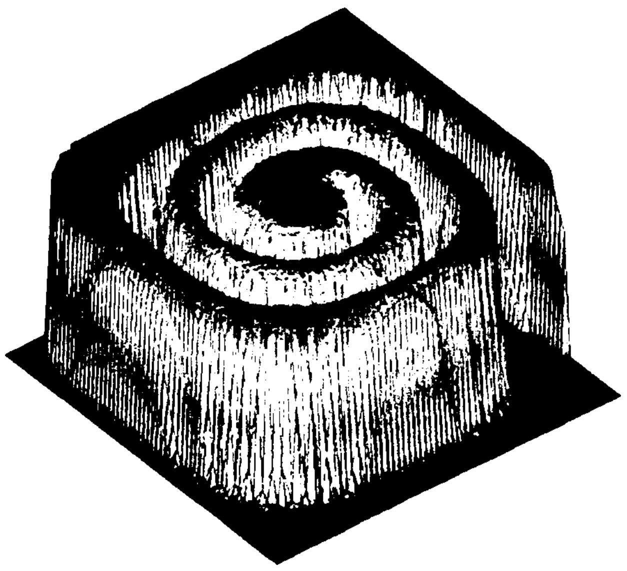
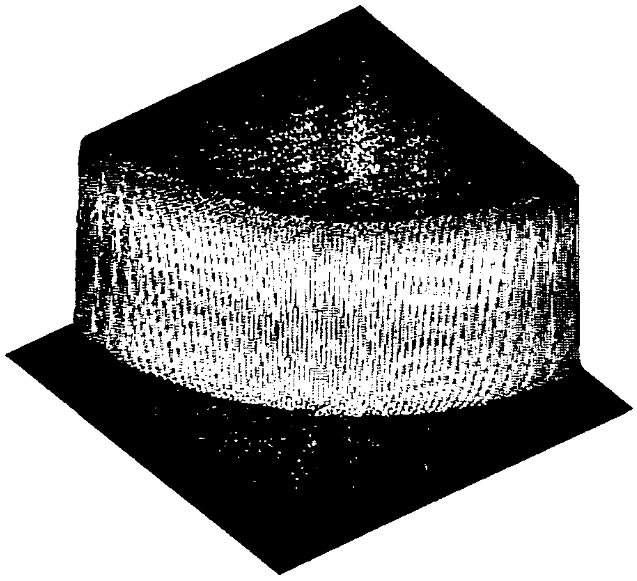


Figure 2.6: Swirling deformation: semi-implicit FEM-FCT, t = 2.5.

initial/exact solution at $t = 1.5$



intermediate solution at $t = 0.75$

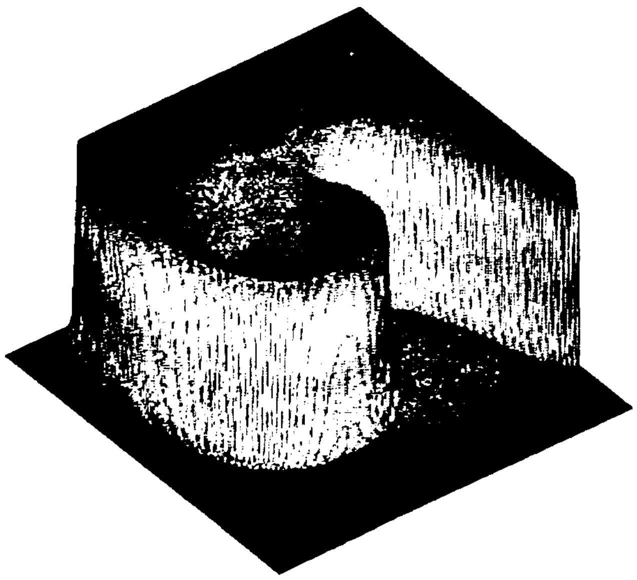


Figure 2.7: Swirling deformation: semi-implicit FEM-FCT, 128 × 128 Q_1−elements.

NLEV	NVT	CPU	NN	NN/Δt	NL	NL/NN	$u_{\min}$	$u_{\max}$
				$\|r\| \leq 10^{-4}$				
5	1,089	6	2,500	1.0	2,500	1.0	0.0	1.0
6	4,225	23	2,500	1.0	5,000	2.0	0.0	1.0
7	16,641	95	2,500	1.0	5,000	2.0	0.0	1.0
8	66,049	422	2,500	1.0	5,000	2.0	0.0	1.0
				$\|r\| \leq 10^{-8}$				
5	1,089	52	46,809	18.72	46,820	1.0	0.0	1.0
6	4,225	200	47,046	18.82	49,546	1.05	0.0	1.0
7	16,641	896	44,827	17.93	51,858	1.17	0.0	1.0
8	66,049	4,212	43,405	17.36	63,425	1.46	0.0	1.0
				$\|r\| \leq 10^{-12}$				
5	1,089	148	142,586	57.03	142,597	1.0	0.0	1.0
6	4,225	676	160,254	64.10	162,754	1.01	0.0	1.0
7	16,641	3,311	177,602	71.04	184,550	1.04	0.0	1.0
8	66,049	15,875	192,895	77.16	212,495	1.10	0.0	1.0

Table 2.3: TP3: semi-implicit FCT with consistent mass matrix, defect correction.

numerical solutions at $t = 0.75$ and $t = 1.5$, which are displayed in Figure 2.7, were calculated on a mesh of 128×128 bilinear finite elements by the semi-implicit FCT algorithm with the consistent mass matrix. The solution profiles resulting from the application of the lumped mass matrix are slightly more diffusive but 'look' quite similar.

For these two benchmark configurations, we performed an in-depth convergence study on four successively refined quadrilateral meshes. A detailed comparison between the standard defect correction method and the discrete Newton approach is presented in Tables 2.3–2.5.

As before, the first two columns display the refinement level NLEV and the number of vertices/nodes NVT. All tests were performed on an Intel Core Duo T2400 (1.83 GHz, FSB 667 MHz) processor with 1024 MB (553 MHz) of system memory. The code was compiled with the Intel Fortran 9.1 Compiler for Linux making use of the `-fast` switch, which yields the best results for this setup. The total CPU time (in seconds) required to reduce the norm of the nonlinear residual to the prescribed tolerance in each time step is given in the third column. In the next four columns, the total number of nonlinear iterations (NN), the number of nonlinear iterations per time step (NL/Δt), the total number of linear iterations (NL) and the number of linear iterations per nonlinear iteration (NL/NN) are displayed in successive order. Given the lack of an exact solution for this benchmark configuration, only the global minimum and maximum of the discrete solution u_h are compared to their analytical values 0 and 1.

NLEV	NVT	CPU	NN	NN/Δt	NL	NL/NN	$u_{\min}$	$u_{\max}$
				$\|r\| \leq 10^{-4}$				
5	1,089	7	2,500	1.0	2,500	1.0	-1.789e-02	1.026
6	4,225	24	2,500	1.0	2,500	1.0	-9.153e-03	1.054
7	16,641	102	2,500	1.0	2,500	1.0	-3.906e-02	1.121
8	66,049	442	2,500	1.0	2,500	1.0	-5.087e-02	1.202
				$\|r\| \leq 10^{-8}$				
5	1,089	32	9,891	3.96	44,547	4.50	-1.319e-11	1.0
6	4,225	138	9,917	3.97	48,379	4.88	-1.430e-09	1.0
7	16,641	611	9,294	3.72	49,464	5.32	-5.427e-12	1.0
8	66,049	2,736	8,974	3.59	50,991	5.68	-8.505e-09	1.0
				$\|r\| \leq 10^{-12}$				
5	1,089	84	25,640	10.26	123,412	4.81	0.0	1.0
6	4,225	369	26,538	10.62	141,645	5.34	0.0	1.0
7	16,641	1,674	25,061	10.02	146,212	5.83	0.0	1.0
8	66,049	7,113	22,287	8.91	139,868	6.28	0.0	1.0

Table 2.4: TP3: semi-implicit FCT with consistent mass matrix, Newton's method, $\eta = 10^{-4}$.

It can be seen from Table 2.3 that the convergence behavior of the standard defect correction scheme deteriorates significantly if the tolerance for the residual norm is reduced from 10^{-8} to 10^{-12}. Moreover, for the latter one, the number of outer iterations increases if the mesh is successively refined. On the other hand, the minimal and maximal solution values perfectly match their analytical bounds 0 and 1 because of the M-matrix property of A.

The convergence behavior of the discrete Newton approach making use of a constant forcing term $\eta = 10^{-4}$ as suggested in [29] is displayed in Table 2.4. This choice is quite restrictive and requires uniformly close approximations of Newton steps in each nonlinear iteration. It reportedly yields local q-linear convergence in some special norm [55]. As compared to the defect correction approach, the number of outer iterations is drastically reduced for all prescribed tolerances and, in addition, it does not increase for finer grids. Based on the moderate number of linear sub-iterations we believe that the ILU-decomposition of the monotone evolution operator A constitutes an appropriate preconditioner for the employed BiCGSTAB algorithm. Importantly, convergence of the fixed-point iteration is a prerequisite for the Newton method to produce a positivity-preserving solution. This is best illustrated by the unsatisfactory minimal and maximal solution values for the loose residual tolerance 10^{-4}.

Let us briefly address the phenomenon of *oversolving* [55] the linear subproblems. To this end, we relax the forcing term $\eta = 10^{-1}$ and leave all other parameters unchanged. The results computed by the discrete Newton method are shown in Table 2.5. The nonlinear convergence behavior is quite similar to that observed for the more restrictive choice $\eta = 10^{-4}$. However, the number of inner iterations is reduced by a factor of 2.5 to 3, which results in a significant reduction of the overall CPU time. Our experiments with different strategies for choosing the forcing term η adaptively [55] and even solving the linear subproblems directly [37] revealed that the simplest choice $\eta = 10^{-1}$ yields the most competitive results in terms of overall performance for this class of time-dependent flows. On one hand, the time step $\Delta t = 10^{-3}$ was chosen moderately small to resolve the temporal evolution with high precision. On the other hand, the amount of antidiffusion accepted by the FCT flux limiter is inversely proportional to Δt so that the computed solution profiles become more diffusive if larger time steps are employed. Consequently, the convergence rates of the defect correction method and of the discrete Newton algorithm improve, but the solution is smeared by numerical diffusion.

It is well known that choosing an appropriate perturbation parameter σ is a delicate task. In our simulations we employed $\sigma = [(1 + \|u\|)\epsilon]^{1/3}$, where ϵ denotes the machine precision, as proposed by Pernice et al. and successfully used in the NITSOL package [123]. In order to investigate the influence of this 'free' parameter we repeated the simulation on mesh level 7 for fixed parameter values $\sigma = \epsilon$ and $\sigma = 0.01$, respectively. Figure 2.8 displays the nonlinear convergence behavior for the different solution strategies. The curve for the defect correction method is marked by stars,

NLEV	NVT	CPU	NN	NN/Δt	NL	NL/NN	$u_{\min}$	$u_{\max}$
				$\|r\| \leq 10^{-4}$				
5	1,089	7	2,500	1.0	2,500	1.0	-1.789e-02	1.026
6	4,225	24	2,500	1.0	2,500	1.0	-9.153e-03	1.054
7	16,641	104	2,500	1.0	2,500	1.0	-3.906e-02	1.121
8	66,049	443	2,500	1.0	2,500	1.0	-5.085e-02	1.202
				$\|r\| \leq 10^{-8}$				
5	1,089	22	10,008	4.0	17,436	1.74	-4.087e-10	1.0
6	4,225	85	9,997	4.0	17,574	1.76	-1.165e-09	1.0
7	16,641	370	9,938	3.98	17,944	1.81	-9.854e-09	1.0
8	66,049	1,597	9,472	3.79	18,005	1.90	-1.797e-07	1.0
				$\|r\| \leq 10^{-12}$				
5	1,089	56	26,448	10.75	45,770	1.70	0.0	1.0
6	4,225	223	27,697	11.08	48,512	1.75	0.0	1.0
7	16,641	939	25,922	10.37	49,030	1.89	0.0	1.0
8	66,049	3,787	22,540	9.02	47,634	2.11	0.0	1.0

Table 2.5: TP3: semi-implicit FCT with consistent mass matrix, Newton's method, $\eta = 10^{-1}$.

whereas circles stand for the rapidly converging Newton method ($\eta = 10^{-1}$, $\sigma = \epsilon$). Using machine precision as perturbation parameter works for this test case, but it is likely to diverge in other situations because of round-off errors, and thus cannot be recommended in general. The strategy proposed by Pernice et al. (square markers) requires slightly more nonlinear steps but turns out to be more robust. Furthermore, the devastating effect of choosing the perturbation parameter too large, e.g., $\sigma = 0.01$, is illustrated by the fourth curve (triangles). If σ is increased even further, the convergence of Newton's method slows down until it resembles that of the defect correction approach.

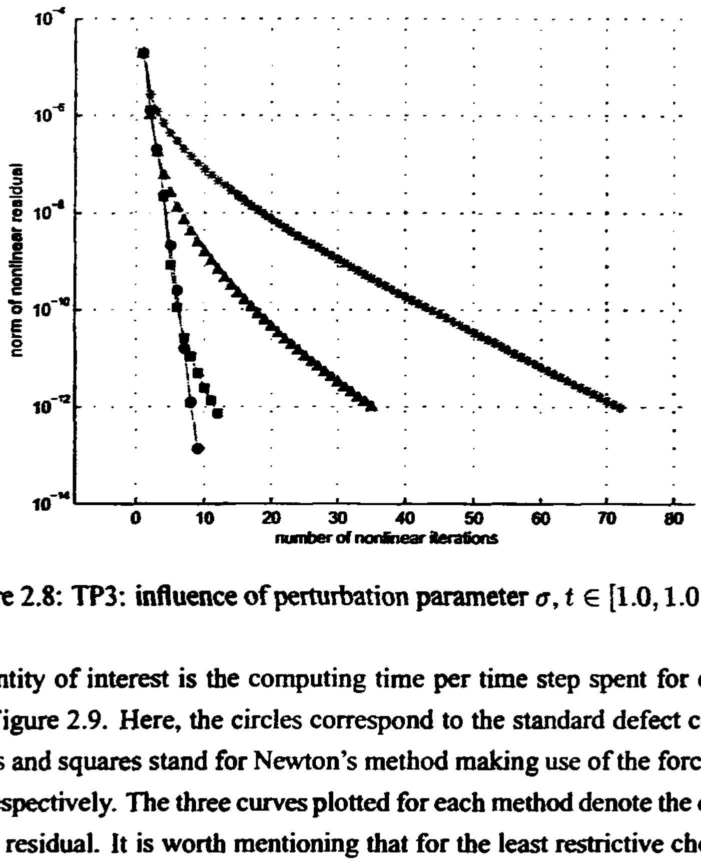


Figure 2.8: TP3: influence of perturbation parameter σ, $t \in [1.0, 1.0 + \Delta t]$.

Another quantity of interest is the computing time per time step spent for each vertex, which is illustrated in Figure 2.9. Here, the circles correspond to the standard defect correction approach whereas triangles and squares stand for Newton's method making use of the forcing term $\eta = 10^{-4}$ and $\eta = 10^{-1}$, respectively. The three curves plotted for each method denote the different tolerances for the nonlinear residual. It is worth mentioning that for the least restrictive choice $\eta = 10^{-1}$, the nodal CPU time remains nearly constant if the number of vertices is increased, whereas a systematic growth is observed for both other methods.

Table 2.6 illustrates the convergence behavior of the different solution methods and the errors of the finite element approximation u_h at time $t = 1.5$ for our benchmark configuration TP4. For all computations, a moderate stopping criterion $\|r\| \leq 10^{-8}$ was used and the approved forcing strategy $\eta = 10^{-1}$ was adopted for Newton's method. Moreover, the perturbation parameter σ was computed as proposed by Pernice et al. [123] and utilized for the divided difference approximation. All other parameter settings, e.g., the configuration of the linear solver, remain unchanged. It can be readily seen that the discrete Newton approach outperforms the standard defect correction scheme in all situations.

Figure 2.10 (top) illustrates the CPU time spent per node in each time step, which remains nearly constant for all mesh levels. As before, the circular markers correspond to the standard defect correction method, whereas squares are used for the discrete Newton approach. Here, the dashed lines represent the lumped-mass versions of the two algorithms. The significant overhead costs of the slowly converging defect correction method are clearly visible. The solution errors, which are virtually the same for both nonlinear solution strategies, exhibit a monotone reduction on sufficiently fine meshes as illustrated in Figure 2.10 (bottom).

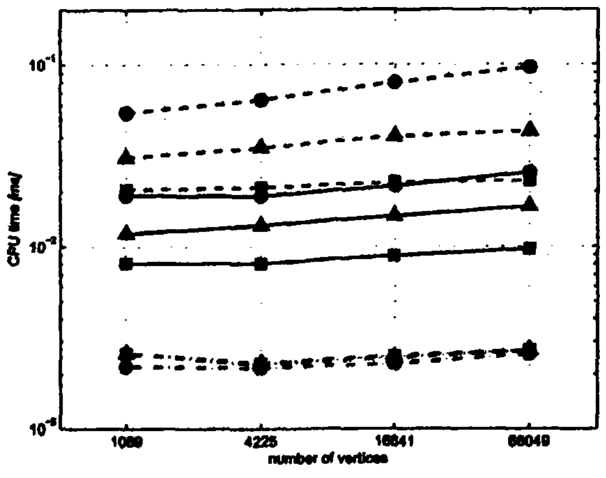
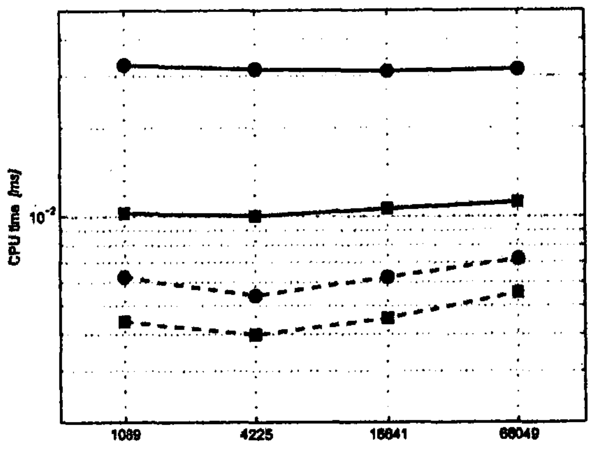


Figure 2.9: TP3: influence of perturbation parameter σ, $t \in [1.0, 1.0 + \Delta t]$.

NLEV	NVT	CPU	NN	NN/Δt	NL	NL/NN	$\|u-u_h\|_1$	$\|u-u_h\|_2$
				defect correction / consistent mass matrix				
5	1,089	89	24,136	16.09	24,136	1.0	2.7748e-2	8.8019e-2
6	4,225	333	22,694	15.13	23,488	1.03	1.5630e-2	6.7038e-2
7	16,641	1,293	19,883	13.26	21,954	1.10	8.8456e-3	5.0641e-2
8	66,049	5,203	17,959	11.97	22,812	1.27	5.1680e-3	3.8747e-2
				defect correction / lumped mass matrix				
5	1,089	17	2,720	1.81	2,720	1.0	4.5446e-2	1.1689e-1
6	4,225	57	2,738	1.83	3,481	1.27	2.9877e-2	9.4992e-2
7	16,641	259	2,804	1.87	3,818	1.36	1.9192e-2	7.5658e-2
8	66,049	1,186	2,953	1.97	4,777	1.62	1.2250e-2	5.9934e-2
				Newton's method / consistent mass matrix				
5	1,089	28	5,506	3.67	9,501	1.73	2.7743e-2	8.8007e-2
6	4,225	106	5,241	3.94	9,074	1.73	1.5624e-2	6.7021e-2
7	16,641	442	4,831	3.22	8,342	1.73	8.8374e-3	5.0609e-2
8	66,049	1,844	4,506	3.0	7,802	1.73	5.1604e-3	3.8719e-2
				Newton's method / lumped mass matrix				
5	1,089	12	1,510	1.01	1,510	1.0	4.5441e-2	1.16882e-1
6	4,225	42	1,513	1.01	1,513	1.0	2.9868e-2	9.4975e-2
7	16,641	188	1,572	1.05	1,572	1.0	1.9175e-2	7.5623e-2
8	66,049	910	1,803	1.20	1,803	1.0	1.2231e-2	5.9887e-2

Table 2.6: TP4: semi-implicit FCT, $\|r\| \leq 10^{-8}$, $\eta = 10^{-1}$.

nodal CPU time *vs.* number of vertices



error reduction in L_1- and L_2-error

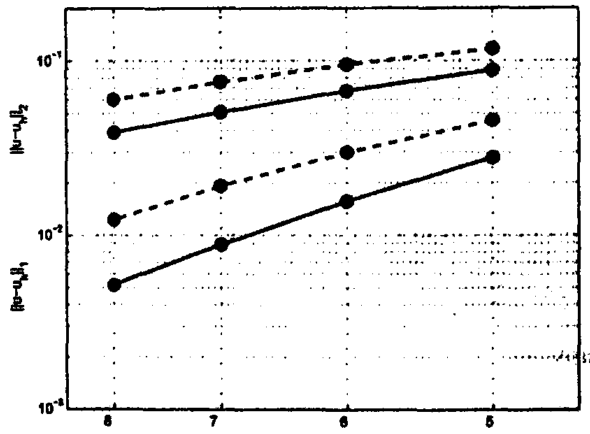


Figure 2.10: TP4: nodal CPU time / error reduction.

2.7 Results and discussion

The semi-implicit approach to flux correction of FCT type leads to a robust and efficient special-purpose algorithm for time-dependent problems discretized in space by the finite element method. The accuracy of the resulting scheme improves as the time step is refined and the consistent mass matrix can be included in a positivity-preserving fashion. The new limiting strategy makes it possible to avoid a repeated computation of the nodal correction factors at each outer iteration. Therefore, the use of an implicit time-stepping method pays off in spite of the CFL-like condition to be satisfied by the time step in the case $\theta < 1$. For sufficiently small time steps, the new algorithm is more accurate and/or efficient than the algebraic flux correction schemes proposed in [98, 99]. On the other hand, it is not to be recommended for steady-state computations, which call for the use of large time steps. In this case, both the limiting strategy and the underlying constraints need to be redefined as explained in [95].

In order to solve the nonlinear algebraic systems, a discrete Newton approach was devised making use of the fact that the underlying sparsity pattern is known *a priori*. The Jacobian matrix was assembled edge-by-edge using numerical differentiation as applied to the low-order operator and to the vector of limited antidiffusive fluxes. The use of a new semi-implicit limiting strategy makes it possible to assemble the Jacobian in a particularly efficient way, resulting in a significant reduction (by a factor of 2.5 to 3.5) of the total CPU time as compared to standard defect correction. The semi-explicit FCT algorithm was found to provide a slightly better accuracy for the test cases considered in the present chapter. However, the high-order system to be solved at the predictor step is extremely ill-conditioned, thus requiring a slowly converging defect correction scheme preconditioned by the low-order operator.

Chapter 3

Solution of Large Sparse Linear Systems with Domain Decomposition Methods

3.1 Introduction

Numerical methods for solving systems of partial differential equations (**PDEs**) usually employ implicit discretization schemes, which always lead to an algebraic system to be solved. The corresponding linear systems are large (involving hundreds of thousands or even several millions of unknowns) but at the same time very sparse. Sparsity here refers to the fact that most of the elements of the matrix involved are zero.

Sparse linear systems of equations can be solved by either *direct sparse linear solvers*, which are sophisticated implementations of LU decomposition and back-substitution, or by *iterative solvers*. Recently, direct sparse linear solvers have been tuned close to perfection and excellent software libraries, serial **CHOLMOD** [34, 36], **UMFPACK** [38, 39, 40, 41] and even parallel **PARDISO** [135, 136], are publicly available.

Sparse direct solvers, are amazingly efficient for linear systems arising from the discretization of PDE problems in two-dimensional domains. In the three-dimensional case, however, their performance deteriorates significantly and at the same time memory resources rapidly become insufficient as the problem size increases.

3.1.1 Motivation: Direct sparse solvers and their shortcomings

Direct sparse linear solvers possess very favorable properties. The solution they provide is accurate up to machine precision, and it is computed very fast once the LU factors are available because they employ optimized computational kernels. At the same time, without sacrificing performance they

PARDISO performance in 2D							
mesh statistics		Elapsed time in seconds					memory
nodes	$h_{\max}$	assembly	init	fact	back-sub	total	MB
4014	3.125e-02	0.004	0.037	0.011	0.001	0.049	1.356
16454	1.563e-02	0.017	0.158	0.052	0.004	0.216	5.857
65175	7.813e-03	0.071	0.704	0.265	0.019	0.988	26.018
263838	3.906e-03	0.299	3.199	1.539	0.111	4.849	117.523
1068112	1.593e-03	1.307	14.766	9.518	0.535	24.819	527.656

Table 3.1: Performance of **PARDISO** direct sparse solver in serial mode for problem (3.1) discretized by linear triangular finite elements.

can provide the solution with the transpose operator. For these reasons among many others, direct sparse solvers have become the standard approach for PDEs in two-dimensional domains. Their use, however, is rather limited for PDEs in three dimensions as the computational resources are quickly exhausted before the spatial accuracy of the discretization scheme becomes sufficient.

We demonstrate next the reasons why direct sparse linear solvers have become a standard tool for 2D problems and the need for more sophisticated solution methods in real-life 3D problems.

2D problems

The complexity of direct sparse linear solvers for two-dimensional problems, being $O(n^{3/2})$ for the LU factorization and $O(n \log n)$ for the back substitution, is nearly optimal, and thus they can hardly be outperformed by other methods. At the same time, the memory requirements are moderate and that makes them perfect black-box solvers for a wide class of two-dimensional problems. Let us illustrate this fact by considering a classical PDE problem, namely the the scalar Poisson equation with homogeneous Dirichlet boundary conditions on the unit square:

$$\begin{aligned} -\nabla^2 u(x) &= 1, \quad x \in \Omega \\ u(x) &= 0, \quad x \in \partial\Omega. \end{aligned} \tag{3.1}$$

We discretize problem (3.1) by the Finite Element method using an unstructured triangular mesh and the P1 family of elements (linear triangles). Table 3.1 summarizes the time spent for the assembly of the finite element matrix and the time **PARDISO** needed for each phase of the solution process. The *initialization* phase involves analysis related to the sparse matrix structure, which is determined by the topology of the mesh and symbolic factorization, while *factorization* and *solution* phases correspond to the sophisticated, highly efficient LU factorization of the matrix and the corresponding back-substitution.

PARDISO performance in 3D							
mesh statistics		Elapsed time in seconds					memory
nodes	$h_{\max}$	assembly	init	fact	back-sub	total	MB
500	2.7e-1	0.001	0.005	0.001	0	0.006	0.298
4001	1.3e-1	0.017	0.062	0.057	0.002	0.121	4.712
32002	6.8e-2	0.180	0.636	2.783	0.069	3.488	80.357
256011	3.9e-2	1.785	6.925	185.469	1.428	193.8	1438.32
2000396	1.7e-2	16.942	76.625	11762.1	21.46	11860	24403

Table 3.2: Performance of **PARDISO** direct sparse solver in serial mode for problem (3.1) discretized by linear tetrahedral finite elements.

For linear steady-state problems, one is interested in the total solution time, while for the transient case, what matters is just the back-substitution time, as initialization and factorization are needed just once. In the nonlinear case, however, factorization and back-substitution times should be taken into account for both steady and transient problems, because the matrix changes within each outer iteration. The initialization phase should be repeated whenever the mesh topology changes (adaptively refined meshes, etc.). We clearly see that initialization and factorization times are greater than for the matrix assembly by less than one order of magnitude. However, the back-substitution is always more than twice as fast as the assembly and therefore the same factorization is frequently used throughout the solution of a nonlinear system. The number of outer iterations (Newton fixed-point steps) thereby increases, but the whole process is considerably accelerated, especially in the transient case when the matrix does not change too much for several time steps. We also see that the memory allocated by the solver is quite low and a problem involving one million of unknowns can be quickly solved on a common laptop. For the reasons described above, direct sparse solvers are frequently the first choice of every researcher when it comes to PDE problems discretized in two dimensions and there is not too much interest in more sophisticated methods for the solution of such problems.

3D problems

The computationally attractive properties that direct sparse linear solvers possess for problems in two spatial dimensions do not carry over to the three-dimensional case. There, the complexity dramatically increases to $O(n^2)$ operations for the factorization and $O(n^{4/3})$ for the back-substitution. Table 3.2 summarizes the performance of **PARDISO** when we solve problem 3.1 on the unit cube using the P1 family of elements (linear tetrahedra).

It is evident that the time spent for the back substitution is still quite encouraging, being usually less than the time needed for the matrix assembly. However, the time associated with the factorization grows rapidly, rendering the total solution process slower than that of the assembly by two orders of magnitude. This is observed even for medium-size problems, where the resolution is still not satisfactory. At the same time the memory needed by the solver grows prohibitively large, restricting severely the size of the problems that can be handled. The memory and CPU time restrictions we just reviewed have motivated practitioners to develop other types of solution approaches that are not so "resource hungry". These are the iterative methods we discuss next.

3.2 Iterative methods

In computational mathematics, an iterative method attempts to solve a problem (for example an equation or system of equations) by finding successive approximations to the solution, starting from an initial guess. Iterative methods are the only choice for nonlinear equations. However, they are very popular for linear problems involving a large number of variables (sometimes of the order of millions), where direct methods would be prohibitively expensive and in some cases impossible even with the best available computing power.

The main classes of iterative methods employed for the solution of linear algebraic systems are the stationary iterative methods, the multigrid methods, and the more popular Krylov subspace methods.

3.2.1 Stationary iterative methods

Stationary iterative methods solve a linear system with an operator approximating the original one. Based on a measure of the error (the residual), they form a correction equation for which this process is repeated. While these methods are simple to derive, implement, and analyze, convergence is only guaranteed for a limited class of matrices. Examples of stationary iterative methods are the Jacobi method, the Gauss-Seidel method and the successive over relaxation (SOR) method. Stationary iterative methods are no longer competitive. However, they are used frequently as **smoothers** in multigrid solvers.

3.2.2 Multigrid methods

Multigrid (MG) methods are a group of algorithms for solving differential equations. They belong to a conceptually different class of iterative methods that take into account the underlying equations (or PDE system), using a hierarchy of discretizations of the problem under consideration. The typical application of multigrid is the numerical solution of elliptic partial differential equations in two or more dimensions [166].

Multigrid methods may be applied in combination with any of the common discretization techniques. In these cases, they are among the fastest solution approaches known today, because their complexity scales linearly with the number of unknowns. In contrast to other methods, they can in general treat arbitrary regions and boundary conditions. Their setting does not depend on the separability of the equations or other special properties. They are also directly applicable to complicated nonsymmetric and nonlinear systems of equations, like the Navier-Stokes equations [151, 153, 154] or even coupled fluid structure interaction PDE systems [80]. The multilevel concept has been successfully adopted by several other areas like graph partitioning [85, 86] and PDE optimization [25, 50, 51, 70, 115].

Other extensions of multigrid include techniques where no PDE and no geometrical problem background is used to construct the multilevel hierarchy. Such algebraic multigrid methods (AMG) construct their hierarchy of operators directly from the system matrix and thus become true black-box solvers for sparse matrices. Algebraic multigrid methods are documented extensively in [150]. High performance scalable libraries are also publicly available [58, 76].

AMG methods are now used as preconditioners (as inner iteration) within Krylov subspace solvers (as outer iteration), which undoubtedly are the most popular iterative solution methods available today.

3.2.3 Krylov subspace methods

The **iterative methods** applied today for solving large-scale linear systems

$$Ax = b \tag{3.2}$$

belong to the class of **preconditioned Krylov subspace solvers**. Krylov subspace solvers are considered black-box solvers, as they do not need any other input except the linear system itself, just like the direct sparse solvers. Unlike direct sparse solvers, however, Krylov subspace solvers do not provide the solution up to machine precision but approximate it instead, up to the desired accuracy defined by some tolerance.

Let x_0 be an initial approximation to the solution of the linear system (3.2) and $r_0 = b - Ax_0$ be the initial residual. Then

$$\mathcal{K}_m(A, r_0) = \text{span}\{r_0, Ar_0, A^2r_0, \ldots, A^{m-1}r_0\} \tag{3.3}$$

is the Krylov subspace of dimension m defined by A and r_0. At the mth step, Krylov subspace methods obtain an approximation x_m to the solution of (3.2) in the space $x_0 + \mathcal{K}_m$ by satisfying a projection or minimizing condition of some kind. Let $r_m = b - Ax_m$ be the residual at the mth step. Some standard conditions are:

- *Petrov-Galerkin condition*

$$r_m \perp \mathcal{R}_m, \tag{3.4}$$

  where $\mathcal{R}_m$ is some m-dimensional subspace.

- *Galerkin condition*, when $\mathcal{R}_m = \mathcal{K}_m$

$$r_m \perp \mathcal{K}_m. \tag{3.5}$$

- *Minimum residual condition*

$$\|r_m\| = \min_{x \in x_0 + \mathcal{K}_m} \|b - Ax\|. \tag{3.6}$$

It can be shown that (3.6) is a Petrov-Galerkin condition when $\mathcal{R}_m = A\mathcal{K}_m$ (Saad [132]).

The construction of an orthonormal basis for the Krylov subspace is carried out by the Arnoldi procedure [7]. When the matrix is symmetric, this procedure simplifies and is due to Lanczos [101, 102]. A two-sided Lanczos procedure also suggested in [102] applies to nonsymmetric matrices. The major advantage of the Lanczos procedures is that the bases can be constructed by a three-term recurrence or two coupled two-term recurrences, respectively. Thus, only two to three previous vectors in each sequence need to be stored, thus keeping the storage requirements low. In contrast, the Arnoldi procedure requires the whole basis to be stored. On the negative side, however, the two-sided Lanczos procedure may break down, and it needs products by the transpose operator A^T, which is usually more computationally expensive to apply than A.

There is a vast literature on Krylov subspace methods, with each trying to cure weaknesses of previous ones. Although methods that target nonsymmetric linear systems can in general be applied equally well to symmetric ones, special methods have been suggested for the symmetric case that are both more effective and computationally efficient.

3.2.4 Symmetric case

The method of choice for symmetric positive definite linear systems is the well known conjugate gradient method (CG) suggested by Hestenes and Stiefel in their monumental paper [77]. We now know that it is effectively based on the symmetric Lanczos process, enforcing the Galerkin condition

(3.5), which can be shown to be equivalent to minimizing the A-norm (energy norm) of the error:

$$\min_{x \in \mathcal{K}_m} \|x - x_*\|_A \,, \tag{3.7}$$

where the A-norm is induced by the A-inner product $\langle x, y \rangle = x^T Ay$. Unlike other methods where the stopping criterion monitors the Euclidean norm of the residual, CG allows sophisticated monitoring of the convergence, based on estimations of the norm of the true error of the linear system (3.2), as later found by Golub and Kent [67] and Arioli [6].

When the matrix A is symmetric but indefinite, CG is no longer applicable. Paige and Saunders [118] devised two methods for this case, both based on the symmetric Lanczos process. In the minimum residual method (MINRES) the minimum residual condition (3.6) is imposed, and only the last two basis vectors for $\mathcal{K}_m$ are needed for the computation of the approximation x_m. By considering the Galerkin condition (3.5) instead, Paige and Saunders in [118] derived the SYMMLQ method for symmetric systems, which usually terminates at the CG approximation if it exists, but along the way generates a different sequence of approximations that minimize the 2-norm of the error over the subspace $\mathcal{K}(A, Ar_0)$. Again only two basis vectors are needed.

The symmetric solvers are effective if the eigenvalues of A are clustered.

3.2.5 Nonsymmetric case

For the solution of linear systems where A is not symmetric, several methods have been proposed. Among them the Generalized Minimal Residual method (GMRES) introduced by Saad and Schultz [133] is among the most popular choices. It computes a sequence of orthogonal vectors (like MINRES) and combines these through a least-squares solve. Unlike MINRES and CG, it uses the whole sequence of basis vectors, and hence a large amount of storage is usually required. Restarted versions of the method, GMRES(m), are frequently used instead, where computation and storage costs are limited by specifying a fixed number of vectors (m) to be generated.

Paige and Saunders in [119, 120] introduced the LSQR method for solving nonsymmetric linear systems and least squares problems. The method employs the Golub and Kahan bidiagonalization procedure. It is analytically equivalent to the symmetric CG method applied to the normal equations $A^T Ax = A^T b$, but it possesses more favorable numerical properties. It can also estimate the condition number of the matrix A and standard errors for x among other quantities of interest. LSQR is effective if the singular values of A are clustered. It is applicable to both square and rectangular systems [119, 120, 134]. On the downside, it requires matrix-vector products with both A and the transpose operator A^T.

Other popular choices for the nonsymmetric case are the conjugate gradient squared method

(CGS) designed by Sonneveld [146] and the Biconjugate Gradient Stabilized method (Bi-CGStab) introduced by van der Vorst [156]. The former is a variant of an older approach, the BiConjugate Gradient method (BiCG), which exhibits irregular convergence and potentially breaks down. Sonneveld managed to avoid BiCG's need for the transpose matrix by applying the update operations for the A-sequence and the A^T-sequences both to the same vectors. Often one observes a convergence rate for CGS about twice that for BiCG, which is in agreement with the observation that the same contraction operator is applied twice. In practice, however, convergence may be more irregular than BiCG. The BiCGStab method remedies the irregular behavior of CGS by using different updates for the A^T sequence. Often it converges about as fast as CGS, sometimes faster and sometimes not.

These and many other properties of iterative Krylov subspace methods are discussed in the monographs by Saad [132], van der Vorst [157], Trefethen [149], and Axelsson [11]. Their algorithmic implementation along with several theoretical details can be found in [14].

Quite frequently for both symmetric and nonsymmetric linear systems, the Krylov methods just reviewed may need several hundreds to many thousands of iterations before they reach the desired accuracy. The situation worsens with increasing problem size and calls for sophisticated solution techniques to accelerate the convergence. This is achieved by what is referred to as **preconditioning techniques**, which are the most crucial component of iterative solution approaches.

3.3 Preconditioning

A **preconditioner** P for a matrix A is a matrix that approximates A in some useful sense. Usually this means that $P^{-1}A$ has a condition number significantly smaller than that of A. Preconditioners are employed by iterative subspace Krylov methods to accelerate their convergence related to the solution of linear systems $Ax = b$ by considering instead the left-preconditioned system

$$P^{-1}Ax = P^{-1}b, \tag{3.8}$$

or the right-preconditioned system

$$AP^{-1}\bar{x} = b, \qquad Px = \bar{x}. \tag{3.9}$$

The products $P^{-1}A$ and AP^{-1} are never formed explicitly. This is prohibitively expensive and would require more storage than direct sparse methods. Instead, the matrix-vector products Av_k required by any iterative method are replaced by products $P^{-1}(Av_k)$ when the preconditioner is applied from the left, or by $A(P^{-1}v_k)$ when the preconditioner is applied from the right.

The preconditioner P should be such that it allows P^{-1} to be applied very efficiently and for a smaller number of iterations, thus reducing the total cost (computing time). At the same same time, especially when we aim at the solution of linear systems obtained from the discretization of PDEs, the preconditioning approach should render the condition number of the preconditioned linear system insensitive to the parameters involved in the adopted discretization scheme (usually the mesh size h or polynomial order) and if possible to the parameters inherent in the problem under consideration (reaction or diffusion coefficients). In cases where this goal is reached, one usually observes that the number of iterations needed by the Krylov solver, in order to reduce the relative residual of the linear system to some threshold, remains the same or marginally increases for a very wide range of the parameters introduced by the discretization approach and those appearing in the underlying PDE system.

Several types of preconditioner have been suggested in the literature. Among the most popular black-box choices are incomplete LU preconditioners, which are extensively documented in the work of Saad [132], as well as very robust multilevel ILU preconditioners, suggested by Bollhöfer [19, 20, 21, 22] and implemented in the publicly available software library ILUPACK.

Motivated by today's widespread availability of cheap multi-processor hardware, we now investigate preconditioning approaches that are designed from the ground up for parallel processing. The class of methods that suit our purpose best is the class of domain decomposition preconditioners.

3.3.1 Domain decomposition preconditioners

Large-scale numerical simulations, like those arising in many areas of physics and engineering, call for parallel solution algorithms. Domain decomposition serves exactly that purpose: to devise parallel algorithms that can benefit strongly from multiprocessor computers.

Domain decomposition methods need a partitioning of the computational domain Ω into subdomains Ω_i, $i = 1, 2, \ldots, p$, which may or may not overlap. The original problem may then be reformulated upon each subdomain Ω_i, yielding a family of subproblems of reduced size, coupled through the values of the unknown solution at the interface of the subdomains; see Figure 3.1.

In many cases, the interface coupling is removed at the expense of introducing an iterative process among subdomains, yielding at each step independent subproblems (of lower complexity) that can be efficiently handled by multiprocessor systems. However, the number of iterations needed to achieve a certain level of accuracy grows rapidly as the number of subdomains and the problem size increase, degrading performance significantly and calling for more sophisticated iterative approaches.

When properly devised, these iterative procedures intrinsically embody a preconditioner for the system induced on the interface unknowns. A distinguishing feature of a domain decomposition method is the property of optimality of such a preconditioner: its ability to generate a sequence that

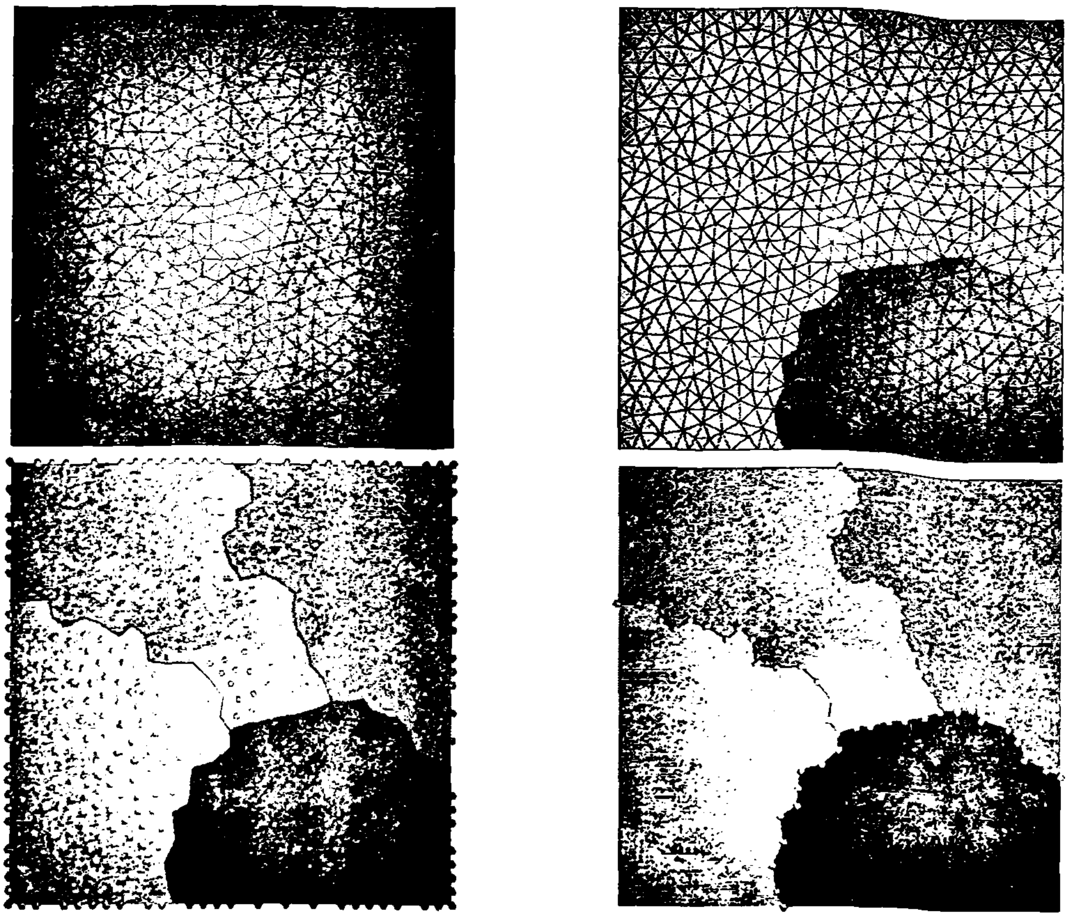


Figure 3.1: The mesh (top left) is partitioned into 4 subdomains (top right). At the bottom left we see the unknowns in the interior and at the bottom right, the unknowns on the interface of the subdomains.

converges at a rate that does not depend on parameters inherent in the discretization approach. Such parameters are usually the size of the original system N or equivalently mesh size h and the number of the partitions, the variability of element sizes and even further diffusion or reaction coefficients (in case of convection diffusion reaction PDE systems).

Traditionally, domain decomposition methods can be broadly classified as either overlapping or nonoverlapping methods. The precise presentation of each method suggested in the literature would require an extended discussion and is beyond our scope. Instead we briefly review the key ideas behind classical domain decomposition methods and provide a more detailed discussion of the key steps of the most recent and competitive ones.

3.3.2 Overlapping methods

One-level methods Methods belonging to this class were introduced and analyzed first by Schwarz in 1890. They require a partitioning of the domain in overlapping subdomains. The level of the overlap may vary. Usually higher levels of overlap improve the quality of the preconditioner by reducing the sensitivity of the solution process to the number of the unknowns. However, the method remains sensitive to the number of partitions. In [137] Schwarz introduced what is now referred to as the *alternating Schwarz method.* No significant advances were reported until one century later when Lions in [105, 106, 107] was responsible for focusing much attention on the subject. Dryja and Windlund (1987) in [47] suggested a modification of the original algorithm to make it significantly more reliable, the *additive Schwarz method.* It was further analyzed in [48] and found to be less sensitive to the level of overlap. Further analysis and convergence estimates were obtained by Bramble, Pasciak, Wang and Xu (1991) in [27].

Multilevel methods The sensitivity of the classical Schwarz methods on the number of partitions was partially removed by a modification of the original algorithm that led to the so-called *multilevel overlapping Schwarz methods.* There, apart from sufficient overlap, a coarse grid is generated where solutions to the original PDE are approximated and interpolated back to the initial fine grid. The coarse grid and the interpolation and restriction operators are constructed in a process similar to classical multigrid methods. The key difference, is that in domain decomposition methods the ratio in mesh refinement between levels may vary between 10 or 100, whereas in classical multigrid methods it is usually 2 or 4. The consideration of the coarse grid dramatically reduces the condition number of the preconditioned system and it renders it practically insensitive to the number of subdomains. With sufficient overlap and a carefull choice of the size of the coarse grid, one usually expects that the iterations needed for the solution of the system will be practically independent of both the number of subdomains and the number of the unknowns. Multilevel versions have been investigated by Dryja and Widlund in [49], by Zhang [171, 172, 173], and by Griebel and Oswald in [71]. Numerical studies of multilevel overlapping Schwarz methods may be found in Gropp and Smith [72] and Bjørstad and Skogen [17]. Recent advances can be found in Chai [30] and Chan and Smith [32].

Overlapping Schwarz methods do not exhibit good scalability in parallel computing systems. The intercommunication between the processors forced by both the overlap and the coarse grid correction inevitably restricts the level of parallelism that can be achieved. Approaches carefully designed to achieve higher scalability in parallel computing systems are those belonging to the class of nonoverlapping domain decomposition methods, discussed next.

3.3.3 Nonoverlapping methods

These methods begin by generating a nonoverlapping partitioning of the domain. At the next step the unknowns that belong to the interior of the subdomains are ordered first and the ones residing on the interface of the subdomains are ordered last; see Figure 3.1. They focus then on devising preconditioners for the Schur-complement system related to the unknowns on the interface of the subdomains.

Neumann-Dirichlet The older methods in this category are the Neumann-Dirichlet method discussed by Bjørstad and Widlund in [18] and later applied to structural mechanics by Bjørstad and Hvidsten [16]. In the case of two subdomains for instance, one has to solve a problem with Neumann boundary conditions on the artificial boundary in the first subdomain, followed by a problem with Dirichlet boundary conditions on the artificial boundary in the second subdomain. Practical extension of this method to more than two subdomains and parallel implementations pose severe restrictions on the partitioning approach by requiring the domain to be partitioned into strips. Then each processor is assigned two adjacent strips. This is not possible for arbitrary domains.

Neumann-Neumann In the Neumann-Neumann preconditioner, proposed in Bourgat et al. [26] and extended later by Le Tallec [147] to arbitrary subdomains, a Dirichlet boundary value problem and a Neumann boundary value problem are solved on each subdomain. This approach extends easily to any number of processors, in contrast to the Neumann-Dirichlet preconditioner. On the downside, for subdomains that reside completely in the interior of the domain, the Neumann boundary value problem to be solved is singular. The standard way to remedy this is to add a zero-order term to the Neumann boundary value problem to render it nonsingular. As with overlapping methods, the standard Neumann-Neumann preconditioner converges slowly for large numbers of subdomains. This observation motivated Mandel to introduce a coarse grid correction, deriving a new preconditioner that we discuss next.

Balancing Neumann-Neumann The balancing Neumann-Neumann method uses a piecewise-constant grid correction to provide global communication among the several processors and it is almost an optimal preconditioner. It was introduced by Mandel in [112]. The convergence depends only mildly on the ratio H/h of the substructure H size to the discretization size h. The standard implementation, however, requires that a Dirichlet and a Neumann boundary value problem be solved exactly for each subdomain at each iteration, which is a potentially expensive operation.

Iterative substructuring Substructuring methods are among the latest and more sophisticated nonoverlapping domain decompositions methods available. The construction of the related Schur-complement preconditioner specializes to the dimension of the problem. We describe a variant suggested by Smith [145] that, unlike previous versions in this category, can be completely parallelized. In three dimensions the interface boundary consists of faces (shared by two adjacent subdomains), edges, and vertices. Edges and vertices may be shared by many faces. The unknowns on the interface boundary u_B are reordered to unknowns on the faces u_F and unknowns on the wirebasket u_W (the union of edges and vertices): $u_B = [u_F \ u_W]^T$. The Schur complement with this type of reordering has the form

$$S = \begin{pmatrix} S_{FF} & S_{FW} \\ S_{WF} & S_{WW} \end{pmatrix} \tag{3.10}$$

and can be assembled from its substructure contributions as

$$S = \sum_{\Omega_i} \bar{R}_i^T S_i \bar{R}_i, \tag{3.11}$$

where S_i may be written as

$$S_i = \begin{pmatrix} S^i_{FF} & S^i_{FW} \\ S^i_{WF} & S^i_{WW} \end{pmatrix}. \tag{3.12}$$

Let T_i^T map the average of the values of the boundary nodes of each face (the adjacent edges and vertices) onto the nodes on the corresponding face. Then S_i may be written as

$$S_i = \begin{pmatrix} I & 0 \\ -T_i & I \end{pmatrix} \begin{pmatrix} S^i_{FF} & \tilde{S}^i_{FW} \\ \tilde{S}^i_{WF} & S^i_{WW} \end{pmatrix} \begin{pmatrix} I & -T_i^T \\ 0 & I \end{pmatrix}. \tag{3.13}$$

The couplings between the various faces and the couplings between the faces and the wirebasket are then dropped. Also S^i_{WW} is replaced by G_i, which is defined via the following minimization problem:

$$v^T G_i v = \min_{\bar{w}_i} (v - \bar{w}_i z_i)^T (v - \bar{w}_i z_i), \tag{3.14}$$

where for scalar PDEs, $z_i = [1, 1, \ldots, 1]^T$. The purpose of the minimization problem (3.14) is to ensure that the null space of G_i is the same as the null space of $\tilde{S}^i_{WW}$, which implies that the null space of the substructure preconditioner B_i to be introduced below is the same as the null space of the Schur-complement contribution of the same substructure S_i. The preconditioner corresponding then to each substructure is

$$B_i^{-1} = \begin{pmatrix} I & 0 \\ -T_i & I \end{pmatrix} \begin{pmatrix} \hat{S}^i_{FF} & 0 \\ 0 & G_i \end{pmatrix} \begin{pmatrix} I & -T_i^T \\ 0 & I \end{pmatrix}. \tag{3.15}$$

Finally the action of the inverse of the related preconditioner can be assembled from individual contributions of each substructure as

$$B^{-1} = \sum_{\Omega_i} \tilde{R}_i^T B_i^{-1} \tilde{R}_i. \tag{3.16}$$

3.4 The $H_{1/2}$ preconditioner

The usefulness of any domain decomposition method (DD) rests on the ability to solve a problem involving a pseudo-differential operator: the Steklov-Poincaré operator. Since under discretisation this gives rise to a system with a dense matrix, for large problems this needs to be solved approximately via a procedure that computes the action of the inverse of the discrete operator on a given vector. To this end, a great number of iterative approaches have been suggested in the literature; classical algorithms include Dirichlet-Neumann, Neumann-Neumann, FETI methods, Schwarz methods, together with two-level and overlapping variants. For descriptions and analyses, see [128, 148].

An alternative that has not been considered to date and can be shown to be competitive is based on a well known property of the discrete Steklov-Poincaré operator: it is norm-equivalent to a Sobolev norm-matrix of index 1/2 [128], the discrete representation of which can be written in terms of the square-root of a discrete Laplacian defined on the union of the boundaries of each subdomain [122]. This discrete norm has a non-sparse representation; however, since only the action of its inverse on a vector is required, we can achieve this using a standard approach based on a Krylov subspace approximation. The resulting algorithm is a generalized Lanczos procedure and the ensuing preconditioning procedure is independent of the size of the problem.

3.5 Problem description

We review below the standard formulation of non-overlapping domain decomposition problems for a general scalar elliptic problem.

3.5.1 Notation and definitions

Throughout this chapter we use the following notation and standard results. Given an open simply-connected domain U in $\mathbb{R}^d$, its boundary is denoted by ∂U. We denote by $C_0^\infty(U)$ the space of infinitely differentiable functions defined on U with compact support in U. We also denote by $L^2(U)$ the Lebesgue space of square-integrable functions defined on U endowed with inner-product $(\cdot,\cdot)$, and by $H^m(U)$ the Sobolev space of order m equipped with norm $\|\cdot\|_{m,U}$ and semi-norm $|\cdot|_{m,U}$ with the convention $H^0(U) = L^2(U)$. The Sobolev spaces of real index $0 \le s \le m$ are defined as interpolation spaces of index $\theta = 1 - s/m$ for the pair $[H^m(U), L^2(U)]$:

$$H^s(\Omega) := [H^m(U), L^2(U)]_\theta \qquad \theta = 1 - s/m.$$

For any s, the space $H_0^s(U)$ denotes the completion of $C_0^\infty(U)$ in $H^s(U)$ (see e.g. [104, p 60]). In particular, we shall be interested in the interpolation space

$$H^{1/2}(U) = [H^1(U), L^2(U)]_{1/2},$$

for which there holds $H_0^{1/2}(U) \equiv H^{1/2}(U)$. Another space of interest is $H_{00}^{1/2}(U)$, a subspace of $H_0^{1/2}(U)$ defined as the interpolation space of index 1/2 for the pair $[H_0^1(U), L^2(U)]$:

$$H_{00}^{1/2}(U) = [H_0^1(U), H^0(U)]_{1/2}.$$

Norms on $H^{1/2}(U), H_{00}^{1/2}(U)$ are denoted by the same notation $|\cdot|_{1/2,U}$ or $\|\cdot\|_{1/2,U}$, with the assumption that it is evident from the context which space is under consideration. We return to the definition of these norms in section 3.6. The dual of $H_{00}^{1/2}(U)$ is denoted by $(H_{00}^{1/2}(U))' \subset H^{-1/2}(U)$, where $H^{-1/2}(U) := (H^{1/2}(U))' \equiv (H_0^{1/2}(U))'$. The duality between $H_{00}^{1/2}(U)$ and its dual is denoted by $\langle\cdot,\cdot\rangle$.

Finally, we make use of the trace operator $\gamma_0 : H^1(U) \to H^{1/2}(\partial U)$, which is known to be surjective and continuous, i.e., there exists a constant $c_\gamma(U)$ such that

$$\|\gamma_0 v\|_{1/2,(\partial U)} \le c_\gamma(U)\|v\|_{1,U} \qquad \forall v \in H^1(U). \tag{3.17}$$

A similar inequality holds if we take $\gamma_0 : H_0^1(U) \to H_{00}^{1/2}(\partial U)$:

$$\|\gamma_0 v\|_{1/2,(\partial U)} \le c_\gamma(U)\|v\|_{1,U} \qquad \forall v \in H_0^1(U). \tag{3.18}$$

We also assume that the following Poincaré inequality holds:

$$\|v\|_{0,U} \le C_P(U)|v|_{1,U}. \tag{3.19}$$

3.5.2 Domain decomposition for scalar elliptic PDE

Let Ω denote an open subset of $\mathbb{R}^d$ with boundary $\partial\Omega$ and consider the problem

$$\begin{cases} \mathcal{L}u = -\mathrm{div}(a\nabla u) + \vec{b}\cdot\nabla u + cu &= f \qquad \text{in } \Omega, \\ u &= 0 \qquad \text{on } \partial\Omega, \end{cases} \tag{3.20}$$

where $f \in L^2(\Omega)$, $c \in L^\infty(\Omega)$, $\vec{b}$ is a vector function whose entries are Lipschitz continuous real-valued functions on $\bar{\Omega}$, and a is a symmetric $d \times d$ matrix whose entries are bounded piecewise-continuous real-valued functions defined on $\bar{\Omega}$, with

$$0 < a_{\min} \le \zeta^T a(\mathbf{x})\zeta \le a_{\max} \quad \forall \zeta \in \mathbb{R}^d, \quad \text{a.e. } \mathbf{x} \in \bar{\Omega}. \tag{3.21}$$

We also assume the following standard condition holds:

$$c - \frac{1}{2}\nabla\cdot\vec{b} \ge c_{\min} \text{ a.e. } \mathbf{x} \in \Omega. \tag{3.22}$$

The weak formulation of problem (3.20) reads

$$\begin{cases} \text{Find } u \in H_0^1(\Omega) \text{ such that for all } v \in H_0^1(\Omega), \\ \qquad B(u,v) = (f,v), \end{cases} \tag{3.23}$$

where the bilinear form $B(\cdot,\cdot) : H_0^1(\Omega) \times H_0^1(\Omega) \to \mathbb{R}$ is defined via

$$B(v,w) = (a\nabla v, \nabla w) + \left(\vec{b}\cdot\nabla v + cv, w\right).$$

Let us assume a partitioning of Ω into N nonoverlapping subdomains Ω_i,

$$\bar{\Omega} = \bigcup_{i=1}^N \bar{\Omega}_i, \qquad \Omega_i \cap \Omega_j \equiv \emptyset \ (i \ne j),$$

and let $\Gamma \subset \mathbb{R}^{d-1}$ denote the set of internal boundaries associated with the above partition of Ω:

$$\Gamma = \bigcup_{i=1}^N \Gamma_i \qquad (\Gamma_i := \partial\Omega_i \setminus \partial\Omega).$$

Given a function v defined on Ω we denote by v_i the restriction of v to Ω_i: $v_i = v\mid_{\Omega_i}$. With this notation, we define the bilinear forms $B_i(\cdot,\cdot) : H_0^1(\Omega_i) \times H_0^1(\Omega_i) \rightarrow \mathbb{R}$ similarly to $B(\cdot,\cdot)$:

$$B_i(v_i, w_i) = (a_i \nabla v_i, \nabla w_i) + \left(\vec{b}_i \cdot \nabla v_i + c_i v_i, w_i\right). \tag{3.24}$$

Now let

$$H_D^1(\Omega_i) = \{w \in H^1(\Omega_i) : w\mid_{\partial\Omega\cap\partial\Omega_i} = 0\}$$

and let $v \in H_0^1(\Omega)$. Then $v_i = v\mid_{\Omega_i} \in H_D^1(\Omega_i)$ and there holds

$$B(u,v) = \sum_{i=1}^{N} B_i(u_i, v_i), \qquad (f,v) = \sum_{i=1}^{N}(f, v_i). \tag{3.25}$$

Let u denote the solution of (3.20) and let $u_i = u\mid_{\Omega_i}$. Assuming the value of the exact solution is known on each Γ_i, say $u_i\mid_{\Gamma_i} = \lambda_i$, problem (3.20) can be equivalently be written as a set of problems defined on Ω_i for all i:

$$\left\{\begin{array}{lll} \mathcal{L}u_i & = f & \text{in } \Omega_i, \\ u_i & = 0 & \text{on } \partial\Omega \cap \partial\Omega_i, \\ u_i & = \lambda_i & \text{on } \Gamma_i. \end{array}\right. \tag{3.26}$$

Under the same assumption that λ_i are known, problems (3.26) can be decoupled into two sets of problems:

$$\left\{\begin{array}{lll} \mathcal{L}u_i^{\{1\}} & = f & \text{in } \Omega_i, \\ u_i^{\{1\}} & = 0 & \text{on } \partial\Omega \cap \partial\Omega_i, \\ u_i^{\{1\}} & = 0 & \text{on } \Gamma_i. \end{array}\right. \qquad \left\{\begin{array}{lll} \mathcal{L}u_i^{\{2\}} & = 0 & \text{in } \Omega_i, \\ u_i^{\{2\}} & = 0 & \text{on } \partial\Omega \cap \partial\Omega_i, \\ u_i^{\{2\}} & = \lambda_i & \text{on } \Gamma_i, \end{array}\right. \tag{3.27}$$

with the solution $u\mid_{\Omega_i} = u_i = u_i^{\{1\}} + u_i^{\{2\}}$.

To find an equation for λ_i we integrate the two sets of problems in (3.27) against $v_i \in H_D^1(\Omega_i)$ and obtain the identities

$$B(u_i^{\{1\}}, v_i) = (f, v_i) + \int_{\Gamma_i} \mathbf{n}_i \cdot a_i \nabla u_i^{\{1\}} v_i ds(\Gamma_i),$$
$$B(u_i^{\{2\}}, v_i) = \int_{\Gamma_i} \mathbf{n}_i \cdot a_i \nabla u_i^{\{2\}} v_i ds(\Gamma_i).$$

Adding them up and then summing over i we find (using (3.25))

$$B(u,v) = (f,v) + \sum_{i=1}^{N} \int_{\Gamma_i} \mathbf{n}_i \cdot a_i \nabla u_i^{\{1\}} v_i ds(\Gamma_i) + \sum_{i=1}^{N} \int_{\Gamma_i} \mathbf{n}_i \cdot a_i \nabla u_i^{\{2\}} v_i ds(\Gamma_i),$$

which yields, using (3.23), the *Steklov-Poincaré equation* for the decomposition (3.26):

$$\sum_{i=1}^{N} \int_{\Gamma_i} \mathbf{n}_i \cdot a_i \nabla u_i^{\{2\}} v_i ds(\Gamma_i) = -\sum_{i=1}^{N} \int_{\Gamma_i} \mathbf{n}_i \cdot a_i \nabla u_i^{\{1\}} v_i ds(\Gamma_i).$$

Now let $\eta \in H_{00}^{1/2}(\Gamma)$ with $\eta_i = \eta\,|_{\Gamma_i}$ and let v_i be the solution to the problem

$$\begin{cases} \mathcal{L}v_i &= 0 \quad \text{in } \Omega_i, \\ v_i &= 0 \quad \text{on } \partial\Omega_i \setminus \Gamma_i, \\ v_i &= \eta_i \quad \text{on } \Gamma_i. \end{cases} \tag{3.28}$$

Note that in the case of the Poisson problem where $\mathcal{L} = -\Delta$, the functions v_i are harmonic extensions of η_i to Ω_i. In general, one can view v_i as $\mathcal{L}$-extensions of the corresponding data η_i to Ω_i. Furthermore, the function v defined via $v\,|_{\Omega_i} = v_i$ can be viewed as a generalized $\mathcal{L}$-extension of the function $\eta \in H_{00}^{1/2}(\Gamma)$ to the domain Ω. Henceforth, such generalized $\mathcal{L}$-extensions of functions η or η_i will be denoted by $E\eta$ and $E_i\eta_i$, respectively. Any other harmonic extensions will be denoted by $F\eta$ and $F_i\eta_i$. We also need the following elliptic regularity result, which is known to hold for the weak solution of (3.28) (see for example [1]):

$$\|v_i\|_{1,\Omega_i} = \|E_i\eta_i\|_{1,\Omega_i} \leq C_e \|\eta_i\|_{1/2,\Gamma_i}, \tag{3.29}$$

where C_e is a constant depending only on the domain Ω_i. We define the *Steklov-Poincaré operator* $\mathcal{S} : H_{00}^{1/2}(\Gamma) \to (H_{00}^{1/2}(\Gamma))'$ as follows. Let $\eta, \mu \in H_{00}^{1/2}(\Gamma)$ with $\eta\,|_{\Gamma_i} =: \eta_i, \mu\,|_{\Gamma_i} =: \mu_i$. We define $\mathcal{S}$ via

$$\langle \mathcal{S}\eta, \mu \rangle = \sum_{i=1}^{N} \int_{\Gamma_i} \mathbf{n}_i \cdot a_i \nabla(E\eta_i)\, \mu_i ds(\Gamma_i) =: \sum_{i=1}^{N} \langle \mathcal{S}_i\eta_i, \mu_i \rangle. \tag{3.30}$$

Using integration by parts, we can give the operator $\mathcal{S}$ the following alternative representation:

$$\langle \mathcal{S}\eta, \mu \rangle = B(E\eta, F\mu) = \sum_{i=1}^{N} B_i(E_i\eta_i, F_i\mu_i) \qquad \forall \eta, \mu \in H_{00}^{1/2}(\Gamma). \tag{3.31}$$

Note that these definitions amend those given in [128, pp. 142–143].

With this definition of $\mathcal{S}$, our model problem can be recast as an ordered sequence of three decoupled sets of problems involving the same operator $\mathcal{L}$ with essential boundary conditions on each subdomain together with a problem set on the interface Γ:

$$\begin{aligned}
&\text{(i)}\begin{cases} \mathcal{L}u_i^{\{1\}} &= f \ \text{ in } \Omega_i, \\ u_i^{\{1\}} &= 0 \ \text{ on } \partial\Omega_i. \end{cases} \\
&\text{(ii)}\begin{cases} \mathcal{S}\lambda = -\sum_{i=1}^{N} \mathbf{n}_i \cdot a_i \nabla u_i^{\{1\}} \ \text{ on } \Gamma. \end{cases} \\
&\text{(iii)}\begin{cases} \mathcal{L}u_i^{\{2\}} &= 0 \ \text{ in } \Omega_i, \\ u_i^{\{2\}} &= \lambda_i \ \text{ on } \Gamma_i, \\ u_i^{\{2\}} &= 0 \ \text{ on } \partial\Omega_i \setminus \Gamma_i. \end{cases}
\end{aligned} \tag{3.32}$$

The resulting solution is $u|_{\Omega_i} = u_i^{\{1\}} + u_i^{\{2\}}$.

We now turn to the properties of the interface operator $\mathcal{S}$. Given representation (3.30) we can immediately see that $\mathcal{S}$ is nonsymmetric unless $\vec{b} = 0$. One can show further that $\mathcal{S}$ is a bounded positive operator on $H_{00}^{1/2}(\Gamma)$.

Lemma 3.5.1. *Let $\mathcal{S}$ be defined by* (3.30) *and let* (3.22) *hold. Then there exist constants α_1, α_2 such that for all $\eta, \mu \in H_{00}^{1/2}(\Gamma)$,*

$$\alpha_1 \|\eta\|_{1/2,\Gamma}^2 \leq \langle \mathcal{S}\eta, \eta \rangle, \qquad \langle \mathcal{S}\eta, \mu \rangle \leq \alpha_2 \|\eta\|_{1/2,\Gamma} \|\mu\|_{1/2,\Gamma}.$$

Proof: Let $v_i = E_i\eta_i, w_i = E_i\mu_i$ satisfy (3.28). We have, using (3.22),

$$\begin{aligned}
\langle \mathcal{S}\eta_i, \eta_i \rangle &= B_i(v_i, v_i) \\
&= (a_i \nabla v_i, v_i) + \left(\vec{b} \cdot \nabla v_i, v_i\right) + (c v_i, v_i) \\
&= (a_i \nabla v_i, v_i) + \left((c - \frac{1}{2}\nabla \cdot \vec{b}) v_i, v_i\right) \\
&\geq a_{\min} |v_i|_{1,\Omega_i}^2 + c_{\min} \|v_i\|_{0,\Omega_i}^2 \\
&\geq \min\{a_{\min}, c_{\min}\} \|v_i\|_{1,\Omega_i}^2 .
\end{aligned}$$

Moreover, using the Poincaré inequality (3.19) we get

$$\begin{aligned}\langle S\eta_i, \mu_i\rangle &= B_i(v_i, w_i)\\ &\le a_{\max}|v_i|_{1,\Omega_i}|w_i|_{1,\Omega_i} + \|\vec{b}\|_{L^\infty(\Omega_i)}|v_i|_{1,\Omega_i}\|w_i\|_{0,\Omega_i} + \|c\|_{L^\infty(\Omega_i)}\|v_i\|_{0,\Omega_i}\|w_i\|_{0,\Omega_i}\\ &\le \max\left\{a_{\max} + \|\vec{b}\|_{L^\infty(\Omega_i)}C_P(\Omega_i), \|c\|_{L^\infty(\Omega_i)}\right\}\|v_i\|_{1,\Omega_i}\|w_i\|_{1,\Omega_i}.\end{aligned}$$

Since $\gamma_0 v_i = \eta_i, \gamma_0 w_i = \mu_i$, the trace inequalities (3.17)–(3.18) read for all $i = 1, \dots, N$

$$\|\eta_i\|_{1/2,\Gamma_i} \le C_\gamma(\Omega_i)\|v_i\|_{1,\Omega_i}, \qquad \|\mu_i\|_{1/2,\Gamma_i} \le C_\gamma(\Omega_i)\|w_i\|_{1,\Omega_i}$$

and the result follows from the regularity estimate (3.29) and definition (3.30) of the operator S. ∎

3.6 Finite element discretisations

Let z be an arbitrary function of H_1 having trace at each of the Ω_i λ_i. In order to write down the weak formulation of problems (3.32) we rewrite the set of equations (3.32,(iii)) as

$$\text{(iii)}\left\{\begin{array}{lll} \mathcal{L}\tilde{u}_i^{\{2\}} & = -\mathcal{L}z_i & \text{in } \Omega_i,\\ \tilde{u}_i^{\{2\}} & = 0 & \text{on } \Gamma_i,\\ \tilde{u}_i^{\{2\}} & = 0 & \text{on } \partial\Omega_i \setminus \Gamma_i,\end{array}\right.$$

where $\tilde{u}_i^{\{2\}} = u_i^{\{2\}} - z$. With this new notation, the weak formulations of problems (3.32) are

$$\begin{array}{ll}\text{(i)}\left\{\begin{array}{l}\text{Find } u_i^{\{1\}} \in H_0^1(\Omega_i) \text{ such that for all } v_i \in H_0^1(\Omega_i),\\ B_i(u_i^{\{1\}}, v_i) = (f_i, v_i).\end{array}\right. & \\ \text{(ii)}\left\{\begin{array}{l}\text{Find } \lambda \in H_{00}^{1/2}(\Gamma) \text{ such that for all } \eta \in H_{00}^{1/2}(\Gamma),\\ s(\lambda, \eta) := \langle S\lambda, \eta\rangle = \sum_{i=1}^N \left[(f_i, F_i\eta_i) - B_i(u_i^{\{1\}}, F_i\eta_i)\right].\end{array}\right. & (3.33)\\ \text{(iii)}\left\{\begin{array}{l}\text{Find } \tilde{u}_i^{\{2\}} = u_i^{\{2\}} - z_i \in H_0^1(\Omega_i) \text{ such that for all } v_i \in H_0^1(\Omega_i),\\ B_i(\tilde{u}_i^{\{2\}}, v_i) = -B_i(z_i, v_i).\end{array}\right. & \end{array}$$

Note that $z_i \in H^1(\Omega_i)$ and $z \in H^1(\Omega)$ with $\gamma_0(\Gamma_i)z_i = \lambda_i$.

Let $P_r(\mathrm{t})$ denote the space of polynomials in d variables of degree r defined on a set $\mathrm{t} \subset \mathbf{R}^d$.

Let

$$V_i^h = V_i^{h,r} := \left\{ w \in C^0(\Omega_i) : w|_{\mathfrak{t}} \in P_k \ \forall \mathfrak{t} \in \mathfrak{T}_h, \ w \,|_{\partial\Omega \cap \partial\Omega_i} = 0 \right\} \subset H_D^1(\Omega_i) \tag{3.34}$$

be a finite-dimensional space of piecewise-polynomial functions defined on some subdivision $\mathfrak{T}_h$ of Ω into simplices $\mathfrak{t}$ of maximum diameter h. Further let $V_{iI}^h, V_{iB}^h \subset V_i^h$ satisfy $V_{iI}^h \oplus V_{iB}^h \equiv V_i^h$. Also let

$$V_{iI}^h = \operatorname{span}\left\{ \phi_k^i, k = 1 \dots n_i^I \right\} \quad \text{and} \quad V_{iB}^h = \operatorname{span}\left\{ \psi_k^i, k = 1 \dots n_i^B \right\}$$

and set $n_I = \sum_i n_i^I$. Further let

$$V_B^h = \bigcup_{i=1}^{N} V_{iB}^h$$

and let $\{\psi_k, k = 1, \dots, n_B\}$ denote a basis for V_B^h. Let $S_i^h = \operatorname{span}\left\{ \gamma_0(\Gamma_i)\psi_k, \ k = 1, \dots, n_i^B \right\}$ with $S^h = \cup_{i=1}^N S_i^h$. Finally, let

$$V^h = \bigcup_{i=1}^{N} V_i^h \subset H_0^1(\Omega).$$

The finite element discretization of the weak formulation (3.23) reads

$$\left\{ \begin{array}{l} \text{Find } u_h \in V^h \text{ such that for all } v_h \in V^h, \\ B(u_h, v_h) = (f, v_h). \end{array} \right. \tag{3.35}$$

The finite element discretization of the weak formulations (3.33) are as follows:

$$\begin{array}{ll} \text{(i)} & \left\{ \begin{array}{l} \text{Find } u_{hi}^{\{1\}} \in V_{iI}^h \text{ such that for all } v_{hi} \in V_{iI}^h, \\ B_i(u_{hi}^{\{1\}}, v_{hi}) = (f_i, v_{hi}). \end{array} \right. \\ \text{(ii)} & \left\{ \begin{array}{l} \text{Find } \lambda_h \in S^h \text{ such that for all } \eta_h \in S^h, \\ s(\lambda_h, \eta_h) = \sum_{i=1}^N \left[(f_i, F_i\eta_{hi}) - B_i(u_{hi}^{\{1\}}, F_i\eta_{hi})) \right]. \end{array} \right. \\ \text{(iii)} & \left\{ \begin{array}{l} \text{Find } \tilde{u}_{hi}^{\{2\}} = u_{hi}^{\{2\}} - z_{hi} \in V_{iI}^h \text{ such that for all } v_{hi} \in V_{iI}^h, \\ B_i(\tilde{u}_{hi}^{\{2\}}, v_{hi}) = -B_i(z_{hi}, v_{hi}). \end{array} \right. \end{array} \tag{3.36}$$

3.6.1 Matrix formulation

Formulation (3.36) amounts to a Schur-complement approach for the solution of the discrete weak formulation (3.35). For completeness of exposition we include this characterization below. Let

$$A\mathbf{u} = \begin{pmatrix} A_{II} & A_{IB} \\ A_{BI} & A_{BB} \end{pmatrix} \begin{pmatrix} \mathbf{u}_I \\ \mathbf{u}_B \end{pmatrix} = \begin{pmatrix} \mathbf{f}_I \\ \mathbf{f}_B \end{pmatrix} = \mathbf{f} \tag{3.37}$$

represent the linear system associated with the discrete formulation (3.35) with $A \in \mathbb{R}^{n \times n}$, $\mathbf{u} \in \mathbb{R}^n$ where $n = n_I + n_B$ and $A_{II} \in \mathbb{R}^{n_I \times n_I}$, $A_{IB}, A_{BI}^T \in \mathbb{R}^{n_I \times n_B}$, $A_{BB} \in \mathbb{R}^{n_B \times n_B}$ are given by

$$A_{II} = \begin{pmatrix} A_{II}^1 & & & & \\ & \ddots & & & \\ & & A_{II}^i & & \\ & & & \ddots & \\ & & & & A_{II}^N \end{pmatrix}, \quad A_{IB} = \begin{pmatrix} A_{IB}^1 \\ \vdots \\ A_{IB}^i \\ \vdots \\ A_{IB}^N \end{pmatrix}, \quad A_{BI} = \begin{pmatrix} A_{BI}^1 & \cdots & A_{BI}^i & \cdots & A_{BI}^N \end{pmatrix}$$

with

$$\begin{aligned} (A_{II}^i)_{kk} &= B_i(\phi_k^i, \phi_k^i), \\ (A_{IB}^i)_{kj} &= B_i(\phi_k^i, \psi_j^i), \\ (A_{BI}^i)_{jk} &= B_i(\psi_l^i, \phi_l^i), \\ (A_{BB})_{ll} &= B(\psi_l, \psi_l), \end{aligned}$$

for all $k = 1, \ldots, n_i^I$, $j = 1, \ldots, n_i^B$, $l = 1, \ldots, n_B$.

Lemma 3.6.1. *In the above notation, the solution of problems* (3.28) *has finite element coefficients*

$$\mathbf{v} = \begin{pmatrix} \mathbf{v}_I \\ \mathbf{v}_B \end{pmatrix} = \begin{pmatrix} -A_{II}^{-1} A_{IB} \mathbf{v}_B \\ \mathbf{v}_B \end{pmatrix},$$

where $\mathbf{v}_B$ *are the coefficients of* η *with respect to the basis of* V_B^h.

Proof: We start by considering the weak formulation of problems (3.28). If we let $w_i|_{\Gamma_i} = \eta_i$ we can re-write our problems as

$$\left\{ \begin{array}{lll} \mathcal{L}\tilde{v}_i & = -\mathcal{L}z_i & \text{in } \Omega_i, \\ \tilde{v}_i & = 0 & \text{on } \partial\Omega_i \setminus \Gamma_i, \\ \tilde{v}_i & = 0 & \text{on } \Gamma_i, \end{array} \right. \tag{3.38}$$

where we set $\tilde{v}_i = v_i - z_i$. These problems have the following discrete weak formulation:

$$\begin{cases} \text{Find } \tilde{v}_{hi} = v_{hi} - z_{hi} \in V_{iI}^h \text{ such that for all } w_{hi} \in V_{iI}^h, \\ B_i(\tilde{v}_{hi}, w_{hi}) = -B_i(z_{hi}, w_{hi}), \end{cases}$$

so the global matrix representation resulting after summing over i is

$$\begin{pmatrix} \mathbf{w}_I^T & 0 \end{pmatrix} \begin{pmatrix} A_{II} & A_{IB} \\ A_{BI} & A_{BB} \end{pmatrix} \begin{pmatrix} \mathbf{v}_I - \mathbf{z}_I \\ 0 \end{pmatrix} = -\begin{pmatrix} \mathbf{w}_I^T & 0 \end{pmatrix} \begin{pmatrix} A_{II} & A_{IB} \\ A_{BI} & A_{BB} \end{pmatrix} \begin{pmatrix} \mathbf{z}_I \\ \mathbf{v}_B \end{pmatrix}$$

and the statement of the lemma follows. ■

Consider now the matrix representation of operator S in the basis $\{\psi_k, k = 1, \ldots, n_B\}$. The discrete form of definition (3.31) is

$$s(\eta_h, \mu_h) = B_i(E_i\eta_{ih}, F_i\mu_{ih}), \tag{3.39}$$

where we recall that F_i is an arbitrary extension operator to Ω_i while E_i is an $\mathcal{L}$-extension to the same domain. If we set $v\,|_{\Gamma_i} = \eta_{ih}$, $w\,|_{\Gamma_i} = \mu_{ih}$, the corresponding discrete representations of these extensions will have the form (using the Lemma 3.6.1)

$$E_i\eta_{ih} = \begin{pmatrix} -A_{II}^{-1}A_{IB}\mathbf{v}_B \\ \mathbf{v}_B \end{pmatrix}, \quad F_i\mu_{ih} = \begin{pmatrix} \mathbf{w}_I \\ \mathbf{w}_B \end{pmatrix}.$$

Hence (3.39) has the representation

$$\mathbf{w}_B^T S \mathbf{v}_B = \begin{pmatrix} \mathbf{w}_I^T & \mathbf{w}_B^T \end{pmatrix} \begin{pmatrix} A_{II} & A_{IB} \\ A_{BI} & A_{BB} \end{pmatrix} \begin{pmatrix} -A_{II}^{-1}A_{IB}\mathbf{v}_B \\ \mathbf{v}_B \end{pmatrix},$$

which is equivalent to

$$\mathbf{w}_B^T S \mathbf{v}_B = \mathbf{w}_B^T (A_{BB} - A_{BI}A_{II}^{-1}A_{IB})\mathbf{v}_B$$

for all $\mathbf{v}_B, \mathbf{w}_B \in \mathbb{R}^{n_B \times n_B}$, so that S is the Schur complement of A_{BB} in the global matrix A. With this notation, (3.36 (i)) has the following matrix formulation:

$$\text{(i) } A_{II}\mathbf{u}_I^{\{1\}} = \mathbf{f}_I,$$

while (3.36 (ii)) becomes

$$\text{(ii) } \mathbf{w}_B^T S \mathbf{u}_B = \begin{pmatrix} \mathbf{w}_I^T & \mathbf{w}_B^T \end{pmatrix} \begin{pmatrix} \mathbf{f}_I \\ \mathbf{f}_B \end{pmatrix} - \begin{pmatrix} \mathbf{w}_I^T & \mathbf{w}_B^T \end{pmatrix} \begin{pmatrix} A_{II} & A_{IB} \\ A_{BI} & A_{BB} \end{pmatrix} \begin{pmatrix} \mathbf{u}_I^{\{1\}} \\ 0 \end{pmatrix},$$

for any $\mathbf{w}_i \in \mathbb{R}^{n_I}, \mathbf{w}_B \in \mathbb{R}^{n_B}$; this equation simplifies to

$$\text{(ii) } S\mathbf{u}_B = \mathbf{f}_B - A_{BI}\mathbf{u}_I^{\{1\}}.$$

Finally, from Lemma 3.6.1 the discrete form of (3.36 (iii)) is seen to be

$$\text{(iii) } \mathbf{u}_I^{\{2\}} = -A_{II}^{-1} A_{IB} \mathbf{u}_B.$$

These equations are easily seen to represent a Schur-complement approach for the original linear system (3.37) with global solution u given by

$$\mathbf{u} = \begin{pmatrix} \mathbf{u}_I \\ \mathbf{u}_B \end{pmatrix} = \begin{pmatrix} \mathbf{u}_I^{\{1\}} \\ \mathbf{u}_B^{\{1\}} \end{pmatrix} + \begin{pmatrix} \mathbf{u}_I^{\{1\}} \\ \mathbf{u}_B^{\{1\}} \end{pmatrix}.$$

3.7 Preconditioners for the Steklov-Poincaré operator

The result of Lemma 3.5.1 holds also in the discrete case for the choice of space S_h introduced in the previous section. In particular it translates into the following coercivity and continuity bounds for $s(\cdot,\cdot) : S_h \times S_h$.

Lemma 3.7.1. *Let* $s(\cdot,\cdot)$ *be defined as in* (3.33) *and let* (3.22) *hold. Then there exist constants* α_1, α_2 *such that for all* $\eta_h, \mu_h \in S_h \subset H_{00}^{1/2}(\Gamma)$,

$$\alpha_1 \|\eta_h\|_{1/2,\Gamma}^2 \leq s(\eta_h, \eta_h), \qquad s(\eta_h, \mu_h) \leq \alpha_2 \|\eta_h\|_{1/2,\Gamma} \|\mu_h\|_{1/2,\Gamma}.$$

We realize from the above discussion that the continuous formulation of the domain decomposition problem is intrinsically related to the Schur complement operator acting on the space $H_{00}^{1/2}(\Gamma)$; a space lying between $H_0^1(\Gamma)$ and $L^2(\Gamma)$. In order to exploit the results of (3.5.1), (3.7.1) for preconditioning purposes, we need to derive corresponding matrix formulations. For this purpose it is essential to familiarize ourselves with the concept of interpolation of Hilbert spaces and their related norms, as well as discrete norms equivalent to the continuous ones, which will finally provide the sought optimal preconditioners. We proceed by recalling related results presented in [4].

3.8 Interpolation spaces

Let $\mathcal{L}, M$ denote two Hilbert spaces with $\mathcal{L} \subset \mathcal{M}$, $\mathcal{L}$ dense in $\mathcal{M}$ with continuous injection. Let $\langle\cdot,\cdot\rangle_{\mathcal{L}}, \langle\cdot,\cdot\rangle_{\mathcal{M}}$ denote the corresponding inner products, and $\|\cdot\|_{\mathcal{L}}, \|\cdot\|_{\mathcal{M}}$ the respective norms. By the Riesz representation theory (see for example [129]) there exists an operator $\mathfrak{J} : \mathcal{L} \to \mathcal{M}$ that is positive and self-adjoint with respect to $\langle\cdot,\cdot\rangle_{\mathcal{M}}$ such that

$$\langle u, v\rangle_{\mathcal{L}} = \langle u, \mathfrak{J}v\rangle_{\mathcal{M}} . \tag{3.40}$$

Using the spectral decomposition of $\mathfrak{J}$ we define the operator

$$\mathfrak{E} = \mathfrak{J}^{1/2} : \mathcal{L} \to \mathcal{M}, \tag{3.41}$$

which in turn is positive self-adjoint. Moreover (see [104, Chapter 1, Section 2.2]), the space $\mathcal{L}$ can be defined to be the domain $D(\mathfrak{E})$ of $\mathfrak{E}$, and the norm of $\mathcal{L}$ is equivalent to the graph norm $\|\cdot\|_{\mathfrak{E}}$:

$$\|u\|_{\mathcal{L}} \sim \|u\|_{\mathfrak{E}} := \left(\|u\|_{\mathcal{M}}^2 + \|\mathfrak{E}u\|_{\mathcal{M}}^2\right)^{1/2} .$$

Similarly, the spectral decomposition of $\mathfrak{E}$ can be used to define any real power of $\mathfrak{E}$. We can now introduce the **interpolation** or *intermediate* spaces of index θ taken from [104].

Definition 1. *Interpolation space of index $\theta \in [0,1]$ for the pair of Hilbert spaces $[\mathcal{L}, \mathcal{M}]$ with $\mathcal{L} \subset \mathcal{M}$, dense and continuously injected in $\mathcal{M}$, is the domain of $\mathfrak{E}^{1-\theta}$ denoted by $[\mathcal{L}, \mathcal{M}]_\theta$, with $\mathfrak{E}$ defined by (3.41), endowed with the inner-product*

$$\langle u, v\rangle_\theta = \langle u, v\rangle_{\mathcal{M}} + \left\langle \mathfrak{E}^{1-\theta}u, \mathfrak{E}^{1-\theta}v\right\rangle_{\mathcal{M}} ,$$

and corresponding norm the graph norm of $\mathfrak{E}^{1-\theta}$:

$$\|u\|_\theta := \left(\|u\|_{\mathcal{M}}^2 + \|\mathfrak{E}^{1-\theta}u\|_{\mathcal{M}}^2\right)^{1/2} . \tag{3.42}$$

In [104] the authors also prove that the space $[\mathcal{L}, \mathcal{M}]_\theta$ is a Hilbert space. We should remark that $[\mathcal{L}, \mathcal{M}]_0 = \mathcal{L}$ and $[\mathcal{L}, \mathcal{M}]_1 = \mathcal{M}$. Moreover, if $0 < \theta_1 < \theta_2 < 1$ then

$$\mathcal{L} \subset [\mathcal{L}, \mathcal{M}]_{\theta_1} \subset [\mathcal{L}, \mathcal{M}]_{\theta_2} \subset \mathcal{M}. \tag{3.43}$$

Now let $\mathfrak{L}(A; B)$ denote the space of continuous linear operators from A into B. The following classic interpolation theorem can be found in [104, Theorem 5.1].

Theorem 3.8.1. *Let $\mathcal{L}_1, \mathcal{M}_1$ be defined as above and let $\mathcal{L}_2, \mathcal{M}_2$ satisfy similar properties. Let $\pi \in \mathfrak{L}(\mathcal{L}_1; \mathcal{L}_2) \cap \mathfrak{L}(\mathcal{M}_1; \mathcal{M}_2)$. Then for all $\theta \in (0,1)$,*

$$\pi \in \mathfrak{L}([\mathcal{L}_1, \mathcal{M}_1]_\theta; [\mathcal{L}_2, \mathcal{M}_2]_\theta).$$

We turn now to the case where the spaces generating the scale of interpolation spaces are finite-dimensional. In particular, we are interested in the discrete norms associated with these spaces.

3.8.1 Finite-dimensional interpolation spaces.

Let $\mathcal{L}_h \subset \mathcal{L}, \mathcal{M}_h \subset \mathcal{M}$ denote two finite-dimensional subspaces of $\mathcal{L}, \mathcal{M}$ respectively, with $n = \dim \mathcal{L}_h = \dim \mathcal{M}_h$. They are Hilbert spaces when endowed with the inner-products $\langle \cdot, \cdot \rangle_{\mathcal{L}}, \langle \cdot, \cdot \rangle_{\mathcal{M}}$. We can similarly define corresponding positive, self-adjoint operators $\mathfrak{J}_h, \mathfrak{C}_h : \mathcal{L}_h \to \mathcal{M}_h$ satisfying

$$\langle u_h, v_h \rangle_{\mathcal{L}} = \langle u_h, \mathfrak{J}_h v_h \rangle_{\mathcal{M}} \qquad u_h, v_h \in \mathcal{L}_h, \tag{3.44}$$

where $\mathfrak{J}_h$ is positive self-adjoint and $\mathfrak{C}_h = \mathcal{J}_h^{1/2}$. We define the discrete interpolation spaces

$$[\mathcal{L}_h, \mathcal{M}_h]_\theta := D(\mathfrak{C}_h^{1-\theta}).$$

Furthermore, we define the scale of discrete norms

$$\|u_h\|_{\theta,h} := \left(\|u_h\|_{\mathcal{M}}^2 + \|\mathfrak{C}_h^{1-\theta} u_h\|_{\mathcal{M}}^2\right)^{1/2}. \tag{3.45}$$

By employing Thm. 3.8.1 the authors in [4] prove the following.

Lemma 3.8.2. *Let $\mathcal{L}_h \subset \mathcal{M}_h, \mathcal{L} \subset \mathcal{M}$ be Hilbert spaces with inner-products $\langle \cdot, \cdot \rangle_{\mathcal{L}}, \langle \cdot, \cdot \rangle_{\mathcal{M}}$ and let $\|\cdot\|_\theta, \|\cdot\|_{\theta,h}$ be defined by* (3.42), (3.45), *respectively. Let us assume that there exists an interpolation operator I_h such that $I_h : \mathfrak{L}(\mathcal{L}; \mathcal{L}_h) \cap \mathfrak{L}(\mathcal{M}; \mathcal{M}_h)$ and $I_h u_h = u_h$ for all $u_h \in \mathcal{L}_h$. Then the norms $\|\cdot\|_\theta, \|\cdot\|_{\theta,h}$ are equivalent on $[\mathcal{L}_h, \mathcal{M}_h]_\theta$ for all $\theta \in (0,1)$.*

3.8.2 Discrete fractional Sobolev norms

We are interested in describing the set of symmetric and positive definite matrices

$$\{H_\theta \in \mathbb{R}^{n \times n} : n \in \mathbb{N},\ 0 \le \theta \le 1\},$$

which induce norms equivalent to $\|\cdot\|_{\theta,h}$ with constants of equivalence independent of n. Let L, M denote the Grammian (or Riesz) matrices corresponding to the inner products $\langle \cdot, \cdot \rangle_{\mathcal{L}}, \langle \cdot, \cdot \rangle_{\mathcal{M}}$. More precisely,

$$L_{ij} = \langle \phi_i, \phi_j \rangle_{\mathcal{L}}, \qquad M_{ij} = \langle \phi_i, \phi_j \rangle_{\mathcal{M}}, \qquad 1 \le i, j \le n, \tag{3.46}$$

where $\{\phi_i\}_{1 \le i \le n}$ denotes a basis of $\mathcal{L}_h$, so that

$$\|u_h\|_{\mathcal{L}} = \|\mathbf{u}\|_L, \qquad \|u_h\|_{\mathcal{M}} = \|\mathbf{u}\|_M,$$

with $\mathbf{u}$ the vector of coefficients of u_h expanded in the basis $\{\phi_i\}$. We first note that the discrete Riesz representation (3.44) becomes

$$\mathbf{u}^T L \mathbf{v} = \mathbf{u}^T M J \mathbf{v}$$

so that $J = M^{-1}L$ is a product of two symmetric and positive definite matrices. In the sequel we will need to define real powers of these matrices. Suppose $H \in \mathbb{R}^{n\times n}$ is a Diagonalizable matrix with a real Positive-definite Spectrum (DPS). Let the eigenvalue decomposition of H be denoted by $H = V^{-1} D_H V$, where D_H is a diagonal matrix with entries λ_i satisfying

$$0 < \lambda_1 < \lambda_2 < \ldots < \lambda_i < \ldots \lambda_n.$$

The following definition [78, 100] will prove very useful in the subsequent discussion.

Definition 2. *Let $\theta \in \mathbb{R}$. The θ power of a DPS matrix $H = V^{-1} D_H V$ is defined via the decomposition $H^\theta := V^{-1} D_H^\theta V$.*

The matrix J will play a key role for several of our results. This matrix is self-adjoint and positive definite in the discrete M-inner-product, because

$$\langle \mathbf{u}, J\mathbf{v} \rangle_M = \mathbf{u}^T M (M^{-1}L)\mathbf{v} = \mathbf{v}^T L \mathbf{u} = \mathbf{v}^T M (M^{-1}L)\mathbf{u} = \langle \mathbf{v}, J\mathbf{u} \rangle_M$$

and

$$\langle \mathbf{u}, J\mathbf{u} \rangle_M = \mathbf{u}^T L \mathbf{u} > 0 \quad \text{for all } \mathbf{u} \neq 0.$$

One can also write explicitly the eigenvalue decomposition of J: since L, M are symmetric and positive-definite, there exists a matrix Q such that ([79, Cor 7.6.2])

$$L = Q^T D Q, \quad M = Q^T Q,$$

where D is a diagonal matrix with positive entries, so that

$$J = M^{-1}L = Q^{-1}DQ.$$

Note that this implies that J is DPS, so that real powers of J can be defined via Definition 2. It is evident that the matrix representation of $\mathfrak{E}_h$ in the basis $\{\phi_i\}$ is the DPS matrix

$$E = Q^{-1}D^{1/2}Q;$$

furthermore, the matrix representation of $\mathfrak{E}_h^{1-\theta}$ is similarly

$$E^{1-\theta} = Q^{-1}D^{(1-\theta)/2}Q.$$

We now turn to the matrix representation of the norm $\|\cdot\|_{\theta,h}$, which we denote by $H_{\theta,h}$. Definition (3.45) yields

$$\|\mathbf{u}\|^2_{H_{\theta,h}} = \|\mathbf{u}\|^2_M + \|E^{1-\theta}\mathbf{u}\|^2_M$$

so that

$$H_{\theta,h} = M + (E^{1-\theta})^T M E^{1-\theta} = M + Q^T D^{1-\theta} Q = M + MJ^{1-\theta} = M + M(M^{-1}L)^{1-\theta}. \tag{3.47}$$

Let us consider now the reduced version of $H_{\theta,h}$:

$$H_\theta = M(M^{-1}L)^{1-\theta}. \tag{3.48}$$

The following proposition allows us to consider the reduced form H_θ as an alternative of $H_{\theta,h}$.

Proposition 3.8.3. *The matrix $H_\theta = M(M^{-1}L)^{1-\theta}$, the reduced form of $H_{\theta,h}$, is equivalent to it with constants of equivalence independent of n.*

Proof: By definition, the norm of $\mathcal{L}_h$ is equivalent to the discrete graph norm (3.45) with $\theta = 0$:

$$Q^T DQ = L \sim H_{0,h} = M + MJ = Q^T(I + D)Q.$$

Hence, there exist two positive real constants α_1, α_2 independent of n such that

$$\alpha_1 D_{ii} \leq (1 + D_{ii}) \leq \alpha_2 D_{ii} \qquad 1 \leq i \leq n.$$

It follows that, setting $\tilde{\alpha}_1 = 1, \tilde{\alpha}_2 = \max\{\alpha_2, 2\}$, there holds

$$\tilde{\alpha}_1 D_{ii}^{1-\theta} \leq (1 + D_{ii}^{1-\theta}) \leq \tilde{\alpha}_2 D_{ii}^{1-\theta} \qquad 1 \leq i \leq n$$

and we deduce that

$$H_{\theta,h} \sim H_\theta := Q^T D^{1-\theta} Q = M J^{1-\theta} = M(M^{-1}L)^{1-\theta}. \tag{3.49}$$

■

Corollary 3.8.4. *Let the assumptions of Lemma* 3.8.2 *hold. Let H_θ be defined as in* (3.49). *Then the norms $\|\cdot\|_\theta, \|\cdot\|_{H_\theta}$ are equivalent on $[\mathcal{L}_h, \mathcal{M}_h]_\theta$ for all $\theta \in (0,1)$.*

In the following, we assume that Γ is the union of planar (straight) faces (segments) Γ_i. We let $\mathcal{M}$ be the space of square-integrable functions defined on Γ and let ∇_Γ denote the tangential gradient of a scalar function $v(\mathbf{x}) : \Omega$:

$$\nabla_\Gamma v(\mathbf{x}) := \nabla v(\mathbf{x}) - \mathbf{n}(\mathbf{n} \cdot \nabla v(\mathbf{x})),$$

i.e., the projection of the gradient of v onto the plane tangent to Γ at $\mathbf{x} \in \Gamma$. We define $\mathcal{L}$ as follows:

$$\mathcal{L} = H_0^1(\Gamma) := \left\{ v \in L^2(\Gamma) : \int_\Gamma |\nabla_\Gamma v|^2 \, ds(\Gamma) < \infty, \; v|_{\Gamma \cap \partial\Omega} = 0 \right\}.$$

We endow $\mathcal{L}$ with the norm

$$|v|_{H_0^1(\Gamma)} = \|\nabla_\Gamma v\|_{L^2(\Gamma)}.$$

Now let $\mathcal{L}_h = (S_h, \|\cdot\|_{H_0^1(\Gamma)}) \subset \mathcal{L}$ and $\mathcal{M}_h = (S_h, \|\cdot\|_{L^2(\Gamma)}) \subset \mathcal{M}$. The norms (3.47), (3.48) for the discrete interpolation space $[\mathcal{L}_h, \mathcal{M}_h]_{1/2}$ have the matrix representations

$$H_{1/2,h} = M + M(M^{-1}L)^{1/2},$$
$$H_{1/2} = M(M^{-1}L)^{1/2},$$

where

$$M_{ij} = (\psi_i, \psi_j)_{L^2(\Gamma)}, \qquad L_{ij} = (\nabla_\Gamma \psi_i, \nabla_\Gamma \psi_j)_{H_0^1(\Gamma)},$$

with $\psi_i \in S^h$ for $i = 1, \ldots, n_B$. With this notation, Lemma 3.8.2 applies with I_h the finite element projector onto S^h. Thus, for all $\lambda_h \in S^h$ with $\lambda_h = \sum_{i=1}^{n_B} \lambda_i \psi_i$ there exist constants κ_1, κ_2 such that $\kappa_1 \|\lambda_h\|_{1/2,\Gamma} \leq \|\lambda\|_{H_{1/2}} \leq \kappa_2 \|\lambda_h\|_{1/2,\Gamma}$. We immediately derive the following result.

Proposition 3.8.5. *Let $s(\cdot,\cdot)$ be defined as in (3.33) and let (3.22) hold. Let $\boldsymbol{\eta}, \boldsymbol{\mu}$ denote the coefficients of η_h, μ_h with respect to the basis $\{\psi_i,\ i = 1, \ldots, n_B\}$ of S^h. Let S denote the matrix representation of $s(\cdot,\cdot)$ with respect to the same basis. Then there exist constants $\tilde{\alpha}_1, \tilde{\alpha}_2$ such that*

$$\tilde{\alpha}_1 \|\boldsymbol{\eta}\|^2_{H_{1/2}} \leq \boldsymbol{\eta}^T S \boldsymbol{\eta}, \qquad \boldsymbol{\mu}^T S \boldsymbol{\eta} \leq \tilde{\alpha}_2 \|\boldsymbol{\eta}\|_{H_{1/2}} \|\boldsymbol{\mu}\|_{H_{1/2}}$$

for all $\eta_h, \mu_h \in S_h \subset H^{1/2}_{00}(\Gamma)$.

We will see below that the above equivalence indicates that $H_{1/2,h}$ and the equivalent norm $H_{1/2}$ can be both used as a preconditioner for the Schur-complement system and that they are optimal in some sense to be described.

3.8.3 Optimal preconditioners

The solution of linear system (3.37) requires an iterative approach in the case of large-scale problems. A useful approach is to consider an iterative solver such as GMRES together with a suitable preconditioning strategy. In our case, one could for example employ a right preconditioner that will incorporate the solution of problems posed on the interior of each domain (achieved in parallel) and the (approximate) solution of a problem involving the discrete Steklov-Poincaré operator. Given the equivalence in Proposition 3.8.5, a candidate right-preconditioner is

$$P_R = \begin{pmatrix} A_{II} & A_{IB} \\ 0 & H_{1/2} \end{pmatrix}.$$

With this choice, the preconditioned system is

$$AP_R^{-1} = \begin{pmatrix} I & 0 \\ A_{BI}A_{II}^{-1} & SH_{1/2}^{-1} \end{pmatrix}.$$

This block structure indicates that the convergence of an iterative algorithm such as GMRES will depend on the ability of $H_{1/2}$ to approximate S. In particular, the eigenvalues of the above preconditioned matrix are either equal to one or coincide with one of the eigenvalues of $SH_{1/2}^{-1}$. In fact, these eigenvalues lie in a region of the complex plane that is independent of the size of the problem and also lies in the right half-plane. To see this, we recall the definition of the H-field of values of a matrix A, given a symmetric and positive-definite matrix H.

Definition 3. *Let $R, H \in \mathbb{R}^{n\times n}$ with H symmetric and positive definite. The H-field of values of the matrix R, denoted by $\mathcal{W}_H(R)$, is a set in the complex plane given by*

$$\mathcal{W}_H(R) = \left\{ z \in \mathbb{C} : z = \frac{\mathbf{x}^* H R \mathbf{x}}{\mathbf{x}^* H \mathbf{x}} = \frac{\langle \mathbf{x}, R\mathbf{x}\rangle_H}{\langle \mathbf{x}, \mathbf{x}\rangle_H},\ \mathbf{x} \in \mathbb{C}^n \setminus \{0\} \right\}.$$

When $H = I$, the set is called the field of values and is denoted by $\mathcal{W}(R)$.

We also need to recall a related result concerning the convergence of GMRES (see [56, 132]).

Lemma 3.8.6. *Let $H \in \mathbb{R}^{n\times n}$ be a positive-definite matrix. Let $R, P \in \mathbb{R}^{n\times n}$ be nonsingular matrices such that the following bounds hold:*

$$\xi_1 \le \frac{\langle \mathbf{x}, RP^{-1}\mathbf{x}\rangle_H}{\langle \mathbf{x}, \mathbf{x}\rangle_H}, \qquad \frac{\|RP^{-1}\mathbf{x}\|_H}{\|\mathbf{x}\|_H} \le \xi_2 \tag{3.50}$$

for some positive constants ξ_1 and ξ_2. Then the GMRES algorithm in the H-inner product yields a residual $\mathbf{r}^k$ after k iterations that satisfies

$$\frac{\|\mathbf{r}^k\|_H}{\|\mathbf{r}^0\|_H} \le \left(1 - \frac{\xi_1^2}{\xi_2^2}\right)^{k/2}. \tag{3.51}$$

The following result provides bounds on the $H_{1/2}^{-1}$-field of values.

Proposition 3.8.7. *Let the hypothesis of Proposition 3.8.5 hold. Then the $H_{1/2}^{-1}$-field of values of $SH_{1/2}^{-1}$ is in the right half-plane and is bounded independently of n_B.*

Proof: The projection on the real line of the $H_{1/2}^{-1}$-field of values is bounded from below by

$$\min_{z\in\mathcal{W}_{H_{1/2}}(SH_{1/2}^{-1})} = \min_{\eta\in\mathbb{R}^{n_B}\setminus\{0\}} \frac{\left\langle \eta, SH_{1/2}^{-1}\eta \right\rangle_{H_{1/2}^{-1}}}{\langle \eta, \eta\rangle_{H_{1/2}^{-1}}} = \min_{\eta\in\mathbb{R}^{n_B}\setminus\{0\}} \frac{\eta^T S \eta}{\eta^T H_{1/2}\eta} \ge \tilde{\alpha}_1 > 0.$$

An upper bound for the field of values is provided by the numerical radius, which in turn is bounded by the maximum $H_{1/2}^{-1}$-singular value. The resulting bound on the $H_{1/2}^{-1}$-field of values of $SH_{1/2}^{-1}$ is

$$|z| \le \max_{\eta\in\mathbb{R}^{n_B}\setminus\{0\}} \frac{\|S\eta\|_{H_{1/2}^{-1}}}{\|\eta\|_{H_{1/2}}} = \max_{\eta\in\mathbb{R}^{n_B}\setminus\{0\}} \max_{\mu\in\mathbb{R}^{n_B}\setminus\{0\}} \frac{\eta S \mu}{\|\eta\|_{H_{1/2}}\|\mu\|_{H_{1/2}}} \le \tilde{\alpha}_2.$$

■

Note that the above bounds also imply the following bounds independent of n_B on the eigenvalues of the preconditioned discrete Steklov-Poincaré operator:

$$\tilde{\alpha}_1 \leq \left|\lambda(SH_{1/2}^{-1})\right| \leq \tilde{\alpha}_2.$$

Given the result of Proposition 3.8.5, a convergence bound can be immediately derived for a system of equations involving the matrix $SH_{1/2}^{-1}$.

Proposition 3.8.8. *Let the hypothesis of Proposition* 3.8.5 *hold. Then GMRES applied to the linear system*

$$SH_{1/2}^{-1}\tilde{\mathbf{y}} = \mathbf{z}, \qquad (\tilde{\mathbf{y}} = H_{1/2}\mathbf{y})$$

in the $H_{1/2}^{-1}$-inner product yields a residual $\mathbf{r}^k$ after k iterations that satisfies

$$\frac{\|\mathbf{r}^k\|_{H_{1/2}}}{\|\mathbf{r}^0\|_{H_{1/2}}} \leq \left(1 - \frac{\tilde{\alpha}_1^2}{\tilde{\alpha}_2^2}\right)^{k/2}. \tag{3.52}$$

The following result is adapted from [109, Thm 3.7].

Proposition 3.8.9. *Let the hypothesis of Proposition* 3.8.5 *hold and let P_R be given by*

$$P_R = \begin{pmatrix} A_{II} & A_{IB} \\ 0 & \rho H_{1/2} \end{pmatrix}. \tag{3.53}$$

Then there exists $\rho_0 > 0$ such that for all $\rho > \rho_0$ conditions (3.50) *hold with R, P replaced by A, P_R and for the choice*

$$H = \begin{pmatrix} A_{II} & 0 \\ 0 & H_{1/2} \end{pmatrix}^{-1}.$$

As before, this result indicates that block triangular preconditioners $P_R(\rho)$ are optimal preconditioners when we use a suitable GMRES iteration to solve the *global linear system.*

3.9 Evaluation of H_θ

The evaluation of H_θ consists of two steps. The first step involves the assembly of the matrices L, M (3.46). These matrices are assembled on the interface of the subdomains, which in a triangular mesh consists of segments and in a tetrahedral mesh consists of triangles. Unlike standard finite element meshes, these segments have vertices with two space coordinates and the triangles have vertices with three space coordinates. The finite element assembly of operators acting on interfaces thus involves a quite non-standard process, explained in what follows.

3.9.1 Assembly on a 2D interface

Let as assume an arbitrary segment on the xy plane and let (x_1, y_1) and (x_2, y_2) be coordinates of its vertices. Our purpose is to obtain the corresponding linear shape functions defined on that segment. As soon as we have them, the rest follow easily as in the case of standard 1D finite elements. For this purpose we need to construct the mapping of any 2D segment to the reference segment defined by the line from (0,0) to (1,0) along the x-axis. We perform a translation of one end of the segment to the origin, followed by a clockwise rotation of the segment around the z-axis to align it with the x-axis. The segment is then scaled to be of unit length. This mapping will provide the sought shape functions defined on 2D segments.

Translation to the origin

Let P define the affine coordinates of the end-points of a general segment in the xy plane. Then P and the required translation matrix T are as follows:

$$P = \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \\ 1 & 1 \end{pmatrix}, \qquad T = \begin{pmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -y_1 \\ 0 & 0 & 1 \end{pmatrix}.$$

The translated segment is now

$$TP = \begin{pmatrix} 0 & x_2 - x_1 \\ 0 & y_2 - y_1 \\ 1 & 1 \end{pmatrix}.$$

Rotation and scaling

The clockwise rotation around the Z axis by an angle θ, followed by scaling of the segment to unit length, are accomplished by applying the following matrices:

$$R_Z = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad S = \begin{pmatrix} 1/l & & \\ & 1/l & \\ & & 1 \end{pmatrix},$$

where $\cos\theta = (x_2 - x_1)/l$, $\sin\theta = (y_2 - y_1)/l$, and $l = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. The total transformation matrix is

$$M = SR_ZT = \begin{pmatrix} (x_2 - x_1)/l^2 & (y_2 - y_1)/l^2 & (-x_1x_2 + x_1^2 - y_1y_2 + y_1^2)/l^2 \\ (-y_2 + y_1)/l^2 & (x_2 - x_1)/l^2 & (x_1y_2 - y_1x_2)/l^2 \\ 0 & 0 & 1 \end{pmatrix},$$

and it satisfies

$$M \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 1 \end{pmatrix}.$$

Shape functions

The linear shape functions corresponding to the end points of the 2D segment have the general form

$$N_i(x, y) = a_ix + b_iy + c_i, \quad i = 1, 2,$$

and the properties

$$N_i(x_j, y_j) = \delta_{ij}, \quad i, j = 1, 2, \tag{3.54}$$
$$N_1(x, y) + N_2(x, y) = 1, \tag{3.55}$$

where δ_{ij} is the Kronecker delta. The coefficients of N_2 are obviously the columns of the first row of the matrix M. The coefficients of N_1 can be easily obtained from (3.55). The sought shape functions are thus

$$N_1(x, y) = -\frac{x_2 - x_1}{l^2}x - \frac{y_2 - y_1}{l^2}y + \frac{x_2^2 + y_2^2 - x_1x_2 - y_1y_2}{l^2} \tag{3.56}$$
$$N_2(x, y) = \frac{x_2 - x_1}{l^2}x + \frac{y_2 - y_1}{l^2}y + \frac{x_1^2 + y_1^2 - x_1x_2 - y_1y_2}{l^2}. \tag{3.57}$$

3.9.2 3D triangles and mapping to the reference

In three dimensions the interface boundary of the subdomains consists of triangles in the 3D space. We follow a similar process, constructing first the matrix that maps an arbitrary 3D triangle to the reference triangle $[(0,0), (1,0), (0,1)]$. Then we use that mapping to obtain the barycentric coordinates of the 3D triangle.

Translation to the origin

Let P define the affine coordinates of the vertices of a general 3D triangle. Then P and the required translation matrix T are as follows:

$$P = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \\ 1 & 1 & 1 \end{pmatrix}, \qquad T = \begin{pmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -y_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

The translated segment is now

$$TP = \begin{pmatrix} 0 & x_2 - x_1 & x_3 - x_1 \\ 0 & y_2 - y_1 & y_3 - y_1 \\ 0 & z_2 - z_1 & z_3 - z_1 \\ 1 & 1 & 1 \end{pmatrix}.$$

Rotations and map to the reference

The translation is followed by three consecutive rotations, counter-clockwise around the z-axis and y-axis and clockwise around the x-axis. After these rotations the 3D triangle lies on the xy-plane with the vertex with coordinates (x_1, y_1, z_1) before any affine transformation takes place located now at the origin. As soon as the triangle is mapped onto the xy-plane the mapping to the reference follows trivially as it is usually done with standard triangular finite elements. The total transformation is expressed by the following matrix:

$$M = \begin{pmatrix} x_2 - x_1 & x_3 - x_1 & ((y_2 - y_1)(z_3 - z_1) - (z_2 - z_1)(y_3 - y_1))/d & x1 \\ y_2 - y_1 & y_3 - y_1 & ((x_1 - x_2)(z_3 - z_1) + (x_3 - x_1)(z_2 - z_1))/d & y1 \\ z_2 - z_1 & z_3 - z_1 & ((x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1))/d & z1 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

where

$$\begin{aligned} d^2 = {} & ((y_2 - y_1)(z_3 - z_1) - (z_2 - z_1)(y_3 - y_1))^2 \\ & + ((x_1 - x_2)(z_3 - z_1) + (x_3 - x_1)(z_2 - z_1))^2 \\ & + ((x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1))^2 . \end{aligned}$$

The determinant of M is equal to d, while $d/2$ is the area of the triangle. The matrix M satisfies

$$M\begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}.$$

Shape functions

The $P1$ shape functions corresponding to the vertices of the 3D triangle have the general form

$$N_i(x,y,z) = a_i x + b_i y + c_i z + d_i, \quad i = 1,2,3$$

and the properties

$$N_i(x_j, y_j, z_j) = \delta_{ij}, \quad i,j = 1,2,3 \tag{3.58}$$

$$N_1(x,y,z) + N_2(x,y,z) + N_3(x,y,z) = 1, \tag{3.59}$$

where δ_{ij} is the Kronecker delta. The coefficients of N_2 are obviously the columns of the first row of the matrix M and the coefficients of N_3 the columns of its second row. The coefficients of N_1 can be easily obtained from the second property (3.59). The sought shape functions are thus

$$N_1(x,y,z) = 1 - N_2(x,y,z) - N_3(x,y,z), \tag{3.60}$$

$$\begin{aligned} N_2(x,y,z) &= (x_2 - x_1)x + (x_3 - x_1)y \\ &+ \frac{1}{d}\left((y_2 - y_1)(z_3 - z_1) - (z_2 - z_1)(y_3 - y_1)\right) z + x_1, \end{aligned} \tag{3.61}$$

$$\begin{aligned} N_3(x,y,z) &= (z_2 - z_1)x + (z_3 - z_1)y \\ &+ \frac{1}{d}\left((x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)\right) z + z_1. \end{aligned} \tag{3.62}$$

3.9.3 Non-integer matrix powers

In order to construct and apply in a practical application any of the discrete norms derived in the previous discussion we are required to evaluate *non-integer* powers of a matrix. This task may be achieved in different ways for different applications. In general, if the dimension of the problem is low, one can employ a direct method based on a generalized eigenvalue decomposition; see for instance [68, chapter 11]. This approach however is prohibitively expensive, as its complexity is of order $O(n^3)$. For large matrices it is recommended only if the matrix has a block-diagonal structure, and the whole process can then take advantage of cache-aware algorithms. When the matrices involved have a Toeplitz structure the desired matrix function can be evaluated via an FFT algorithm as suggested by Peisker [122]. The complexity then drops to $O(n \log n)$. In both cases the storage requirements are of order $O(n^2)$. Newton's method has attractive convergence properties but it will always be outperformed by the approach employing the eigenvalue decomposition.

In the realm of preconditioning we usually desire an approximation of H_θ or of its action applied to a given vector. There are several approaches found in the literature designed for this problem. In [74] Hale, Higham and Trefethen propose a method based on contour integrals. Any matrix function can be represented as a contour integral and then this integral is evaluated using the periodic trapezoid rule with conformal maps involving Jacobi elliptic functions; the suggested algorithms show geometric convergence. Another approach is to construct approximations of Krylov type, which take advantage of the sparsity of the matrices involved. Several authors have considered this approach for general matrix functions; see for instance [2, 28, 44, 45, 46, 52, 130]. Convergence analysis of the proposed algorithms is provided for the computation of the square root function in [45].

In what follows we describe a robust Krylov based approximation we adopted for any real value of θ.

3.9.4 Generalized Lanczos algorithms

The Lanczos method introduced in [101] is a technique that allows us to solve large sparse eigenvalue problems $Ax = \lambda x$. The method involves partial tridiagonalizations of the given matrix A. Important information about A's extremal eigenvalues emerges long before the tridiagonalization is complete. This makes the Lanczos algorithm particularly useful in situations where a few of A's largest or smallest eigenvalues are desired. For a remarkable derivation of the underlying algorithm through optimization of the Rayleigh quotient and a detailed discussion about its convergence properties and applications, see [68, chapter 9].

Suppose that we are given a symmetric and positive definite matrix A. The standard Lanczos algorithm of Krylov space dimension $k < n$ constructs a set of orthogonal vectors v_i, $i = 1, \dots, k+1$. The columns of the matrix $V_k = [v_1, v_2, \dots, v_k]$ are known as the Lanczos vectors and they satisfy the identity

$$AV_k = V_k T_k + \beta_{k+1} v_{k+1} e_k^T, \quad V_k^T V_k = I_k,$$

where $I_k \in \mathbb{R}^{k \times k}$ is the identity matrix with kth column denoted by e_k, while $T_k \in \mathbb{R}^{k \times k}$ is a symmetric and tridiagonal matrix defined in (3.64) below.

We are interested in the generalized eigenvalue problem

$$Ax = \lambda Bx, \tag{3.63}$$

where A and B are symmetric positive-definite matrices. The generalized eigenvalue algorithm [121, 155] constructs a set of B-orthogonal vectors v_i such that

$$AV_k = BV_kT_k + \beta_{k+1}Bv_{k+1}e_k^T, \quad V_k^TBV_k = I_k,$$

where T_k and I_k are defined as before.

Table 3.3 lists the standard and generalized Lanczos algorithms. For the standard algorithm one needs to supply the symmetric positive definite matrix A, an arbitrary starting vector v, and the Lanczos space dimension k. For the generalized Lanczos algorithm we need to supply the matrix B as well. After termination, Lanczos provides the tridiagonal matrix T_k and the matrix V_k.

```
LANCZOS($A, v, k$)
$\beta_1 = 0, v_0 = 0, v_1 = v/\|v\|_2$
for $i = 1 : k$
    $w_i = Av_i - \beta_i v_{i-1}$
    $\alpha_i = (w_i, v_i)$
    $w_{i+1} = w_i - \alpha_i v_i$
    $\beta_{i+1} = \|w_{i+1}\|_2$
    if $\beta_{i+1} = 0$ stop
    $v_{i+1} = w_{i+1}/\beta_{i+1}$
end for
```

```
GLANCZOS($A, B, v, k$)
$\beta_1 = 0, v_0 = 0, v_1 = v/\|v\|_B$
for $i = 1 : k$
    $w_i = B^{-1}Av_i - \beta_i v_{i-1}$
    $\alpha_i = (w_i, v_i)_B$
    $w_{i+1} = w_i - \alpha_i v_i$
    $\beta_{i+1} = \|w_{i+1}\|_B$
    if $\beta_{i+1} = 0$ stop
    $v_{i+1} = w_{i+1}/\beta_{i+1}$
end for
```

Table 3.3: Modifications to the standard Lanczos algorithm (left) that lead to the generalized version (right).

We should note that T_k can be seen as the projection of A onto the space spanned by the B-orthogonal columns of V_k. In exact arithmetic:

$$V_k^TAV_k = T_k, \quad V_k^TBV_k = I_k.$$

In both cases described above the explicit form of T_k is

$$T_k = \begin{pmatrix} \alpha_1 & \beta_2 & 0 & 0 \\ \beta_2 & \alpha_2 & \ddots & 0 \\ 0 & \ddots & \ddots & \beta_k \\ 0 & 0 & \beta_k & \alpha_k \end{pmatrix}. \tag{3.64}$$

In exact arithmetic, when $k = n$, the algorithm can be seen as providing simultaneous factorizations of the matrix pair (B, A):

$$A = V_n^{-T} T_n V_n^{-1}, \quad B = V_n^{-T} V_n^{-1}. \tag{3.65}$$

Estimation of $f(A)\, b$ and $f(B^{-1}A)\, b$

Suppose $f(\cdot)$ is an analytic function. Let us assume exact arithmetic and suppose the generalized Lanczos algorithm has terminated without breakdown, giving V_n and T_n. Using (3.65) we have

$$B^{-1}A = V_n T_n V_n^{-1}$$

and therefore

$$f(B^{-1}A)\, b = V_n f(T_n) V_n^{-1}\, b = V_n f(T_n) V_n^T B\, b. \tag{3.66}$$

In practical applications, especially in the preconditioning realm, we do not need to proceed up to $k = n$. A few Lanczos vectors $k \ll n$ will suffice. In that case (3.66) becomes

$$f(B^{-1}A)\, b \approx V_k f(T_k) V_k^T B\, b. \tag{3.67}$$

If Lanczos starts with an arbitrary vector v, then the estimation provided by (3.67) will be poor and will converge only for sufficiently large values of k (values close to n). However, if the starting vector v provided to Lanczos is b itself, then from the B-orthogonality of the columns of V_k, (3.67) simplifies to

$$f(B^{-1}A)\, b \approx V_k\, f(T_k)\, \|b\|_B\, e_1 \tag{3.68}$$

and the convergence of the estimated value given by (3.68) to the exact value of $f(A)\, b$ is significantly accelerated. Finally any matrix function on the matrix T_k can be evaluated by computing the eigenvalue decomposition of $T_k = U \Lambda U^T$, where U is the matrix of the eigenvectors and Λ is a diagonal matrix of the real and positive eigenvalues. The corresponding algorithm[1] is described in [43] and has complexity of $O(kn)$. Since the matrix is only $k \times k$, where k in practical applications is really small ($k < 100$), the eigenvalue decomposition of the matrix T_k takes only a few milliseconds. After the eigenvalue decomposition of T_k has been computed, any function of T_k can be evaluated as

$$f(T_k) = U f(\Lambda) U^T. \tag{3.69}$$

The standard case may be recovered from the generalized one by changing B to the identity matrix I. Then formula (3.68) becomes

$$f(A)\, b \approx V_k\, f(T_k)\, \|b\|_2\, e_1. \tag{3.70}$$

The preconditioner described in section 3.4 involves inverse powers of $B^{-1}A$. In that case after substitution of (3.69) into equations (3.68), (3.70) we obtain

$$\begin{aligned}(B^{-1}A)^{-\theta}\, b &\approx V_k\, U\Lambda^{-\theta}U^T\, \|b\|_B\, e_1,\\ A^{-\theta}\, b &\approx V_k\, U\Lambda^{-\theta}U^T\, \|b\|_2\, e_1.\end{aligned} \tag{3.71}$$

Finally, the action of each of our preconditioners in section 3.4 on an arbitrary vector b is estimated by

$$H_1^{-\theta}\, b = L^{-\theta}\, b \approx V_k\, U\Lambda^{-\theta}U^T\, \|b\|_2\, e_1, \tag{3.72}$$

$$H_2^{-\theta}\, b = (M^{-1}L)^{-\theta}\, M^{-1}\, b \approx V_k\, U\Lambda^{-\theta}U^T\, \|b\|_M\, e_1, \tag{3.73}$$

$$H_3^{-\theta}\, b = (I + (M^{-1}L)^{\theta})^{-1}\, M^{-1}\, b \approx V_k\, U(I+\Lambda^{\theta})^{-1}U^T\, \|b\|_M\, e_1. \tag{3.74}$$

In (3.72) Lanczos iteration starts with b. In (3.73) and (3.74), however, the starting vector is $M^{-1}\, b$.

Speeding up Lanczos

The Lanczos iteration converges very quickly to the largest eigenvalues of the matrix A. But since we are interested in inverse powers of the matrix A, the largest eigenvalues approximated first by Lanczos will have a small reciprocal and consequently convergence to the final result will be observed as soon as Lanczos has sufficiently resolved the smallest eigenvalues.

The eigenvalues of A^{-1} are the reciprocals of the eigenvalues of A. If we consider instead the

[1]The LAPACK routine used for this purpose is DSTEMR, which is specifically designed for symmetric tridiagonal eigenvalue problems.

inverse of the matrix A in Lanczos, the largest eigenvalues of A^{-1} or the smallest eigenvalues of A will be recovered soon and the application of $A^{-\theta}$ or $(B^{-1}A)^{-\theta}$ will converge much sooner. The corresponding eigenvalue problems are

$$\begin{aligned} A^{-1}\,x &= \frac{1}{\lambda}\,x, \\ B\,x &= \frac{1}{\lambda}A\,x. \end{aligned} \tag{3.75}$$

The eigenvalues of the modified problem (3.75) are the reciprocals of the original problem (3.63) as well. Everything we discussed before holds here, with the exception that the standard Lanczos algorithm is carried out with A^{-1} instead of A, while for the generalized eigenvalue problem one has to swap B and A. The formulas (3.71) then become

$$\begin{aligned} (B^{-1}A)^{-\theta}\,b &\approx V_k\,U\Lambda^{\theta}U^T\,\|b\|_B\,e_1, \\ A^{-\theta}\,b &\approx V_k\,U\Lambda^{\theta}U^T\,\|b\|_2\,e_1. \end{aligned} \tag{3.76}$$

Finally, the action of each one of our preconditioners introduced in section 3.4 on an arbitrary vector b, is estimated by

$$H_1^{-\theta}\,b = L^{\theta}\,b \approx V_k\,U\Lambda^{-\theta}U^T\,\|b\|_2\,e_1, \tag{3.77}$$

$$H_2^{-\theta}\,b = (M^{-1}L)^{-\theta}\,M^{-1}\,b \approx V_k\,U\Lambda^{\theta}U^T\,\|b\|_M\,e_1, \tag{3.78}$$

$$H_3^{-\theta}\,b = (I + (M^{-1}L)^{\theta})^{-1}\,M^{-1}\,b \approx V_k\,U(I + \Lambda^{-\theta})^{-1}U^T\,\|b\|_M\,e_1. \tag{3.79}$$

In (3.77) Lanczos iteration starts with b, while in (3.78) and (3.79) the starting vector is $M^{-1}\,b$. We would like to note here that the matrices $\Lambda^{-\theta}$, Λ^{θ} or $(I+\Lambda^{\theta})^{-1}$, $(I+\Lambda^{-\theta})^{-1}$ are diagonal matrices and their inversion wherever it appears is trivial. For instance, the diagonal entries of the matrix $\Lambda^{-} = (I + \Lambda^{-\theta})^{-1}$ are $\lambda_{ii}^{-} = (1 + \lambda_{ii}^{-\theta})^{-1}$ while the entries of the matrix $\Lambda^{+} = (I + \Lambda^{\theta})^{-1}$ are $\lambda_{ii}^{+} = (1 + \lambda_{ii}^{\theta})^{-1}$.

3.10 Flexible Krylov subspace solvers

The Lanczos subspace iteration we employ for the approximation of the action of the square root on a vector $\mathcal{L}(A, v_m, k)$, at the mth step of the outer Krylov subspace iteration, generates different sequences of approximations for different starting vectors v_m. It cannot be considered a fixed preconditioner unless the accuracy of the action of the square root on v_m has been computed up to machine precision. This would require, however, a very large k and reorthogonalization of the

Lanczos vectors, as orthogonality is usually lost very soon [68]. Apart from the large amounts of storage that would be needed, the application of the preconditioner would become very inefficient. That leaves no choice but to consider a few Lanczos vectors (small k) per application of the preconditioner, at the expense of having a preconditioner that is no longer fixed but changes within each outer iteration.

The classical Krylov subspace methods we reviewed in section 3.2.3 have not been designed for variable preconditioners. GMRES, for instance, may reduce the relative residual below the prescribed threshold, but the solution we end up with may be totally wrong. As in our case, there are many other examples where the preconditioner is too expensive to apply. There, an approximate solution $\hat{z}$ whose residual 2-norm falls below some prescribed (inner) tolerance ($\|u_m - M\hat{z}\| < \epsilon_m$) usually works fine [57]. Another application is to replace the preconditioning step by some other Krylov subspace iteration. The most important application, however, is to allow a preconditioner to be updated with newly computed information as in Eirola and Nevanlinna [54] and Weiss [165].

These challenges call for more sophisticated Krylov subspace solvers specifically designed to allow variable preconditioners. It was demonstrated by several authors that existing Krylov subspace methods are capable of providing that flexibility with minor modifications to the original algorithm. These methods are called ***Flexible Krylov subspace methods.*** We briefly describe a flexible variant of GMRES, proposed by Saad [131] and named flexible GMRES (**FGMRES**).

Let us suppose that we use a right-preconditioner. Then the approximation x_m at the last (mth) step of GMRES can be written as

$$
\begin{aligned}
x_m &= x_0 + M^{-1} V_m \, y_m, \\
V_m &= [v_1, v_2, \dots, v_m].
\end{aligned}
\tag{3.80}
$$

When the preconditioner M is variable, then at the last step of GMRES it is clear that $M_m^{-1} V_m \neq [M_1^{-1} v_1, M_2^{-1} v_2, \dots, M_m^{-1} v_m]$. Therefore, the final approximate solution x_m cannot be recovered by means of (3.80). Instead, during the recurrence, one computes and stores $z_k = M_k^{-1} v_k$, $k = 1, 2, \dots, m$, in $Z_m = [z_1, z_2, \dots, z_m]$. The final solution can then be computed as

$$
x_m = x_0 + Z_m y_m. \tag{3.81}
$$

The corresponding Arnoldi relation is $AZ_m = V_{m+1} H_{m+1,m}$ and the vector y_m is obtained by the same minimization, but the subspace $\mathcal{R}(Z_m)$ is not necessarily a Krylov subspace. Unlike the situation in several other classical Krylov methods, there may not exist any Krylov subspace containing $\mathcal{R}(Z_m)$. Nevertheless, as long as $\mathcal{R}(Z_m) \subset \mathcal{R}(Z_{m+1})$, and thus the subspace keeps growing, there is no breakdown and the sequence $\|r_m\|_2$ is non increasing [53]. On the downside, the storage is essentially doubled because both Z_m and V_m need to be stored.

Based on the same framework, flexible variants of other preconditioned algorithms have been devised. These are variable preconditioned CG [69], flexible CG [117], which allows at the same time sparsification [59], and flexible versions of the BiCG and Bi-CGStab algorithms [162]. Interestingly enough, **FGMRES** can assist the solution of large sparse linear systems by direct sparse solvers as well. In [3] the authors consider the triangular factorization of matrices in single-precision arithmetic and show how these factors can be used to obtain a backward stable solution. They even manage to obtain double-precision accuracy even when the system is ill-conditioned. This technique allows direct sparse solvers to operate in single precision, thereby reducing the memory requirements by half and speeding up considerably both the factorization and solution phases.

```
GMRES($A, M, b, tol$)
$x_0 = M^{-1}b,\ r_0 = b - Ax_0,$
$\beta = \|r_0\|_2\ u_1 = \frac{r_0}{\beta},\ k = 0$
while $\|r_k\|_2 > \beta\ tol$
    $k = k + 1$
    $z_k = M^{-1}v_k,\ w = Az_k$
    for $i = 1, 2, \dots, k$ do
        $h_{i,k} = u_i^T w$
        $w = w - h_{i,k}u_i$
    end for
    $h_{k+1,k} = \|w\|_2$
    $u_{k+1} = \frac{w}{h_{k+1,k}}$
    $V_k = [v_1, v_2, \dots, v_k]$

    $H_k = \{h_{i,j}\},\ 1 \le i \le j+1,\ 1 \le j \le k$
    $y_k = \text{argmin}_y \|\beta e_1 - H_k\, y\|_2$
    $x_k = x_0 + M^{-1}V_k\, y_k,\ r_k = b - A\, x_k$
end while
```

```
FGMRES($A, M_i, b, tol$)
$x_0 = M_0^{-1}b,\ r_0 = b - Ax_0,$
$\beta = \|r_0\|_2\ u_1 = \frac{r_0}{\beta},\ k = 0$
while $\|r_k\|_2 > \beta\ tol$
    $k = k + 1$
    $z_k = M_k^{-1}v_k,\ w = Az_k$
    for $i = 1, 2, \dots, k$ do
        $h_{i,k} = u_i^T w$
        $w = w - h_{i,k}u_i$
    end for
    $h_{k+1,k} = \|w\|_2$
    $u_{k+1} = \frac{w}{h_{k+1,k}}$
    $V_k = [v_1, v_2, \dots, v_k]$
    $Z_k = [z_1, z_2, \dots, z_k]$
    $H_k = \{h_{i,j}\},\ 1 \le i \le j+1,\ 1 \le j \le k$
    $y_k = \text{argmin}_y \|\beta e_1 - H_k\, y\|_2$
    $x_k = x_0 + Z_k\, y_k,\ r_k = b - A\, x_k$
end while
```

Table 3.4: Modifications to the classical GMRES algorithm (left) that render it flexible, FGMRES (right). The update (left) at the last step marked with blue is replaced by the two steps at the right marked with red.

3.11 Computational experiments covered by the theory

In what follows the performance of our preconditioning approach is thoroughly investigated through several numerical experiments. Our main purpose is to verify the basic theoretical outcome of section 3.4 claiming independence of the condition number of the preconditioned system on the mesh size h. Although this is traditionally achieved by sophisticated multilevel overlapping methods, we will demonstrate that it is also feasible by the single-level approach we introduced in section 3.4. We defined the discrete version of our preconditioner as

$$H_{1/2,h} = M + M\left(M^{-1}L\right)^{1/2}, \tag{3.82}$$

and we also considered its equivalent

$$H_{1/2} = M\left(M^{-1}L\right)^{1/2}, \tag{3.83}$$

with M and L being the mass matrix and the negative Laplace operator, assembled on the interface of the subdomains with the mass matrix also replaced by its diagonal. In what follows we will also consider the following version which performed quite efficiently in some of benchmark cases:

$$H^{1}_{1/2} = L^{1/2}. \tag{3.84}$$

The action of the inverse square root of the matrices involved during the preconditioning step is obtained in most cases as described in section 3.9.4, where we employ the Lanczos iteration, but with A replaced by its inverse, which drastically reduces the number of iterations needed.

The set of the numerical experiments that follow try to withdraw any confusion possibly introduced by the polyparametric setting of our preconditioner. More precisely they are carefully chosen to clarify the following issues:

1. **Preconditioner type**: Which of the three available versions of the preconditioner should be preferred and under what circumstances should we switch from one to another?

2. **Mass matrix form**: It is common practice to replace the mass matrix by a lumped version. In our case, doing so will clearly decrease the cost of each iteration, but what will the effect be on the number of iterations?

3. **Lanczos k**: What is the optimal parameter k for the Lanczos subspace iteration and is there any systematic way to obtain it?

Whenever the geometry of our domain and the partitioning are such that the matrix on the interface of subdomains obtains a block-diagonal structure obvious benefits arise:

1. **Exact evaluation of the desired matrix function (square root)**: The block-diagonal structure of the interface operators allows for a separate eigenvalue decomposition of each block. The evaluation of any matrix function then becomes trivial and at the same time exact up to machine precision, thus avoiding any approximation errors, which may potentially influence the convergence rate of the iterative solution process.

2. **Fixed preconditioner**: The preconditioner obtained with the block-diagonal eigenvalue decomposition becomes fixed throughout the iterative process, allowing us to use one of the classical iterative methods like the computationally more efficient preconditioned conjugate gradient method (PCG).

3. **Ideal setting for parallel processing**: The block-diagonal structure of the interface operator is ideal for parallel processing on shared memory or distributed systems, as every part of the solution process is completely decoupled and can be exclusively assigned to different processors.

3.11.1 Benchmark components

For the discretization of all problems considered, we employed the finite element method using the Lagrangian P1 family of elements; that is, linear triangles for two-dimensional problems and linear tetrahedra for PDEs discretized in three dimensions.

Mesh generation

Finite element meshes in two dimensions were obtained by the unstructured mesh generator **Triangle** [138, 139, 140], which generates exact Delaunay triangulations and high quality triangular meshes. Unstructured meshes in three dimensions were obtained by the tetrahedral mesh generator TetGen [142, 143, 144], which among others, provides quality tetrahedral Delaunay meshes.

Krylov subspace solver and stopping criterion

As the Krylov subspace solver, we used the FGMRES method because our Lanczos-based preconditioner changes at each step. Whenever we use a block-diagonal eigenvalue decomposition for the approximation of the square root, we chose the PCG method, which is computationally more efficient.

Our stopping criterion requires the relative residual at the kth iteration $\|r_k\|/\|r_0\|$ to drop below a specific threshold, set to 10^{-6} throughout our numerical experiments. This threshold is in most cases small enough, because usually the error norms associated with the accuracy of finite element approximation (L^2 , H^1 norms) have converged long before the relative residual reaches that value.

Ideally the stopping criterion should monitor the convergence of the error of the finite element solution. Although this approach is feasible for some simple cases involving symmetric operators [6] and has been extended recently to the nonsymmetric case provided the preconditioner is chosen in a specific way [5], the general case remains an open problem.

Lanczos matrix function approximation

When the partitioning and geometry are such that the separators of the subdomains intersect, the discrete operator on the interface does not have a block-diagonal structure and the Lanczos procedure is employed for the evaluation of the action of the desired matrix function on a vector. The dimension of the Lanczos subspace k clearly influences the convergence of the iterative solution process. Small values of k result in poor approximation of the square root or any other matrix function, thus increasing the iterations needed by the Krylov method to reduce the relative residual below the specified threshold.

We should note that the specific choice of k need not be the same for all mesh levels considered, but it may change depending on the resolution of the mesh (h). Since we are only interested in iteration counts at this point and not in the cost of each preconditioning step, we fix the value of k so that the number of iterations needed to solve the linear system at the finest level does not drop further if k is increased. To illustrate the sensitivity of the number of iterations on k we consider for some benchmark cases several different values of k.

Partitioning approach

Quite frequently in the domain-decomposition literature, suggested methods consider a standard domain and partitioning approach, where they discretize and solve the PDE problem under consideration. These are the unit square for 2D problems subdivided into square subdomains, or the unit cube for 3D problems subdivided into cubic subdomains. Apart from the convenience related to the construction of the partitions, this standard benchmark problem also provides an easy way to obtain information from a coarser level and project it back to the fine level. The general partitioning scheme provided by a graph partitioning tool is examined in section 3.12.

3.11.2 Poisson's equation

We consider Poisson's equation with Dirichlet boundary conditions:

$$\begin{aligned} -\nabla^2 u(x) &= 1, \quad x \in \Omega \\ u(x) &= 0, \quad x \in \partial\Omega. \end{aligned} \tag{3.85}$$

In what follows we solve this equation in several different domain configurations in both two and three-dimensional domains. The operator L that appears in the definition of the family of preconditioners we consider here is the discrete version of the $-\nabla^2$ operator assembled on the interface of the subdomains.

Long pipe 2D

A very long pipe of height $l = 1000$ aligned with the Y axis and width $a = 1$ is considered here. It is manually partitioned into slices parallel to the X axis, as shown in Figure 3.2. This benchmark setting meets the requirements discussed in section 3.11 and reveals the optimal performance of the preconditioner. In the general two-dimensional case, the performance of the preconditioner (in terms of iteration counts) will always be inferior to the one experienced here.

The mesh levels used for this benchmark and their characteristics are listed in Table 3.6. The number of unknowns on the interface of the subdomains for each of the partitions and mesh levels considered can be found in Table 3.5. In Tables 3.7, 3.8 we compare the iterations needed for reducing the initial residual of the Schur-complement system by six orders of magnitude for several different values of the Lanczos space dimension k (3, 5, 10, 15). The mass matrix in the two versions of the preconditioner 3.82- 3.83 has been replaced by its diagonal.

It is clear that the number of iterations for this example is influenced only marginally by the specific value of k. Apart from that it is obviously h-independent as predicted theoretically. For the first two mesh levels and/or the first 2 partitionings (2, 4), the number of iterations is one or two less than the dominant value. But this is to be expected because for these cases the number of unknowns of the Schur-complement system is too small; see Table 3.5.

Moreover we see that it is unexpectedly independent of the number of the partitions, which is a property of only a few sophisticated multilevel domain decomposition methods.

The second version of the preconditioner $M(M^{-1}L)^{1/2}$ performs asymptotically better than the third version $M + M(M^{-1}L)^{1/2}$ by one iteration. We proceed with the 3D equivalent of this benchmark problem.

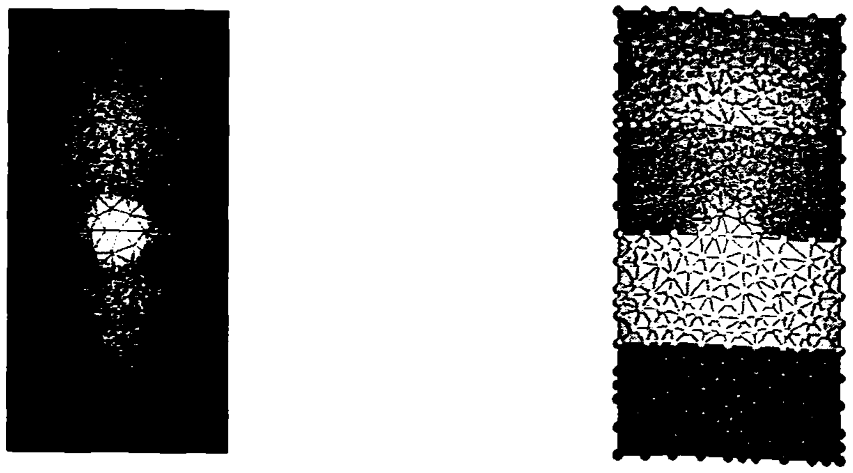


Figure 3.2: A piece of the 2D pipe (left) manually partitioned to 4 subdomains (right). We can see the unknowns on the interface (gray) and the unknowns in the interior of each subdomain.

	mesh level		
subdomains	1	2	3
2	7	17	33
4	25	51	98
8	60	119	227
16	124	254	492
32	258	522	1015
64	513	1055	2065
128	1051	2140	4143
256	2088	4301	8306
512	4236	8593	16684
1024	8185	17230	33319

Table 3.5: Nodes on the interface of the subdomains for all mesh levels and partitionings.

2 subdomains			
level	elements	nodes	$h_{\max}$
1	99041	55920	2.653e-01
2	416805	221985	1.328e-01
3	1712315	883797	6.532e-02
4 subdomains			
1	99062	55962	2.672e-01
2	416646	221833	1.313e-01
3	1711548	883447	6.481e-02
8 subdomains			
1	99303	56043	2.616e-01
2	417312	222154	1.312e-01
3	1713167	884135	6.457e-02
16 subdomains			
1	99641	56214	2.637e-01
2	416788	221901	1.319e-01
3	1713905	884585	6.478e-02
32 subdomains			
1	99017	55938	2.651e-01
2	417472	222273	1.298e-01
3	1711285	883223	6.471e-02

64 subdomains			
level	elements	nodes	$h_{\max}$
1	99573	56207	2.605e-01
2	418268	222648	1.312e-01
3	1711017	883080	6.553e-02
128 subdomains			
1	99121	56031	2.660e-01
2	417866	222493	1.301e-01
3	1714048	884537	6.488e-02
256 subdomains			
1	98446	55653	2.608e-01
2	417578	222396	1.319e-01
3	1712177	883789	6.519e-02
512 subdomains			
1	98898	56166	2.558e-01
2	417982	222680	1.308e-01
3	1716068	885924	6.446e-02
1024 subdomains			
1	94236	53272	2.515e-01
2	419456	223843	1.278e-01
3	1711392	883275	6.461e-02

Table 3.6: Mesh levels used for each one of the ten partitions. For each mesh level we see the number of elements, number of nodes and maximum edge length.

mesh level	1	2	3
subdomains	Iterations		
$k = 3$	$L^{1/2}$		
2	5	7	9
4	6	7	9
8	7	8	10
16	7	7	9
32	7	8	9
64	7	8	10
128	7	12	10
256	7	10	12
512	7	8	12
1024	7	8	11
diagonal	$M(M^{-1}L)^{1/2}$		
2	5	6	8
4	6	6	8
8	7	7	8
16	7	7	8
32	7	7	8
64	7	7	8
128	7	7	9
256	7	7	9
512	7	7	9
1024	7	7	8
diagonal	$M + M(M^{-1}L)^{1/2}$		
2	5	7	9
4	7	7	9
8	7	8	9
16	8	8	9
32	8	8	9
64	8	8	9
128	8	8	9
256	8	8	9
512	7	8	9
1024	8	8	9

mesh level	1	2	3
subdomains	Iterations		
$k = 5$	$L^{1/2}$		
2	5	6	7
4	6	6	8
8	6	7	8
16	6	7	8
32	7	7	8
64	7	7	9
128	7	10	8
256	7	8	9
512	7	7	10
1024	7	7	9
diagonal	$M(M^{-1}L)^{1/2}$		
2	5	6	7
4	6	6	7
8	6	7	7
16	7	7	7
32	7	7	7
64	7	7	7
128	7	7	7
256	7	7	7
512	7	7	7
1024	7	7	7
diagonal	$M + M(M^{-1}L)^{1/2}$		
2	5	7	8
4	7	7	8
8	7	8	8
16	8	8	8
32	8	8	8
64	8	8	8
128	8	8	8
256	8	8	8
512	8	8	8
1024	8	8	8

Table 3.7: Iterations needed for reducing the initial residual of the Schur-complement system by a factor of 10^{-6}, for all three versions of the preconditioner with the diagonal version of the mass matrix M. Lanczos $k = 3, 5$.

mesh level	1	2	3
subdomains	Iterations		
$k = 10$	$L^{1/2}$		
2	5	6	6
4	6	6	7
8	6	7	7
16	6	7	7
32	7	7	7
64	7	7	8
128	7	9	7
256	7	8	8
512	7	7	8
1024	7	7	7
diagonal	$M(M^{-1}L)^{1/2}$		
2	5	6	6
4	6	6	6
8	6	7	6
16	7	7	7
32	7	7	7
64	7	7	7
128	7	7	7
256	7	7	7
512	7	7	7
1024	7	7	7
diagonal	$M + M(M^{-1}L)^{1/2}$		
2	5	7	7
4	7	7	7
8	7	8	7
16	8	7	7
32	8	8	8
64	8	8	8
128	8	8	8
256	8	8	8
512	8	8	8
1024	8	8	8

mesh level	1	2	3
subdomains	Iterations		
$k = 15$	$L^{1/2}$		
2	5	6	6
4	6	6	6
8	6	7	7
16	6	7	7
32	7	7	7
64	7	7	7
128	7	9	7
256	7	8	8
512	7	7	8
1024	7	7	7
diagonal	$M(M^{-1}L)^{1/2}$		
2	5	6	6
4	6	6	6
8	6	7	7
16	7	7	7
32	7	7	7
64	7	7	7
128	7	7	7
256	7	7	7
512	7	7	7
1024	7	7	7
diagonal	$M + M(M^{-1}L)^{1/2}$		
2	5	7	7
4	7	7	7
8	7	8	7
16	8	7	7
32	8	8	8
64	8	8	8
128	8	8	8
256	8	8	8
512	8	8	8
1024	8	8	8

Table 3.8: Iterations needed for reducing the initial residual of the Schur-complement system by a factor of 10^{-6}, for all three versions of the preconditioner with the diagonal version of the mass matrix M. Lanczos $k = 10, 15$.

	mesh levels		
subdomains	1	2	3
4	77	199	647
8	155	473	1581
16	325	988	3390
32	660	2022	6977
64	1403	4181	14202
128	2809	8349	28634
256	5685	16865	57668
512	11114	33784	115306
1024	22439	68742	232067
2048	46439	137762	464165

Table 3.9: Nodes on the interface of the subdomains

Long pipe 3D

A very long pipe of length $l = 1000$ aligned with the Z axis, with square cross-section of side $a = 1$ is considered here. It is manually partitioned in slices parallel to the XY plane as shown in Figure 3.3. This benchmark setting meets the requirements discussed in section 3.11.

The mesh levels used for this benchmark and their characteristics are listed in Table 3.10. The number of unknowns on the interface of the subdomains for each of the partitions and the mesh levels considered can be found in Table 3.9. In Tables 3.11, 3.12 we compare the iterations needed for reducing the initial residual of the Schur-complement system by six orders of magnitude for several different values of the Lanczos space dimension k $(3, 5, 10, 15)$. The mass matrix in the last two versions of the preconditioner has been replaced by its diagonal.

It is clear that the number of iterations for this example is influenced only marginally by the specific value of k. Moreover, although it is not h-independent as pronouncedly as we experienced in the 2D case, it increases only marginally as the mesh is refined. The dependence on the number of partitions, however, seems to have vanished, especially for the versions of the preconditioner that involve the mass matrix. We also verify here that the second version of the preconditioner $M(M^{-1}L)^{1/2}$ performs asymptotically better than the third version $M + M(M^{-1}L)^{1/2}$ by one iteration. Undoubtedly the number of iterations for this problem is a lower bound for the performance of the preconditioner in terms of iteration counts. We do not expect better performance in the experiments that follow, but this benchmark will allow us to see how much room there is for improvement.

We proceed next by examining the performance of the preconditioner with a different domain partitioning, where the interface of subdomains intersect while the interfaces remain planar.

level	elements	nodes	$h_{\max}$
4 subdomains			
1	207442	59055	6.774e-01
2	1627405	348350	3.371e-01
3	12795605	2346496	1.701e-01
8 subdomains			
1	206327	58585	6.907e-01
2	1628901	348670	3.390e-01
3	12796247	2346282	1.719e-01
16 subdomains			
1	200054	55843	6.924e-01
2	1620898	345834	3.343e-01
3	12780812	2343077	1.736e-01
32 subdomains			
1	196452	53723	6.904e-01
2	1613664	343359	3.354e-01
3	12783539	2342304	1.699e-01
64 subdomains			
1	198963	54559	6.923e-01
2	1621957	345579	3.311e-01
3	12798434	2346124	1.730e-01

level	elements	nodes	$h_{\max}$
128 subdomains			
1	204029	56184	6.924e-01
2	1634357	348796	3.358e-01
3	12831683	2353549	1.753e-01
256 subdomains			
1	214623	59396	6.653e-01
2	1652785	353208	3.322e-01
3	12883967	2364871	1.698e-01
512 subdomains			
1	218351	60403	6.407e-01
2	1681057	359518	3.321e-01
3	12958955	2380370	1.720e-01
1024 subdomains			
1	220865	61606	6.566e-01
2	1739687	370650	3.372e-01
3	13110417	2408687	1.696e-01
2048 subdomains			
1	264365	74543	6.429e-01
2	1810639	386516	3.268e-01
3	13406875	2469553	1.718e-01

Table 3.10: Mesh levels used. For each level we see the number of elements, number of nodes, maximum and minimum edge length.

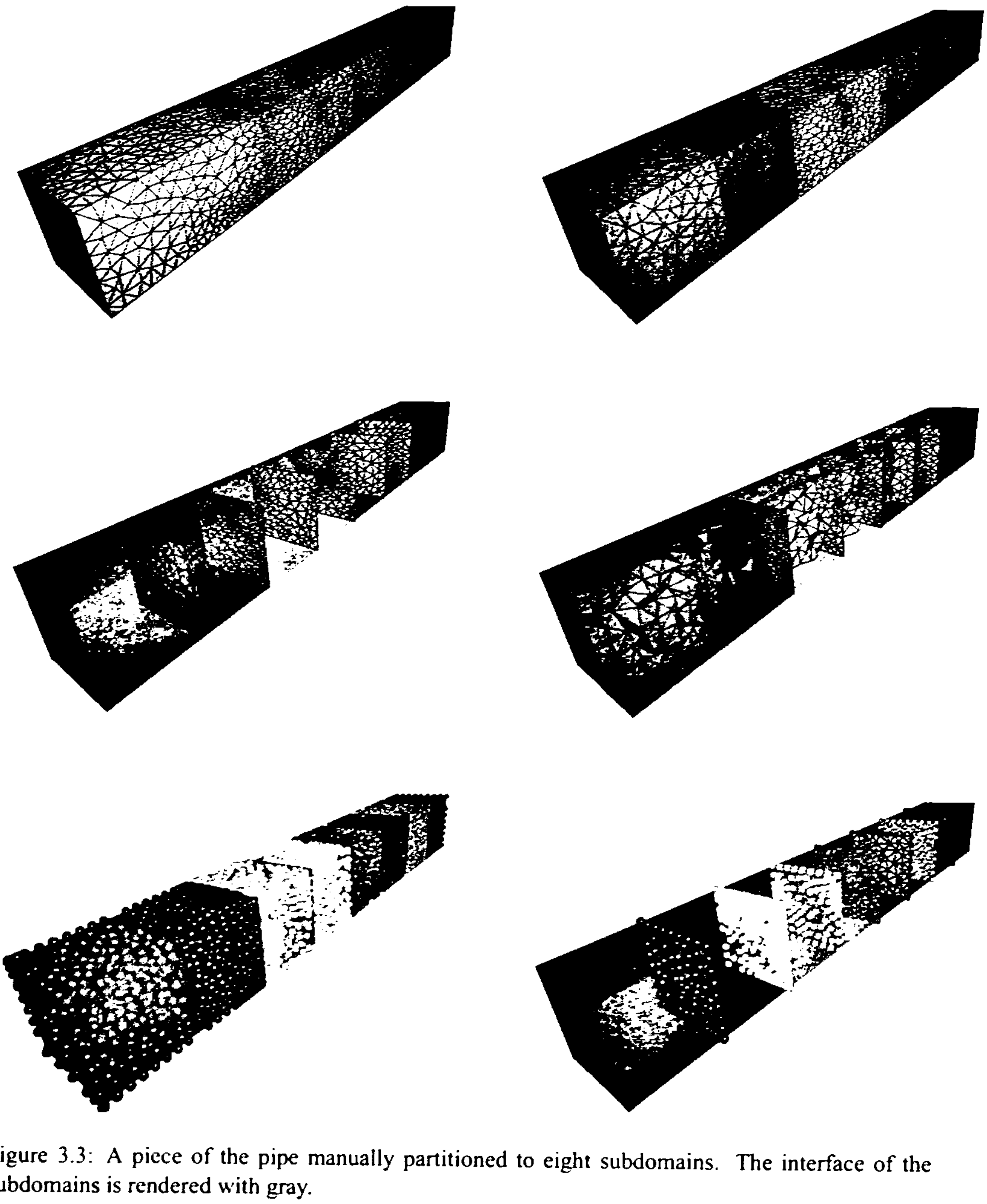


Figure 3.3: A piece of the pipe manually partitioned to eight subdomains. The interface of the subdomains is rendered with gray.

mesh level	1	2	3
subdomains	Iterations		
$k = 3$	$L^{1/2}$		
4	11	11	12
8	11	12	14
16	10	11	14
32	10	14	15
64	13	15	16
128	15	16	16
256	15	17	17
512	16	17	17
1024	14	16	18
2048	18	17	19
diagonal	$M(M^{-1}L)^{1/2}$		
4	9	10	11
8	9	11	12
16	10	11	12
32	10	12	12
64	11	12	13
128	10	12	13
256	11	12	13
512	11	12	13
1024	11	13	13
2048	15	14	15
diagonal	$M + M(M^{-1}L)^{1/2}$		
4	9	11	12
8	9	12	12
16	10	11	13
32	10	13	12
64	11	12	13
128	10	13	14
256	12	13	13
512	12	12	14
1024	12	14	14
2048	16	15	16

mesh level	1	2	3
subdomains	Iterations		
$k = 5$	$L^{1/2}$		
4	11	11	11
8	11	11	13
16	10	11	13
32	10	14	14
64	13	15	15
128	14	16	15
256	14	16	16
512	16	16	17
1024	14	16	17
2048	18	17	18
diagonal	$M(M^{-1}L)^{1/2}$		
4	9	10	11
8	9	11	12
16	10	11	12
32	10	12	12
64	11	12	13
128	10	12	13
256	11	12	13
512	12	12	13
1024	11	13	13
2048	15	14	15
diagonal	$M + M(M^{-1}L)^{1/2}$		
4	9	11	12
8	9	12	13
16	10	11	13
32	10	13	13
64	11	12	14
128	11	13	14
256	12	13	14
512	12	13	14
1024	12	14	14
2048	16	15	16

Table 3.11: Iterations needed for reducing the initial residual of the Schur-complement system by a factor of 10^{-6}, for all three versions of the preconditioner with the diagonal version of the mass matrix M. Lanczos $k = 3, 5$.

mesh level	1	2	3
subdomains	Iterations		
$k = 10$	$L^{1/2}$		
4	11	10	11
8	11	11	13
16	10	11	13
32	10	14	14
64	13	15	15
128	14	16	14
256	14	16	16
512	15	16	16
1024	14	16	17
2048	18	17	17
diagonal	$M(M^{-1}L)^{1/2}$		
4	9	10	11
8	9	11	12
16	10	11	12
32	10	12	12
64	11	12	13
128	10	12	13
256	11	12	13
512	12	12	13
1024	11	13	14
2048	15	14	15
diagonal	$M + M(M^{-1}L)^{1/2}$		
4	9	11	12
8	9	11	13
16	10	11	13
32	10	12	13
64	11	13	14
128	11	13	14
256	12	13	14
512	12	13	14
1024	12	13	14
2048	16	15	16

mesh level	1	2	3
subdomains	Iterations		
$k = 15$	$L^{1/2}$		
4	11	10	11
8	11	11	13
16	10	11	13
32	10	14	14
64	13	15	15
128	14	16	14
256	14	16	15
512	15	16	16
1024	14	16	17
2048	18	17	17
diagonal	$M(M^{-1}L)^{1/2}$		
4	9	10	11
8	9	11	12
16	10	11	12
32	10	12	12
64	11	12	13
128	10	12	13
256	11	12	13
512	12	12	13
1024	11	13	13
2048	15	14	15
diagonal	$M + M(M^{-1}L)^{1/2}$		
4	9	11	12
8	9	11	13
16	10	11	13
32	10	12	13
64	11	13	14
128	11	13	14
256	12	13	14
512	12	13	14
1024	12	13	14
2048	16	15	16

Table 3.12: Iterations needed for reducing the initial residual of the Schur-complement system by a factor of 10^{-6}, for all three versions of the preconditioner with the diagonal version of the mass matrix M. Lanczos $k = 10, 15$.

Unit square

The domain of definition in this example is the unit square. It is manually split into 4, 16, 64 and 256 equal size squares as shown in Figure 3.4. Then for each subdivision we consider five mesh levels, generated in such a way that the maximum edge length $h_{\max}$ is approximately the same for the same levels of all partitioning configurations. Details about the mesh levels considered can be found in Table 3.13, while the numbers of unknowns on the interface of subdomains are listed in Table 3.14.

Table 3.15 lists the iterations needed for reducing the initial residual of the Schur-complement system by six orders of magnitude ($\|r_k\|_2 \leq 10^{-6} \|r_0\|_2$) for all three versions of the preconditioner and with the mass matrix both as full or replaced by its diagonal. For the square root we use the Lanczos approach with the inverse matrix and $k = 140$ Lanczos vectors. Clearly seen is that the iterations for all the cases considered do not depend on the mesh refinement level or equivalently on the mesh size $h_{\max}$. However, they do depend on the number of the subdomains. The second version of the preconditioner $M(M^{-1}L)^{1/2}$ with the mass matrix M replaced by its diagonal is once more the most efficient.

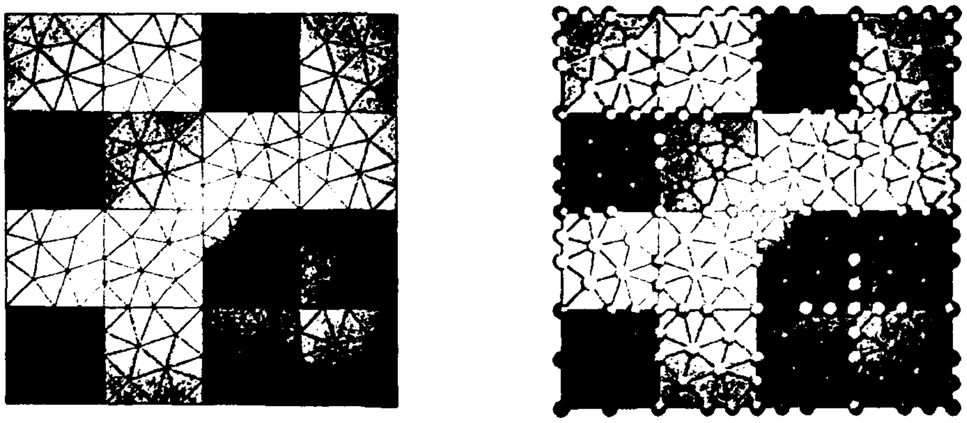


Figure 3.4: Unit square partitioned manually into 16 subdomains. We can see the nodes on the interface of the subdomains (gray) and those in the interior.

4 subdomains				64 subdomains			
level	elements	nodes	$h_{\max}$	level	elements	nodes	$h_{\max}$
1	7867	4056	3.125e-02	1	7867	4056	3.125e-02
2	32319	16409	1.562e-02	2	32319	16409	1.562e-02
3	130206	65590	7.812e-03	3	130206	65590	7.812e-03
4	525641	263801	3.906e-03	4	525641	263801	3.906e-03
16 subdomains				256 subdomains			
level	elements	nodes	$h_{\max}$	level	elements	nodes	$h_{\max}$
1	7867	4056	3.125e-02	1	7867	4056	3.125e-02
2	32319	16409	1.562e-02	2	32319	16409	1.562e-02
3	130206	65590	7.812e-03	3	130206	65590	7.812e-03
4	525641	263801	3.906e-03	4	525641	263801	3.906e-03

Table 3.13: Mesh levels used. For each level we see the number of elements, number of nodes, maximum and minimum edge length.

	mesh level			
subdomains	1	2	3	4
4	157	257	573	1160
16	394	826	1639	3367
64	860	1787	3729	7703
256	1891	3596	7484	15716

Table 3.14: Nodes on the interface of the subdomains.

mesh level	1	2	3	4	5
subdomains	Iterations				
	$L^{1/2}$				
4	9	9	10	10	10
16	12	12	13	13	13
64	16	16	16	16	16
256	23	21	22	22	22
full	$M(M^{-1}L)^{1/2}$				
4	12	12	13	13	13
16	15	15	16	16	15
64	19	20	20	20	20
256	27	27	26	26	26
diagonal	$M(M^{-1}L)^{1/2}$				
4	8	9	9	10	9
16	12	12	12	12	13
64	14	14	14	14	15
256	19	19	19	18	18
full	$M + M(M^{-1}L)^{1/2}$				
4	13	13	14	14	14
16	17	17	17	17	17
64	21	21	21	21	21
256	29	29	29	28	28
diagonal	$M + M(M^{-1}L)^{1/2}$				
4	10	10	10	10	10
16	13	13	13	13	14
64	17	17	17	17	17
256	24	23	23	23	23

Table 3.15: Iterations needed for reducing the initial residual of the Schur-complement system by a factor of 10^{-6} for all three versions of the preconditioner with Lanczos $k = 140$.

Unit cube

We proceed with the 3D version of the previous experiment. The unit cube is internally split into $8, 64, 512$ cubes of equal size as shown in Figure 3.5. The resulting piecewise linear complex (PLC) for each of the three cases is then forwarded to the tetrahedral mesh generator **TetGen**. Four different mesh levels are generated for each case, in such a way that the maximum edge length $h_{\max}$ for each level is about half the one of the previous level. The maximum volume constraint for the tetrahedra of each level, an input parameter to **TetGen**, was adjusted for each case such that the maximum edge lengths are approximately the same, among the same levels of each of the three cases. Details about the meshes are listed in Table 3.16, while the nodes on the interface of the subdomains for each of the cases examined here can be found in Table 3.17.

Subdomains 8				
l	elements	nodes	$h_{\min}$	$h_{\max}$
1	17963	3482	0.0624	0.1398
2	165268	28603	0.0156	0.0696
3	1399060	229041	0.0078	0.0348
4	11832730	1884996	0.0034	0.0174

Subdomains 64			
elements	nodes	$h_{\min}$	$h_{\max}$
21444	4090	0.026	0.1398
172805	29943	0.014	0.0692
1457693	238839	0.006	0.0348
11933280	1902206	0.003	0.0174

Subdomains 512				
l	elements	nodes	$h_{\min}$	$h_{\max}$
1	21966	4387	0.0624	0.1398
2	199361	34821	0.0123	0.0699
3	1554746	255606	0.0061	0.0349
4	12149976	1939420	0.0032	0.0175

Table 3.16: Mesh levels generated for the cube manually partitioned to 8, 64 and 512 subdomains.

Tables 3.18, 3.19 list the iteration counts obtained for all three versions of our preconditioner with either the full or diagonal version of the mass matrix and with $k = 5, 10, 15, 20$ to demonstrate the effect of the Lanczos space dimension on the number of iterations. We observe that the first version of the preconditioner $L^{1/2}$, the most efficient in terms of cost per iteration, is also the more efficient in terms of iteration counts. Most importantly, the diagonal version of the mass matrix

	mesh levels			
subdomains	1	2	3	4
8	747	3214	12880	53460
64	2194	9231	38980	158733
512	3907	20579	89677	364470

Table 3.17: Nodes on the interface of the subdomains

mesh level	1	2	3	4
$k=5$	Iterations			
subdomains	$L^{1/2}$			
8	17	22	23	31
64	23	25	29	36
512	20	32	35	41
full	$M(M^{-1}L)^{1/2}$			
8	24	28	30	35
64	28	31	37	44
512	28	41	42	46
diagonal	$M(M^{-1}L)^{1/2}$			
8	21	24	24	28
64	24	26	30	36
512	24	34	34	38
full	$M+M(M^{-1}L)^{1/2}$			
8	25	29	30	37
64	30	33	38	45
512	29	42	44	49
diagonal	$M+M(M^{-1}L)^{1/2}$			
8	21	24	25	29
64	24	27	31	37
512	24	35	35	39

mesh level	1	2	3	4
$k=10$	Iterations			
subdomains	$L^{1/2}$			
8	17	21	19	25
64	24	24	26	30
512	22	32	32	35
full	$M(M^{-1}L)^{1/2}$			
8	24	27	26	31
64	29	30	33	37
512	28	41	40	43
diagonal	$M(M^{-1}L)^{1/2}$			
8	21	23	22	26
64	24	25	27	31
512	24	34	33	35
full	$M+M(M^{-1}L)^{1/2}$			
8	24	28	27	32
64	30	31	34	38
512	30	43	43	45
diagonal	$M+M(M^{-1}L)^{1/2}$			
8	21	23	23	26
64	24	26	28	32
512	25	35	34	36

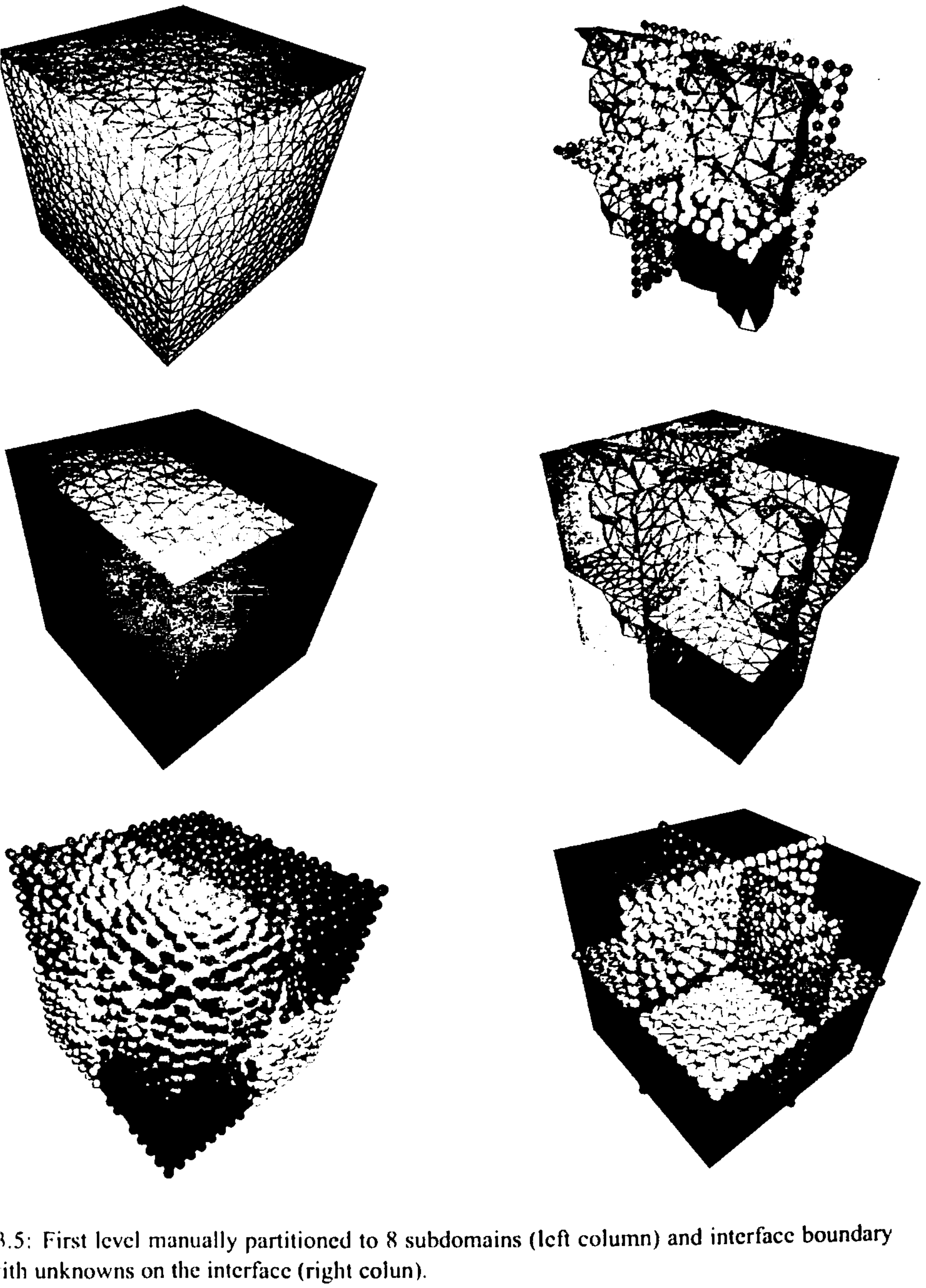
Table 3.18: Iterations needed for reducing the initial residual of the Schur-complement system by a factor of 10^{-6} for all three versions of the preconditioner with Lanczos $k = 5$ (left) and $k = 10$ (right) for the case of manually constructed partitions.

outperforms by far (3 to 10 iterations) the corresponding versions of the preconditioner involving the full version of the mass matrix. The difference becomes more pronounced as the number of subdomains increases and is more prominent for the third version of our preconditioner. Finally the second version of the preconditioner as in the 2D case performs slightly better than the third (1 to 3 iterations) in terms of iteration counts. The difference reduces to one iteration when the diagonal version of the mass matrix is considered.

mesh level	1	2	3	4
$k = 15$	Iterations			
subdomains	$L^{1/2}$			
8	17	21	19	23
64	24	24	26	27
512	22	32	32	33
full	$M(M^{-1}L)^{1/2}$			
8	24	27	26	30
64	29	30	32	36
512	29	41	41	42
diagonal	$M(M^{-1}L)^{1/2}$			
8	21	23	22	24
64	24	25	27	30
512	24	34	33	34
full	$M + M(M^{-1}L)^{1/2}$			
8	24	28	27	31
64	30	31	34	37
512	30	45	45	44
diagonal	$M + M(M^{-1}L)^{1/2}$			
8	21	23	23	25
64	24	26	27	30
512	25	35	35	35

mesh level	1	2	3	4
$k = 20$	Iterations			
subdomains	$L^{1/2}$			
8	17	21	19	23
64	24	25	26	26
512	22	32	31	33
full	$M(M^{-1}L)^{1/2}$			
8	24	27	26	29
64	29	30	32	35
512	29	41	41	42
diagonal	$M(M^{-1}L)^{1/2}$			
8	21	23	22	24
64	24	25	27	29
512	24	34	33	34
full	$M + M(M^{-1}L)^{1/2}$			
8	24	28	26	30
64	30	31	34	37
512	30	43	44	44
diagonal	$M + M(M^{-1}L)^{1/2}$			
8	21	23	23	25
64	24	26	28	30
512	25	35	34	35

Table 3.19: Iterations needed for reducing the initial residual of the Schur-complement system by a factor of 10^{-6} for all three versions of the preconditioner with Lanczos $k = 15$ (left) and $k = 20$ (right) for the case of manually constructed partitions.



3.5: First level manually partitioned to 8 subdomains (left column) and interface boundary ith unknowns on the interface (right colun).

3.11.3 Reaction diffusion

We proceed with the investigation of reaction-diffusion problems in two and three dimensions. The underlying PDE problem is the following:

$$\begin{aligned} -\nabla^2 u(x) + \alpha u(x) &= 1, \quad x \in \Omega \\ u(x) &= 0, \quad x \in \partial\Omega, \end{aligned} \tag{3.86}$$

where α is the reaction coefficient: a new parameter introduced by this problem. When α approaches zero, the reaction-diffusion operator becomes just a perturbation to the Laplace operator we have investigated previously. For this reason we restrict our study to values of $\alpha \geq 1$. We should note that the matrix L in what follows is no longer the discretization of the $-\nabla^2$ as in the Poisson problem but the discretization of the full operator $-\nabla^2 + \alpha I$.

Long pipe 2D

We begin by solving problem (3.86) in two dimensions in the same configuration we used for the Poisson problem of section 3.11.2. Our domain is manually partitioned as shown in Figure 3.2. As in the Poisson case we do not expect better performance for reaction diffusion problems in 2D than what we will experience here.

Details about the mesh levels used for this benchmark are listed in Table 3.6. The number of unknowns on the interface of the subdomains for each of the partitions and mesh levels considered can be found in Table 3.5. In Tables 3.20, 3.21 we compare the iterations needed for reducing the initial residual of the Schur-complement system by six orders of magnitude for four different values of α $(1, 10, 100, 1000)$. The mass matrix in the last two versions of the preconditioner has been replaced by its diagonal.

Independence of the number of iterations with respect to the level of refinement and the number of subdomains is prominent for all three versions of the preconditioner but especially for those involving the mass matrix. The preconditioners incorporating the full mass matrix are also h- and subdomains-independent but the number of iterations increases by one or two compared with those where the mass matrix is replaced by its diagonal. All versions of the preconditioners seem to be insensitive on the parameter α as well.

mesh level	1	2	3
subdomains	Iterations		
$\alpha = 1$	$L^{1/2}$		
2	5	6	6
4	6	6	7
8	6	6	7
16	6	6	7
32	6	7	7
64	6	7	8
128	7	9	7
256	7	8	8
512	6	6	9
1024	6	7	8
diagonal	$M(M^{-1}L)^{1/2}$		
2	5	6	6
4	6	6	6
8	6	6	6
16	7	6	6
32	7	6	6
64	7	6	6
128	7	6	6
256	7	6	6
512	6	6	6
1024	6	6	6
diagonal	$M + M(M^{-1}L)^{1/2}$		
2	5	6	6
4	7	6	6
8	7	7	7
16	7	7	7
32	7	7	7
64	8	7	7
128	8	7	7
256	7	7	7
512	7	7	7
1024	7	7	7

mesh level	1	2	3
subdomains	Iterations		
$\alpha = 10$	$L^{1/2}$		
2	5	6	6
4	6	6	7
8	6	6	7
16	6	6	7
32	6	7	7
64	6	7	7
128	7	9	7
256	7	8	7
512	6	6	8
1024	6	7	8
diagonal	$M(M^{-1}L)^{1/2}$		
2	5	6	6
4	6	6	6
8	6	6	6
16	7	6	6
32	7	6	6
64	7	6	6
128	7	6	6
256	7	6	6
512	6	6	6
1024	6	6	6
diagonal	$M + M(M^{-1}L)^{1/2}$		
2	5	6	6
4	7	6	6
8	7	7	6
16	7	7	7
32	7	7	7
64	7	7	7
128	7	7	7
256	7	7	7
512	7	7	7
1024	7	7	7

Table 3.20: Iterations needed for reducing the initial residual of the Schur-complement system by a factor of 10^{-6}, for all three versions of the preconditioner with the diagonal version of the mass matrix M and $\alpha = 1, 10$. Lanczos $k = 5$.

mesh level	1	2	3
subdomains	Iterations		
$\alpha = 100$	$L^{1/2}$		
2	5	6	6
4	6	6	6
8	6	6	7
16	6	6	6
32	6	6	6
64	7	7	7
128	7	9	7
256	7	8	7
512	6	6	8
1024	6	6	8
diagonal	$M(M^{-1}L)^{1/2}$		
2	5	6	6
4	6	6	6
8	7	6	6
16	7	6	6
32	7	6	6
64	7	7	6
128	7	6	6
256	7	7	6
512	6	6	6
1024	7	6	6
diagonal	$M + M(M^{-1}L)^{1/2}$		
2	5	6	6
4	7	6	6
8	7	6	6
16	7	7	6
32	7	7	6
64	7	7	6
128	7	7	6
256	7	7	7
512	7	7	6
1024	7	7	6

mesh level	1	2	3
subdomains	Iterations		
$\alpha = 1000$	$L^{1/2}$		
2	5	6	6
4	7	7	6
8	8	7	6
16	7	7	6
32	8	7	6
64	8	7	7
128	8	10	7
256	8	9	7
512	7	7	8
1024	7	7	8
diagonal	$M(M^{-1}L)^{1/2}$		
2	5	6	6
4	7	7	6
8	8	7	6
16	8	7	6
32	8	7	6
64	8	7	6
128	8	7	7
256	8	7	7
512	7	7	7
1024	7	7	6
diagonal	$M + M(M^{-1}L)^{1/2}$		
2	5	6	6
4	7	7	6
8	8	7	6
16	8	7	6
32	8	7	6
64	8	7	6
128	8	7	7
256	8	7	7
512	7	7	7
1024	7	7	7

Table 3.21: Iterations needed for reducing the initial residual of the Schur-complement system by a factor of 10^{-6}, for all three versions of the preconditioner with the diagonal version of the mass matrix M and $\alpha = 100, 1000$. Lanczos $k = 5$.

Long pipe 3D

We proceed with the 3D equivalent of the previous problem. The configuration is the same one we used in section 3.11.2. The pipe here is manually partitioned as well in slices parallel to the XY plane, as shown in Figure 3.3. The mesh levels used for this benchmark and their characteristics are listed in Table 3.10. The number of unknowns on the interface of the subdomains for each of the partitions and mesh levels considered can be found in Table 3.9.

In Tables 3.22 and 3.23 below, we compare the iterations needed for reducing the initial residual of the Schur-complement system by six orders of magnitude for several different values of the parameter α $(1, 10, 100, 1000)$. The mass matrix in the last two versions of the preconditioner has been replaced by its diagonal.

We see that the versions of the preconditioner that involve the mass matrix outperform the mass-matrix-free version $L^{1/2}$. On the other hand, all versions are clearly h-independent. The dependence on the number of the partitions seems to have vanished, especially for the versions of the preconditioner that involve the mass matrix.

We verify here as well that the second version of the preconditioner $M(M^{-1}L)^{1/2}$ performs asymptotically better than the third version $M + M(M^{-1}L)^{1/2}$ by one iteration. Undoubtedly the numbers of iterations in this problem are a lower bound for the performance of the preconditioner in terms of iteration counts. We do not expect better performance in the experiments that follow, but this benchmark allows us to see how much room there is for improvement.

mesh level	1	2	3
subdomains	Iterations		
$\alpha = 1$	$L^{1/2}$		
4	12	11	11
8	11	11	13
16	10	11	13
32	10	13	14
64	13	14	15
128	14	15	15
256	15	16	15
512	15	15	16
1024	14	15	16
2048	18	17	17
diagonal	$M(M^{-1}L)^{1/2}$		
4	9	10	10
8	9	11	12
16	10	10	11
32	10	11	12
64	11	11	12
128	10	12	13
256	11	12	12
512	11	12	13
1024	11	12	13
2048	15	14	14
diagonal	$M + M(M^{-1}L)^{1/2}$		
4	9	11	11
8	9	11	12
16	10	11	12
32	10	12	12
64	12	12	13
128	11	13	13
256	12	13	13
512	12	12	13
1024	12	13	13
2048	16	15	15

mesh level	1	2	3
subdomains	Iterations		
$\alpha = 10$	$L^{1/2}$		
4	12	11	11
8	11	11	13
16	10	11	13
32	10	13	14
64	13	14	15
128	14	15	14
256	15	16	15
512	15	15	16
1024	14	15	16
2048	18	17	16
diagonal	$M(M^{-1}L)^{1/2}$		
4	9	10	11
8	9	11	12
16	10	10	11
32	9	11	12
64	11	11	12
128	10	12	13
256	11	12	12
512	12	12	13
1024	11	12	13
2048	15	13	13
diagonal	$M + M(M^{-1}L)^{1/2}$		
4	9	10	11
8	9	11	12
16	10	11	12
32	10	12	12
64	12	12	13
128	11	13	13
256	12	13	13
512	12	12	13
1024	11	13	13
2048	15	14	14

Table 3.22: Iterations needed for reducing the initial residual of the Schur-complement system by a factor of 10^{-6}, for all three versions of the preconditioner with the diagonal version of the mass matrix M and $\alpha = 1, 10$. Lanczos $k = 20$.

mesh level	1	2	3
subdomains	Iterations		
$\alpha = 100$	$L^{1/2}$		
4	12	10	11
8	11	11	13
16	10	11	13
32	9	13	14
64	13	14	15
128	15	15	14
256	15	16	15
512	16	15	16
1024	15	15	16
2048	17	17	16
diagonal	$M(M^{-1}L)^{1/2}$		
4	9	10	11
8	9	11	12
16	10	11	11
32	9	11	12
64	11	12	12
128	10	12	13
256	12	12	12
512	12	12	13
1024	11	12	13
2048	14	13	13
diagonal	$M + M(M^{-1}L)^{1/2}$		
4	9	10	11
8	9	11	12
16	10	11	12
32	9	12	12
64	11	12	13
128	10	12	13
256	12	12	13
512	12	12	13
1024	11	13	13
2048	14	13	13

mesh level	1	2	3
subdomains	Iterations		
$\alpha = 1000$	$L^{1/2}$		
4	13	9	10
8	13	10	13
16	10	10	12
32	9	12	13
64	15	14	15
128	16	14	14
256	16	15	15
512	16	16	17
1024	14	14	16
2048	17	16	17
diagonal	$M(M^{-1}L)^{1/2}$		
4	10	9	11
8	11	9	12
16	9	11	11
32	9	11	12
64	12	12	12
128	12	12	13
256	13	12	13
512	12	12	13
1024	11	12	13
2048	14	12	13
diagonal	$M + M(M^{-1}L)^{1/2}$		
4	10	9	11
8	11	9	12
16	9	11	11
32	9	11	12
64	12	12	12
128	12	12	13
256	13	12	13
512	12	12	13
1024	11	12	13
2048	14	12	13

Table 3.23: Iterations needed for reducing the initial residual of the Schur-complement system by a factor of 10^{-6}, for all three versions of the preconditioner with the diagonal version of the mass matrix M and $\alpha = 100, 1000$. Lanczos $k = 20$.

Unit square

We proceed with the unit square, which we discretize on the same meshes we used for the corresponding Laplace problem. The domain is manually split into $4, 16, 64$ and 256 equal size squares as shown in Figure 3.4. Details about the mesh levels considered can be found in Table 3.13, while the numbers of unknowns on the interface of subdomains are listed in Table 3.14.

We consider four different PDE problems corresponding to different values of the reaction coefficient α $(1, 10, 100, 1000)$. As for the Laplace problem we verify here as well that the number of iterations is independent of the mesh size h. For this case, however, the performance in terms of iterations of the second version of the preconditioner is slightly better than for the other two, in contrast to what we experienced for the Laplace problem. Moreover the number of iterations on the number of subdomains decreases as the reaction coefficient grows. Finally we see that for $\alpha = 1000$ it drops $(p = 64, 256)$ slightly when the number of subdomains increases. Table 3.24 lists the iteration counts for each of the four problems. We should note that the matrix L appearing in the definition of our preconditioner is the discretization of the operator $-\nabla^2 + \alpha I$.

Unit cube

We proceed with the 3D version of the previous experiment. The unit cube is internally split to $8, 64, 512$ cubes of equal size as shown in Figure 3.5. The finite element meshes are constructed as in the corresponding Poisson problem in section 3.11.2. Details about the meshes are listed in Table 3.16, while the nodes on the interface of the subdomains for each one of the cases examined here can be found in Table 3.17. As in the previous case the matrix L appearing in the definition of our preconditioner is the discretization of the operator $-\nabla^2 + \alpha I$. We use here as well the Lanczos iteration with the inverse matrix. There are no essential differences for values of k greater than $k = 5$. However in our computations we used $k = 20$.

In the Tables 3.25, 3.26, we see the iterations needed for reducing the initial residual of the Schur-complement system by six orders of magnitude, for each one of the four PDE problems corresponding to the four different values of the parameters α $(1, 10, 100, 1000)$.

mesh level	1	2	3	4	5
$\alpha = 1$	Iterations				
subdomains	$L^{1/2}$				
4	9	10	11	11	11
16	11	12	13	13	13
64	16	16	16	17	17
256	23	21	21	22	23
diagonal	$M(M^{-1}L)^{1/2}$				
4	9	9	10	10	10
16	12	12	13	13	14
64	14	15	15	15	16
256	20	20	20	20	21
diagonal	$M + M(M^{-1}L)^{1/2}$				
4	9	9	10	10	10
16	12	13	13	13	14
64	15	16	15	16	16
256	21	21	21	21	22
$\alpha = 10$	Iterations				
subdomains	$L^{1/2}$				
4	9	10	11	10	11
16	11	12	13	13	13
64	15	16	16	16	17
256	22	21	21	22	23
diagonal	$M(M^{-1}L)^{1/2}$				
4	9	9	9	10	10
16	12	12	13	13	13
64	14	15	15	15	16
256	20	20	20	20	21
diagonal	$M + M(M^{-1}L)^{1/2}$				
4	9	9	10	10	10
16	12	12	13	13	13
64	15	15	15	16	16
256	20	20	21	21	22

mesh level	1	2	3	4	5
$\alpha = 100$	Iterations				
subdomains	$L^{1/2}$				
4	8	9	11	10	10
16	11	11	12	12	12
64	13	14	14	14	14
256	19	18	19	19	19
diagonal	$M(M^{-1}L)^{1/2}$				
4	8	9	9	9	9
16	11	11	12	12	12
64	13	13	14	14	14
256	18	18	19	19	19
diagonal	$M + M(M^{-1}L)^{1/2}$				
4	8	9	9	9	9
16	11	11	12	12	12
64	14	14	14	14	14
256	19	19	19	19	20
$\alpha = 1000$	Iterations				
subdomains	$L^{1/2}$				
4	13	14	15	14	15
16	14	15	17	16	17
64	14	15	14	15	16
256	14	12	13	13	14
diagonal	$M(M^{-1}L)^{1/2}$				
4	12	12	12	12	12
16	16	15	15	14	15
64	14	15	15	15	15
256	13	12	12	13	13
diagonal	$M + M(M^{-1}L)^{1/2}$				
4	12	12	12	12	12
16	15	14	14	14	15
64	13	14	14	14	14
256	12	11	12	12	13

Table 3.24: Unit square. Iterations needed for reducing the initial residual of the Schur-complement system by a factor of 10^{-6} for all three versions of the preconditioner. The mass matrix is replaced by its diagonal and L is the discretization of the operator of the original problem. Lanczos $k = 15$.

mesh level	1	2	3	4
$\alpha = 1$	Iterations			
subdomains	$L^{1/2}$			
8	17	21	19	23
64	24	25	26	26
512	22	32	31	32
full	$M(M^{-1}L)^{1/2}$			
8	24	27	25	29
64	28	30	32	35
512	28	41	41	42
diagonal	$M(M^{-1}L)^{1/2}$			
8	21	23	22	24
64	24	25	27	29
512	24	34	33	34
full	$M + M(M^{-1}L)^{1/2}$			
8	24	28	26	30
64	30	31	34	36
512	30	43	44	44
diagonal	$M + M(M^{-1}L)^{1/2}$			
8	21	23	23	25
64	24	26	27	30
512	25	35	34	35

mesh level	1	2	3	4
$\alpha = 10$	Iterations			
subdomains	$L^{1/2}$			
8	17	21	19	23
64	24	24	25	26
512	21	31	31	32
full	$M(M^{-1}L)^{1/2}$			
8	23	27	25	29
64	28	30	32	35
512	28	41	40	42
diagonal	$M(M^{-1}L)^{1/2}$			
8	21	23	22	24
64	24	25	27	29
512	24	33	33	33
full	$M + M(M^{-1}L)^{1/2}$			
8	24	28	26	29
64	30	31	34	36
512	29	42	42	44
diagonal	$M + M(M^{-1}L)^{1/2}$			
8	21	23	23	24
64	24	26	27	29
512	24	34	34	35

Table 3.25: Unit cube. Iterations needed for reducing the initial residual of the Schur-complement system by a factor of 10^{-6} for all three versions of the preconditioner with Lanczos $k = 20$ for the case of manually constructed partitions and $\alpha = 1, 10$.

mesh level	1	2	3	4
$\alpha = 100$	Iterations			
subdomains	$L^{1/2}$			
8	16	20	18	22
64	23	23	25	25
512	18	28	28	28
full	$M(M^{-1}L)^{1/2}$			
8	23	26	25	27
64	27	28	30	32
512	26	37	37	38
diagonal	$M(M^{-1}L)^{1/2}$			
8	20	22	21	23
64	23	24	25	27
512	22	31	30	31
full	$M + M(M^{-1}L)^{1/2}$			
8	23	27	25	27
64	27	29	31	33
512	27	39	39	39
diagonal	$M + M(M^{-1}L)^{1/2}$			
8	20	22	21	24
64	23	24	26	27
512	22	32	31	32

mesh level	1	2	3	4
$\alpha = 1000$	Iterations			
subdomains	$L^{1/2}$			
8	15	19	16	20
64	21	22	23	24
512	14	25	25	25
full	$M(M^{-1}L)^{1/2}$			
8	21	24	23	25
64	25	25	28	29
512	19	30	30	31
diagonal	$M(M^{-1}L)^{1/2}$			
8	19	21	20	22
64	22	22	24	25
512	17	26	25	26
full	$M + M(M^{-1}L)^{1/2}$			
8	22	24	23	26
64	25	26	28	29
512	20	30	30	31
diagonal	$M + M(M^{-1}L)^{1/2}$			
8	19	21	20	22
64	22	22	24	25
512	17	26	25	26

Table 3.26: Unit cube. Iterations needed for reducing the initial residual of the Schur-complement system by a factor of 10^{-6} for all three versions of the preconditioner with Lanczos $k = 20$ for the case of manually constructed partitions and $\alpha = 100, 1000$.

3.12 Computational experiments beyond the theory

Here we consider computational experiments not necessarily in accordance with the theoretical assessments. Everything discussed in section 3.11 related to the cases considered and the benchmark components remain as before. The main differences arise from the following modifications.

Partitioning approach The standard partitioning approach employed in section 3.11 is not always applicable to real-life problems set up in three spatial dimensions, where the underlying geometries are far from trivial. For this reason, we consider the case where our domain is automatically partitioned by a standard mesh-partitioning tool. A multilevel k-way scheme is employed, provided by the software library METIS [85, 86], which is a graph-partitioning tool. The purpose of the underlying algorithm for our case is to obtain equal size partitions minimizing the total edge-cut and consequently the total number of nodes on the interface of subdomains.

Matrix exponent The optimal matrix exponent predicted by the analysis (square root appearing in all three versions of the preconditioner) is no longer considered fixed. We examine different matrix exponents and suggest the best one for each problem considered.

Long pipe 3D

We consider again the example of section 3.11.2. This time the partitioning does not produce planar separators parallel to the XY plane. The separators and the partitioning to eight partitions are shown in Figure 3.6. The mesh levels used for this benchmark and their characteristics are listed in Table 3.27. The number of unknowns on the interface of the subdomains for each of the partitions and mesh levels considered can be found in Table 3.28.

mesh level				
level	elements	nodes	$h_{\min}$	$h_{\max}$
1	207442	59055	9.556e-02	6.774e-01
2	1627405	348350	5.732e-02	3.371e-01
3	12795605	2346496	2.953e-02	1.701e-01

Table 3.27: Mesh levels used. For each level we see the number of elements, number of nodes, maximum and minimum edge lengths.

In Table 3.29 we compare the iterations needed for reducing the initial residual of the Schur-complement system by six orders for both METIS and manually generated partitions. The value k has been fixed to $k = 15$ while the mass matrix in the last two versions of the preconditioner

	mesh level		
subdomains	1	2	3
4	54	175	639
8	125	409	1490
16	278	891	3190
32	582	1851	6559
64	1204	3753	13382
128	2521	7636	27011
256	5015	16384	53997
512	10057	31387	108586
1024	19996	62081	218885
2048	28796	102563	389332

Table 3.28: Nodes on the interface of the subdomains for the partitions generated by METIS.

has been replaced by its diagonal. We consider first the $1/2$ version of the preconditioners. The partitioning generated by METIS seems to maintain the h-independence. The number of iterations, however, has become more sensitive to the number of subdomains and has increased.

We then modify the exponent and investigate its effect on the number of iterations. The exponent that performs best was found to be 0.7. The iterations for this new family of preconditioner and both manual and METIS-generated partitions are listed in Table 3.30 for Lanczos $k = 15$. The number of iterations has decreased for both partitioning types. The h-independence is maintained and even enhanced for all versions of the preconditioner.

For the METIS-generated partitions we observe that the number of iterations increases rapidly after some point (64 subdomains). We see in what follows that in most cases considered, the iterations are around 16 to 17 even for a very large number of subdomains with partitions generated by METIS.

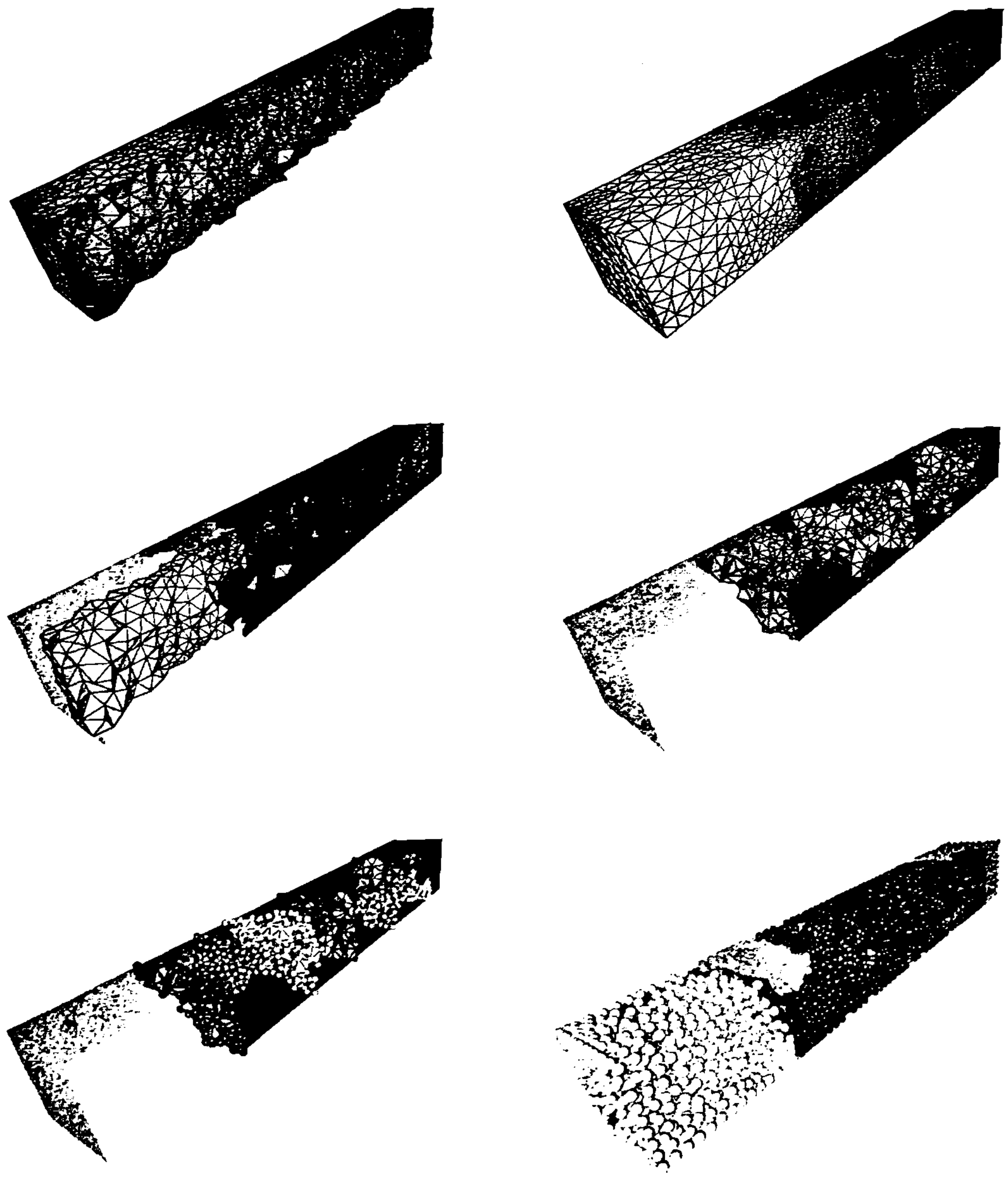


Figure 3.6: A piece of the pipe manually into eight subdomains by METIS. Note the interface of the subdomains (gray), the nodes on the interface of the subdomains and the nodes in the interior.

Manual Partitions			
mesh level	1	2	3
subdomains	Iterations		
$k = 15$	$L^{1/2}$		
4	11	10	11
8	11	11	13
16	10	11	13
32	10	14	14
64	13	15	15
128	14	16	14
256	14	16	15
512	15	16	16
1024	14	16	17
2048	18	17	17
diagonal	$M(M^{-1}L)^{1/2}$		
4	9	10	11
8	9	11	12
16	10	11	12
32	10	12	12
64	11	12	13
128	10	12	13
256	11	12	13
512	12	12	13
1024	11	13	13
2048	15	14	15
diagonal	$M + M(M^{-1}L)^{1/2}$		
4	9	11	12
8	9	11	13
16	10	11	13
32	10	12	13
64	11	13	14
128	11	13	14
256	12	13	14
512	12	13	14
1024	12	13	14
2048	16	15	16

METIS Partitions			
mesh level	1	2	3
subdomains	Iterations		
$k = 15$	$L^{1/2}$		
4	9	10	11
8	9	11	12
16	11	11	13
32	12	11	12
64	12	12	14
128	13	14	14
256	14	15	14
512	16	15	15
1024	17	15	16
2048	20	19	18
diagonal	$M(M^{-1}L)^{1/2}$		
4	9	11	12
8	10	12	13
16	10	12	14
32	14	12	14
64	14	13	14
128	14	17	15
256	15	17	15
512	15	17	16
1024	17	17	17
2048	21	21	21
diagonal	$M + M(M^{-1}L)^{1/2}$		
4	9	12	13
8	10	12	14
16	11	13	15
32	14	13	15
64	14	13	15
128	15	18	16
256	15	18	16
512	15	18	17
1024	18	18	18
2048	21	22	22

Table 3.29: Manually constructed versus METIS-generated partitions. Exponent is set to $1/2$ while the mass matrix is replaced by its diagonal and Lanczos $k = 15$.

Manual Partitions			
mesh level	1	2	3
subdomains	Iterations		
$k = 15$	$L^{0.7}$		
4	11	9	9
8	11	10	12
16	9	10	11
32	9	12	12
64	12	13	13
128	13	14	13
256	13	14	14
512	14	15	14
1024	12	14	15
2048	17	15	15
diagonal	$M(M^{-1}L)^{0.7}$		
4	8	9	9
8	9	9	10
16	9	9	10
32	8	10	10
64	10	11	11
128	10	11	11
256	10	11	11
512	11	11	11
1024	10	11	11
2048	13	11	12
diagonal	$M + M(M^{-1}L)^{0.7}$		
4	8	9	9
8	8	9	10
16	9	9	10
32	8	10	10
64	10	11	11
128	10	11	11
256	10	11	11
512	11	11	11
1024	10	11	11
2048	14	12	12

METIS Partitions			
mesh level	1	2	3
subdomains	Iterations		
$k = 15$	$L^{0.7}$		
4	8	9	10
8	10	10	11
16	11	10	12
32	13	11	12
64	13	11	13
128	13	14	13
256	14	14	14
512	17	16	14
1024	18	14	16
2048	19	18	17
diagonal	$M(M^{-1}L)^{0.7}$		
4	8	10	11
8	10	11	12
16	11	11	13
32	13	11	13
64	13	11	13
128	14	17	13
256	15	16	14
512	16	17	14
1024	18	16	16
2048	20	19	18
diagonal	$M + M(M^{-1}L)^{0.7}$		
4	8	10	11
8	10	11	12
16	10	11	13
32	13	11	13
64	13	11	13
128	14	16	13
256	14	16	14
512	15	17	14
1024	18	16	16
2048	20	19	18

Table 3.30: Iterations needed for reducing the initial residual of the Schur-complement system by a factor of 10^{-6}, with the 0.7-family of preconditioners and both manually and METIS-generated partitions. Mass matrix is replaced by its diagonal and Lanczos $k = 15$.

Unit cube

We study here the performance of the preconditioners on partitions produced by METIS. The benchmark was performed on the mesh levels described in Table 3.31. The partitioner allows greater flexibility in the number of desired subdomains and thus we were able to perform a more extensive test in terms of number of subdomains and mesh levels. The first mesh level consists of 500 nodes and 2079 tetrahedra, while the final level consists of 20 million nodes and approximately 127 million tetrahedra. At the same time the number of subdomains ranges from 2 to 16384. We should note, however, that only a subset of the number of partitions was considered for each level. This is because partitions with very few unknowns are not desirable in practice because the number of unknowns on the interface increases considerably, with an adverse effect on the performance. The number of interface unknowns for each level and partitioning is listed in Table 3.32. Figure 3.7 visualizes the second mesh level partitioned by METIS into 64 subdomains along with the interface boundary of the subdomains.

level	elements	nodes	h_{min}	h_{max}
1	2079	500	9.8858e-02	2.7123e-01
2	20869	4001	3.1216e-02	1.3442e-01
3	185963	32002	1.5604e-02	6.8082e-02
4	1567874	256011	7.7944e-03	3.3546e-02
5	12568378	2000396	3.7695e-03	1.7208e-02
6	127400903	20000768	1.7757e-03	8.0723e-03

Table 3.31: Mesh levels for METIS generated partitions. For each level we see the number of elements, number of nodes, maximum and minimum edge length.

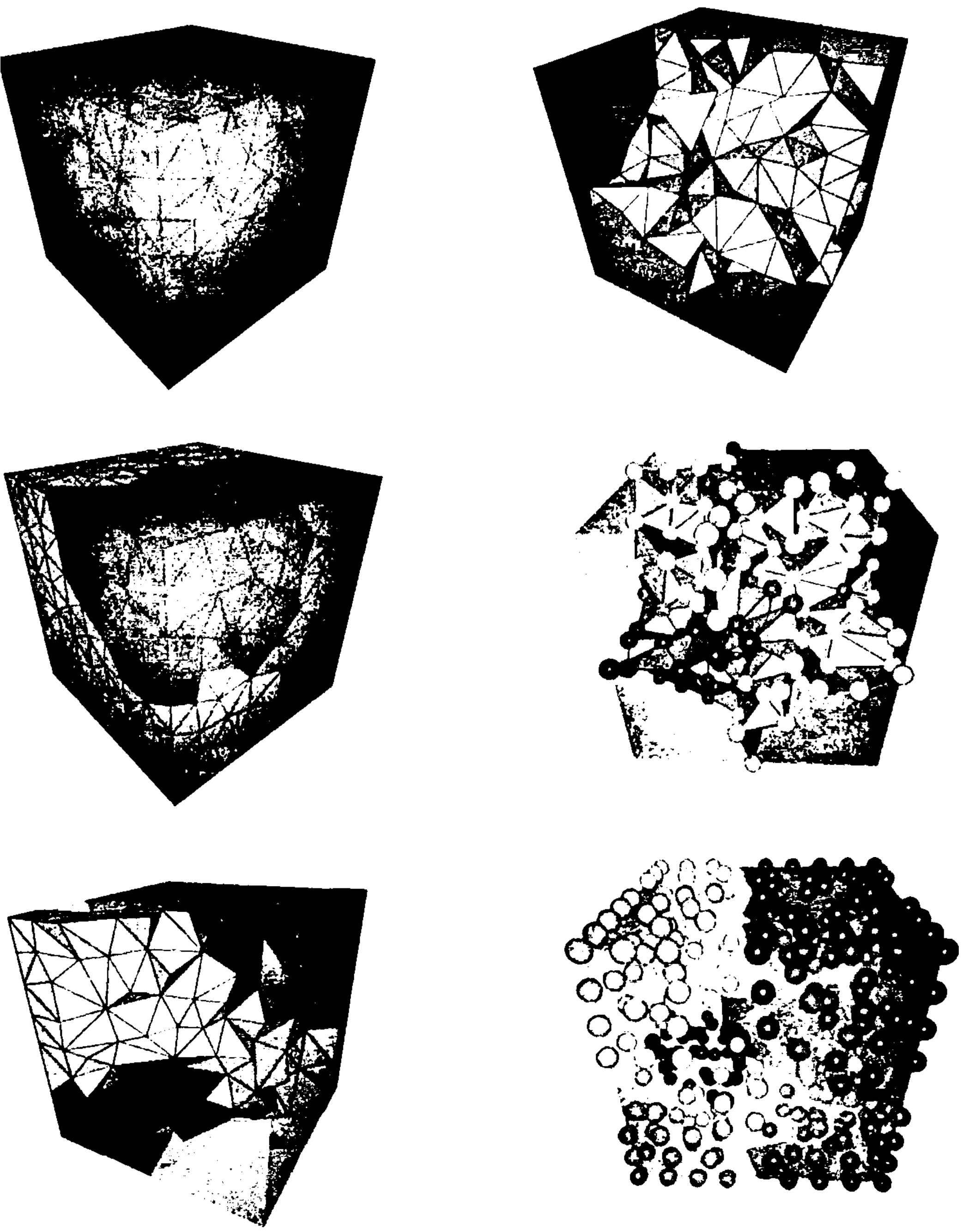
We begin by considering the first five levels of Table 3.31 and partitions ranging from 2 to 16384. We benchmark the three versions of our preconditioner with the mass matrix replaced by its diagonal, which was found to perform better than the full version by 2 to 3 iterations for every single case and furthermore does not require a linear system to be factorized and solved at each iteration. The results are summarized in Table 3.33 for two different values of Lanczos k $(3, 20)$. We see that in most cases we achieve h-independence for all three versions of the preconditioner. Wherever this is not evident, the number of iterations increases logarithmically with the refinement level. The sensitivity to the number of subdomains is also clear. The most efficient version of the preconditioner for this problem, in terms of iterations, is the first one $(L^{1/2})$. In what follows we present how the sensitivity of the number of iterations with respect to the number of subdomains can be diminished without any additional computational cost.

	mesh level					
subdomains	1	2	3	4	5	6
2	71	290	1183			
4	124	551	2314	9621		
8	189	785	3552	13780	55259	
16		1190	5216	21686	88701	
32		1565	7160	29961	124023	
64			9213	39476	160484	
128			11770	51465	216556	1056946
256			14800	65696	278533	1364200
512				83015	353147	1745987
1024					447034	2222309
2048					559011	2796220
4096					692256	3503214
8192						4369353
16384						5419460

Table 3.32: Nodes on the interface of the subdomains for the unit cube.

Towards h- and subdomains-independence

We proceed by investigating the effect of the exponent of the family of preconditioners on their efficiency. The same test as before is considered here on all mesh levels summarized in Table 3.31 for the first version of our preconditioner, which was found to be the most efficient one in the previous benchmark. This time, however, the exponent is not the one suggested theoretically. Numerical experiments showed that the optimal value is 0.77. The preconditioner here is therefore $L^{0.77}$. We consider four different values of Lanczos k $(3, 5, 10, 20)$. The results are summarized in Table 3.34. For large numbers of subdomains we observe that the number of iterations decreases as the mesh is refined. A similar phenomenon is observed for dependence on the number of partitions. The iterations seem to decrease as the number of partitions grows and then they increase again slowly. This behavior is quite nonstandard in the Domain Decomposition realm. Usually iterations either increase or remain constant (for very sophisticated algorithms) with mesh refinement level h or with the number of subdomains.



3.7: A sample cube partitioned by METIS to 8 subdomains. The column at the right shows rface boundary (top) the unknowns on the interface (middle) and the unknowns in the interior ubdomains (bottom).

mesh level	1	2	3	4	5
subdomains	Iterations				
$k = 3$	$L^{1/2}$				
4	12	15	19	24	35
8	14	16	21	28	36
16		19	23	28	39
32		20	25	30	41
64		22	27	36	46
128			29	38	51
256			33	42	56
512			37	47	62
1024				54	69
2048				60	71
4096				66	78
diagonal	$M(M^{-1}L)^{1/2}$				
4	14	16	19	22	26
8	18	18	20	24	28
16		20	22	23	27
32		21	22	25	28
64		26	26	25	28
128			28	26	30
256			33	34	30
512			39	38	32
1024				44	44
2048				51	52
4096				59	58
diagonal	$M + M(M^{-1}L)^{1/2}$				
4	14	17	19	22	26
8	18	18	20	25	28
16		21	22	24	28
32		23	23	25	28
64		27	27	26	28
128			30	27	31
256			35	37	32
512			42	41	35
1024				49	48
2048				54	56
4096				63	62

mesh level	1	2	3	4	5
subdomains	Iterations				
$k = 20$	$L^{1/2}$				
4	12	14	15	16	18
8	14	16	18	18	20
16		18	19	20	21
32		19	20	21	23
64		21	22	23	25
128			25	25	27
256			27	28	30
512			31	32	34
1024				37	40
2048				43	43
4096				50	50
diagonal	$M(M^{-1}L)^{1/2}$				
4	14	15	18	19	20
8	18	18	20	22	22
16		21	21	23	23
32		22	23	24	25
64		26	26	26	27
128			28	29	30
256			32	33	34
512			39	37	38
1024				45	44
2048				52	50
4096				59	57
diagonal	$M + M(M^{-1}L)^{1/2}$				
4	14	16	19	20	21
8	18	19	21	22	23
16		21	22	24	24
32		23	25	25	27
64		27	27	27	29
128			30	30	32
256			34	35	37
512			42	40	41
1024				49	48
2048				55	54
4096				64	60

Table 3.33: Iterations needed for reducing the initial residual of the Schur-complement system by a factor of 10^{-6}, for all three versions of the 1/2-family of preconditioners and the diagonal version of the mass matrix M. Lanczos $k = 3, 20$.

The effect of Lanczos k						
mesh level	1	2	3	4	5	6
subdomains	Iterations					
	$k=3$					
2	9	11	12			
4	11	11	13	16	19	
8	14	13	14	16	20	
16		14	14	16	21	
32		15	14	16	20	
64		16	15	15	19	
128			16	15	19	25
256			17	16	19	24
512			20	18	19	25
1024				20	21	25
2048				22	23	26
4096				25	25	26
8192					28	30
16384					33	36
	$k=5$					
2	10	11	12			
4	11	11	13	15	17	
8	14	13	13	15	18	
16		14	13	15	18	
32		15	14	15	17	
64		16	15	14	16	
128			16	14	17	20
256			17	15	16	19
512			20	17	16	20
1024				19	17	19
2048				21	18	20
4096				23	21	20
8192					23	23
16384					25	26

The effect of Lanczos k					
1	2	3	4	5	6
Iterations					
$k=10$					
9	11	12			
11	12	13	15	17	
14	13	14	15	18	
	14	14	15	18	
	15	15	15	17	
	16	16	14	15	
		16	14	15	18
		18	15	15	17
		20	17	15	18
			19	16	17
			21	17	17
			24	20	18
				22	19
				24	21
$k=20$					
9	11	12			
11	12	13	15	17	
14	13	14	16	18	
	14	14	16	19	
	15	15	15	17	
	16	16	15	16	
		16	15	16	19
		18	15	15	18
		20	17	15	17
			19	17	16
			21	17	16
			24	20	17
				22	18
				25	21

Table 3.34: The $L^{0.77}$ preconditioner and Lanczos $k = 3, 5, 10, 20$.

3.12.1 More complex geometries

We proceed now with harder geometrical configurations corresponding to real-life problems. In several cases, a finite element mesh may consist of very large and very small elements. This usually happens when a very peculiar shape has to be resolved by the corresponding surface mesh or when adaptive refinement algorithms are adopted to reduce the computational cost as much as possible. In such cases the largest and smallest volumes of the corresponding tetrahedra may differ by several orders of magnitude. In the cases studied so far we did not experience such problems because the ratio of the maximum to minimum edge length $h_{\max}/h_{\min}$ was less than one order of magnitude. Most preconditioners encountered in the literature such as multigrid solvers and several domain decomposition preconditioners either fail to converge or their performance in most of the cases is significantly deteriorated. Others do not touch this problem at all. In the problems that follow we consider meshes with large variations of the mesh size h. The benchmark problems reveal under which circumstances the mass matrix is essential in the definition of the preconditioners, and shed some light on the choice of optimal exponent.

We suspect that the optimal exponent is related to the regularity of the solution. The regularity of the solution of the Poisson equation is influenced not only by the non-homogeneous term but also by the regularity of the surface of the domain. In non-convex domains with a rough surface the regularity of the solution inevitably drops compared with that observed for the unit cube. If our assumption is correct that the optimal exponent is related to the regularity of the solution, we expect that the optimal exponent for the problems that follow will be less than the optimal exponent for the cube case, which was 0.77, and will drop as the domain's surface regularity decreases.

Brain

We begin with the non-trivial geometry of the human brain depicted in Figure 3.8. It is partitioned with METIS into 8 and 16 subdomains. The ratio $h_{\max}/h_{\min}$ is around two orders of magnitude, as we can see in Table 3.35. The numbers of unknowns on the interface of the subdomains can be found in Table 3.36. The linear system corresponding to the first mesh level consists of about half a million unknowns, while the unknowns for the last level rise to 26 million.

Table 3.37 lists the iterations needed for reducing the relative residual by a factor of 10^{-6} for four different values of Lanczos k $(5, 10, 20, 30)$. We see that the iterations for $k = 30$ do not differ too much from those obtained with $k = 5$. We consider only the second version of the preconditioner because it outperforms the others. The optimal exponent for this problem was 0.7, but the performance is similar with the 0.6 and slightly better than 0.5.

In most cases we experience h-independence. The dependence on the number of subdomains is not monotone. Iterations increase only slightly, if at all, as the number of partitions is doubled.

mesh level				
level	elements	nodes	$h_{\min}$	$h_{\max}$
1	2077212	468474	4.650e-05	1.101e-02
2	2112476	473730	4.643e-05	5.472e-03
3	31753520	5120357	2.148e-05	2.531e-03
4	164363725	25973106	4.643e-05	1.216e-03

Table 3.35: Mesh levels used. For each level we see the number of elements, number of nodes, maximum and minimum edge lengths.

	mesh level			
subdomains	1	2	3	4
2	1873	2064		
4	4290	4511		
8	7749	7976	51532	
16	13013	13314	91333	
32	18230	19077	133778	
64	25225	27327	192083	
128	35335	37159	267648	
256	49104	51010	372033	
512	66434	68949	504671	1543907
1024			679160	2067967
2048			895170	2737064
4096			1172815	3602083
8192				4676182
16384				5995113

Table 3.36: Nodes on the interface of the subdomains for the brain.

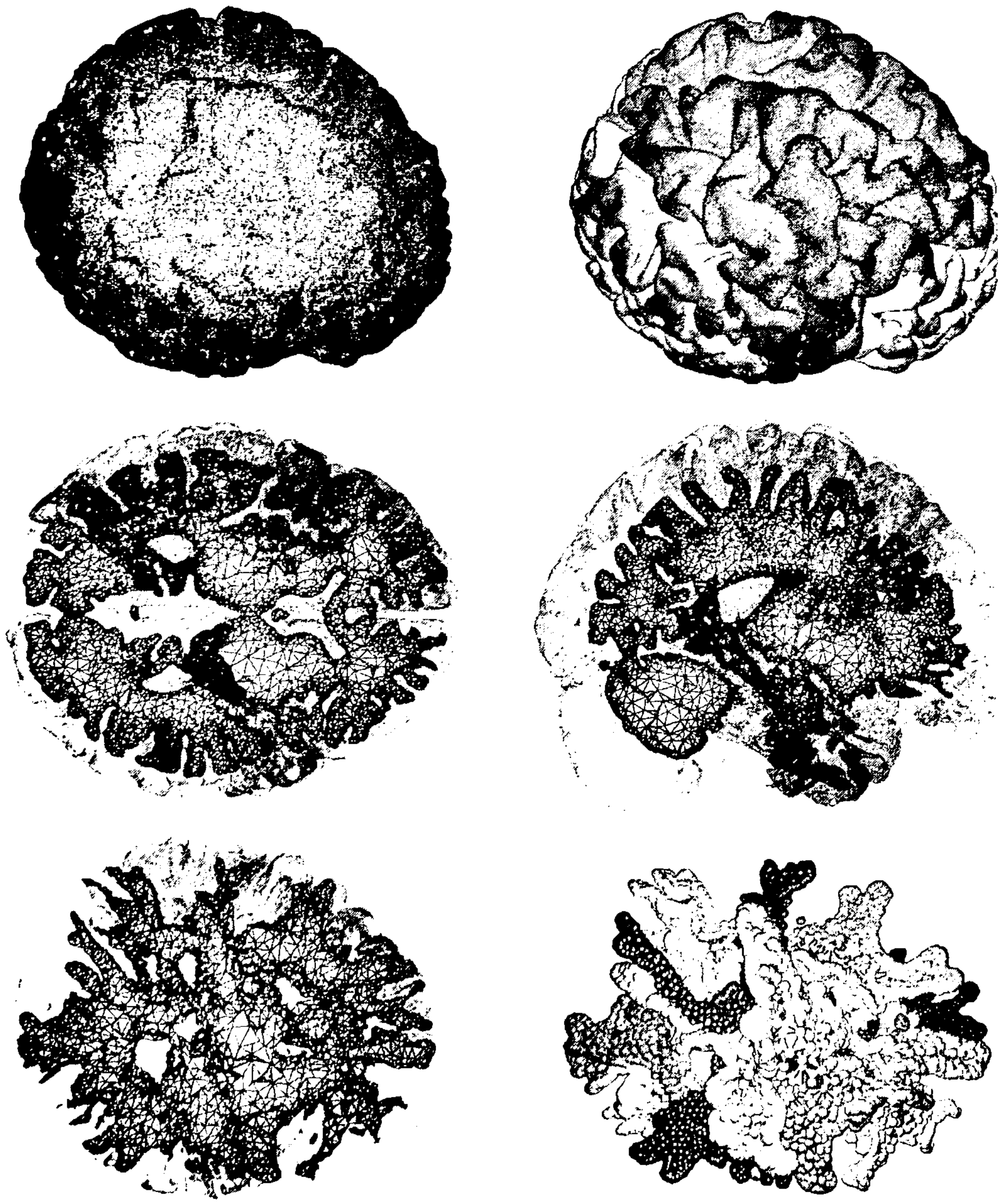


Figure 3.8: First level partitioned by METIS into 16 subdomains along with mesh slices, interface boundary and interface unknowns.

meshlevel	1	2	3	4
subdomains	$M(M^{-1}L)^{0.7}$			
$k = 5$	Iterations			
2	13	13		
4	14	15		
8	16	15	18	
16	19	17	25	
32	19	18	20	
64	19	19	21	
128	21	20	21	
256	23	21	21	
512	23	22	21	24
1024			23	24
2048			23	25
4096			24	24
8192				26
16384				26
$k = 10$	Iterations			
2	13	13		
4	14	14		
8	16	15	16	
16	19	16	24	
32	19	18	18	
64	19	19	19	
128	21	20	19	
256	23	21	20	
512	23	22	20	21
1024			22	22
2048			22	24
4096			24	23
8192				25
16384				26

meshlevel	1	2	3	4
subdomains	$M(M^{-1}L)^{0.7}$			
$k = 15$	Iterations			
2	13	13		
4	14	14		
8	16	15	16	
16	19	16	23	
32	19	18	18	
64	19	19	19	
128	21	20	19	
256	23	21	20	
512	22	22	20	21
1024			22	22
2048			22	23
4096			24	23
8192				25
16384				26
$k = 20$	Iterations			
2	13	13		
4	14	14		
8	16	15	16	
16	19	16	23	
32	19	18	18	
64	19	19	19	
128	21	20	19	
256	23	21	20	
512	22	22	20	21
1024			22	22
2048			22	23
4096			24	23
8192				25
16384				26

Table 3.37: Iterations needed for reducing the initial residual of the Schur-complement system by a factor of 10^{-6} for all three versions of the preconditioner with Lanczos $k = 5, 10, 20, 30$.

Boeing 747

We consider here the Boeing 747 depicted in Figure 3.9. The corners and cross-sections in the wings, tail, and engines should result in a less regular solution of the Poisson problem. Thus we expect an optimal exponent smaller than for the previous problem. This is indeed the case: the optimal exponent for this problem was found to be 0.62. The mesh levels we used are listed in Table 3.38 and the numbers of unknowns on the interface of the subdomains are in Table 3.39.

We observe as well that the iterations decrease as the mesh is refined. In several cases the independence of the number of partitions is striking. For instance for $k = 3, 5$ we see for the fourth mesh level, consisting of almost nine million unknowns, that the number of iterations either drops as the number of subdomains increases, or increases very slowly.

mesh level				
level	elements	nodes	$h_{\min}$	$h_{\max}$
1	231000	51366	2.580e-03	8.121e+00
2	416085	82170	2.557e-03	4.232e+00
3	4685686	780255	2.568e-03	2.158e+00
4	56071121	8987420	2.559e-03	1.821e+00

Table 3.38: Mesh levels used. For each level we see the number of elements, number of nodes, maximum and minimum edge lengths.

	mesh level			
subdomains	1	2	3	4
2	605	534		
4	519	1707		
8	952	1938	15420	
16	1582	3917	27076	
32	3328	6418	44339	
64	4387	9294	64618	345687
128		13275	89350	487023
256		18939	119486	654231
512			157176	867909
1024			204224	1129954
2048			258662	1458420

Table 3.39: Number of nodes on the interface of the subdomains for the Boeing 747.

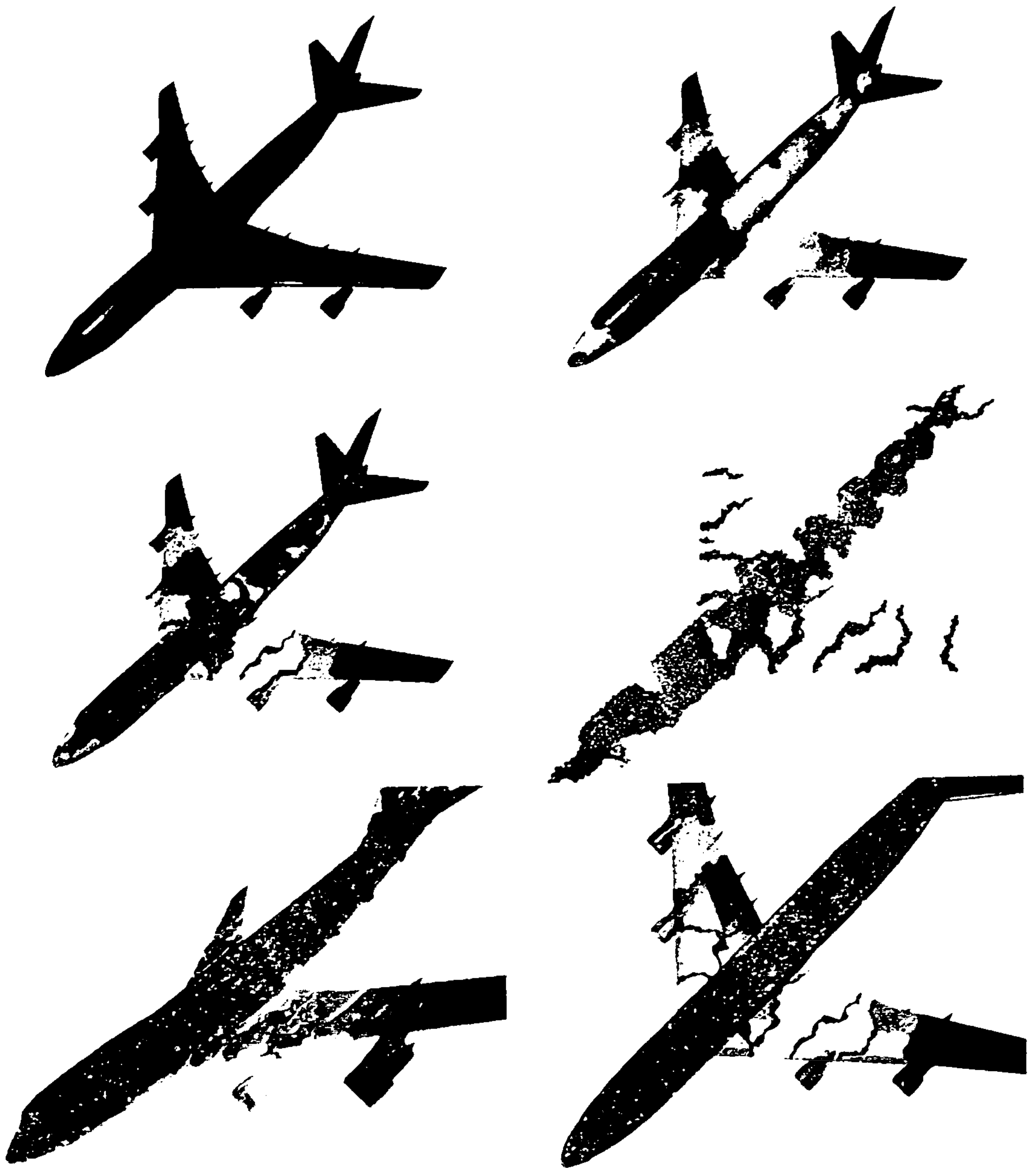


Figure 3.9: Second mesh level partitioned by METIS to 33 subdomains.

meshlevel	1	2	3	4
subdomains	$M(M^{-1}L)^{0.62}$			
$k = 3$	Iterations			
2	13	13		
4	12	14		
8	15	13	17	
16	18	16	17	
32	30	18	18	
64	27	23	20	23
128		24	20	28
256		34	20	24
512			23	24
1024			26	23
2048			24	22
$k = 5$	Iterations			
2	13	12		
4	12	14		
8	14	13	15	
16	17	16	15	
32	28	17	16	
64	24	22	18	21
128		24	17	24
256		32	18	21
512			23	22
1024			26	22
2048			24	22

meshlevel	1	2	3	4
subdomains	$M(M^{-1}L)^{0.62}$			
$k = 10$	Iterations			
2	13	12		
4	11	14		
8	13	13	14	
16	16	16	15	
32	26	17	16	
64	22	20	17	19
128		21	17	24
256		29	19	19
512			23	20
1024			26	22
2048			25	23
$k = 20$	Iterations			
2	13	12		
4	11	14		
8	13	13	14	
16	16	16	15	
32	24	17	16	
64	21	18	17	18
128		20	17	24
256		26	20	19
512			24	21
1024			26	22
2048			27	24

Table 3.40: Iterations needed for reducing the initial residual of the Schur-complement system by a factor of 10^{-6} for all three versions of the preconditioner with Lanczos $k = 3, 5, 10, 20$.

Crystal

We try here an even more peculiar shape: the crystal shown in Figure 3.10, where the first of the mesh levels considered is partitioned with METIS into 8 and 16 subdomains. The ratio $h_{\max}/h_{\min}$ is more than five orders of magnitude, as we can see in Table 3.41. At the same time the boundary of the crystal is very rough and we expect that the regularity of the solution will drop considerably compared with the regularity of the solution of the cube problem. As a result the optimal exponent is expected to be smaller than the optimal 0.77 exponent we found for the cube problem. The number of unknowns on the interface of the subdomains can be found in Table 3.42.

The numbers of iterations needed for these problems are listed in Table 3.43. We consider only the first two versions of the preconditioner because the last one always performs worse than the second in terms of iterations. The standard exponent 0.5 and the optimal exponent 0.6 for this problem display more or less similar performance, as the iterations with the latter are only one to five less than those with the former, unlike the previous problem, where we observed differences up to 24 iterations. This is expected because the regularity of the solution of this problem is much smaller than that of the cube; it is much closer to the optimal value predicted theoretically, which is 0.5. We also observe that unlike the cube case, where the most efficient version of the preconditioner was the mass-matrix-free one, consideration of the mass matrix here dramatically increases the performance of the preconditioner. This fact clearly states that **the mass matrix version of the preconditioner is needed whenever very large and tiny elements coexist in the same finite element mesh.**

In most of the cases we experience h-independence or the iterations decrease as the mesh is refined, especially for the 0.6 exponent. Irregular behavior is observed for the dependence of iterations on the number of partitions. They either remain constant or decrease or increase as the number of partitions grows.

Crystal				
level	elements	nodes	$h_{\min}$	$h_{\max}$
1	344679	67799	1.2841e-07	4.4729e-02
2	1054204	240832	1.2841e-07	2.7993e-02
3	12374659	2521753	1.2841e-07	1.4046e-02

Table 3.41: Mesh levels used. For each level we show the number of elements, number of nodes, maximum and minimum edge length.

	mesh level		
subdomains	1	2	3
2	648	730	
4	1416	2984	12430
8	2483	4353	35690
16	3963	6407	34043
32	5346	9720	55465
64	7940	14920	83462
128	10870	21286	120709
256		30414	169924
512		41839	232152
1024			315250

Table 3.42: Nodes on the interface of the subdomains for the crystal.

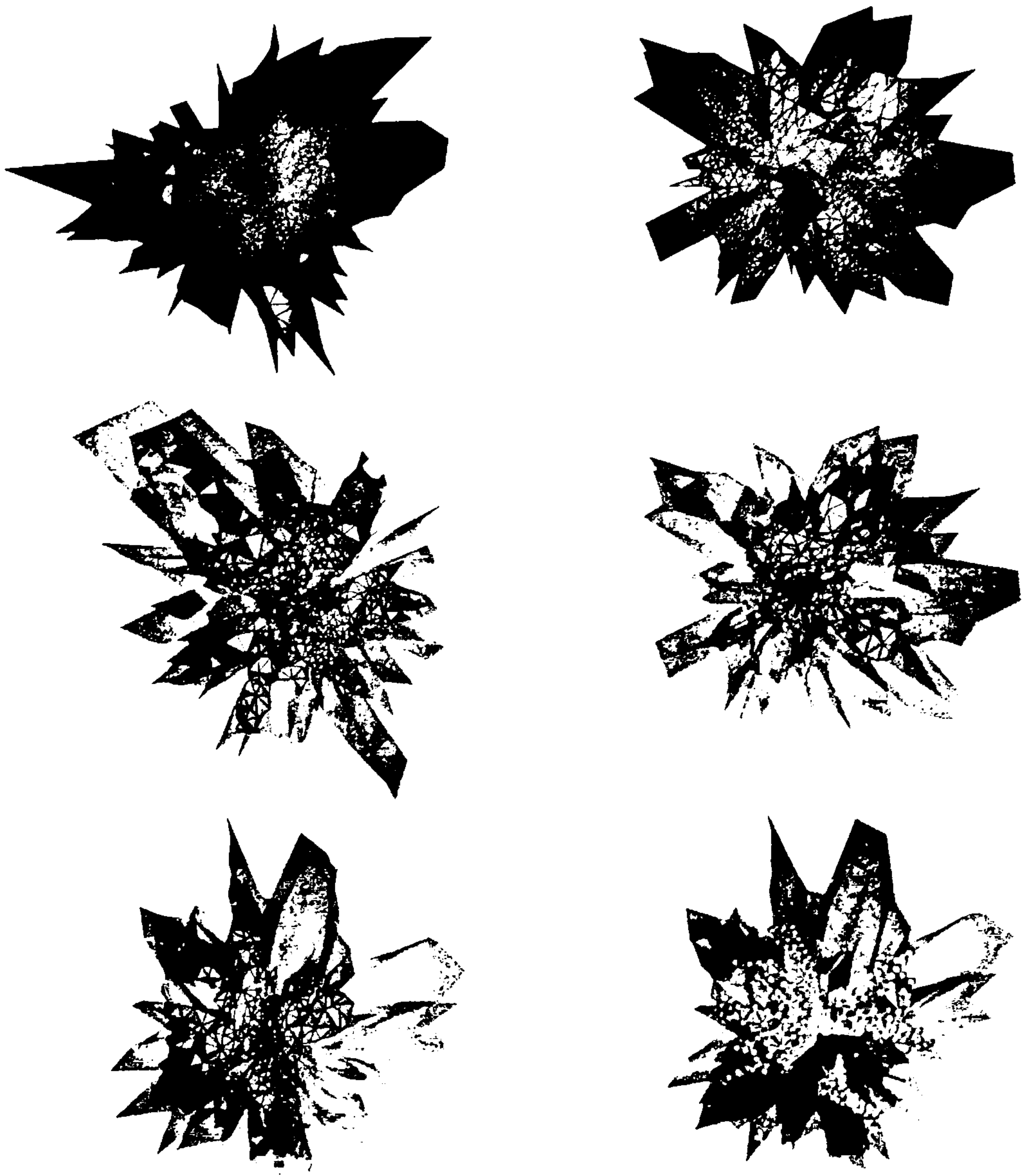


Figure 3.10: First level partitioned by METIS to 16 subdomains, a mesh-slice, interface boundary and unknowns on the interface.

mesh level	1	2	3	1	2	3	1	2	3	1	2	3
subdomains	$L^{1/2}$			$M(M^{-1}L)^{1/2}$			$L^{0.6}$			$M(M^{-1}L)^{0.6}$		
$k = 20$	Iterations			Iterations			Iterations			Iterations		
4	112	57	30	17	16	17	109	56	30	19	17	17
8	124	63	131	21	18	37	120	62	135	21	18	36
16	160	110	28	26	23	20	157	110	28	29	23	18
32	150	100	30	25	26	21	148	99	30	25	25	19
64	159	104	71	29	31	30	156	99	69	31	34	32
128	181	113	55	33	30	25	177	109	53	36	33	23
256		141	66		37	27		137	64		39	29
512		146	79		40	30		144	76		42	29
1024			80			32			77			36
$k = 40$	Iterations			Iterations			Iterations			Iterations		
4	112	57	29	16	16	17	109	56	30	17	16	16
8	124	63	131	19	16	32	120	62	135	20	16	34
16	160	110	28	22	18	21	157	110	28	26	18	18
32	150	100	30	22	21	22	148	99	30	24	22	18
64	159	104	71	25	25	26	156	99	69	27	26	27
128	181	113	55	28	25	25	177	109	53	30	26	21
256		141	65		32	28		137	64		33	26
512		147	78		32	29		144	75		32	26
1024			79			33			76			30
$k = 60$	Iterations			Iterations			Iterations			Iterations		
4	112	57	30	16	16	17	109	56	30	17	16	16
8	124	63	131	19	17	32	120	62	135	20	16	29
16	160	110	28	21	17	21	157	110	28	23	18	18
32	150	100	30	21	21	22	148	99	30	23	21	18
64	159	104	71	23	24	25	156	99	69	26	25	23
128	181	113	55	25	25	25	177	109	53	28	25	20
256		141	65		31	29		137	64		31	25
512		147	77		30	29		144	75		30	24
1024			79			33			76			27

Table 3.43: Iterations needed for reducing the initial residual of the Schur-complement system by a factor of 10^{-6}. The mass-matrix-free version of the preconditioner is not recommended when very large and tiny tetrahedra coexist on the same mesh. Lanczos $k = 20, 40, 60$.

F-16 Fighting Falcon

We proceed with a harder test case, the F-16 Fighting Falcon depicted in Figure 3.11. The corners and cross-sections in the wings, tail, and missiles should reduce the regularity of the solution even further. In this case we expect the optimal exponent to be smaller than for the Boeing 747 benchmark. This is indeed the case: the optimal exponent for this problem was found to be 0.54. The mesh levels we used are listed in Table 3.44 and the numbers of unknowns on the interface of the subdomains are in Table 3.45.

Table 3.46 lists the iterations needed for the second version of our preconditioner $M(M^{-1}L)^{\theta}$ with four different values for the exponent: $\theta = 0.5, 0.54, 0.6, 0.65$. The Lanczos subspace dimension k was fixed for all test cases to the value $k = 30$. Although we cannot claim independence of the number of subdomains, with the exception of the second column for all exponents, we can see especially for the exponent $\theta = 0.54$ practical independence of the mesh size h for all rows between 32 and 512 subdomains.

mesh level				
level	elements	nodes	$h_{\min}$	$h_{\max}$
1	436429	104381	1.222e-06	5.863e-01
2	2136627	404520	1.222e-06	3.700e-01
3	2882141	531103	1.222e-06	1.892e-01
4	25055374	4135416	1.222e-06	1.187e-01

Table 3.44: Mesh levels used. For each level we see the number of elements, number of nodes, maximum and minimum edge lengths.

	mesh level			
subdomains	1	2	3	4
2	412			
4	492	6075	3652	
8	1637	2916	5757	
16	2011	7440	14438	
32	3891	11535	15605	135288
64	5425	17435	24885	193570
128	8418	25936	37269	263479
256		39615	54443	351571
512		62602	79607	464651
1024				607313
2048				787200

Table 3.45: Number of nodes on the interface of the subdomains for the F-16.

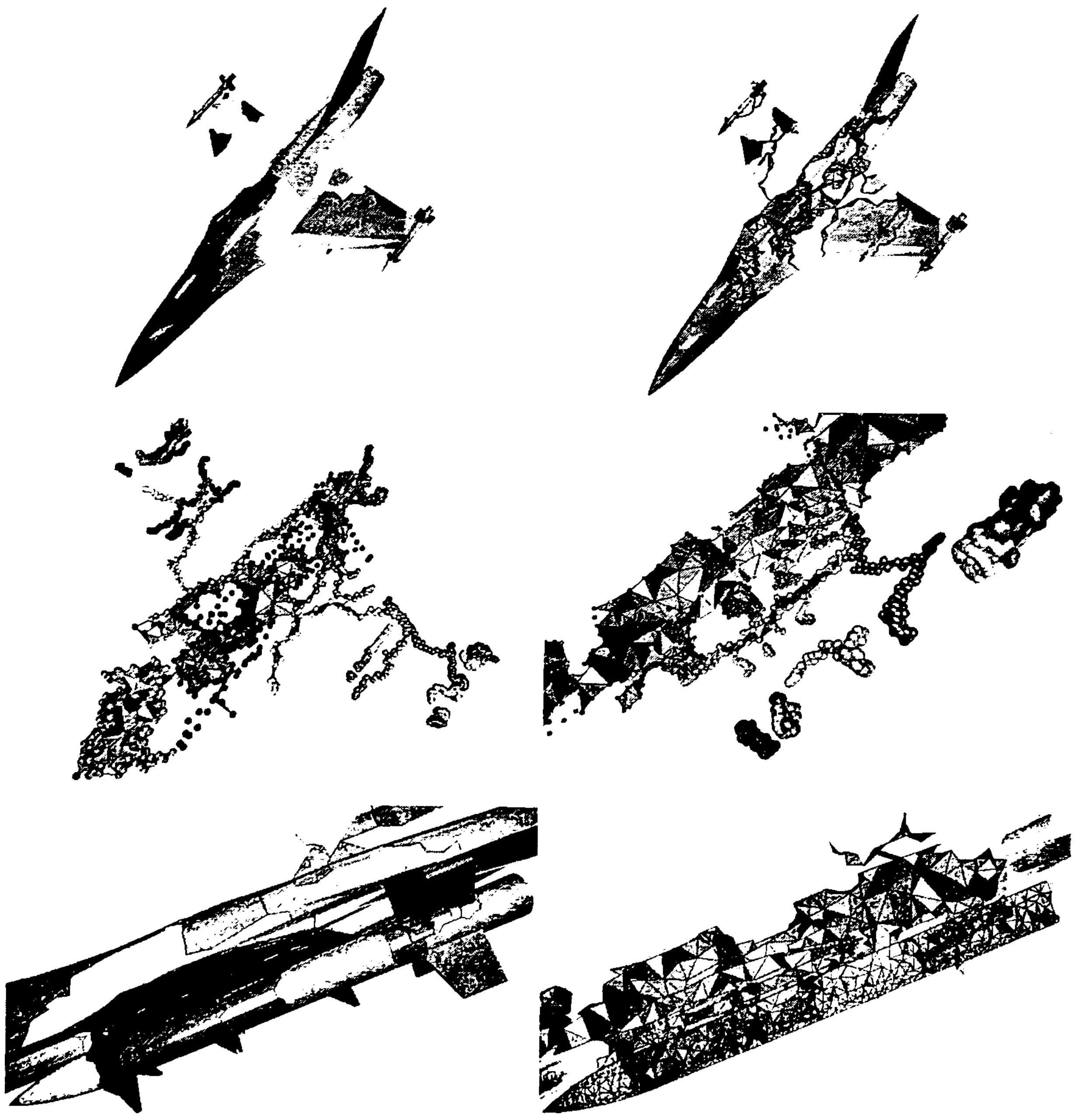


Figure 3.11: First mesh level partitioned by METIS to 55 subdomains.

meshlevel	1	2	3	4
	$M(M^{-1}L)^{0.5}$			
subdomains	Iterations			
2	12			
4	12	19	15	
8	16	17	23	
16	19	21	35	
32	20	22	23	24
64	22	22	25	24
127	26	25	27	27
256		28	31	30
512		31	32	33
1024				38
2048				40
	$M(M^{-1}L)^{0.54}$			
subdomains	Iterations			
2	12			
4	11	19	14	
8	15	16	24	
16	18	22	36	
32	19	22	22	22
64	22	22	25	22
127	25	24	27	24
256		29	31	28
512		31	32	29
1024				33
2048				35

meshlevel	1	2	3	4
	$M(M^{-1}L)^{0.6}$			
subdomains	Iterations			
2	12			
4	12	20	16	
8	16	15	23	
16	19	24	38	
32	19	22	23	21
64	22	22	26	22
127	25	25	27	22
256		29	33	28
512		31	34	27
1024				31
2048				31
	$M(M^{-1}L)^{0.65}$			
subdomains	Iterations			
2	13			
4	14	22	17	
8	17	14	23	
16	20	25	38	
32	20	23	23	20
64	23	25	26	22
127	26	28	28	21
256		31	32	28
512		34	35	26
1024				29
2048				31

Table 3.46: Iterations needed for reducing the initial residual of the Schur-complement system by a factor of 10^{-6} for the second version of the preconditioner with four different exponents $\theta = 0.5, 0.54, 0.6, 0.65$. Lanczos space dimension $k = 30$.

3.13 H_θ versus iterative substructuring in three dimensions

It would be interesting to compare the performance of our $H^{1/2}$ preconditioner with other popular domain decomposition approaches. The most competitive one is an iterative substructuring algorithm suggested by Smith [145], carefully designed to address problems in 3D and providing a fully parallel algorithm without sacrificing the robustness of optimal sequential iterative substructuring algorithms.

The second benchmark problem considered in [145] is a really tempting one as it considers a rather general case involving two different types of boundary conditions. The boundary $\partial\Omega$ of the domain of definition Ω, which is the unit cube, is split into two parts. On the first part Γ_1 corresponding to $x = 0$, we impose homogeneous Dirichlet conditions. On the remaining part of the boundary $\Gamma_2 = \partial\Omega - \Gamma_1$, we assume homogeneous Neumann boundary conditions. The boundary value problem involves the Poisson equation

$$-\nabla^2 u = f, \quad \text{in } \Omega, \tag{3.87}$$

where the non-homogeneous term f is given by $f(x, y, z) = ze^x \sin(y)$. The cube in [145] is partitioned into 64 subcubes. Details about the mesh levels considered are listed in Table 3.47. We use the same type of partitions as the ones considered in [145]. In addition we test the performance of our preconditioners with METIS-generated partitions.

Table 3.48 lists the iterations needed for each case. The first column shows the number of iterations obtained by the 2-level iterative substructuring algorithm suggested in [145]. The Lanczos space dimension k for all our preconditioners was set to $k = 40$, but $k = 10, 20$ give similar results. The number of iterations of the best performing preconditioner (marked with blue) for the $1/2$ exponent and the 0.7 exponent that was found to be optimal for this test case, follows for both manually and METIS-generated partitions. Once again the 0.7 exponent outperforms the one suggested in theory. As before this is attributed to the fact that the H^1 regularity is too pessimistic for the solution of the boundary problem (3.87). However, we see that consideration of Neumann boundary conditions reduced the optimal exponent for the specific geometry, which was found to be 0.77 for the pure Dirichlet problem.

Consideration of the optimal exponent for this problem renders our single-level preconditioner more robust than the two-level approach suggested by Smith. In the case when the partitions have been generated by METIS, our algorithm is pronouncedly h-independent, outperforming by 11 iterations the iterative substructuring algorithm for the last level consisting of more than two million unknowns. Comparable performance is observed for sufficiently fine meshes from the $1/2$ class of preconditioners.

Manually generated meshes			
level	elements	nodes	$h_{\max}$
1	202323	34991	7.013e-02
2	1657357	271437	3.408e-02
3	3258917	528933	2.668e-02
4	5880586	945761	2.238e-02
5	14261180	2271852	1.650e-02

mesh levels used with METIS			
level	elements	nodes	$h_{\max}$
1	201316	34514	6.573e-02
2	1665255	271408	3.267e-02
3	3271340	527962	2.636e-02
4	5645964	904953	2.201e-02
5	13786443	2190085	1.685e-02

Table 3.47: Details about the manually and METIS-generated mesh levels for the cube partitioned into 64 subdomains. For each level we see the number of elements, number of nodes, maximum and minimum edge lengths.

64 subdomains	Smith's Parallel Iterative Substructuring 3D Algorithm				
n	34,848	270,400	524,880	903,264	2,130,048
Smith	17	23	25	28	32
	H^{θ} with Manual Partitioning to bricks				
n	34,991	271,437	528,933	945,761	2,271,852
n_b	10,690	43,882	70,958	102,150	182,992
$L^{1/2}$	32	31	33	34	32
$M(M^{-1}L)^{0.7}$	24	25	25	26	26
	H^{θ} with METIS Partitioning				
n	34,514	271,408	527,962	904,953	2,190,085
n_b	9,704	41,136	65,085	94,994	171,245
$L^{1/2}$	29	33	33	33	34
$L^{0.7}$	20	19	21	19	21

Table 3.48: H_{θ} versus Smith's iterative substructuring algorithm [145]. Smith partitions the unit cube into 64 subcubes. Iterations needed for reducing the relative residual of the Schur-complement system by six orders of magnitude. Both manual and METIS-generated partitions are considered for H_{θ}, and for each case we show the best performing preconditioner. The number of unknowns is denoted by n while n_b stands for the number of unknowns on the interface. The Lanczos dimension was set to $k = 40$.

3.14 Parallel performance

Domain decomposition methods have been designed from the ground up for parallel computing. Thus, it is really tempting to explore the degree of parallelism that can be achieved by the method we investigated in the previous sections. We restrict our parallel investigations to symmetric multiprocessor computing systems (SMP). The parallel code uses OpenMP directives mainly for portability reasons but also because several compiler vendors[2] have acknowledged the benefits of OpenMP and provide OpenMP implementations even for distributed systems (Beowulf clusters).

The degree of parallelism achieved in SMP systems is strongly dependent on the architectural design of CPU, the operating system, and the compiler itself. We will see, however, that it is also influenced by the optimized BLAS libraries the code links to. Our purpose here is to find the combination of platform, compiler and BLAS-LAPACK libraries that provides the highest scalability and also the most efficient code. For this reason, we have to consider a test case that is ideal for parallel computing and provides equally balanced workload among all the processors of the system. Without doubt, such a test case is the example we investigated in section 3.11.2, where all subdomains and all separators consist of equal number of unknowns. The block-diagonal structure of the interface operator is ideal for parallel evaluation of the desired matrix power. The preconditioner we used is the $M(M^{-1}L)^{0.7}$ with the mass matrix M replaced by its diagonal, which proved to be the most robust one in almost all the examples considered in sections 3.11 and 3.12. The domain was partitioned into several subdomains starting from 16 up to 2048 in powers of 2, and for each case we investigated the scalability and runtime performance of several components of the code. The code was compiled with all the compilers mentioned below and linked to optimized BLAS libraries that were available for each platform. The direct sparse solver we used for the subdomain matrices was CHOLMOD.

We have used the following platforms for our benchmark:

[2]Intel Cluster OpenMP: `http://www.intel.com/cd/software/products/asmo-na/eng/375500.htm`

Linux AMD64

BLAS Libraries

1. Intel MKL
2. AMD ACML

COMPILERS

1. g++-4.2.2
2. intel icpc
3. Sun CC

Sun Solaris

BLAS Libraries

1. AMD ACML

COMPILERS

1. Sun CC

AIX 5.3

BLAS Libraries

1. ESSL

COMPILERS

1. xlC_r

3.14.1 Linux

We begin our investigation on the Linux operating system, where more compilers and software libraries are available. Our computing platform is an SMP system with 8 Dual Core Opteron™ Processors 870 and 1MB L2 cache per core. We used a 64-bit Linux distribution with the kernel recompiled and tuned for the specific platform. The compilers and their compilation flags used follow:

GNU (GCC) 4.2.2
g++-4.2.2 -O3 -march=native -fopenmp

Intel (cce) 10.1.015
icpc -O3 -xW -openmp

CC: Sun Ceres 5.10 C++ Linux_i386 2008/07/10
CC -xtarget=native -xO5 -fast -m64 -xopenmp

The LAPACK routine used for the eigenvalue decomposition of the block diagonal boundary operator was DSYEVR, which is the fastest algorithm for symmetric eigenproblems when both eigenvalues and eigenvectors are desired. We should note, however, that the corresponding implementation of LAPACK[3] is not thread-safe, in contrast to the implementations provided by the MKL and ACML libraries. Thus, unpredictable results are observed if the DSYEVR routine provided by LAPACK is called by many threads simultaneously. Regarding the Sun performance library, it operates in parallel mode whenever the environmental variable **OMP_NUM_THREADS** is set to a number greater than one. This variable controls the number of threads created during the execution

[3]LAPACK: `http://www.netlib.org/lapack/`

of a parallel OpenMP program. When parallel OpenMP applications are linked to SUNPERF, the latter always operates in parallel mode, resulting in segmentation faults in most cases. This is why we did not consider it in our benchmarks.

In the figures that follow we show the total runtime and achieved speedup on 16 cores, for all combinations of compilers and optimized BLAS libraries used. We compare several phases of the total solution process.

At the left of Figure 3.12 we see the total runtime corresponding to the combined initialization, factorization and solving phases, while at the right we show the achieved speedup when the code runs on 16 cores. The initialization phase refers to the symbolic factorization performed by the direct sparse solvers for the matrices corresponding to the diagonal blocks A_{II}. The factorization phase refers to the total factorization (numerical factorization phase of direct sparse solvers corresponding to the diagonal blocks and preconditioner's eigenvalue decomposition) while the solution phase involves every other step of the solution process.

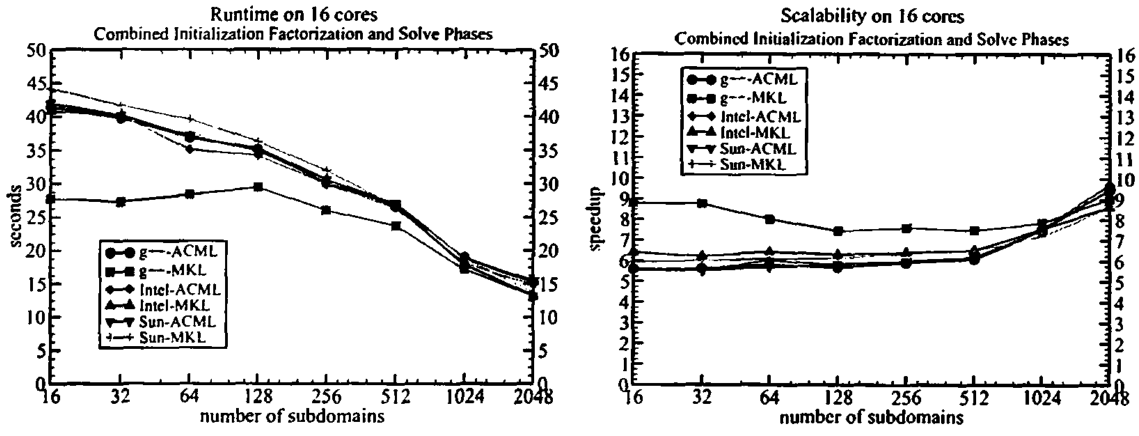


Figure 3.12: Runtime in seconds for the combined initialization, factorization, and solution phases (left) and their scalability (right) when the code runs on 16 cores.

Figure 3.13 shows the runtime and scalability observed for the combined factorization and solution phases, while Figure 3.14 concerns the solution phase only. It is evident that the highest performance and scalability are achieved when the code is compiled with g++ and linked with the Intel-MKL library. For this combination of compiler and BLAS we proceed by examining the scalability of each single phase of the solution algorithm.

In Figure 3.15 we see the scalability of the initialization phase (left) and the scalability of the numerical factorization phase (right) corresponding to the subdomain matrices A_{II}. We see that both phases scale well independently of the number of the partitions, with a speedup of 7 when the code runs on 8 cores. The speedup observed for the initialization phase on 16 cores is quite

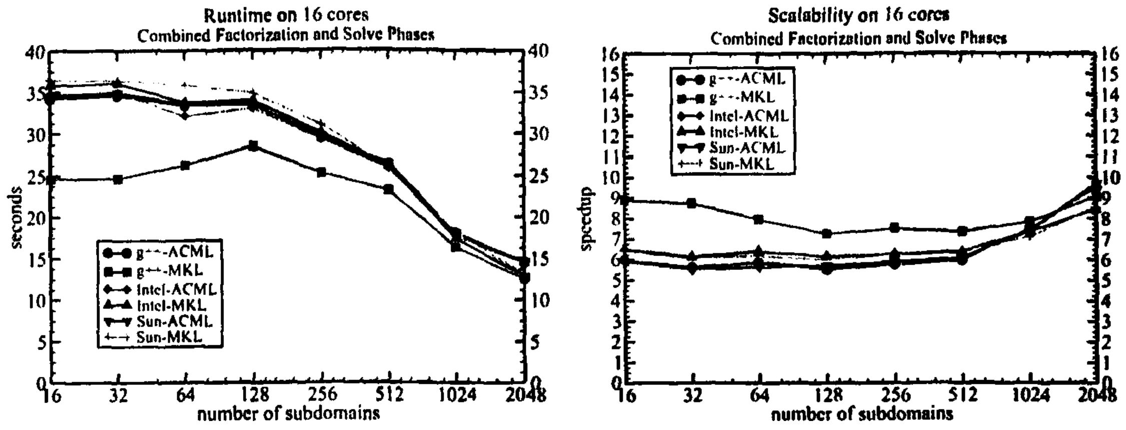


Figure 3.13: Runtime in seconds for the combined factorization and solution phases (left) and their scalability (right) when the code runs on 16 cores.

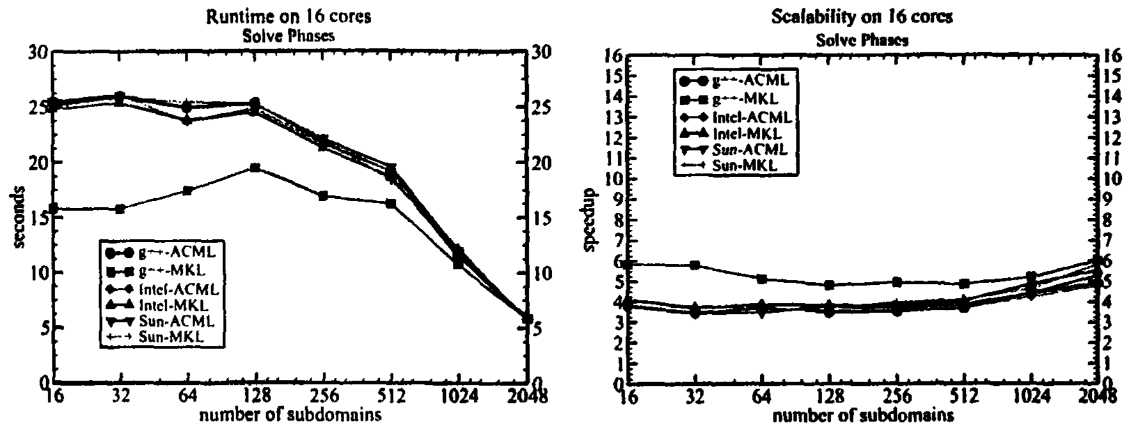


Figure 3.14: Runtime in seconds for the solution phase (left) and scalability of the solution phase (right) when the code runs on 16 cores.

irregular however. It strongly depends on the number of partitions with a maximum speedup close to 14 observed for 512 subdomains. On the other hand the factorization phase scales pretty well even when the code runs on 16 cores, showing for every subdomain a speedup between 13 and 14. We should note here that the irregular speedup of the initialization phase occurs for every single combination of compiler and BLAS library the code links to when it runs on 16 cores, in contrast to the factorization phase, which scales smoothly with respect to the number of partitions.

In Figure 3.16 we see the speedup of the solution phase corresponding to the backsubstitution of the direct sparse solvers for the block matrices A_{II}. Although the code runs on an AMD platform,

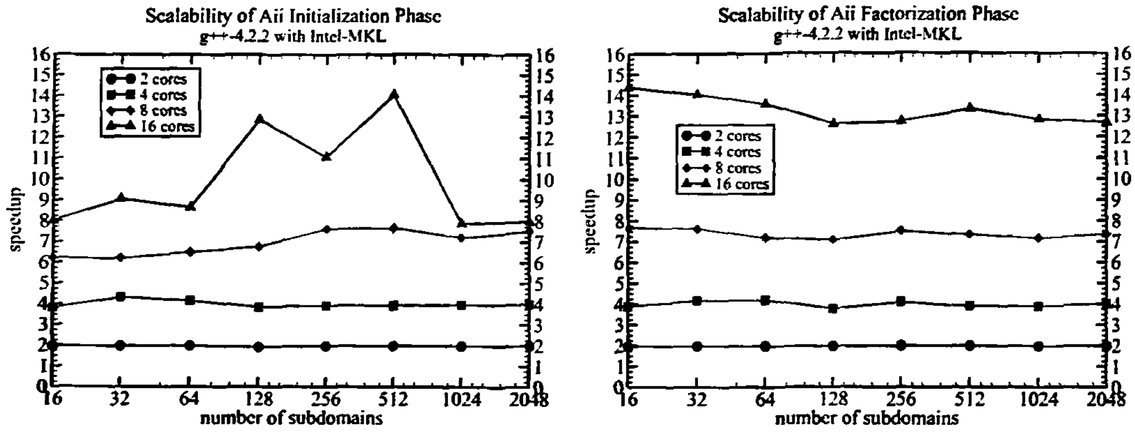


Figure 3.15: Scalability of the symbolic factorization (left) and numerical factorization phase (right) performed by the direct sparse solvers on the diagonal blocks A_{II}.

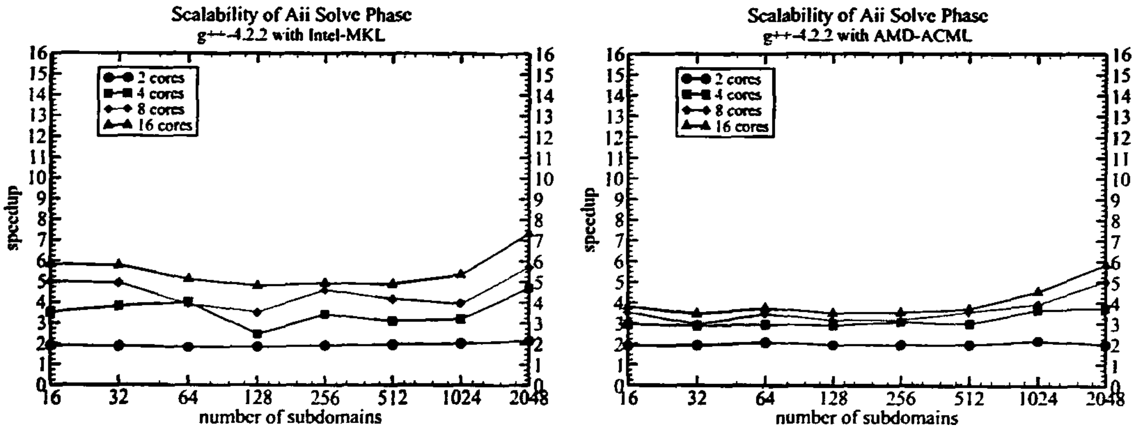


Figure 3.16: Scalability of solutions performed by the direct sparse solvers on the subdomain matrices A_{II}. Intel-MKL BLAS (left) and AMD-ACML BLAS (right).

the Intel-MKL BLAS (left) provides better scalability than the AMD-ACML BLAS (right). Similar speedups were also obtained for every other combination of compiler and BLAS. It is evident that the scalability of the solution phase is not what one expects. This is a well known problem believed to be due to bandwidth limitations of the $O(n^2)$ BLAS2 kernels used in the solution phases, as opposed to the $O(n^3)$ BLAS3 kernels used in the factorization phases. We discuss it more later. Finally in Figure 3.17 we see the speedup for the two main phases of the preconditioning. The factorization phase (corresponding to the block eigenvalue decomposition), which is the most time-consuming phase (left), scales much better because the complexity of the related algorithms is

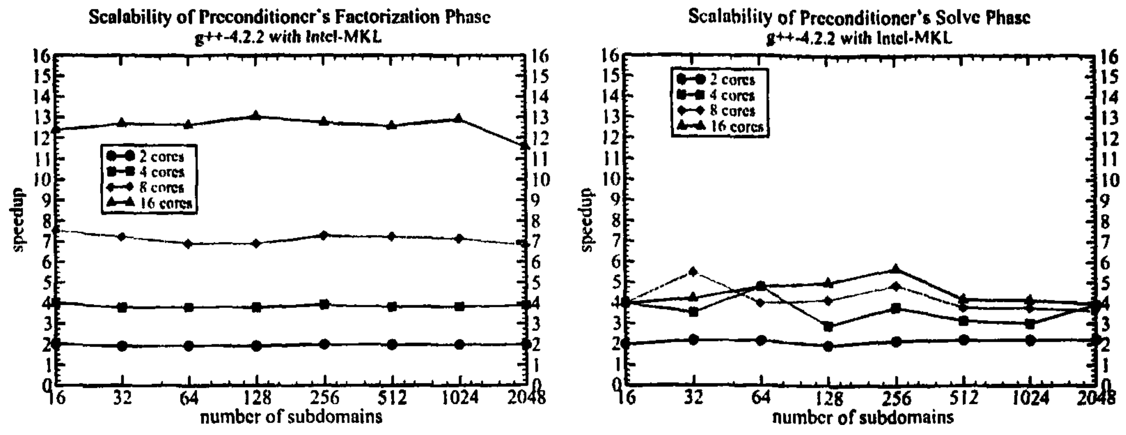


Figure 3.17: Scalability of the preconditioner's factorization phase, (left) and solve phase (right) when the code runs on 16 cores.

$O(n^3)$, whereas the complexity of the solution phase drops to $O(n^2)$ and the code hardly scales, with a maximum speedup of 6 on 16 cores.

3.14.2 Sun Solaris

The latest version of the Solaris Express Community Edition[4] was installed on the same SMP system that we benchmarked Linux. Here we have only one option of compilers and math libraries. The latest version of the Sun CC compiler was used. The code was compiled with the same flags as for the Linux case and linked to the AMD-ACML BLAS library. The Sun performance library (SUNPERF) was not an option here for the same reason we mentioned previously. Thus we perform a side by side comparison of the runtime and scalability under Solaris and Linux.

At the left of Figure 3.18 we see the total runtime for the initialization, factorization, and solution phases. The corresponding speedup is shown at the right. It is evident, especially on a single core but also on any number of cores, that the code runs faster under the Solaris operating system. As regards scalability, we observe slightly better speedup under Solaris, especially on 8 and 16 cores for any number of partitions. As the number of partitions increases, the differences seem to vanish. As we experienced before, solution phases do not scale well and the performance of the two operating systems is more or less identical.

Figure 3.19 shows runtime and speedups for the factorization and solution phases.

[4]`http://www.opensolaris.org/os/downloads/sol_ex_dvd_1/`

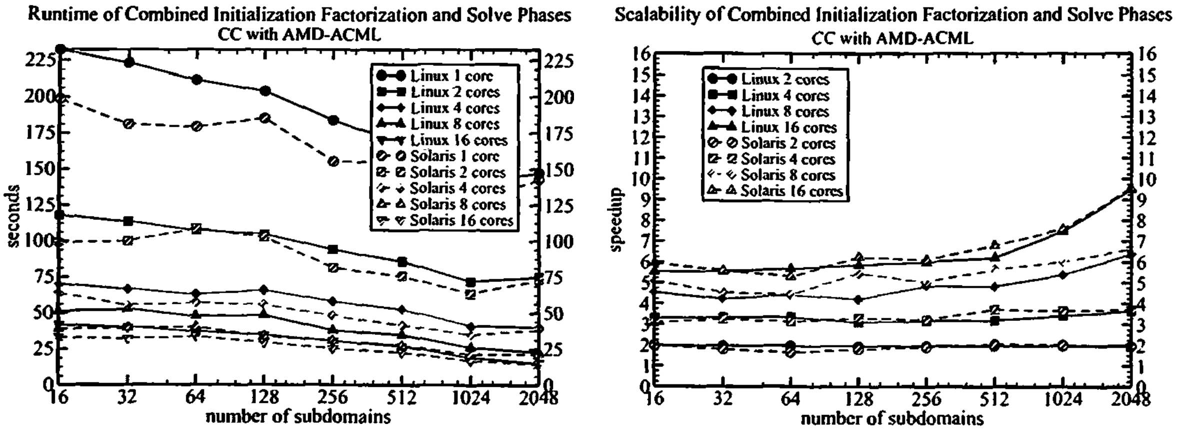


Figure 3.18: Total runtime in seconds for the combined initialization, factorization, and solution phases (left) and the speedup observed as a function of the number of subdomains (right), under both Linux and Solaris.

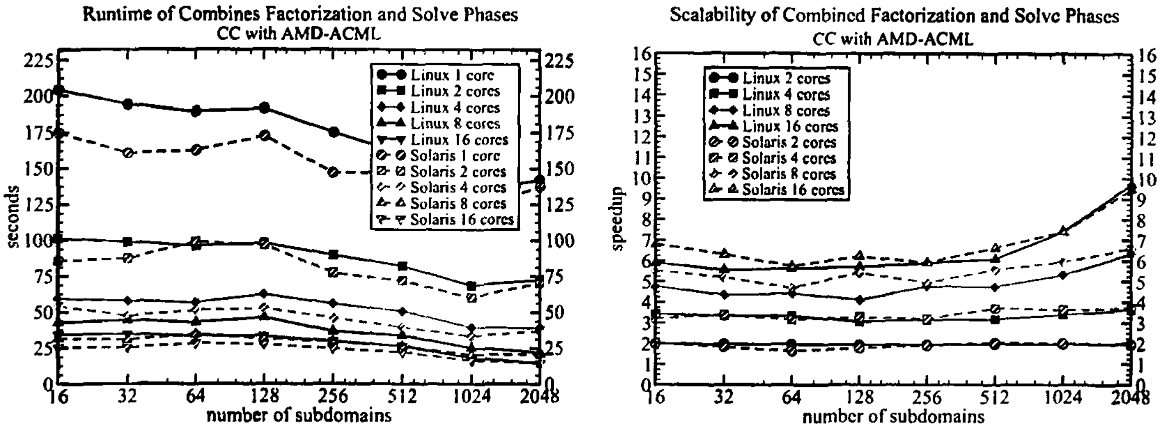


Figure 3.19: Total runtime in seconds for the factorization and solution phases (left) and the speedup observed as a function of the number of subdomains (right), under both Linux and Solaris.

3.14.3 AIX

The AIX operating system version 5.3 was run on a POWER PC consisting of 16 Power5 processors at 1900 MHz each. The code was compiled with the following IBM xlC_r C++ compiler and flags:

IBM(R) XL C/C++ Enterprise Edition V8.0 for AIX(R)

xlC_r -O3 -qstrict -qsmp=omp -q64 -qtune=auto

and linked to the IBM Engineering and Scientific Subroutine Library version (ESSL)[5]. The serial thread-safe version of ESSL was the only available BLAS[6].

Unfortunately ESSL does not provide an implementation of the LAPACK DSYEVR routine that we used earlier for the solution of symmetric eigenvalue problems. Since there is no version of Intel's MKL or AMD's ACML available for AIX and since using DSYEVR from LAPACK is not an option because of the thread-safety problems we described previously, we are forced to use LAPACK's DSYEV routine. The latter is the fastest routine for symmetric eigenvalue problems when only eigenvalues are desired but it is about two times slower than DSYEVR when both eigenvalues and eigenvectors are desired. The performance of the eigenvalue decomposition here should not be directly compared with that observed on other platforms.

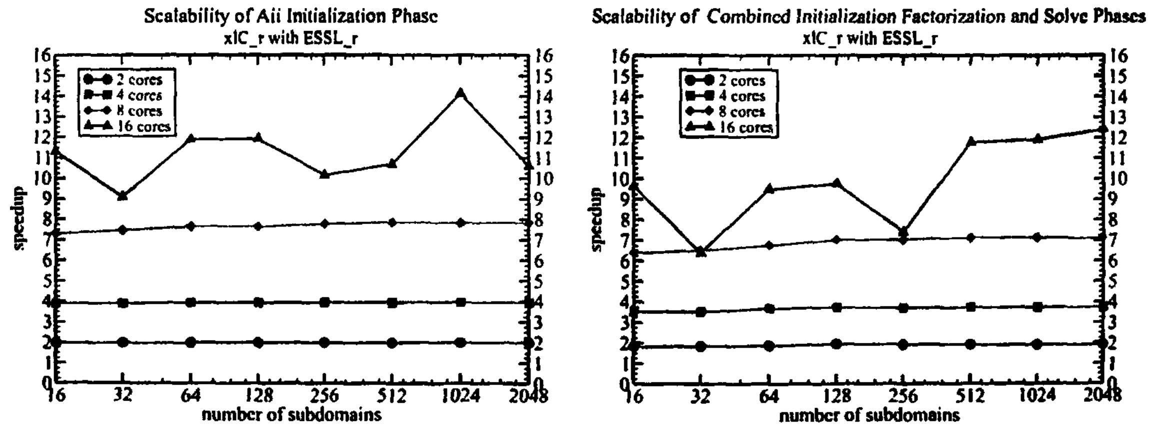
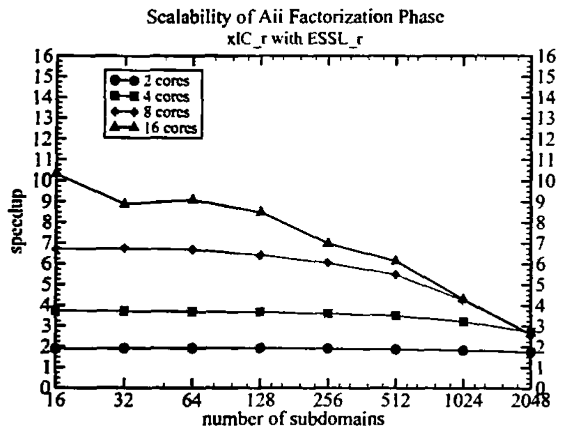


Figure 3.20: Scalability of symbolic factorization phase (left) and solve phase (right) of the direct sparse solvers on the subdomain matrices A_{II}.

In Figure 3.20 we see that the symbolic factorization (left) and solution (right) for the subdomain matrices A_{II} scale pretty well. A speedup of almost 8 for the initialization and around 7 for the solution phase are observed independently of the number of partitions when the code runs on 8

[5]ESSL version 4.2
[6]Thread-safe ESSL: the code is linked with libessl_r



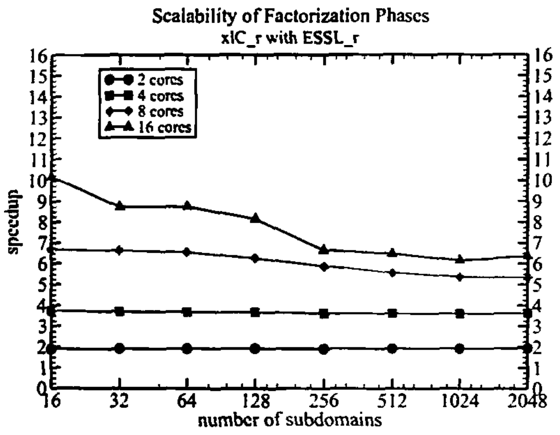


Figure 3.21: Scalability of numerical factorization for the subdomain matrices A_{II} (left) and total (combined A_{II} and preconditioner's) factorization phases (right).

cores. On 16 cores the speedup depends on the number of partitions. The situation however is quite different for factorization, as we can see in Figure 3.21. The speedup of the total factorization and the factorization of the subdomain matrices drops as the number of partitions increases, especially with 8 or 16 cores. This is in contrast to our experience on the Opteron platform under Solaris or Linux. Thus we realize that the classical argument "*BLAS2 algorithms do not scale well because of memory bandwidth limitations, while BLAS3 algorithms do*" is not quite true. Much depends on the CPU architecture and/or the BLAS implementation. However, we have no means of isolating the real problem because we do not have a BLAS implementation that is common to all architectures.

3.14.4 Linux vs Solaris vs AIX

We realize that scalability on different platforms is a very delicate issue influenced by several factors. The same parallel code compiled with the same compiler and linked to the same libraries does not show uniform scalability under different operating systems. The best combination of compiler and BLAS on the Linux platform was the **g++-4.2.2** compiler and the **Intel-MKL** BLAS. Solaris performs better than Linux with a specific combination of compiler and BLAS. On Linux, however, one has more options for compilers and optimized BLAS libraries, which counterbalances the advantage of Solaris. On the other hand, IBM AIX finds it very hard to parallelize the factorization phases (which scale quite well on Linux and Solaris), yet it provides quite good speedup in the solution phases (which do not scale well on Linux or Solaris). Whether this is a problem of the operating system or the hardware is not clear, though we believe it is related to hardware and not software components.

3.15 Results and extensions

The main contribution of this work is that it achieved what was considered to be impossible: a single-level domain decomposition approach for the Poisson and reaction-diffusion problems insensitive to the size of the system and the number of subdomains. However, independence of the preconditioner's performance on the number of subdomains is achieved with fractional powers greater than 1/2. Although this result is not yet explained theoretically, it has been verified numerically in a number of real-life applications and we believe that it is due to higher regularity of the solution of the benchmarked problems than the H^1 regularity guaranteed by theory for the general case.

Our numerical experiments indicate that there is still room for improvement, even without embracing the multilevel concept. We highlight a number of issues where further improvements are desirable and will potentially boost further the performance and robustness of our preconditioner.

Evaluation of H_θ A major challenge is the efficient and yet scalable evaluation of matrix real powers. The Lanczos approach we described in section 3.9.4 is very robust, especially when it uses the inverse of the matrix. However, for hard cases like the brain problem or the crystal problem 3.12.1, a large number of Lanczos vectors ($k \approx 40$) may be needed to ensure h-independence. A possible alternative is the method suggested in [74]. However the scalability, as in the Lanczos case, depends on the scalability of the corresponding direct sparse linear solver. A more scalable approach would take into account the special structure of the interface boundary, resulting in a sufficiently sparse matrix having an almost block-triangular structure. Then block versions of the symmetric tridiagonal eigenvalue decomposition suggested by Dhillon [43] may be what we are looking for. The extra efficiency will come from the optimal bandwidth utilization of the $O(n^3)$ algorithms associated with block-matrix operations. Apart from the cache efficiency we will definitely experience sufficiently high scalability, at least for Linux-Opteron platforms, which are quite popular these days.

Extensions of our preconditioning approach to more complicated PDE systems are also quite tempting.

Stokes flows The extension of our approach to Stokes equations is straightforward. Assuming a Taylor-Hood approximation of the velocity and pressure spaces in order to satisfy the LBB condition, the preconditioner will have the following block-diagonal structure:

$$\begin{pmatrix} L^\theta & 0 \\ 0 & M^\theta \end{pmatrix},$$

where L is the discrete $P2$ Laplacian (quadratic triangles/tetrahedra) assembled on the boundary and M is the $P1$ (linear triangles/tetrahedra) discretization of the identity operator (mass matrix) assembled on the boundary, while θ will be the optimal fractional power we are familiar with.

Convection-Diffusion problems Extension of our approach to convection-diffusion problems is not so straightforward. Preliminary results in 2D with the $H_{1/2}$ preconditioner show h-independence and a mild sensitivity to the number of partitions, which nevertheless grows quickly as the Peclet number increases. This is expected because our preconditioner is symmetric and we do not expect it to be able to precondition well an increasingly nonsymmetric operator, like the convection-diffusion one.

Navier-Stokes flows Extension of our approach to Navier-Stokes equations will definitely be based on the preconditioner for Stokes but it will also be subject to the same difficulties as convection-diffusion PDEs. However the problem is nonlinear and that allows us to take advantage of adaptive preconditioning schemes suggested by Loghin et al. [108], which have been proved quite effective as they reduce the total number of iterations throughout the solution of a nonlinear system by a factor of four. Any successful extension of the theory to convection-diffusion problems directly applies here as well.

Structural mechanics Applications encountered in structural mechanics involve symmetric operators because of the symmetry of the stress tensor. Although the theory of H_θ has not yet been extended to this class of problems, we believe that some fractional power of the differential operator assembled on the boundary will work here as well.

Multilevel It will be really interesting to see how much the performance of our preconditioner is improved by adopting the multilevel approach. The latter can be applied in many ways. Apart from the classical additive approach providing global communication, we could for instance use the coarse level for preconditioning the unknowns on the wirebasket, decoupling completely the application of the matrix power. This will allow a block-diagonal eigenvalue decomposition for the remaining unknowns on the decoupled faces, and this can be completely parallelized.

Parallel scalability As regards the scalability of our code, there is still room for improvement. We experienced higher scalability for the factorization and initialization phases than for the solution phase on Linux platforms. On AIX we have exactly the opposite behavior. Apart from more extensive tests that need to be done, as we described in section 3.14 there are several tools and libraries we could use to improve parallel performance. These tools **numactl**

and **libnuma** under Linux[7] and **pbind** and **liblgrp** under Solaris[8] control the sheduling of our code, process, thread and memory affinity. However, such a detailed investigation for pushing the scalability a bit further would make sense only after the original algorithm incorporates all the enhancements we discussed above.

Minimizing total work Finally, an algorithm must be introduced to allow the preconditioner to find the optimal number of partitions. Optimality here refers to the runtime of the solution, factorization, and solution phases, or all three phases including initialization. In memory-limited situations, tuning the number of partitions so that the memory consumption is minimized should also be an option.

This process involves describing in terms of the number of subdomains and the system size the complexity of the desired objective. Then for a given number of unknowns one has to solve a nonlinear algebraic equation to find the optimal number of partitions. An intermediate step in this direction is to be able to describe the growth of the number of unknowns on the interface of the subdomains as a function of the system size and the number of partitions. However, this is not possible for an arbitrary-shaped domain. One has to make crude assumptions or apply a partitioner on the mesh for several subdomains and describe by nonlinear regression analysis the growth of the boundary unknowns.

Moreover, if the Lanczos approach with the inverse matrix is employed for matrix-power evaluation, we need to be able to tell, for a given h and number of boundary unknowns, how the number of iterations for solving the system up to a certain threshold grows as the Lanczos k decreases. Clearly a larger k results in fewer iterations, while a smaller k results in a less accurate approximation of the desired matrix power but at the same time more efficient application of the preconditioner. For this reason, more robust and efficient evaluation methods of matrix powers are desirable and the block-diagonal approach combined with a coarse level on the wirebasket or a block-tridiagonal eigenvalue decomposition are methods that have to be taken seriously into account. Apart from their design, which favors parallel implementations, their complexity can be explicitly described and optimization becomes easier.

For these reasons we did not attempt something like that, because our results would be carried out for an ideal problem as in section 3.11.2 (for which such an analysis is straightforward), or would be based on crude assumptions regarding the Lanczos k.

[7]libnuma project: `http://oss.sgi.com/projects/libnuma/`
[8]Solaris Numa: `http://opensolaris.org/os/community/performance/numa/`

Bibliography

[1] S. Agmon, A. Douglis, and L. Nirenberg. Estimates near the boundary for solutions of elliptic partial differential equations satisfying general boundary conditions. I. *Comm. Pure Appl. Math.*, 12:623–727, 1959.

[2] E. Allen, J. Baglama, and S. Boyd. Numerical approximation of the product of the square root of a matrix with a vector. *Linear Algebra and its Applications*, 310:167–181, 2000.

[3] M. Arioli and I. S. Duff. Using FGMRES to obtain backward stability in mixed precision. Technical report, Rutherford Appleton Laboratory, 2008.

[4] M. Arioli and D. Loghin. Discrete interpolation norms with applications. Technical Report Technical Report RAL-TR-2008-012, Rutherford Appleton Laboratory, Didcot, UK, 2008.

[5] M. Arioli, D. Loghin, and A. J. Wathen. Stopping criteria for iterations in finite element methods. *Numer. Math.*, 99(3):381–410, 2005.

[6] Mario Arioli. A stopping criterion for the conjugate gradient algorithm in the finite element method framework. *Numerische Mathematik*, 97(1):1–24, 2004.

[7] Walter E. Arnoldi. The principal of minimized iteration in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, 9:17–29, 1951.

[8] K. A. Awada, D. R. Jackson, S. B. Baumann, J. T. Williams, D. R. Wilton, S. B. Baumann, and A. C. Papanicolaou. Computational aspects of finite element modeling in EEG source localization. *IEEE Trans. Biomedical Engineering*, 44(8):736–752, 1997.

[9] K. A. Awada, D. R. Jackson, S. B. Baumann, J. T. Williams, D. R. Wilton, P. W. Fink, and B. R. Prasky. Effect of conductivity uncertainties and modeling errors on EEG source localization using a 2-D model. *IEEE Trans. Biomedical Engineering*, 45(9):1135–1145, 1998.

[10] K. A. Awada, D. R. Jackson, S. B. Baumann, D. R. Wilton, and J. T. Williams. Closed-form evaluation of flux integrals appearing in a finite element solution of the 3D poisson equation with dipole sources. *Electromagnetics*, 20(3):167–185, 2000.

[11] Owe Axelsson. *Iterative Solution Methods*. Cambridge University Press, Cambridge and New York, 1994.

[12] D. H. Bailey. High-precision floating-point arithmetic in scientific computation. *Lawrence Berkeley National Laboratory. Paper LBNL-57487*, 2004.

[13] D. H. Bailey, Hida Yozo, Xiaoye S. Li, and Brandon Thompson. ARPREC: An arbitrary precision computation package. *Lawrence Berkeley National Laboratory. Paper LBNL-53651.*, 2002. `http://repositories.cdlib.org/lbnl/LBNL-53651.`

[14] Richard Barrett, Michael W. Berry, Tony F. Chan, James Demmel, June Donato, Jack Dongarra, Victor Eijkhout, Roldan Pozo, Charles Romine, and Henk A. van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, 1993.

[15] W. W. Bell. *Special functions for scientists and engineers*. D. Van Nostrand Company Ltd, 1967.

[16] Petter E. Bjørstad and Anders Hvidsten. Iterative methods for substructured elasticity problems in structural analysis. In Ronald Glowinski, Gene H. Golub, Gérard A. Meurant, and Jacques Périaux, editors, *Domain Decomposition Methods for Partial Differential Equations*, Philadelphia, PA, 1988.

[17] Petter E. Bjørstad, Randi Moe, and Morten Skogen. Parallel domain decomposition and iterative refinement algorithms. In Wolfgang Hackbusch, editor, *Parallel Algorithms for PDEs, Proceedings of the 6th GAMM-Seminar held in Kiel, Germany, January 19–21, 1990*, 1990.

[18] Petter E. Bjørstad and Olof B. Widlund. Iterative methods for the solution of elliptic problems on regions partitioned into substructures. *SIAM J. Numer. Anal.*, 23(6):1093–1120, 1986.

[19] Matthias Bollhöfer. A robust ilu based on monitoring the growth of the inverse factors. *Linear Algebra Appl.*, 338(1-3):201–218, 2001.

[20] Matthias Bollhöfer. On the impact of inverse-based dropping on ilus derived from direct methods. *SIAM J. Sci. Comput.*, 25(1):86–103, 2003.

[21] Matthias Bollhöfer and Youcef Saad. Ilus and factorized approximate inverses are strongly related: Part ii: Applications to stabilization. *SIAM J. Matrix Anal. Appl.*, 23(3):692–705, 2001.

[22] Matthias Bollhöfer and Youcef Saad. On the relations between ilus and factored approximate inverses. *SIAM J. Matrix Anal. Appl.*, 24(1):219–237, 2002.

[23] J. P. Boris and D. L. Book. Flux-corrected transport. I. SHASTA, A fluid transport algorithm that works. *J. Computational Physics*, 11:38–69, 1973.

[24] J. Borwein and D. Bailey. *Mathematics by Experiment: Plausible Reasoning in the 21st Century*. A. K. Peters, 2004.

[25] A. Borzì and V. Schulz. Multigrid methods for PDE optimization. *SIAM Review*, to appear.

[26] J. F. Bourgat, Roland Glowinski, Patrick Le Tallec, and Marina Vidrascu. Variational formulation and lagorithm for trace operator in domain decomposition calculations. In Tony F. Chan, Ronald Glowinski, Jacques Périaux, and Olof B. Widlund, editors, *Domain Decomposition Methods*, Philadelphia, PA, 1989. SIAM.

[27] James H. Bramble, Joseph E. Pasciak, Junping Wang, and Jinchao Xu. Convergence estimates for product iterative methods with applications to domain decomposition. *Math. Comp.*, 57(195):1–21, 1991.

[28] C. Cabos. Evaluation of matrix functions with the block lanczos algorithm. *Comput. Math. Appl.*, 33(1-2):45–57, 1997.

[29] X. C. Cai, W. D. Gropp, D. E. Keyes, and M. D. Tidriri. Newton-Krylov-Schwarz methods in CFD. *Proceedings of the International Workshop on the Navier-Stokes Equations, Notes in Numerical Fluid Mechanics*, 1994. Vieweg.

[30] Xiao-Chuan Cai. The use of pointwise interpolation in domain decomposition methods. *SIAM J. Sci. Comput.*, 16:250–256, 1995.

[31] P. J. Capon and P. K. Jimack. An inexact Newton method for systems arising from the finite element method. *Appl. Math. Lett.*, 10(3):9–12, 1997.

[32] Tony F. Chan and Barry Smith. Multigrid and domain decomposition methods for unstructured meshes. In *Proceedings of the Third International Conference on Advances in Numerical Methods and Applications*, pages 53–62. World Scientific, 1994.

[33] A. Charalambopoulos and G. Dassios. On the Vekua pair in spheroidal geometry and its role in solving boundary value problems. *Applicable Analysis*, 81:85–113, 2002.

[34] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam. Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. *ACM Trans. Math. Softw.*, 35(3), 2009.

[35] S. Turek D. Kuzmin, R. Löhner. *Flux-Corrected Transport: Principles, Algorithms, and Applications*. Springer, 2005.

[36] T. A. Davis and W. W. Hager. Dynamic supernodes in sparse cholesky update/downdate and triangular solves. *ACM Trans. Math. Softw.*, 35(4), 2009.

[37] Timothy A. Davis. Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.*, 30(2):196–199, 2004.

[38] Timothy A. Davis. Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.*, 30(2):196–199, 2004.

[39] Timothy A. Davis. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.*, 30(2):165–195, 2004.

[40] Timothy A. Davis and Iain S. Duff. An unsymmetric-pattern multifrontal method for sparse LU factorization. *SIAM J. on Matrix Anal. Appl.*, 18(1):140–158, 1997.

[41] Timothy A. Davis and Iain S. Duff. A combined unifrontal/multifrontal method for unsymmetric sparse matrices. *ACM Trans. Math. Softw.*, 25(1):1–20, 1999.

[42] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM J. Numer. Anal.*, 19:400–408, 1982.

[43] Inderjit S. Dhillon and Beresford N. Parlett. Multiple representations to compute orthogonal eigenvectors of symmetric tridiagonal matrices. *Linear Algebra and its Applications*, 387:1–28, 2004.

[44] V. Druskin, A. Greenbaum, and L. Knizhnerman. Using nonorthogonal lanczos vectors in the computation of matrix functions. *SIAM J. Sci. Comput.*, 19(1):38–54, 1998.

[45] V. Druskin and L. Knizhnerman. Two polynomial methods of calculating functions of symmetric matrices. *U.S.S.R. Comput. Maths. Math. Phys.*, 29(6):112–121, 1989.

[46] V. Druskin and L. Knizhnerman. Extended Krylov subspaces: approximations of the matrix square root and related functions. *SIAM J. Matrix Anal. Appl.*, 19(3):755–771, 1998.

[47] Maksymilian Dryja and Olof B. Widlund. An additive variant of the Schwarz alternating method for the case of many subregions. *Technical Report*, 339, 1987.

[48] Maksymilian Dryja and Olof B. Widlund. On the optimality of an additive iterative refinement method. In *Fourth Copper Mountain Conference on Multigrid Methods*, pages 161–170, Philadelphia, PA, 1989. SIAM.

[49] Maksymilian Dryja and Olof B. Widlund. Multilevel additive methods for elliptic finite element problems. In *Parallel Algorithms for Partial Differential Equations, Proceedings of the Sixth GAMM-Seminar*, pages 58–69, 1990.

[50] D. Echeverría. *Multi-Level Optimization: Space Mapping and Manifold Mapping.* PhD thesis, University of Amsterdam, Amsterdam, The Netherlands, 2007.

[51] D. Echeverría and P. W. Hemker. Space mapping and defect correction. *Comp. Methods in Appl. Math.*, 5(2):107–136, 2005.

[52] M. Eiermann and O. G. Ernst. A restarted Krylov subspace method for the evaluation of matrix functions. *SIAM J. Numer. Anal.*, 44(6):2481–2504, 2006.

[53] Michael Eiermann and Oliver G. Ernst. Geometric aspects in the theory of Krylov subspace methods. *Acta Numerica*, 10:251–312, 2001.

[54] Timo Eirola and Olavi Nevanlinna. Accelerating with rank-one updates. *J. Computational and Applied Mathematics*, 123:261–292, 2000.

[55] S. C. Eisenstat and H. F. Walker. Choosing the forcing term in an inexact Newton method. *SIAM J. Sci. Comput.*, 17:16–32, 1996.

[56] H. C. Elman. *Iterative Methods for Large Sparse Non-symmetric Systems of Linear Equations.* PhD thesis, Yale University, New Haven, 1982.

[57] Howard C. Elman, Oliver G. Ernst, and Dianne P. O'Leary. A multigrid method enhanced by Krylov subspace iteration for discrete Helmholtz equations. *SIAM J. on Scientific Computing*, 23:1290–1314, 2001.

[58] R. D. Falgout, J. E. Jones, and U. M. Yang. The design and implementation of *hypre*, a library of parallel high performance preconditioners. In A. M. Bruaset, P. Bjørstad, and

A. Tveito, editors, *Numerical Solution of Partial Differential Equations on Parallel Computers*. Springer-Verlag, 2006.

[59] Peter Feldmann, Ronald W. Freund, and Emrah Acar. Power grid analysis using a flexible conjugate gradient algorithm with sparsification. *Unpublished manuscript*, 2006.

[60] L. Formaggia, A. Moura, and F. Nobile. On the stability of the coupling of 3D and 1D fluid-structure interaction models for blood flow simulations. *Math. Mod. Numer. Anal.*, 41(4):743–769, 2007.

[61] L. Formaggia, F. Nobile, A. Quarteroni, and A. Veneziani. Multiscale modelling of the circulatory system: a preliminary analysis. *Comput. Visual. Sci 2*, 2(3):75–83, 1999.

[62] Y. C. Fung. *Biomechanics: Mechanical Properties of Living Tissues*. Springer, 2nd edition, June 1993.

[63] Y. C. Fung. *Biomechanics: Circulation*. Springer, 2nd edition, November 1996.

[64] Y. C. Fung. *Biomechanics: Motion, Flow, Stress, and Growth*. Springer, 2nd edition, October 1998.

[65] Klaus Gärtner. Existence of bounded discrete steady state solutions of the van roosbroeck system on boundary conforming delaunay grids. *SIAM SISC*, 2008. accepted.

[66] L. N. Gergidis, D. Kourounis, S. Mavratzas, and A. Charalambopoulos. Acoustic Scattering in Prolate Spheroidal Geometry via Vekua Tranformation - Theory and Numerical results. *CMES: Computer Modeling in Engineering & Sciences*, 21:157–175, 2007.

[67] Gene H. Golub and Mark D. Kent. Estimates of eigenvalues for iterative methods. *Mathematics of Computation*, 53(188):619–626, 1989.

[68] Gene H. Golub and Charles Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 3rd edition, 1996.

[69] Gene H. Golub and Qiang Ye. Inexact preconditioned conjugate gradient method with inner-outer iteration. *SIAM J. on Scientific Computing*, 21:1305–1320, 2001.

[70] S. Gratton, A. Sartenaer, and Ph. L. Toint. Recursive trust-region methods for multiscale nonlinear optimization. *SIAM J. on Optimization*, 19(1):414–444, 2008.

[71] M. Griebel and P. Oswald. Remarks on the abstract theory of additive and multiplicative Schwarz algorithms. *Numer. Math*, 70:163–180, 1995.

[72] William D. Gropp and Barry F. Smith. Experiences with domain decomposition in three dimensions: Overlapping Schwarz methods. In *Domain Decomposition Methods in Science and Engineering*, pages 323–333. AMS, 1992.

[73] R. Hackman. The transition matrix for acoustic and elastic wave scattering in prolate spheroidal coordinates. *J. Acoust. Soc. Am.*, 75:35–45, 1984.

[74] N. Hale, N. J. Higham, and L. N. Trefethen. Computing $a^{\alpha}, \log(a)$ and related matrix functions by contour integrals. *J. Numer. Anal.*, 46(5):2505–2523, 2008.

[75] Y. S. He, Z. K. Xie, and Q. Y. Ye. Radiation and scattering of sound waves by a screened prolate spheroid. *Wave motion*, 26:85–96, 1997.

[76] Van Emden Henson and Ulrike Meier Yang. BoomerAMG: a parallel algebraic multigrid solver and preconditioner. *Appl. Numer. Math.*, 41(1):155–177, 2002.

[77] Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Natl. Bur. Stand*, 49:409–436, 1952.

[78] N. J. Higham. *Functions of Matrices: Theory and Computation.* SIAM, Philadelphia, USA, 2008.

[79] R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, 1986.

[80] J. Hron and S. Turek. A monolithic multigrid FEM solver for FSI, June 2005. FSW Workshop, Erlangen.

[81] A. Jameson. Computational algorithms for aerodynamic analysis and design. *Applied Numerical Mathematics*, 13:383–422, 1993.

[82] A. Jameson. Positive schemes and shock modelling for compressible flows. *International J. for Numerical Methods in Fluids*, 20:743–776, 1995.

[83] T. Jongen and Y. P. Marx. Design of an unconditionally stable, positive scheme for the $K - \varepsilon$ and two-layer turbulence models. *Computers and Fluids*, 26(5):469–487, 1997.

[84] Eugenius S. B. C. Ang Jr, Vicko Gluncic, Alvaro Duque, Mark E Schafer, and Pasko Rakic. Prenatal exposure to ultrasound waves impacts neuronal migration in mice. *Proc. Natl. Acad. Sci.*, 103(34):12903–12910, 2006.

[85] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1998.

[86] George Karypis and Vipin Kumar. Parallel multilevel k-way partitioning scheme for irregular graphs. *SIAM Review*, 41(2):278–300, 1999.

[87] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. SIAM, Philadelphia, 1995.

[88] H. Kieler, S. Cnattingius, J. Palmgren B. Haglund, and O. Axelsson. Ultrasound screening in pregnancy: a systematic review of the clinical effectiveness, cost-effectiveness and women's views. *Health technology assessment*, 4(16:i-vi):1–193, 2000.

[89] H. Kieler, S. Cnattingius, J. Palmgren B. Haglund, and O. Axelsson. Sinistrality–a side-effect of prenatal sonography: a comparative study of young men. *Epidemiology*, 12(6):618–623, 2001.

[90] D. A. Knoll and D. A. Keyes. Jacobian-free Newton-Krylov methods: A survey of approaches and applications. *J. Computational Physics*, 193:357–397, 2004.

[91] Milos Kojic, Nenad Filipovic, Boban Stojanovic, and Nikola Kojic. *Computer Modeling in Bioengineering: Theoretical Background, Examples and Software*. Wiley, 1st edition, June 2008.

[92] E. Konukoglu, X. Pennec, O. Clatz, and N. Ayache. Tumor growth modeling in oncological image analysis. In Isaac Bankman, editor, *Handbook of Medical Image Processing and Analysis 2*. Elsevier, to be published.

[93] E. Konukoglu, M. Sermesant, O. Clatz, J.-M. Peyrat, H. Delingette, and N. Ayache. A recursive anisotropic fast marching approach to reaction diffusion equation: Application to tumor growth modeling. In *IPMI*, Kerkrade, Netherlands, 2007.

[94] D. Kuzmin. Positive finite element schemes based on the flux-corrected transport procedure. *Computational Fluid and Solid Mechanics*, pages 887–888, 2001.

[95] D. Kuzmin. On the design of general-purpose flux limiters for implicit FEM with a consistent mass matrix. *SIAM J. Numerical Analysis*, 219:513–531, 2006.

[96] D. Kuzmin and M. Möller. Algebraic fux correction i. scalar conservation laws. *Flux-Corrected Transport: Principles, Algorithms, and Applications*, pages 155–206, 2005.

[97] D. Kuzmin, M. Möller, and S. Turek. Multidimensional FEM-FCT schemes for arbitrary time-stepping. *International J. for Numerical Methods in Fluids*, 42:265–295, 2003.

[98] D. Kuzmin, M. Möller, and S. Turek. FEM-FCT schemes for multidimensional conservation laws. *Computer Methods in Applied Mechanics and Engineering*, 193:4915–4946, 2004.

[99] D. Kuzmin and S. Turek. Flux correction tools for finite elements. *J. Computational Physics*, 175:525–558, 2002.

[100] P. Lancaster and M. Tismenetsky. *The Theory of Matrices*. Academic Press, New York, 2nd edition, 1985.

[101] Cornelius Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Natl. Bur. Stand*, 45:255–282, 1950.

[102] Cornelius Lanczos. Solution of linear equations by minimized iterations. *J. Research of the National Bureau of Standards*, 49:33–53, 1952.

[103] R. J. LeVeque. High-resolution conservative algorithms for advection in incompressible flow. *SIAM J. Numerical Analysis*, 33:627–665, 1996.

[104] J. L. Lions and E. Magenes. *Problêmes aux limites non homogènes et applications*. Dunod, Paris, 1968.

[105] Pierre Louis Lions. Interprétation stochastique de la méthode alternée de Schwarz. *C. R. Acad. Sci. Paris*, 268:325–328, 1978.

[106] Pierre Louis Lions. On the Schwarz alternating method I. In Ronald Glowinski, Gene H. Golub, Gérard A. Meurant, and Jacques Périaux, editors, *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 1–42, Philadelphia, 1988. SIAM.

[107] Pierre Louis Lions. On the Schwarz alternating method II. In Tony F. Chan, Ronald Glowinski, Jacques Périaux, and Olof B. Widlund, editors, *Domain Decomposition Methods*, pages 47–70, Philadelphia, 1989. SIAM.

[108] D. Loghin, D. Ruiz, and A. Touhami. Adaptive preconditioners for nonlinear systems of equations. *J. Computational and Applied Mathematics*, 189(1):362–374, 2006.

[109] D. Loghin and A. J. Wathen. Analysis of preconditioners for saddle-point problems. *SIAM J. Sci. Comp.*, 25(6):2029–2049, 2004.

[110] R. Löhner and J. D. Baum. 30 years of FCT: Status and directions. *Flux-Corrected Transport: Principles, Algorithms, and Applications*, pages 251–296, 2005.

[111] R. Löhner, K. Morgan, J. Peraire, and M. Vahdati. Finite element flux-corrected transport (FEM-FCT) for the Euler and Navier-Stokes equations. *International J. for Numerical Methods in Fluids*, 7:1093–1109, 1987.

[112] Jan Mandel. Balancing domain decomposition. *Communications in Numerical Methods in Engineering*, 9(3), 1993.

[113] J. A. Meijerink and H. A. van der Vorst. Iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Math. Comput.*, 31:148–162, 1997.

[114] M. Möller. Efficient solution techniques for implicit finite element schemes with flux limiters. *International J. for Numerical Methods in Fluids*, 55(7):611–635, 2007.

[115] S. Nash. A multigrid approach to discretized optimization problems. *J. Optimization Methods and Software*, 14:99–116, 2000.

[116] E. J. Nielsen, W. K. Anderson, R. W. Walters, and D. E. Keyes. Application of Newton-Krylov methodology to a three-dimensional unstructured Euler code. *AIAA*, 1995.

[117] Yvan Notay. Flexible conjugate gradient. *SIAM J. on Scientific Computing*, 22:1444–1460, 2000.

[118] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12:617–629, 1975.

[119] C. C. Paige and M. A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Softw.*, 8(1):43–71, 1982.

[120] C. C. Paige and M. A. Saunders. Algorithm 583; LSQR: Sparse linear equations and least-squares problems. *ACM Trans. Math. Softw.*, 8(2):195–209, 1982.

[121] B. N. Parlett. The Symmetric Eigenvalue Problem. *Classics in Applied Mathematics*, 20, 1998.

[122] P. Peisker. On the numerical solution of the first biharmonic equation. *Mathematical Modelling and Numerical Analysis*, 22(4):655–676, 1988.

[123] M. Pernice and H. F. Walker. NITSOL: A Newton iterative solver for nonlinear systems. *SIAM J. Sci. Comput.*, 19:302–318, 1998.

[124] Chris Plummer, A. Simon Harvey, and Mark Cook. EEG source localization in focal epilepsy: Where are we now? *Epilepsia*, 49(2):201–218, 2008.

[125] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C++*. Cambridge University Press, 2nd edition, 2002.

[126] Vasilios C. Protopappas, Dimitrios I. Fotiadis, and Konstantinos N. Malizos. Guided ultrasound wave propagation in intact and healing long bones. *Ultrasound in Med. & Biol.*, 32(5):693–708, 2006.

[127] Vasilios C. Protopappas, Iraklis C. Kourtis, Lampros C. Kourtis, Konstantinos N. Malizos, Christos V. Massalas, and Dimitrios I. Fotiadis. Three-dimensional finite element modeling of guided ultrasound wave propagation in intact and healing long bones. *J. Acoust. Soc. Am.*, 121(6):3907–3921, 2006.

[128] A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Numerical Mathematics and Scientific Computation. Oxford University Press, 1999.

[129] F. Riesz and B. Sz-Nagy. *Functional Analysis*. Blackie and Son Limited, 1956.

[130] Y. Saad. Analysis of some Krylov subspace approximations to the matrix exponential operator. *SIAM J. Numer. Anal.*, 29(1):209–228, 1992.

[131] Yousef Saad. A flexible inner-outer preconditioned GMRES. *SIAM J. on Scientific Computing*, 14:461–469, 1993.

[132] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, 2003.

[133] Yousef Saad and Martin H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. on Scientific and Statistical Computing*, 7:856–869, 1986.

[134] M. A. Saunders. Solution of sparse rectangular systems using LSQR and CRAIG. *BIT*, 5:588–604, 1995.

[135] O. Schenk and K. Gärtner. On fast factorization pivoting methods for symmetric indefinite systems. *Elec. Trans. Numer. Anal.*, 23:158–179, 2006.

[136] Olaf Schenk and Klaus Gärtner. Solving unsymmetric sparse systems of linear equations with PARDISO. *Future Gener. Comput. Syst.*, 20(3):475–487, 2004.

[137] H. A. Schwarz. Gesammelte mathematische abhandlungen. *J. Computational and Applied Mathematics*, 2:133–143, 1890.

[138] Jonathan Richard Shewchuk. Robust Adaptive Floating-Point Geometric Predicates. In *Proceedings of the Twelfth Annual Symposium on Computational Geometry*, pages 141–150. Association for Computing Machinery, May 1996.

[139] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, may 1996. From the First ACM Workshop on Applied Computational Geometry.

[140] Jonathan Richard Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry Theory and Applications*, 22(1–3):21–74, May 2002.

[141] R. Shewchuk. Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates. *Discrete & Computational Geometry*, 18:305–363, 1997.

[142] H. Si. TetGen. `http://tetgen.berlios.de`, 2007.

[143] H. Si. Adaptive tetrahedral mesh generation by constrained Delaunay refinement. *International J. for Numerical Methods in Engineering*, 75(7):856–880, 2008.

[144] H. Si. *Three Dimensional Boundary Conforming Delaunay Mesh Generation.* PhD thesis, Faculty II - Mathematics and Natural Sciences, Technische Universität Berlin, 2008.

[145] Barry F. Smith. A parallel implementation of an iterative substructuring algorithm for problems in three dimensions. *SIAM J. Sci. Stat. Comp.*, 14:406–423, 1993.

[146] Peter Sonneveld. CGS: A fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 10:36–52, 1989.

[147] Patrick Le Tallec, Yann-Hervé De Roeck, and Marina Vidrascu. Domain decomposition methods for large linearly elliptic three dimensional problems. *J. Computational and Applied Mathematics*, 34, 1991.

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

[149] L. N. Trefethen and D. Bau. *Numerical Linear Algebra.* SIAM, Philadelphia, 1997.

[150] Ulrich Trottenberg, Cornelius W. Osterlee, and Anton Schuller. *Multigrid.* Academic Press, 1st edition, December 2000.

[151] S. Turek. Multigrid techniques for a divergence-free finite element discretization . *East-West J. Numerical Mathematics*, 2(3):229–255, 1994.

[152] S. Turek. *Efficient Solvers for Incompressible Flow Problems: An Algorithmic and Computational Approach.* Springer, 1999.

[153] S. Turek and H. Oswald. A parallel multigrid algorithm for solving the incompressible Navier-Stokes equations with nonconforming finite elements in three dimensions. In *Proc. Parallel CFD'96, Capri/Italy*, 1996.

[154] S. Turek, A. Ouazzi, and R. Schmachtel. Multigrid methods for stabilized nonconforming finite elements for incompressible flow involving the deformation tensor formulation. *J. Numer. Math.*, 10:235–248, 2002.

[155] H. A. van der Vorst. A generalized Lanczos scheme. *Mathematics of Computation*, 39(160):559–561, 1982.

[156] Henk A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 13:631–644, 1992.

[157] Henk A. van der Vorst. *Iterative Krylov Methods for Large Linear Systems.* Cambridge University Press, Cambridge and New York and Melbourne, 2003.

[158] V. K. Varadan and V. V. Varadan. Computation of rigid body scattering by prolate spheroids using the T-matrix approach. *J. Acoust. Soc. Am.*, 71:22–25, 1982.

[159] I. N. Vekua. Solutions of the equation $\Delta u + \lambda^2 u = 0$. *Soobshcheniga Akademii Nauk Gruz. SSSR*, 3:307–314, 1942.

[160] I. N. Vekua. Inversion of an integral transformation and some applications. *Soobshcheniga Akademii Nauk Gruz. SSSR*, 6:177–183, 1945.

[161] I. N. Vekua. *New Methods for Solving Elliptic Equations.* North Holland Publishing Co., 1967.

[162] Judith A. Vogel. Flexible BiCG and flexible Bi-CGSTAB for nonsymmetric linear systems. *Applied Mathematics and Computation*, 188(1):226–233, 2007.

[163] H. Wang, Q. Wu, X. He, and L. Li. Computation of wave scattering problems from spheric body: Derivation of the new Sommerfeld-Watson transformation. *Progress In Electromagnetics Research Symposium 2005, Hangzhou, China*, pages 22–26, 2005.

[164] P. C. Waterman. New formulation of acoustic scattering. *J. Acoust. Soc. Am.*, 45:1417–1429, 1969.

[165] Rüdiger Weiss. *Parameter-Free Iterative Linear Solvers*, volume 97. Akademie-Verlag, Berlin, 1996.

[166] Pieter Wesseling. *An Introduction to Multigrid Methods*. R. T. Edwards, Inc., 1st edition, December 2004.

[167] Wolfram Research, Inc. *Mathematica Edition: Version 5.1*, 2004.

[168] S. T. Zalesak. Fully multidimensional flux-corrected transport algorithms for fluids. *J. Computational Physics*, 31:335–362, 1979.

[169] S. T. Zalesak. *The design of Flux-Corrected Transport (FCT) algorithms for structured grids*. Flux-Corrected Transport: Principles, Algorithms, and Applications. Springer, 2005.

[170] S. Zhang and J. Jin. *Computation of Special functions*. Wiley Interscience, 1996.

[171] Xuejun Zhang. Domain decomposition algorithms for the biharmonic dirichlet problem. In *Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 119–126. SIAM, 1992.

[172] Xuejun Zhang. Multilevel additive Schwarz methods. 63(4):521–539, 1992.

[173] Xuejun Zhang. Studies in domain decomposition: Multilevel methods and the biharmonic dirichlet problem. Technical report, University of Maryland, Department of Computer Science, 1992.