NTNU

Norwegian University of
Science and Technology

Finite element solutions to the wave equation in non-convex domains

A relaxation of the CFL condition in the
presence of local mesh refinement

Andreas Borgen Longva

Master of Science in Physics and Mathematics
Submission date: March 2017
Supervisor: Ulrik Skre Fjordholm, IMF
Co-supervisor: Daniel Peterseim, University of Bonn
Mira Schedensack, University of Bonn

Norwegian University of Science and Technology
Department of Mathematical Sciences

Abstract

Finite element solutions in non-convex domains generally suffer from reduced convergence rates due to reduced regularity. This is also the case for the wave equation. To remedy the situation, (strong) local mesh refinement can restore the optimal convergence rates associated with smooth solutions in convex domains. However, this is problematic for the application of explicit time integrators such as the Leapfrog method, for which time steps are required to satisfy the well-known CFL condition.

In this thesis, we study a method proposed by Peterseim and Schedensack which promises to recover the stability of the Leapfrog method while maintaining the optimal convergence rate associated with convex domains. The method has two stages. In the offline phase, a basis for a *corrected* finite element space is constructed. In the online phase, the wave equation is solved as usual with the mass- and stiffness matrices from the corrected space.

The main contribution of this thesis is the efficient implementation and subsequent evaluation of the practical applicability of the method by Peterseim and Schedensack. We show through numerical experiments that the cost of computing the basis is reasonable in the sense that it can be within the same order of magnitude as the cost of online computations. Moreover, we show that the method clearly outperforms the solution methods considered with respect to the cost of online computations.

In addition, we propose an augmented version of the Leapfrog method which is shown to perform very well in numerical experiments, and we prove that the method is stable under the same CFL condition as the standard finite element space on the quasi-uniform mesh.

The thesis concludes with a discussion of settings for which the method may be particularly well-suited and lists scenarios for which the method is expected to perform poorly.

Sammendrag

Løsninger av differensialligninger med endelig-element-metoden fører ofte til reduserte konvergensrater på grunn av redusert regularitet av den eksakte løsningen. For å motvirke dette kan man anvende lokalt forfinede mesh med det hensyn å gjenoppnå den optimale konvergensraten som man assosierer med glatte løsninger i konvekse domener. Det viser seg derimot at dette skaper problemer for eksplisitte tidsintegratorer slik som Leapfrog-metoden, som krever at tidsstegene oppfyller den velkjente CFL-betingelsen.

I denne oppgaven studerer vi en metode som nylig ble foreslått av Peterseim og Schedensack, og som lover å gjenoppnå stabilitet for Leapfrog-metoden samtidig som man oppnår den optimale konvergensraten man er kjent med fra konvekse domener. Metoden har to stadier. I offline-fasen konstruerer man en basis for et *korrigert* endelig-element-rom. I online-fasen løser man bølgeligningen på vanlig måte ved hjelp av masse- og stivhetsmatrisen fra det korrigerte rommet.

Hovedbidraget ved denne oppgaven er en effektiv implementasjon av metoden til Peterseim og Schedensack, samt en evaluering av metodens praktiske anvendbarhet. Vi viser gjennom numeriske eksperimenter at kostnaden ved å regne ut basisen er rimelig i den forstand at den kan være av samme størrelsesorden som kostnaden av online-fasen. Videre viser vi at metoden åpenbart yter bedre i online-fasen enn de andre løsningsmetodene vi vurderer.

I tillegg foreslår vi en modifisert Leapfrog-metode som viser seg å fungere veldig bra i numeriske eksperimenter, og vi viser at denne metoden er stabil gitt den samme CFL-betingelsen som det vanlige endelig-element-rommet på det kvasi-uniforme meshet.

Oppgaven avsluttes med en diskusjon av situasjoner hvor det kan tenkes at metoden er spesielt godt egnet, samt scenarioer hvor metoden antas å fungere dårlig.

Acknowledgements

There are in particular three individuals to which I owe a significant debt of gratitude to. They have each played a crucial role in my work on this thesis, so I will not mention them in order of importance.

The first is my local supervisor at NTNU, Professor Ulrik Skre Fjordholm. I think it is fair to say that I owe a great portion of what I know about academic writing to his persistently excellent advice and his detailed feedback. It was also his course on the Finite Element Method that initially sparked my interest in pursuing finite element methods further.

The second is Professor Daniel Peterseim at the Institute for Numerical Simulation (INS) at the University of Bonn. Professor Peterseim graciously allowed me to write my thesis as a visiting student at the University of Bonn, working on a topic that tightly integrates with his research.

The third is Dr. Mira Schedensack, also at the INS. Dr. Schedensack has largely been responsible for patiently correcting my misconceptions and mistakes as I gradually absorbed the ideas of their research. I entered most meetings in a state of great confusion, yet almost always emerged with a clear idea of the road ahead.

I would also like to thank Professor Anton Evgrafov at NTNU for his advice on numerical linear algebra during the course of a few brief exchanges.

Finally, I have relied on a number of open-source software libraries in my work on this thesis. While I am grateful to all authors of these libraries, I would especially like to mention a few developers in particular. First, I am grateful to the developers of the `Eigen` library for their extremely impressive response time when they patched a bug which was blocking my work. I am also grateful to Luis Pedro Coelho for his work on the excellent task-based parallelization framework `Jug`, without which I have no idea how I would have completed my numerical experiments, and also for his personal assistance on one occasion. Finally, I would like to thank Denis Demidov for his helpfulness when I wanted to integrate his library `amgcl` into my work.

Remarks

I want to point out that the work behind Chapter 3 — in which I prove stability and derive error estimates for the Leapfrog and Crank-Nicolson methods — was for the most part carried out in the semester project that came before this Master's thesis. I have included it in this thesis because I could not find appropriate references which would give me the exact results I needed.

For the purposes for this thesis, I have however rephrased and generalized the convergence results so that they would also be applicable to the corrected finite element space defined in Chapter 5.

Contents

Chapter 1

Introduction

In this thesis, we will consider the solutions of the second-order wave equation in polygonal domains with finite element methods. The equation can be written in its classical form

$$u_{tt} - \Delta u = f.$$

We consider the domain of interest Ω to be a bounded, polygonal domain. Because we wish to study problems in which classical derivatives may not exist, we must instead consider the functional-analytic *weak formulation*

$$(u_{tt}, v) + (\nabla u, \nabla v) = (f, v) \qquad v \in H_0^1(\Omega)$$

with appropriate initial and boundary conditions. This will be precisely defined in Section 2.1.

One of the most popular approaches for solving the wave equation with the Finite Element Method (FEM) in space uses the Leapfrog method [1][2][3] to advance the solution in time. The Leapfrog method is an explicit method, and along with other explicit time integrators for the wave equation, its stability depends on the relation between the time step Δt and the resolution of the computational mesh. More precisely, one has the requirement that Δt must satisfy

$$\Delta t \leq C h_{\min}$$

for some constant $C > 0$ independent of $h_{\min}$, which relates to the size of the smallest element in the mesh. For linear finite elements, we will see that given sufficient regularity of the exact solution u, the method attains an error bound of $\mathcal{O}(h^2 + (\Delta t)^2)$ in the $L^2(\Omega)$ norm and $\mathcal{O}(h + (\Delta t)^2)$ in the H^1 norm. Here h denotes the mesh resolution. For quasi-uniform mesh families, we usually have that $h/h_{\min}$ is not too large, and so we see that the above requirement for stability is typically a reasonable condition.

For convex domains with sufficiently smooth initial conditions and right-hand side f, it can be shown that the exact solution u is smooth enough to attain these convergence

rates with quasi-uniform meshes. However, if the domain Ω is not convex and has re-entrant corners - corners whose interior angle exceeds π radians - it is possible that even with smooth data, the exact solution u exhibits singularities in the re-entrant corners. In this case, the regularity of u is reduced, and the standard piecewise linear polynomial finite element space with a quasi-uniform mesh will fail to give the optimal convergence rate $\mathcal{O}(h^2)$ in the L^2 norm and $\mathcal{O}(h)$ in the H^1 norm.

It is well-known from elliptic problems (see e.g. [4][5]) that appropriate local mesh refinement in the vicinity of re-entrant corners may help to restore the optimal convergence rate associated with smooth solutions. It turns out that this is the case also for the wave equation [6], and by way of local refinement one can recover the optimal rate $\mathcal{O}(h) = \mathcal{O}(h_{\max})$ in the H^1 norm. However, the resulting locally refined meshes are heavily graded towards the re-entrant corners, and as a result, it is observed that

$$h_{\min} \ll h_{\max}.$$

In fact, the size of the smallest element may be many orders of magnitude smaller than the largest. From the above discussion, it is clear that this is severely detrimental to the stability of the Leapfrog method. The result is that the method is essentially unusable in the presence of such local mesh refinement.

To overcome this difficulty, the most straightforward approach is to use an unconditionally stable implicit scheme. To this end, we will show how one can adapt the classical Crank-Nicolson method [7] for parabolic PDEs so that it gives rise to an unconditionally stable method for the wave equation. Here we have mostly adapted and extended the techniques used to develop the theory of the Leapfrog method to prove the fundamental properties of the Crank-Nicolson-derived scheme.

The downside to using an implicit method is increased computational complexity associated with solving a possibly ill-conditioned linear system at each time step. In the pursuit to achieve greater computational efficiency, several techniques [8][9][10][11] that in different ways restore the usefulness of explicit time integrators have been developed. The main objective of this thesis is to study a method proposed by Peterseim and Schedensack [12], which uses techniques from multi-scale modeling and numerical homogenization to combine the reasonable stability region of the quasi-uniform mesh with the optimal convergence rate of the locally refined mesh.

The gist of this method is the computation of a number of *correctors*, which are used to augment the standard basis associated with the quasi-uniform mesh in a way such that it is able to exploit information from the locally refined mesh. The result is a new finite element space in which the Leapfrog method is both stable and is able to attain the optimal convergence rate in the H^1 norm.

The computational efficiency of the method is however somewhat of an open question. The computation of the correctors involves the solution to a large number of elliptic problems. A central question is thus whether the computation of these correctors is practically feasible, in the sense that the computation can be performed in reasonable time. Moreover, the mass- and stiffness matrices associated with the new finite

element space might have considerably higher density than the system matrices associated with the standard finite element space. A second important question is then whether the increased density is so significant that the method is ultimately rendered inefficient.

The goal of this thesis is to at least partially answer both of these questions. To that end, we will describe an efficient implementation of the method, and in the process we propose two different methods that allow the efficient computation of correctors. It is also important to note that these two methods are not mutually exclusive. While one method which leverages a sparse direct solver is perhaps particularly suited to solving the smaller elliptic problems, the second method based on GMRES [13] may be better suited for the larger problems.

During the course of running numerical experiments, an augmented version of the Leapfrog method was discovered to perform very well in practice. The augmented version reduces the cost of computing load vectors and allows the usage of a much sparser mass matrix than the original method proposed by Peterseim and Schedensack. Stability for this new method is proved, but theoretical error estimates have not been obtained.

We will finally present numerical experiments which show that the method of Peterseim and Schedensack can attain high efficiency, and for the model problem considered, it is clearly the most efficient approach of the ones considered. However, it is likely that the performance of the Crank-Nicolson method which we have used as a baseline is tainted by an inappropriate preconditioner, and it is possible that the outcome would not be so clear cut given a better preconditioner. We conclude the thesis by a discussion of the advantages and limitations of the method, as well as an outlook on future topics of further research.

A prototype software library named `crest` providing a complete implementation of the method of Peterseim and Schedensack is also released as part of this thesis.

1.1 Outline of the thesis

A brief outline of the contents of each chapter follows.

- Chapter 2 introduces the weak formulation of the problem, as well as basic language and notions associated with finite element theory.
- Chapter 3 introduces the Leapfrog and Crank-Nicolson methods, and develops the theory for the energy conservation, stability and convergence properties of each method.
- Chapter 4 discusses triangulation of the domain, and demonstrates how local refinement can mitigate the detrimental effects caused by non-convex domains.
- Chapter 5 discusses the theory of the method proposed by Peterseim and Schedensack.

- Chapter 6 discusses practical aspects of the aforementioned method, and proposes two methods for solving the *corrector problems* that are associated with the method.
- Chapter 7 proposes an augmented Leapfrog method which has less computational complexity than the original Leapfrog method when used in conjunction with the method by Peterseim and Schedensack, and stability is proved for this augmented method.
- Chapter 8 presents an experimental error analysis of the numerical methods, as well as providing empirical evidence for the runtime characteristics of the methods studied.
- Chapter 9 summarizes some of the most important results and observations in the thesis, and mentions some possible directions for future research.

1.2 Notation

For the convenience of the reader, some of the notation that is frequently used is presented here, in addition to wherever it is introduced.

- C is often used as a generic constant, and is typically different between individual theorems and lemmas. However, within the same context, an effort has been made to clearly distinguish different constants by incrementally indexing new constants by C_0, C_1, etc.
- $C(u, T)$ emphasizes the fact that the constant depends only on data of the problem, which includes T, f, u_0, v_0.
- Ω represents an open, bounded polygonal domain.
- $H^s = H^s(\Omega) = W^{s,2}(\Omega)$ for any s, unless otherwise stated, where $W^{s,2}$ is standard notation for Sobolev spaces. Similarly, $L^2 = L^2(\Omega)$.
- The gradient ∇ and Laplacian Δ refer only to *spatial* derivatives.
- For a Hilbert space V,

$$L^q(0,T;V) = \left\{ g : [0,T] \to V \text{ such that } g \text{ is measurable and } \int_0^T \|g(t)\|_V^q \, \mathrm{d}t < \infty \right\}.$$

- $(\cdot,\cdot) = (\cdot,\cdot)_{L^2} = (\cdot,\cdot)_{L^2(\Omega)}$ denotes the L^2 inner product.
- $\|\cdot\| = \|\cdot\|_{L^2} = \|\cdot\|_{L^2(\Omega)}$ denotes the L^2 norm.
- $a(\cdot,\cdot) = (\nabla\cdot, \nabla\cdot)_{L^2(\Omega)}$.
- $C_c^\infty(Q;R)$ denotes an infinitely differentiable mapping from Q to R with compact support, and $C_c^\infty(Q) = C_c^\infty(Q;\mathbb{R})$.

- Δt represents a *constant* time step for a fully discretized scheme.
- $t^n = n\Delta t$ for $n \geq 0$.
- $f^n = f(t^n)$ and similarly for other functions.
- u_h^n represents a fully discretized (in space and time) approximation of the exact solution $u(t^n)$.
- $\delta u^{n+1/2} = (u^{n+1} - u^n)/\Delta t$.
- $u^{n+1/2} = (u^{n+1} + u^n)/2$.
- $\delta^2 u_h^n = \frac{u_h^{n+1} - 2u_h^n + u_h^{n-1}}{(\Delta t)^2}$.
- $\mathcal{T}_h$ is a mesh with mesh resolution h, e.g. a set of elements. In Chapters 5 and onwards, $\mathcal{T}_H$ and $\mathcal{T}_h$ refer to a pair of meshes where $\mathcal{T}_H$ is quasi-uniform and $\mathcal{T}_h$ is a local refinement of $\mathcal{T}_H$.
- $\mathcal{N}(\mathcal{T}_h)$ is the set of vertices in the mesh $\mathcal{T}_h$, and $\#\mathcal{N}(\mathcal{T}_h)$ denotes the number of vertices in the mesh.
- $S^p(\mathcal{T}_h)$ and $S_0^p(\mathcal{T}_h)$ are piecewise polynomial finite element spaces of degree p on the mesh $\mathcal{T}_h$ as defined by Definition 2.2.1.
- $N_h = \dim S_0^1(\mathcal{T}_h)$ and similarly $N_H = \dim S_0^1(\mathcal{T}_H)$.
- λ_i or λ_z corresponds to a basis function (often a Lagrangian basis function) associated with the node index i in the associated finite element space or vertex z in the associated computational mesh.
- $\lambda_{H,i} \in S_0^1(\mathcal{T}_H)$ and $\lambda_{h,i} \in S_0^1(\mathcal{T}_h)$ refer to Lagrangian basis functions in the coarse and fine polynomial space, respectively.

Chapter 2

Mathematical foundation

In this chapter, we will introduce some essential basic notions associated with the solutions of partial differential equations with the Finite Element Method (FEM). In order to avoid having to repeat large amounts of standard finite element theory, we will assume that the reader is acquainted with the solution of time-dependent problems with the FEM by the method of lines, but only to the extent of basics one would find in any textbook on the subject.

We will not discuss properties of the wave equation itself. For this, we refer the reader to the standard monograph by Evans [14]. The thesis should otherwise be largely self-contained, although some experience with numerical linear algebra may be required to fully appreciate the discussion in Chapter 6.

2.1 Weak formulation

This section introduces the formal definition of the problem by way of a *weak formulation* of the wave equation. We refer the reader to [14] for the motivation and justification for the following formulation of the problem, as well as the corresponding theory of existence and uniqueness.

Definition 2.1.1 (Weak formulation)
Let $\Omega \subseteq \mathbb{R}^d$ for $d = 2, 3$ be a bounded, polygonal domain. For a function u which satisfies

$$u \in L^2(0,T;H_0^1(\Omega)), \qquad \dot{u} \in L^2(0,T;L^2(\Omega)), \qquad \ddot{u} \in L^2(0,T;H^{-1}(\Omega)),$$

we say that u is a *weak solution* to the wave equation if

1. $f \in L^2(0,T;L^2(\Omega))$.
2. $u(0) = u_0 \in H_0^1(\Omega)$.
3. $\dot{u}(0) = v_0 \in L^2(\Omega)$.
4. for almost every $t \in [0,T]$,

$$(\ddot{u}(t), v) + a(u(t), v) = (f(t), v) \qquad \forall v \in H_0^1(\Omega), \tag{2.1}$$

where

$$a(u,v) := (\nabla u, \nabla v)\,. \tag{2.2}$$

Remark. In the above definition, as is always the case with Sobolev spaces, the derivatives must be understood in a distributional sense.

By the Poincaré inequality, it is easy to see that $a(\cdot,\cdot)$ induces an inner product on the space $H_0^1(\Omega)$, and consequently $\sqrt{a(\cdot,\cdot)}$ is a norm on the space $H_0^1(\Omega)$. The following lemma follows naturally.

Lemma 2.1.2 (Equivalence of norms)
There exist constants $C_0, C_1 > 0$ such that for all $v \in H_0^1(\Omega)$,

$$C_0 \|v\|_{H^1} \leq \sqrt{a(v,v)} \leq C_1 \|v\|_{H^1}\,. \tag{2.3}$$

Throughout this thesis, we will use both the notation $(\nabla u, \nabla v)$ and $a(u,v)$ interchangably.

2.2 Finite element spaces

In order to apply the FEM to Definition 2.1.1, we let $\mathcal{T}_h$ denote a *mesh* of our computational domain Ω. We will make a more precise statement on the geometrical setting in Chapter 4. For now, it is sufficient to let $\mathcal{T}_h$ denote a set of *elements*, and that for each element $T \subseteq \Omega$, $d+1$ vertices are associated with it. Here, the mesh resolution parameter h relates to the size of the largest element. For the purposes of this thesis, we

consider a *finite element space* to be a finite-dimensional subspace of $H_1(\Omega)$ generated from the computational mesh $\mathcal{T}_h$.

We now introduce the notation we use for the standard piecewise polynomial finite element space, which coincides with the notation used in [12].

Definition 2.2.1 (Piecewise polynomial finite element spaces)
Given a computational mesh $\mathcal{T}_h$, we define the standard piecewise polynomial finite element space $S_p(\mathcal{T}_h)$ of order p

$$S^p(\mathcal{T}_h) := \left\{ v_h \in C^0(\Omega) \mid v_h\big|_K \in \mathbb{P}_p(K) \, \forall K \in \mathcal{T}_h \right\} \tag{2.4}$$

where $\mathbb{P}_p(K)$ denotes the set of polynomials on K of total degree less or equal to p. We furthermore define the piecewise polynomial space $S_0^p(\Omega)$ to be the corresponding space which vanishes at the boundary of Ω by

$$S_0^p(\mathcal{T}_h) := S^p(\mathcal{T}_h) \cap H_0^1(\Omega). \tag{2.5}$$

In principle, any basis for this space will do. However, for the sake of computational complexity it is desirable to use a basis with small, local support. To this end we define the notation for the standard Lagrangian basis below.

Definition 2.2.2 (Lagrangian basis)
Given a polynomial finite element space $S^p(\mathcal{T}_h)$, we define the Lagrangian basis function $\lambda_i \in S^p(\mathcal{T}_h)$ associated with node $n_i \in \Omega$ by the property

$$\lambda_i(n_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{2.6}$$

for each node $n_j \in \Omega$ associated with a degree of freedom in $S^p(\Omega)$.

Remark. Because our purpose here is merely to make clear the notation used for the Lagrangian basis functions, we will avoid the exact definition of what constitutes a node of $S^p(\mathcal{T}_h)$, but it suffices to say that it coincides with the usual notions.

The next definition introduces a property of finite element spaces that is of crucial importance in the derivation of a stability estimate for the Leapfrog method, which we will introduce in Chapter 3.

Definition 2.2.3 (Inverse property)
For a finite element space $X_h \subseteq H_0^1(\Omega)$, we say that it has the *inverse property* if and only if there exists a constant $C_I > 0$ independent of h such that

$$\|\nabla v_h\|_{L^2(\Omega)} \leq C_I \, h^{-1} \, \|v_h\|_{L^2(\Omega)} \quad \forall v_h \in X_h. \tag{2.7}$$

Remark. The above definition for the inverse property is a specialized version of the more general definition found in [3]. Moreover, Quarteroni notes in [15] that X_h has the inverse property if Ω is triangulated by a regular and quasi-uniform family of meshes $\{\mathcal{T}_h \mid h > 0\}$. These properties will be properly introduced in the later section on triangulation.

The following lemma will be useful in our later proofs, and is a direct consequence of the preceding definition and the Cauchy-Schwarz inequality.

Lemma 2.2.4 (Inverse property for the bilinear form)
If $X_h \subseteq H_0^1(\Omega)$ has the inverse property, there exists $C_{\text{inv}} > 0$ independent of h such that

$$a(u_h, v_h) \leq \frac{C_{\text{inv}}}{h^2} \, \|u_h\| \, \|v_h\| \tag{2.8}$$

for any $u_h, v_h \in X_h$.

On occasion, it will be necessary to approximate a given function on $H_0^1(\Omega)$ in a finite element space. For example, we might need to construct finite-dimensional approximations of the initial conditions u_0 and v_0. If u_0 and v_0 are continuous, it is usually sufficient to use the standard nodal interpolator. Throughout this thesis, we will frequently make use of the *elliptic projection*, sometimes also called the *Ritz interpolator*.

Definition 2.2.5 (Elliptic projection)
Let X_h be a finite element space and V be a subspace of $H_0^1(\Omega)$, and assume that $X_h \subseteq V \subset H_0^1(\Omega)$. The elliptic projection $R_h : V \to X_h$ is defined as the orthogonal projection of V onto X_h with respect to the $a(\cdot, \cdot)$ inner product, in the sense that, for any $v \in V$,

$$a(R_h v, v_h) = a(v, v_h) \qquad \forall v_h \in X_h. \tag{2.9}$$

The elliptic projection is a special case of a projection. We will give a formal definition from [16] of a projection below, as we will use it in later chapters.

Definition 2.2.6 (Projection)
Given a vector space V and a subspace $W \subseteq V$, an operator $P : V \to W$ is a *projection* if $P^2 = P$, or equivalently

$$Pw = w \quad \forall w \in P(V),$$

where $P(V) \subseteq W$ denotes the image of P.

In order to estimate the error of the Leapfrog and Crank-Nicolson methods in Chapter 3, we will require error estimates for the elliptic projection in piecewise polynomial spaces.

Lemma 2.2.7 (Error of elliptic projection in polynomial spaces)
For $X_h = S_0^p(\mathcal{T}_h)$ and $2 \leq s \leq p+1$, there exist constants $C_0, C_1 > 0$ independent of h such that for all $v \in H^s(\Omega) \cap H_0^1(\Omega)$,

$$\|v - R_h v\|_{L^2(\Omega)} \leq C_0 h^s \|v\|_{H^s(\Omega)} \tag{2.10}$$

$$\|v - R_h v\|_{H^1(\Omega)} \leq C_1 h^{s-1} \|v\|_{H^s(\Omega)}. \tag{2.11}$$

Proof. See [17]. □

The reason that the elliptic projection plays such an important role in this thesis is that it is closely related to the *best approximation error* in H^1 for a given finite element space. To define precisely what we mean by this, consider the following lemma, which is a simple consequence of Céa's lemma.

Lemma 2.2.8
Given any $u \in H_0^1(\Omega)$, there exists a constant $C > 0$ such that the elliptic projection onto a finite element space X_h satisfies

$$\|u - R_h u\|_{H^1} \leq C \inf_{v_h \in X_h} \|u - v_h\|_{H^1} \tag{2.12}$$

Lemma 2.2.8 states that the elliptic projection of $u \in H_0^1(\Omega)$ onto the finite element space X_h is *the best approximation of u in X_h* up to a constant, with respect to the H^1 norm. We emphasize this point because it has great significance for Chapter 4 and 5.

In order to link the functional-analytic definitions with algebraic formulations that are suitable for direct computation, we will need to be able to relate the quantities that appear in the weak (Galerkin) formulation of the problem with algebraic equivalents.

In order to do so, we will need to introduce the usual *mass-* and *stiffness* matrices (collectively referred to as *system matrices*).

Definition 2.2.9 (System matrices)
For a finite element space X_h with dimension $N_h := \dim X_h$ and a basis $\Lambda = \{\lambda_i \mid i = 1, \dots, N_h\} \subseteq X_h$, we define the *mass matrix* $M \in \mathbb{R}^{N_h \times N_h}$ and *stiffness matrix* $A \in \mathbb{R}^{N_h \times N_h}$ by

$$M_{ij} = (\lambda_j, \lambda_i)\,, \tag{2.13}$$
$$A_{ij} = a(\lambda_j, \lambda_i) \tag{2.14}$$

for $i, j = 1, \dots, N_h$.

As in the case of elliptic equations, we define the concept of *load vectors* which relate to the right-hand side term f.

Definition 2.2.10 (Load vectors)
For a finite element space X_h with dimension $N_h := \dim X_h$ and a basis $\Lambda = \{\lambda_i \mid i = 1, \dots, N_h\} \subseteq X_h$, we define the (time-dependent) *load vector* $b(t) \in \mathbb{R}^{N_h}$ by

$$b_i(t) = (f(t), \lambda_i)\,. \tag{2.15}$$

for $i = 1, \dots, N_h$.

Note that neither the definition for the system matrices M and A nor the definition of the load vector $b(t)$ are limited to Lagrangian basis functions. In Chapter 5 we will introduce a class of finite element spaces for which the basis we will use is not Lagrangian.

Chapter 3

Discretization

In this thesis, we consider spatial discretization with the Finite Element Method. For the discretization in time, we consider two different methods which have similar convergence and energy conservation properties, but differ in computational complexity and conditions for stability.

We denote by $u_h^n \approx u(t^n)$ the approximation of the exact solution u at time $t^n = n\Delta t$, and the two methods under study are both two-step recurrence relations, which means that u_h^{n+1} depends on u_h^n and u_h^{n-1}.

Due to its simplicity and approximation properties, the Leapfrog method is a very natural fit for the wave equation, and hence is arguably the most popular. It is straightforward to implement, conserves the energy of the system, offers relatively low computational complexity and sports a quadratic convergence rate in time. However, as we shall see, its stability depends on the fulfillment of the well-known *CFL* condition.

The literature search for citable, rigorous and complete results for the expected convergence rate for the Leapfrog method in polynomial spaces was unfortunately not entirely fruitful. While Christiansen [1] and Joly [2] both discuss the method, their discussion takes place from a more general perspective, and convergence results are not ultimately shown specifically for polynomial spaces, though attainable results are hinted at. However, this leaves open the questions of what assumptions must be made on the regularity of the exact solution and how to approximate the u_h^0 and u_h^1 for optimal convergence rates to be attained. On the other hand, there is a fairly detailed proof in [3], but it makes the arguably very impractical assumption that $u_0 = v_0 = 0$. For the purposes of this thesis we also had the additional constraint of making sure that we could prove convergence also for the corrected finite element space that will be introduced in Chapter 5. Because of all these considerations, we will here present the complete stability and error analysis of the Leapfrog method for any (sufficiently regular) u_0 and v_0. The proof of convergence presented here relies on techniques from the aforementioned literature, but ultimately follows its own line of reasoning to the

final result.

Because the stability of the Leapfrog method is generally not preserved in the presence of strong local mesh refinement, an unconditionally stable method with comparable convergence properties was sought as a baseline with which to compare the method that will be presented in Chapter 5. To this end, we will derive a method similar to the Crank-Nicolson method for parabolic PDEs. It turns out, however, that the results we need for the purposes of this thesis are not easily found in the existing literature. Dupont [18] and Baker [19] study a method which is in some sense mathematically equivalent, but it relies on the common practice of rewriting the second-order wave equation as a first-order system. For our purposes, this is less than ideal because it doubles the number of variables in the system, and so is not a completely appropriate comparison with the Leapfrog method. In contrast, the method we present in this section is based directly on the second-order formulation, and so is very similar to the Leapfrog method. The method appears in [20] in relation to a multi-scale method for the wave equation, but the results seem not to be directly applicable to our present situation. An almost identical method is also presented by Larsson [17], but it has a slightly different right-hand side. In this case, optimal convergence rates are presented without proof, but curiously the method does not seem to work as advertised in numerical experiments unless $f = 0$. In light of these difficulties, we will - as in the case of the Leapfrog method - present the full stability and convergence analysis for the Crank-Nicolson method. In order to make the process a little easier on the reader, we have sought to develop proofs that closely follow the flow of the corresponding proofs for the Leapfrog method.

It must also be noted that there exists a French paper by Bamberger et al. [21] which seems to cover a detailed analysis of various discretization schemes for the wave equation, but due to the risk of quite literally getting lost in translation, we have elected not to use the results of this paper.

Our strategy for the analysis will be as follows. For each of the two methods, we will first present a functional-analytic formulation similar to the weak formulation (2.1) for the wave equation. We will go on to show how this formulation is equivalent to an algebraic formulation which leads to a practically computable method. Next, we define an appropriate notion of *discrete energy* for each method, and go on to show that this energy is conserved, but only under certain assumptions for the Leapfrog method. The energy conservation is furthermore shown to lead to stability of the method. Finally, we complete the analysis by proving convergence in terms of errors of the elliptic projection defined in Definition 2.2.5, and subsequently demonstrating how this leads to expected convergence rates for piecewise polynomial finite element spaces.

3.1 Finite difference operators

The discretization methods we present rely on finite difference operators for the temporal discretization. In order to prove convergence, we will need some results involving these operators. For brevity, we will omit the proofs of these results, as they are easily obtained from the theory of Taylor series by using mean-value forms of the remainder, along with applications of the intermediate value theorem where appropriate.

Lemma 3.1.1 (First-order difference operators)
Given $t_0 \in \mathbb{R}$, $k > 0$ and a function $g \in C([t_0 - k, t_0 + k]) \cap C^2((t_0 - k, t_0 + k))$, the first-order forward and backward operators satisfy

$$\delta^+ g(t_0) := \frac{g(t_0 + k) - g(t_0)}{k} = \dot{g}(t_0) + \frac{k}{2}\ddot{g}(t_+), \tag{3.1}$$

for the forward operator, and

$$\delta^- g(t_0) := \frac{g(t_0) - g(t_0 - k)}{k} = \dot{g}(t_0) - \frac{k}{2}\ddot{g}(t_-), \tag{3.2}$$

for the backward operator, for some $t_+ \in (t_0, t_0 + k)$ and $t_- \in (t_0 - k, t_0)$. If in addition, $g \in C^3((t_0 - k, t_0 + k))$, the first-order central operator satisfies

$$\delta g(t_0) := \frac{g(t_0 + k) - g(t_0 - k)}{2k} = \dot{g}(t_0) + \frac{k^2}{6}\frac{\mathrm{d}^3 g}{\mathrm{d}t^3}(t_c), \tag{3.3}$$

for some $t_c \in (t_0 - k, t_0 + k)$.

Lemma 3.1.2 (Second order central difference operator)
Given $t_0 \in \mathbb{R}$, $k > 0$ and a function $g \in C([t_0 - k, t_0 + k]) \cap C^4((t_0 - k, t_0 + k))$, the second order central difference operator satisfies

$$\delta^2 g(t_0) := \frac{g(t_0 + k) - 2g(t_0) + g(t_0 - k)}{k^2} = \ddot{g}(t_0) + \frac{k^2}{12}\frac{\mathrm{d}^4 g}{\mathrm{d}t^4}(t_c) \tag{3.4}$$

for some $t_c \in (t_0 - k, t_0 + k)$.

In addition to using the above difference notations for continuous functions, we will

also use it to denote fully discrete quantities. More precisely, we have that

$$\delta u_h^{n+1/2} := \frac{u_h^{n+1} - u_h^{n-1}}{\Delta t},$$
$$\delta^2 u_h^n := \frac{u_h^{n+1} - 2u_h^n + u_h^{n-1}}{(\Delta t)^2}.$$

3.2 The Leapfrog method

We will now define the Leapfrog method in terms of a functional-analytic weak formulation. We will go on to give a fully algebraic representation more suitable for direct computation. Furthermore, we will show that under conditions on the size of the time step Δt, the method preserves a discrete approximation of energy, which leads to stability. Finally, we will prove convergence of the method for any finite element space in terms of the approximation properties of the elliptic projection onto the space.

Definition 3.2.1 (Leapfrog method for the wave equation)
Given a finite element space $X_h \subseteq H_0^1(\Omega)$, the weak formulation of the fully discretized Leapfrog method is for $t = n\Delta t$ and integer $n \geq 1$ given by

$$\left(\frac{u_h^{n+1} - 2u_h^n + u_h^{n-1}}{(\Delta t)^2}, v_h \right) + a(u_h^n, v_h) = (f^n, v_h) \qquad \forall v_h \in X_h. \tag{3.5}$$

The next lemma gives an equivalent, computable representation as a linear system, which lets us solve the problem numerically.

Lemma 3.2.2 (Algebraic formulation of the Leapfrog method)
Let u_h^n be defined as in Definition 3.2.1 *and let $\Lambda = \{\lambda_i \mid i = 1, \ldots, N_h\}$ be a basis of X_h, with $N_h = \dim X_h$. Define $\xi^n \in \mathbb{R}^{N_h}$ such that*

$$u_h^n = \sum_{j=1} \xi_j^n \lambda_j. \tag{3.6}$$

Then the Leapfrog method is for $n \geq 1$ equivalent to the linear system

$$M\xi^{n+1} = (\Delta t)^2 \left(b^n - A\xi^n\right) + M\left(2\xi^n - \xi^{n-1}\right), \tag{3.7}$$

where the system matrices M and A are defined by Definition 2.2.9 *and the load vector $b^n := b(t^n)$ by Definition* 2.2.10.

Proof. Let $v_h = \lambda_i$ in (3.5) and insert (3.6). □

We note that a very favorable property of the Leapfrog method is that it only requires the solution of the mass matrix M at every time step. It is well known that the mass matrix is generally well-conditioned and reasonably cheap to invert. Furthermore, we will see in Section 3.4 that the mass matrix can be accurately approximated by a diagonal matrix, which makes the method fully explicit.

Our next goal is to show that the Leapfrog method conserves a certain discrete energy quantity, which we define below.

Definition 3.2.3 (Discrete energy for the Leapfrog method)
We define the discrete energy for the Leapfrog method by

$$\hat{\mathcal{E}}_h^{n+1/2} := \frac{1}{2}\left\|\delta u_h^{n+1/2}\right\|_{L^2(\Omega)}^2 + \frac{1}{2}a(u_h^n, u_h^{n+1}), \tag{3.8}$$

where

$$\delta u_h^{n+1/2} := \frac{u_h^{n+1} - u_h^n}{\Delta t}. \tag{3.9}$$

From the above definition, we note that, due to the second term, the discrete energy is in general not a non-negative quantity. For the stability analysis of the Leapfrog method, it is desirable to determine a condition for which the discrete energy is non-negative. The next lemma will enable us to do exactly that.

Lemma 3.2.4 (CFL and the non-negativity of the Leapfrog discrete energy)
Assume that X_h has the inverse property from Definition 2.2.3*, and that Δt is chosen sufficiently small such that*

$$1 - \frac{C_{\text{inv}}(\Delta t)^2}{2h^2} \geq \lambda > 0 \tag{3.10}$$

for some $\lambda \in (0,1)$, and C_{inv} the same as in Lemma 2.2.4*. Then*

$$\hat{\mathcal{E}}_h^{n+1/2} \geq \frac{\lambda}{2}\left\|\delta u_h^{n+1/2}\right\|_{L^2(\Omega)}^2 + \frac{1}{4}\left[a(u_h^{n+1}, u_h^{n+1}) + a(u_h^n, u_h^n)\right] \geq 0. \tag{3.11}$$

(3.10) *is referred to as the* CFL condition.

Proof. By assumption, X_h has the inverse property. We begin by rewriting the second

term from (3.8) in terms of symmetric arguments, before applying Lemma 2.2.4:

$$\begin{aligned} 2a(u_h^n, u_h^{n+1}) &= a(u_h^{n+1}, u_h^{n+1}) + a(u_h^n, u_h^n) - a(u_h^{n+1} - u_h^n, u_h^{n+1} - u_h^n) \\ &= a(u_h^{n+1}, u_h^{n+1}) + a(u_h^n, u_h^n) - (\Delta t)^2 a(\delta u_h^{n+1/2}, \delta u_h^{n+1/2}) \\ &\geq a(u_h^{n+1}, u_h^{n+1}) + a(u_h^n, u_h^n) - \frac{C_{\text{inv}}(\Delta t)^2}{h^2} \left\| \delta u_h^{n+1/2} \right\|_{L^2(\Omega)}^2 . \end{aligned}$$

Inserting this into (3.8) we get

$$\begin{aligned} \hat{\mathcal{E}}_h^{n+1/2} &\geq \frac{1}{2} \left(1 - \frac{C_{\text{inv}}(\Delta t)^2}{2h^2} \right) \left\| \delta u_h^{n+1/2} \right\|_{L^2(\Omega)}^2 + \frac{1}{4} \left[a(u_h^{n+1}, u_h^{n+1}) + a(u_h^n, u_h^n) \right] \\ &\geq \frac{\lambda}{2} \left\| \delta u_h^{n+1/2} \right\|_{L^2(\Omega)}^2 + \frac{1}{4} \left[a(u_h^{n+1}, u_h^{n+1}) + a(u_h^n, u_h^n) \right] , \end{aligned}$$

which is a sum of non-negative quantities.

□

Theorem 3.2.5 (Energy conservation for the Leapfrog method)
Assume that the conditions in Lemma 3.2.4 *hold. Then, if* $f = 0$*, the Leapfrog method conserves energy in the sense that, for all* $n \geq 0$*,*

$$\hat{\mathcal{E}}_h^{n+1/2} = \hat{\mathcal{E}}_h^{1/2}. \tag{3.12}$$

If $f \neq 0$*, we have, for all* $n \geq 0$*,*

$$\sqrt{\hat{\mathcal{E}}_h^{n+1/2}} \leq \sqrt{\hat{\mathcal{E}}_h^{1/2}} + \sum_{k=1}^{n} \frac{\Delta t}{\sqrt{2\lambda}} \left\| f^k \right\| . \tag{3.13}$$

Remark. The estimate blows up if λ tends to zero. In practical applications, this is problematic as it is often desirable to let λ be close to zero, since it admits larger time steps. An estimate of the energy which does not blow up can be found in [2].

Proof of Theorem 3.2.5. Let $v_h = u_h^{n+1} - u_h^{n-1} = (u_h^{n+1} - u_h^n) + (u_h^n - u_h^{n-1})$ in the Leapfrog weak formulation (3.5). We obtain

$$\left(\frac{u_h^{n+1} - 2u_h^n + u_h^{n-1}}{(\Delta t)^2}, u_h^{n+1} - u_h^{n-1} \right) + a(u_h^n, u_h^{n+1} - u_h^{n-1}) = \left(f^n, u_h^{n+1} - u_h^{n-1} \right) .$$

Expanding the terms we may rewrite this as

$$\begin{aligned} 2\left(\hat{\mathcal{E}}_h^{n+1/2} - \hat{\mathcal{E}}_h^{n-1/2} \right) &= \left\| \delta u_h^{n+1/2} \right\|_{L^2(\Omega)}^2 + a(u_h^n, u_h^{n+1}) - \left\| \delta u_h^{n-1/2} \right\|_{L^2(\Omega)}^2 - a(u_h^{n-1}, u_h^n) \\ &= \left(f^n, u_h^{n+1} - u_h^{n-1} \right) . \end{aligned}$$

If $f = 0$, we have that $f^k = 0$ for all $k \in \mathbb{N}_0$. Summing over the integers $k = 1, \dots, n$ then yields $\hat{\mathcal{E}}_h^{n+1/2} = \hat{\mathcal{E}}_h^{1/2}$, which proves the first part of the theorem. Otherwise, if $f \neq 0$, we continue the above derivation to obtain

$$\begin{aligned} 2\left(\hat{\mathcal{E}}_h^{n+1/2} - \hat{\mathcal{E}}_h^{n-1/2}\right) &= \left(f^n, u_h^{n+1} - u_h^n\right) + \left(f^n, u_h^n - u_h^{n-1}\right) \\ &= \Delta t \left(f^n, \delta u_h^{n+1/2}\right) + \Delta t \left(f^n, \delta u_h^{n-1/2}\right) \\ &\leq \Delta t \left\|f^n\right\| \left(\left\|\delta u_h^{n+1/2}\right\| + \left\|\delta u_h^{n-1/2}\right\|\right). \end{aligned}$$

Since the CFL condition is assumed to be satisfied, we may apply Lemma 3.2.4 to the right side of the above inequality, which yields

$$\hat{\mathcal{E}}_h^{n+1/2} - \hat{\mathcal{E}}_h^{n-1/2} \leq \frac{\Delta t}{\sqrt{2\lambda}} \left\|f^n\right\| \left(\sqrt{\hat{\mathcal{E}}_h^{n+1/2}} + \sqrt{\hat{\mathcal{E}}_h^{n-1/2}}\right).$$

If $\hat{\mathcal{E}}_h^{n+1/2} \neq 0$, we may perform the necessary division to obtain

$$\sqrt{\hat{\mathcal{E}}_h^{n+1/2}} - \sqrt{\hat{\mathcal{E}}_h^{n-1/2}} = \frac{\hat{\mathcal{E}}_h^{n+1/2} - \hat{\mathcal{E}}_h^{n-1/2}}{\sqrt{\hat{\mathcal{E}}_h^{n+1}} + \sqrt{\hat{\mathcal{E}}_h^{n-1/2}}} \leq \frac{\Delta t}{\sqrt{2\lambda}} \left\|f^n\right\|.$$

If, on the other hand, $\hat{\mathcal{E}}_h^{n+1/2} = 0$, it follows trivially from the fact that $\hat{\mathcal{E}}_h^{n-1/2}$ is a non-negative quantity that $\sqrt{\hat{\mathcal{E}}_h^{n+1/2}} \leq \sqrt{\hat{\mathcal{E}}_h^{n-1/2}}$, and the above inequality still holds. Summing over $k = 1, \dots, n$, we're consequently left with

$$\sqrt{\hat{\mathcal{E}}_h^{n+1/2}} \leq \sqrt{\hat{\mathcal{E}}_h^{1/2}} + \sum_{k=1}^{n} \frac{\Delta t}{\sqrt{2\lambda}} \left\|f^k\right\|.$$

□

Theorem 3.2.6 (Stability of the Leapfrog method)
Assume that the conditions in Lemma 3.2.4 *hold. Then the Leapfrog method is stable in the sense that there exists some* $C > 0$ *independent of* h *and* Δt *such that*

$$\begin{aligned} \left\|\delta u_h^{n+1/2}\right\|_{L^2} + \left\|u_h^{n+1}\right\|_{H^1} \leq C\Big(&\left\|\delta u_h^{1/2}\right\|_{L^2} + \left\|u_h^0\right\|_{H^1} + \left\|u_h^1\right\|_{H^1} \\ &+ \sum_{k=1}^{n} \Delta t \left\|f^k\right\|_{L^2} \Big). \end{aligned} \tag{3.14}$$

Proof. We seek to bound the norms on the left-hand side of (3.14) by the discrete energy. Recall the equivalence of norms from Lemma 2.1.2, which lets us obtain, for constants $C_0, C_1 > 0$ independent of h and Δt,

$$\begin{aligned}\left\|\delta u_h^{n+1/2}\right\|_{L^2} + \left\|u_h^{n+1}\right\|_{H^1} &\leq \left\|\delta u_h^{n+1/2}\right\|_{L^2} + C_0\sqrt{a(u_h^{n+1}, u_h^{n+1})} \\ &\leq \sqrt{\frac{2}{\lambda}\hat{\mathcal{E}}_h^{n+1/2}} + C_0\sqrt{4\hat{\mathcal{E}}_h^{n+1/2}} \\ &\leq C_1\sqrt{\hat{\mathcal{E}}_h^{n+1/2}}.\end{aligned}$$

We can write

$$2a(u,v) = a(u,u) + a(v,v) - a(u-v, u-v) \leq a(u,u) + a(v,v),$$

which together with Theorem 3.2.5 and the equivalence of norms from Lemma 2.1.2 gives us

$$\begin{aligned}\sqrt{\hat{\mathcal{E}}_h^{n+1/2}} &\leq \sqrt{\hat{\mathcal{E}}_h^{1/2}} + \sum_{k=1}^{n} \frac{\Delta t}{\sqrt{2\lambda}} \left\|f^k\right\| \\ &= \sqrt{\frac{1}{2}\left\|\delta u_h^{1/2}\right\|^2 + \frac{1}{2}a(u_h^0, u_h^1)} + \sum_{k=1}^{n} \frac{\Delta t}{\sqrt{2\lambda}} \left\|f^k\right\| \\ &\leq \sqrt{\frac{1}{2}\left\|\delta u_h^{1/2}\right\|^2 + \frac{1}{4}a(u_h^0, u_h^0) + \frac{1}{4}a(u_h^1, u_h^1)} + \sum_{k=1}^{n} \frac{\Delta t}{\sqrt{2\lambda}} \left\|f^k\right\| \\ &\leq \sqrt{\frac{1}{2}\left\|\delta u_h^{1/2}\right\|^2 + C_2\left\|u_h^0\right\|_{H^1}^2 + C_2\left\|u_h^1\right\|_{H^1}^2} + \sum_{k=1}^{n} \frac{\Delta t}{\sqrt{2\lambda}} \left\|f^k\right\| \\ &\leq C_3\left(\left\|\delta u_h^{1/2}\right\| + \left\|u_h^0\right\|_{H^1} + \left\|u_h^1\right\|_{H^1} + \sum_{k=1}^{n} \Delta t \left\|f^k\right\|\right)\end{aligned}$$

for suitable constants $C_2, C_3 > 0$ independent of h and Δt. Combining the preceding inequality with the above bound for the norms completes the proof. □

We are now ready to prove error estimates for the Leapfrog method. We will first give an estimate in terms of the elliptic projection defined in Definition 2.2.5 that applies to any finite element space. Afterwards, we will use the error estimates for the elliptic projection in polynomial spaces (Lemma 2.2.7) to determine the expected convergence rate of the method given certain regularity assumptions on the exact solution u.

Theorem 3.2.7 (Error estimates for the Leapfrog method)
Given a finite element space X_h, assume that the following conditions are satisfied:

- $u \in C^4(0,T;H_0^1(\Omega))$.
- *The conditions of Lemma* 3.2.4 *are satisfied.*

Furthermore, let $\epsilon_h^n := u_h^n - R_h u(t^n)$, $r_h := u - R_h u$ and define $e_h^n := u_h^n - u(t^n)$, the error at time $t = n\Delta t$. Then the following error estimate holds for the Leapfrog method:

$$\begin{aligned}\left\|\delta e_h^{n+1/2}\right\|_{L^2} + \left\|e_h^{n+1}\right\|_Q + \|e_h^n\|_Q \leq C \Bigg(& \left\|\delta \epsilon_h^{1/2}\right\|_{L^2} + \left\|\epsilon_h^1\right\|_{H^1} + \left\|\epsilon_h^0\right\|_{H^1} \\ & + (\Delta t)^2 + \sup_{0\leq z\leq T} \|r_h(z)\|_Q \\ & + \sum_{i=1}^{2} \sup_{0\leq z\leq T} \left\| \frac{\partial^i r_h}{\partial t^i}(z) \right\|_{L^2} \Bigg),\end{aligned} \tag{3.15}$$

for $Q = L^2$ and $Q = H^1$, and $C = C(u,T) > 0$ is independent of h and Δt.

Proof. Writing $r_h^n := r_h(t^n) = u(t^n) - R_h u(t^n)$ we can reformulate the error as

$$e_h^n = u_h^n - u(t^n) = [u_h^n - R_h u(t^n)] - [u(t^n) - R_h u(t^n)] = \epsilon_h^n - r_h^n.$$

Consequently, by replacing u_h^n with ϵ_h^n in the Leapfrog weak formulation (3.5) and using the definition of the elliptic projection (2.9), we observe that

$$(\delta^2 \epsilon_h^n, v_h) + a(\epsilon_h^n, v_h) = \overbrace{(\delta^2 u_h^n, v_h) + a(u_h^n, v_h)}^{(f(t^n), v_h)} - (\delta^2 R_h u(t^n), v_h) - \overbrace{a(R_h u(t^n), v_h)}^{a(u(t^n), v_h)}.$$

By using (2.1) to replace the term involving f, we get

$$\begin{aligned}(\delta^2 \epsilon_h^n, v_h) + a(\epsilon_h^n, v_h) &= (\ddot{u}(t^n) - \delta^2 R_h u(t^n), v_h) \\ &= (\ddot{u}(t^n) - \delta^2 u(t^n) + \delta^2 u(t^n) - \delta^2 R_h u(t^n), v_h) \\ &= (\tau^n + \delta^2 r_h^n, v_h),\end{aligned}$$

where $\tau^n := \ddot{u}(t^n) - \delta^2 u(t^n)$ is the discretization error from the second order central difference operator. We may now use our stability result from Theorem 3.2.6 to bound

the error term ϵ_h^n:

$$\begin{aligned}\left\|\delta\epsilon_h^{n+1/2}\right\|_{L^2} + \left\|\epsilon_h^{n+1}\right\|_{L^2} + \|\epsilon_h^n\|_{L^2} &\leq \left\|\delta\epsilon_h^{n+1/2}\right\|_{L^2} + \left\|\epsilon_h^{n+1}\right\|_{H^1} + \|\epsilon_h^n\|_{H^1} \\ &\leq C_1\Bigg(\left\|\delta\epsilon_h^{1/2}\right\|_{L^2} + \left\|\epsilon_h^1\right\|_{H^1} + \left\|\epsilon_h^0\right\|_{H^1} \\ &\qquad + \Delta t \sum_{k=1}^{n} \left\|\tau^k + \delta^2 r_h^k\right\|_{L^2} \Bigg).\end{aligned}$$

The forth term needs a little work. The triangle inequality lets us easily treat τ^k and $\delta^2 r_h^k$ separately. We begin with τ^k, and from (3.4) we have, for some $t_\tau \in (t^{k-1}, t^{k+1})$,

$$\|\tau^k\| = \left\| -\frac{(\Delta t)^2}{12}\frac{\partial^4 u}{\partial t^4}(t_\tau)\right\| \leq \frac{(\Delta t)^2}{12} \sup_{0\leq z\leq T}\left\|\frac{\partial^4 u}{\partial t^4}(z)\right\|$$

Next, we may again use (3.4) to obtain the following estimate, where $t_r \in (t^{k-1}, t^{k+1})$:

$$\begin{aligned}\left\|\delta^2 r_h^k\right\|_{L^2} = \left\|\delta^2 r_h(t^k)\right\|_{L^2} &= \left\|\frac{\partial^2 r_h}{\partial t^2}(t^k) + \frac{(\Delta t)^2}{12}\frac{\partial^4 r_h}{\partial t^4}(t_r)\right\|_{L^2} \\ &\leq \left\|\frac{\partial^2 r_h}{\partial t^2}(t^k)\right\| + \frac{(\Delta t)^2}{12}\sup_{0\leq z\leq T}\left\|\frac{\partial^4 r_h}{\partial t^4}(z)\right\|_{L^2}.\end{aligned}$$

It is easily seen that the definition of R_h leads to, for any $t^* \in [0, T]$,

$$\left\|\frac{\partial^4 r_h}{\partial t^4}(t^*)\right\|_{L^2} \leq \left\|R_h\frac{\partial^4 u}{\partial t^4}(t^*)\right\|_{H^1} + \left\|\frac{\partial^4 u}{\partial t^4}(t^*)\right\|_{H^1} \leq 2\left\|\frac{\partial^4 u}{\partial t^4}(t^*)\right\|_{H^1},$$

which when combined with the above yields

$$\left\|\delta^2 r_h^k\right\|_{L^2} \leq \sup_{0\leq z\leq T}\left\|\frac{\partial^2 r_h}{\partial t^2}(z)\right\| + \frac{(\Delta t)^2}{6}\sup_{0\leq z\leq T}\left\|\frac{\partial^4 u}{\partial t^4}(z)\right\|_{H^1}.$$

Denoting $N_t = T/\Delta t$ the number of time steps, the preceding results let us write

$$\begin{aligned}\Delta t\sum_{k=1}^{n}\left\|\tau^k + \delta^2 r_h^k\right\|_{L^2} &\leq \Delta t\sum_{k=1}^{N_t}\left\|\tau^k + \delta^2 r_h^k\right\|_{L^2} \\ &\leq \cancel{\Delta t}\,\frac{T}{\cancel{\Delta t}}\left(\sup_{0\leq z\leq T}\left\|\frac{\partial^2 r_h}{\partial t^2}(z)\right\|_{L^2} + \frac{(\Delta t)^2}{4}\sup_{0\leq z\leq T}\left\|\frac{\partial^4 u}{\partial t^4}(z)\right\|_{H^1}\right)\end{aligned}$$

Having obtained estimates for ϵ_h, we turn to the task of estimating $\left\|\delta r_h^{n+1/2}\right\|$. Using a procedure analogous to the one we used to estimate $\left\|\delta^2 r_h^k\right\|$ along with the finite difference estimate (3.3), we have for $t_r \in (t^n, t^{n+1})$,

$$\begin{aligned}\left\|\delta r_h^{n+1/2}\right\|_{L^2} &= \left\|\frac{\partial r_h}{\partial t}(t^{n+1/2}) + \frac{(\Delta t)^2}{24}\frac{\partial^3 r_h}{\partial t^3}(t_r)\right\|_{L^2} \\ &\leq \sup_{0\leq z\leq T}\left\|\frac{\partial r_h}{\partial t}(z)\right\|_{L^2} + \frac{(\Delta t)^2}{12}\sup_{0\leq z\leq T}\left\|\frac{\partial^3 u}{\partial t^3}(z)\right\|_{H^1}.\end{aligned} \tag{3.16}$$

Finally, we collect all our partial results to conclude the theorem:

$$\begin{aligned}\left\|\delta e_h^{n+1/2}\right\| + \left\|e_h^{n+1}\right\|_{L^2} + \|e_h^n\|_{L^2} \leq{}& \left\|\delta \epsilon_h^{n+1/2}\right\| + \left\|\epsilon_h^{n+1}\right\|_{L^2} + \|\epsilon_h^n\|_{L^2} \\ &+ \left\|\delta r_h^{n+1/2}\right\| + \left\|r_h^{n+1}\right\|_{L^2} + \|r_h^n\|_{L^2},\end{aligned}$$

for the L^2 norms, and

$$\begin{aligned}\left\|e_h^{n+1}\right\|_{H^1} + \|e_h^n\|_{H^1} \leq{}& \left\|\epsilon_h^{n+1}\right\|_{H^1} + \|\epsilon_h^n\|_{H^1} \\ &+ \left\|r_h^{n+1}\right\|_{H^1} + \|r_h^n\|_{H^1},\end{aligned}$$

for the H^1 norms.

□

Corollary 3.2.8 (Error estimates for the Leapfrog method in polynomial spaces) *Given a finite element space $X_h = S_0^p(\mathcal{T}_h)$, assume that the following conditions are met:*

- $u \in C^4(0,T;H_0^1(\Omega)) \cap C^2(0,T;H^s(\Omega))$, *for some* $s \in [2, p+1]$.
- *The conditions of Lemma* 3.2.4 *are satisfied.*
- u_h^0 *and* u_h^1 *are chosen such that for some* $C_0 > 0$ *independent of* h *and* Δt

$$\left\|\delta \epsilon_h^{1/2}\right\|_{L^2} + \left\|\epsilon_h^0\right\|_{H^1} + \left\|\epsilon_h^1\right\|_{H^1} \leq C_0(h^s + (\Delta t)^2), \tag{3.17}$$

  where $\epsilon_h^n = u_h^n - R_h u(t^n)$.

Then the following error estimates hold for the Leapfrog method:

$$\left\|\delta e_h^{n+1/2}\right\|_{L^2} + \left\|e_h^{n+1}\right\|_{L^2} + \|e_h^n\|_{L^2} \leq C\left(h^s + (\Delta t)^2\right), \tag{3.18}$$

$$\left\|e_h^{n+1}\right\|_{H^1} + \|e_h^n\|_{H^1} \leq C\left(h^{s-1} + (\Delta t)^2\right), \tag{3.19}$$

where $e_h^n = u_h^n - u(t^n)$ denotes the error at time $t = n\Delta t$ and $C = C(u,T) > 0$ is independent of h and Δt.

Remark. One can obtain the same results or similar with relaxed regularity assumptions on u. However, the theorem as presented here is sufficient for the purposes of this thesis while avoiding additional tedious technicalities.

Proof of Corollary 3.2.8. This result is a straightforward consequence of Theorem 3.2.7 along with the the given assumptions. Because $u, \dot{u}, \ddot{u} \in H^s(\Omega)$, the error estimates for the elliptic projection found in 2.2.7 can be applied to the terms involving the elliptic projection in Theorem 3.2.7. □

3.3 The Crank-Nicolson method

In this subsection we will derive an implicit method for the wave equation that closely resembles the well-known Crank Nicolson method for parabolic partial differential equations. We will follow the recipe of the derivation of the properties for the Leapfrog method, in that we will first define a functional-analytic definition followed by an algebraic formulation. We will go on to show that the method conserves a discrete energy quantity with no restrictions on the time step Δt, which leads to unconditional stability. Finally, we will derive error estimates and formulate the results in a similar fashion as for the Leapfrog method.

We note first that for a scalar, parabolic PDE

$$\frac{\partial u}{\partial t} = F(x, t, u, \nabla u, \Delta u),$$

a discretization in time similar to the Crank-Nicolson method [7] gives

$$\frac{u^{n+1} - u^n}{\Delta t} = \frac{1}{2}\left[F(x, t^{n+1}, u^{n+1}, \nabla u^{n+1}, \Delta u^{n+1}) + F(x, t^n, u^n, \nabla u^n, \Delta u^n)\right]$$

Next, consider the scalar wave equation $u_{tt} - \Delta u = f$. Let $w = u_t$ and $U = [u, w]^T$ to obtain the vector-form PDE

$$\begin{aligned} \frac{\partial}{\partial t}\begin{pmatrix} u \\ w \end{pmatrix} &= \begin{pmatrix} 0 \\ f \end{pmatrix} + \begin{pmatrix} w \\ c^2 \Delta u \end{pmatrix} \\ \implies \frac{\partial U}{\partial t} &= F(x, t, U, \Delta U). \end{aligned}$$

In this form we apply the vector-equivalent of the above scalar Crank-Nicolson method for two consecutive values of n to obtain

$$\begin{aligned} u^{n+1} - u^n &= \frac{\Delta t}{2}\left(w^{n+1} + w^n\right) \\ w^{n+1} - w^n &= \frac{\Delta t}{2}\left(f^{n+1} + c^2 \Delta u^{n+1} + f^n + c^2 \Delta u^n\right) \\ u^n - u^{n-1} &= \frac{\Delta t}{2}\left(w^n + w^{n-1}\right) \\ w^n - w^{n-1} &= \frac{\Delta t}{2}\left(f^n + c^2 \Delta u^n + f^{n-1} + c^2 \Delta u^{n-1}\right). \end{aligned}$$

Finally, forming a recurrence relation by combining the above equations and then taking the inner product with a test function $v_h \in X_h$ motivates the following definition.

Definition 3.3.1 (Crank-Nicolson method for the wave equation)
Given a finite element space $X_h \subseteq H_0^1(\Omega)$, the weak formulation of the fully discretized Crank-Nicolson method is for $t = n\Delta t$ and integer $n \geq 1$ given by

$$\left(\frac{u_h^{n+1} - 2u_h^n + u_h^{n-1}}{(\Delta t)^2}, v_h\right) + \frac{1}{4}a(u_h^{n+1} + 2u_h^n + u_h^{n-1}, v_h) = \frac{1}{4}\left(f^{n+1} + 2f^n + f^{n-1}, v_h\right) \tag{3.20}$$

for all $v_h \in X_h$.

For practical implementation, we require a different representation that is better suited to computation, which the next lemma provides.

Lemma 3.3.2 (Algebraic formulation of the Crank-Nicolson method)
Let u_h^n be defined as in Definition 3.3.1 *and let $\Lambda = \{\lambda_i \mid i = 1, \ldots, N_h\}$ be a basis of X_h, with $N_h = \dim X_h$. Define $\xi^n \in \mathbb{R}^{N_h}$ such that*

$$u_h^n = \sum_{j=1} \xi_j^n \lambda_j. \tag{3.21}$$

Then the Crank-Nicolson method is for $n \geq 1$ equivalent to the linear system

$$\left(M + \frac{(\Delta t)^2}{4}A\right)\xi^{n+1} = \frac{(\Delta t)^2}{4}(b^{n+1} + 2b^n + b^{n-1}) + M(2\xi^n - \xi^{n-1}) - \frac{(\Delta t)^2}{4}A(2\xi^n + \xi^{n-1}), \tag{3.22}$$

where the system matrices M and A are defined by Definition 2.2.9 *and the load vector $b^n := b(t^n)$ by Definition* 2.2.10.

Proof. Let $v_h = \lambda_i$ in (3.20) and insert (3.21). □

We note that the coefficient matrix $M + (\Delta t)^2 A/4$ associated with the Crank-Nicolson method is in general not so easy to invert as the mass matrix which needs to be inverted for the Leapfrog method. However, in the experience of the author, the method is competitive to the Leapfrog method without mass lumping when the computational mesh is quasi-uniform and the time step is sufficiently small (as a rule of thumb, within the region of stability for the Leapfrog method).

From experience with the Crank-Nicolson method applied to parabolic equations, we expect it to be unconditionally stable and quadratically convergent given sufficient regularity of the solution and a sufficiently accurate spatial discretization. Indeed, we will shortly confirm these properties. Before we can prove stability for the Crank-Nicolson method, however, we need a suitable concept for discrete energy.

Definition 3.3.3 (Discrete energy for the Crank-Nicolson method)
The discrete energy for the Crank-Nicolson method is defined by

$$\bar{\mathcal{E}}_h^{n+1/2} := \frac{1}{2}\left\|\delta u_h^{n+1/2}\right\|^2 + \frac{1}{2}a(u_h^{n+1/2}, u_h^{n+1/2}), \tag{3.23}$$

where $\delta u_h^{n+1/2} = (u_h^{n+1} - u_h^n)/\Delta t$ and $u_h^{n+1/2} = (u_h^{n+1} + u_h^n)/2$.

Remark. Unlike the discrete energy for the Leapfrog method (Definition 3.2.3), the discrete energy for Crank-Nicolson is unconditionally non-negative. Indeed, as we will see in the following theorem, this is the reason for its unconditional stability.

Theorem 3.3.4 (Energy conservation for the Crank-Nicolson method)
If $f = 0$, the Crank-Nicolson method conserves energy in the sense that, for all $n \geq 0$,

$$\bar{\mathcal{E}}_h^{n+1/2} = \bar{\mathcal{E}}_h^{1/2}. \tag{3.24}$$

If $f \neq 0$, we have, for all $n \geq 0$,

$$\sqrt{\bar{\mathcal{E}}_h^{n+1/2}} \leq \sqrt{\bar{\mathcal{E}}_h^{1/2}} + \sum_{k=1}^{n} \frac{\Delta t}{\sqrt{2}} \left\| \frac{1}{4}(f^{k+1} + 2f^k + f^{k-1}) \right\|. \tag{3.25}$$

Proof. Choose v_h in (3.20) such that

$$v_h = u_h^{n+1} - u_h^{n-1} = (u_h^{n+1} - u_h^n) + (u_h^n - u_h^{n-1}) = (u_h^{n+1} + u_h^n) - (u_h^n + u_h^{n-1}),$$

and rewrite the first term on the left side as

$$\left(\frac{(u_h^{n+1} - u_h^n) - (u_h^n - u_h^{n-1})}{(\Delta t)^2}, (u_h^{n+1} - u^n) + (u^n - u^{n-1}) \right) = \left\|\delta u_h^{n+1/2}\right\|^2 - \left\|\delta u_h^{n-1/2}\right\|^2,$$

and the second term as

$$\frac{1}{4}a\left[(u_h^{n+1} + u_h^n) + (u_h^n + u_h^{n-1}), (u_h^{n+1} + u_h^n) - (u_h^n + u_h^{n-1})\right] = a(u_h^{n+1/2}, u_h^{n+1/2}) - a(u_h^{n-1/2}, u_h^{n-1/2}).$$

This allows us to rewrite the left-hand side in terms of the difference in the discrete energy, and we get

$$2(\bar{\mathcal{E}}_h^{n+1/2} - \bar{\mathcal{E}}_h^{n-1/2}) = \left(\frac{1}{4}(f^{n+1} + 2f^n + f^{n-1}), (u_h^{n+1} - u_h^n) + (u_h^n - u_h^{n-1})\right).$$

If $f = 0$, then $f^n = 0$ for all $n \in \mathbb{N}$, so it follows that $\bar{\mathcal{E}}_h^{n+1/2} = \bar{\mathcal{E}}_h^{n-1/2}$, and summing over $k = 1, \dots, n$ proves the first part of the theorem. Conversely, if $f \neq 0$, we bound the right-side of the above equation to obtain

$$\begin{aligned} 2(\bar{\mathcal{E}}_h^{n+1/2} - \bar{\mathcal{E}}_h^{n-1/2}) &\leq \sqrt{2}\Delta t \left\| \frac{1}{4}(f^{n+1} + 2f^n + f^{n-1}) \right\| \left(\frac{\left\| \delta u_h^{n+1/2} \right\|}{\sqrt{2}} + \frac{\left\| \delta u_h^{n-1/2} \right\|}{\sqrt{2}} \right) \\ &\leq \sqrt{2}\Delta t \left\| \frac{1}{4}(f^{n+1} + 2f^n + f^{n-1}) \right\| \left(\sqrt{\bar{\mathcal{E}}_h^{n+1/2}} + \sqrt{\bar{\mathcal{E}}_h^{n-1/2}} \right). \end{aligned}$$

From here, we make the same kind of argument as in the proof of Theorem 3.2.5 and proceed in almost exactly the same way to complete the proof. □

Theorem 3.3.5 (Stability of the Crank-Nicolson method)
The Crank-Nicolson method from Definition 3.3.1 *is stable in the sense that*

$$\begin{aligned} \left\| \delta u_h^{n+1/2} \right\|_{L^2} + \left\| u_h^{n+1/2} \right\|_{H^1} \leq C\Big(&\left\| \delta u_h^{1/2} \right\|_{L^2} + \left\| u_h^{1/2} \right\|_{H^1} \\ &+ \Delta t \sum_{k=1}^{n} \left\| \frac{1}{4}(f^{k+1} + 2f^k + f^{k-1}) \right\|_{L^2} \Big). \end{aligned} \tag{3.26}$$

for some $C > 0$ independent of h and Δt.

Remark. The theorem only bounds the H^1 norm of the average of two consecutive time steps, which is in some sense slightly weaker than what we proved for the Leapfrog method.

Remark. Note that we do not need to assume that the finite element space X_h has the inverse property, as we did for the stability of the Leapfrog method.

Proof of Theorem 3.3.5. From the equivalence of norms in Lemma 2.1.2 and the energy conservation from Theorem 3.3.4, it follows directly that, for suitable constants

$C_1, C_2, C > 0$,

$$\begin{aligned}\left\|\delta u_h^{n+1/2}\right\| + \left\|u_h^{n+1/2}\right\|_{H^1} &\leq \left\|\delta u_h^{n+1/2}\right\| + C_1\sqrt{a(u_h^{n+1/2}, u_h^{n+1/2})} \\ &\leq C_2\sqrt{\bar{\mathcal{E}}_h^{n+1/2}} \\ &\leq C_2\left(\sqrt{\bar{\mathcal{E}}_h^{1/2}} + \frac{\Delta t}{\sqrt{2}}\sum_{k=1}^{n}\left\|\frac{1}{4}(f^{k+1} + 2f^k + f^{k-1})\right\|.\right) \\ &\leq C\Big(\left\|\delta u_h^{1/2}\right\| + \left\|u_h^{1/2}\right\|_{H^1} \\ &\qquad + \Delta t\sum_{k=1}^{n}\left\|\frac{1}{4}(f^{k+1} + 2f^k + f^{k-1})\right\|.\Big),\end{aligned}$$

which concludes the proof.

□

We are finally ready to give error estimates for the Crank-Nicolson method. As in the case of the Leapfrog method, we first give estimates in terms of the error of the the elliptic projection, and then proceed to give concrete results for polynomial spaces under appropriate regularity assumptions.

Theorem 3.3.6 (Error estimates for the Crank-Nicolson method)
Given a finite element space $X_h \subseteq H_0^1(\Omega)$, assume that the solution u satisfies $u \in C^4(0, T; H_0^1(\Omega))$. Furthermore, let $\epsilon_h^n := u_h^n - R_h u(t^n)$, $r_h := u - R_h u$ and define $e_h^n := u_h^n - u(t^n)$, the error at time $t^n = n\Delta t$. Then the following error estimate holds for the Crank-Nicolson method:

$$\begin{aligned}\left\|\delta e_h^{n+1/2}\right\|_{L^2} + \left\|e_h^{n+1/2}\right\|_Q \leq C\Bigg(&\left\|\delta\epsilon_h^{1/2}\right\|_{L^2} + \left\|\epsilon_h^{1/2}\right\|_{H^1} + (\Delta t)^2 \\ &+ \sup_{0\leq z\leq T}\|r_h(z)\|_Q + \sum_{i=1}^{2}\sup_{0\leq z\leq T}\left\|\frac{\partial^i r_h}{\partial t^i}(z)\right\|_{L^2}\Bigg),\end{aligned} \tag{3.27}$$

for $Q = L^2$ and $Q = H^1$, $e_h^{n+1/2} = (e_h^{n+1} + e_h^n)/2$, and $C = C(u, T) > 0$ is independent of h and Δt.

Proof. Our method of proof is deliberately very similar to the proof of convergence for the Leapfrog method, presented in Theorem 3.2.7, but we must make some modifications. As in the case of the Leapfrog method, write

$$e_h^n = u_h^n - u(t^n) = [u_h^n - R_h u(t^n)] - [u(t^n) - R_h u(t^n)] = \epsilon_h^n - r_h(t^n).$$

Insert ϵ_h^n into the left hand side of the Crank-Nicolson weak formulation (3.20) in place of u_h^n and leverage the discrete weak formulation of u_h^n (3.20), as well as the elliptic projection property (2.9) and the weak formulation of u (2.1) to obtain

$$\begin{aligned}
\big(\delta^2\epsilon_h^n, v_h\big) &+ \frac{1}{4}a(\epsilon_h^{n+1} + 2\epsilon_h^n + \epsilon_h^{n-1}, v_h) \\
&= \overbrace{(\delta^2 u_h^n, v_h) + \frac{1}{4}a\Big(u_h^{n+1} + 2u_h^n + u_h^{n-1}, v_h\Big)}^{\left(\frac{1}{4}(f^{n+1}+2f^n+f^{n-1}),v_h\right)} - \big(\delta^2 R_h u(t^n), v_h\big) \\
&\quad - \underbrace{\frac{1}{4}a\Big(R_h\big(u(t^{n+1}) + 2u(t^n) + u(t^{n-1})\big), v_h\Big)}_{\left(\frac{1}{4}(f^{n+1}+2f^n+f^{n-1}),v_h\right) - \frac{1}{4}(\ddot{u}(t^{n+1})+2\ddot{u}(t^n)+\ddot{u}(t^{n-1}),v_h)} \\
&= \frac{1}{4}\big(\ddot{u}(t^{n+1}) + 2\ddot{u}(t^n) + \ddot{u}(t^{n-1}), v_h\big) - \big(\delta^2 R_h u(t^n), v_h\big) \\
&= \big(\ddot{u}(t^n) - \delta^2 u(t^n), v_h\big) + \big(\delta^2 u(t^n) - \delta^2 R_h u(t^n), v_h\big) \\
&\quad + \frac{1}{4}\big(\ddot{u}(t^{n+1}) - 2\ddot{u}(t^n) + \ddot{u}(t^{n-1}), v_h\big) \\
&= \big(\tau^n + \delta^2 r_h^n + \frac{\Delta t}{4}[\delta^+\ddot{u}(t^n) - \delta^-\ddot{u}(t^n)], v_h\big).
\end{aligned}$$

Here we have used the short-hand notation $\delta^2\epsilon_h^n$ for the second-order central difference operator from (3.4), as well as the forward and backward operators from (3.1) and (3.2). As in the proof of convergence for the Leapfrog method, $\tau^n = \ddot{u}(t^n) - \delta^2 u(t^n)$ is the time discretization error of the second-order central difference operator. We may now use the main result of Theorem 3.3.5 to immediately obtain

$$\begin{aligned}
\left\|\delta\epsilon_h^{n+1/2}\right\|_{L^2} + \left\|\epsilon_h^{n+1/2}\right\|_{H^1} \leq C_1\Big(&\left\|\delta\epsilon_h^{1/2}\right\|_{L^2} + \left\|\epsilon_h^{1/2}\right\|_{H^1} \\
&+ \Delta t\sum_{k=1}^{n}\left\|\tau^k + \delta^2 r_h^k + \frac{\Delta t}{4}[\delta^+\ddot{u}(t^k) - \delta^-\ddot{u}(t^k)]\right\|_{L^2}\Big).
\end{aligned}$$

We can use the triangle inequality to bound the third term. From the proof of convergence for the Leapfrog method, we have that

$$\Delta t\sum_{k=1}^{n}\left\|\tau^n + \delta^2 r_h^n\right\| \leq T\left(\sup_{0\leq z\leq T}\left\|\frac{\partial^2 r_h}{\partial t^2}(z)\right\|_{L^2} + \frac{(\Delta t)^2}{4}\sup_{0\leq z\leq T}\left\|\frac{\partial^4 u}{\partial t^4}(z)\right\|_{H^1}\right).$$

Next, we use the properties of the forward and backward finite operators from (3.1) and (3.2), as well as the intermediate value theorem, to write, for some $t_+, t_-, t_c \in$

(t^{n-1}, t^{n+1}),

$$\begin{aligned}\frac{\Delta t}{4}\left\|\delta^+\ddot{u}(t^n) - \delta^-\ddot{u}(t^n)\right\| &= \frac{\Delta t}{4}\left\|\frac{\partial^3 u}{\partial t^3}(t^n) + \frac{\Delta t}{2}\frac{\partial^4 u}{\partial t^4}(t_+) - \frac{\partial^3 u}{\partial t^3}(t^n) + \frac{\Delta t}{2}\frac{\partial^4 u}{\partial t^4}(t_-)\right\| \\ &= \frac{(\Delta t)^2}{4}\left\|\frac{\partial^4 u}{\partial t^4}(t_c)\right\| \\ &\leq \frac{(\Delta t)^2}{4}\sup_{0\leq z\leq T}\left\|\frac{\partial^4 u}{\partial t^4}(z)\right\|.\end{aligned}$$

Employing a similar procedure as used in the Leapfrog convergence proof, we obtain an appropriate estimate in the following form, where we have denoted $N_t = T/\Delta t$ the number of time steps,

$$\Delta t\sum_{k=1}^{n}\left\|\frac{\Delta t}{4}[\delta^+\ddot{u}(t^k) - \delta^-\ddot{u}(t^k)]\right\|_{L^2} \leq \cancel{\Delta t}\frac{T}{\cancel{\Delta t}}\frac{(\Delta t)^2}{4}\sup_{0\leq z\leq T}\left\|\frac{\partial^4 u}{\partial t^4}(z)\right\|_{L^2}.$$

What remains is to estimate the error terms involving r_h. Noting that

$$\left\|r_h^{n+1/2}\right\|_Q \leq \frac{1}{2}\Big(\left\|r_h^{n+1}\right\|_Q + \left\|r_h^n\right\|_Q\Big) \leq \sup_{0\leq z\leq T}\|r_h(z)\|_Q$$

for $Q = L^2$ and $Q = H^1$, we obtain the necessary estimate for $\delta r_h^{n+1/2}$ from (3.16). To conclude the proof, one only needs to combine the partial estimates to arrive at the desired result. □

Corollary 3.3.7 (Error estimates in polynomial spaces)
Given a finite element space $X_h = S_0^p(\mathcal{T}_h)$, assume that the following conditions are met:

- $u \in C^4(0,T;H_0^1(\Omega)) \cap C^2(0,T;H^s(\Omega))$, *for some* $s \in [2, p+1]$.
- u_h^0 *and* u_h^1 *are chosen such that*

$$\left\|\delta\epsilon_h^{1/2}\right\|_{L^2} + \left\|\epsilon_h^{1/2}\right\|_{H^1} \leq C_0(h^s + (\Delta t)^2), \tag{3.28}$$

  where $\epsilon_h^n = u_h^n - R_h u(t^n)$ *for integer* n, $\epsilon_h^{1/2} = (\epsilon_h^1 + \epsilon_h^0)/2$, *and* $C_0 > 0$ *is a constant independent of* h *and* Δt.

Then the following error estimate holds for the Crank-Nicolson method:

$$\left\|\delta e_h^{n+1/2}\right\|_{L^2} + \left\|e_h^{n+1/2}\right\|_{L^2} + h\left\|e_h^{n+1/2}\right\|_{H^1} \leq C(u,T)\left(h^s + (\Delta t)^2\right), \tag{3.29}$$

where $e_h^n = u_h^n - u(t^n)$ *denotes the error at time* $t = n\Delta t$ *for integer* n, $e_h^{n+1/2} = (e_h^{n+1} + e_h^n)/2$, *and* $C(u,T) > 0$ *is independent of* h *and* Δt.

Proof. The result is a direct consequence of Theorem 3.3.6, along with the assumptions and error estimates for the elliptic projection (Lemma 2.2.7) applied to u, $\dot{u}$ and $\ddot{u}$. □

3.4 Mass lumping

It turns out that there exists a very simple technique which drastically increases the computational efficiency of the Leapfrog method. Under certain conditions, the procedure of *mass lumping* replaces the mass matrix M by a diagonal matrix which sufficiently accurately approximates the application of M. A short explanation of the main idea can be found in [2], including the procedure for piecewise linear elements. For a more thorough treatment which includes the mass lumping of higher-order finite element spaces, see [22]. Diving into the details is beyond the scope of this thesis, but the main result for linear piecewise elements will be of significant importance for our numerical experiments and subsequent discussion of efficiency.

Lemma 3.4.1 (Mass lumping)
Given a Lagrangian basis for the piecewise linear finite element space $S_0^1(\mathcal{T}_h)$, the mass matrix M can be approximated by a diagonal matrix D whose elements are defined by

$$D_{ii} = \sum_{j=1}^{N_h} M_{ij} \qquad i = 1, \dots, N_h. \tag{3.30}$$

The resulting modification to the Leapfrog method satisfies the same convergence properties as the standard Leapfrog method presented in Definition 3.2.1.

The above lemma paves the way for a modified Leapfrog method which is *fully explicit* in the sense that it merely requires elementary matrix-vector operations and the inversion of a diagonal matrix.

Definition 3.4.2 (Lumped Leapfrog)
Let u_h^n and ξ^n be defined as in Lemma 3.2.2, and assume that Λ is a *Lagrangian* basis for $S_0^1(\mathcal{T}_h)$. Then the mass-lumped Leapfrog method is for $n \geq 1$ equivalent to the linear system

$$\xi^{n+1} = 2\xi^n - \xi^{n-1} + (\Delta t)^2 D^{-1} \left(b^n - A\xi^n \right), \tag{3.31}$$

where the stiffness matrix A is defined by Definition 2.2.9, load vector $b^n := b(t^n)$ by Definition 2.2.10 and D is the diagonal matrix defined in Lemma 3.4.1.

We conclude the section by noting that the mass lumping modification only works for

the Leapfrog method, because it only requires inversion of the mass matrix M. The matrix that needs to be inverted at every step for Crank-Nicolson does not admit a similar technique to be used.

3.5 Initialization: Taking the first step

Until now, we have defined numerical methods that depend on values of u_h^0 and u_h^1 in order to compute u_h^2. In order to form complete numerical methods, we need to complete the methods we have presented so far by defining a way of computing these values. We will refer to this process as *initialization*.

The convergence results for the Leapfrog method in Theorem 3.2.7 and the Crank-Nicolson method in Theorem 3.3.6 suggest that somehow approximating our initial data with the elliptic projection R_h may be a feasible way to approach the problem of initialization, which motivates the following lemma.

Lemma 3.5.1 (Initialization by elliptic projection)
If $u \in C^3(0,T;H_0^1(\Omega))$ and u_h^0 and u_h^1 are chosen such that

$$u_h^0 = R_h u_0, \qquad u_h^1 = R_h\Big(u_0 + \Delta t v_0 + \frac{(\Delta t)^2}{2}\ddot{u}(0)\Big), \tag{3.32}$$

where R_h is the elliptic projection from Definition 2.2.5, then the following estimate holds for some constant $C = C(u,T) > 0$ independent of h and Δt:

$$\left\|\delta\epsilon_h^{1/2}\right\|_{L^2} + \left\|\epsilon_h^{1/2}\right\|_{H^1} \leq \left\|\delta\epsilon_h^{1/2}\right\|_{L^2} + \left\|\epsilon_h^0\right\|_{H^1} + \left\|\epsilon_h^1\right\|_{H^1} \leq C(u,T)(\Delta t)^2, \tag{3.33}$$

with $\epsilon_h^n = u_h^n - R_h u(t^n)$ and $\epsilon_h^{n+1/2} = (\epsilon_h^{n+1} + \epsilon_h^n)/2$.

Proof. Obviously, $\left\|\epsilon_h^0\right\| = 0$. Using the equivalence of norms from Lemma 2.1.2 and the definition of the elliptic projection in Definition 2.2.5, we first note that

$$\|R_h v\|_{H^1} \leq C_0 \|v\|_{H^1}$$

for some $C_0 > 0$. Using this result, along with a Taylor expansion of $u(\Delta t)$, we have

that, for some $t_* \in (0, \Delta t)$,

$$\begin{aligned}
\left\|\epsilon_h^1\right\|_{H^1} &= \left\|u_h^1 - R_h u(\Delta t)\right\|_{H^1} \\
&= \left\| R_h\left(u_0 + \Delta t v_0 + \frac{(\Delta t)^2}{2}\ddot{u}(0)\right)\right. \\
&\qquad \left. - R_h\left(u_0 + \Delta t v_0 + \frac{(\Delta t)^2}{2}\ddot{u}(0) + \frac{(\Delta t)^3}{6}\frac{\partial^3 u}{\partial t^3}(t_*)\right)\right\|_{H^1} \\
&= \left\| R_h\left(\frac{(\Delta t)^3}{6}\frac{\partial^3 u}{\partial t^3}(t_*)\right)\right\|_{H^1} \\
&\leq C_1(\Delta t)^3 \sup_{0\leq z\leq T}\left\|\frac{\partial^3 u}{\partial t^3}(z)\right\|_{H^1} = C_2(u,T)(\Delta t)^3.
\end{aligned}$$

Finally, we use the above results to write

$$\left\|\delta\epsilon_h^{1/2}\right\|_{L^2} = \left\|\frac{\epsilon_h^1 - \epsilon_h^0}{\Delta t}\right\|_{L^2} = \frac{1}{\Delta t}\left\|\epsilon_h^1\right\|_{L^2} \leq \frac{1}{\Delta t}\left\|\epsilon_h^1\right\|_{H^1} = C_2(u,T)(\Delta t)^2,$$

which concludes the proof. □

Evidently, the initialization procedure defined in the previous lemma satisfies the requirements prescribed by the convergence theorems for the Leapfrog method (Corollary 3.2.8) and the Crank-Nicolson method (Corollary 3.3.7) in polynomial spaces.

We close this chapter by noting that while the initialization procedure defined in Lemma 3.5.1 yields optimal convergence rates when paired with the Leapfrog or Crank-Nicolson methods, it is very impractical to implement, particularly because it requires knowledge of the spatial derivatives of u_0 and v_0. However, in practice the nodal interpolator is often sufficiently accurate.

Chapter 4

The geometrical setting

Throughout the previous chapters, very little attention was paid to the geometrical setting. The main topic of this thesis is a discussion of finite element solutions in non-convex domains, and so we need to introduce some geometrical considerations.

We will first introduce some standard definitions and language for triangulations. We move on to introduce the *newest-vertex bisection* (NVB) [23] algorithm for mesh refinement. Next, we will discuss the issue of reduced regularity of the exact solution to the wave equation in non-convex domains, which has detrimental effects on the convergence rate associated with the standard polynomial spaces generated from quasi-uniform meshes for smooth solutions. We will furthermore see how a particular algorithm for local mesh refinement is able to recover the usual convergence rates in the H^1 norm. We end the chapter with an important result on the regularity of the exact solution.

4.1 Triangulation and simple bisection

In this thesis we will only consider two-dimensional, triangular elements. We will assume that the reader is not a stranger to triangulation, but for completeness we will recall a few important definitions, which we have borrowed from [15].

Definition 4.1.1
For a triangulation $\mathcal{T}_h$ of a bounded polygonal domain $\Omega \subseteq \mathbb{R}^2$, we have that

$$\bar{\Omega} = \bigcup_{K \in \mathcal{T}_h} K,$$

where $\bar{\Omega}$ denotes the closure of Ω. We say that

- for any triangle $K \in \mathcal{T}_h$, we define its *diameter* h_K as the longest edge of the triangle.
- the mesh-size h is given by

$$h = \max_{K \in \mathcal{T}_h} h_K.$$

- $\mathcal{T}_h$ is *conforming* if for any $K_1, K_2 \in \mathcal{T}_h$ where $F = K_1 \cap K_2 \neq \emptyset$ and $K_1 \neq K_2$, F is either a whole edge or a single vertex in the triangulation.
- the mesh family $\{\mathcal{T}_h \mid h > 0\}$ is (shape) *regular* if there exists $\gamma > 0$ independent of h such that, for any $\mathcal{T}_h$,

$$\frac{h_K}{\rho_K} \leq \gamma \quad \forall K \in \mathcal{T}_h, \tag{4.1}$$

  where ρ_K denotes the diameter of the inscribed circle of the element K.
- the mesh family $\{\mathcal{T}_h \mid h > 0\}$ is *quasi-uniform* if there exists a constant $\beta > 0$ such that,

$$\min_{K \in \mathcal{T}_h} h_K \geq \beta h \quad \forall h > 0.$$

The shape regularity condition is often rephrased in terms of a *minimum angle*, stating that the minimum angle of any triangle in the mesh is bounded from below by a constant independent of h.

Definition 4.1.2
$\mathcal{N}(\mathcal{T}_h)$ denotes the set of vertices in a triangulation, and $\#\mathcal{N}(\mathcal{T}_h)$ denotes the number of vertices. In addition, we denote the number of triangles in the triangulation by $\#\mathcal{T}_h$.

While convergence properties are naturally described in terms of the mesh resolution h for quasi-uniform meshes, this formulation offers little intuition for the computational complexity associated with a given mesh resolution. Moreover, it is often not sufficient

to describe convergence rates where local refinement is used to better approximate the solution. To this end, the number of vertices in the mesh is typically a much better measure. The following is a common result that can be readily proved in terms of the above definitions, but for brevity we will omit the proof.

Lemma 4.1.3
For any conforming mesh family $\mathbb{T}_h$, there exists a constant $C_1 > 0$ such that, for any $\mathcal{T}_h \in \mathbb{T}_h$,

$$\#\mathcal{N}(\mathcal{T}_h)^{-1/2} \leq C_1 h. \tag{4.2}$$

If in addition $\mathbb{T}_h$ is (shape) regular and quasi-uniform, then there exists a constant $C_2 > 0$ such that

$$h \leq C_2(\#\mathcal{N}(\mathcal{T}_h))^{-1/2}. \tag{4.3}$$

We will now go on to define a simple algorithm for mesh refinement. The idea is to define a set of *marked* triangles which we wish to refine. The marked triangles are passed to the algorithm for refinement, and in the process it may refine additional triangles to maintain conformity of the mesh. This forms the basis for a cycle in the form of

$$\texttt{MARK} \rightarrow \texttt{REFINE} \rightarrow \texttt{REPEAT}$$

with repetition until some criterion is fulfilled. For example, if we wish to make sure all triangles have diameter below some threshold, we would continue to mark triangles whose diameter is larger than the threshold. When all triangles have diameters below the threshold, the cycle ends by virtue of the last set of marked triangles being empty.

The algorithm below is typically referred to as *newest-vertex bisection* (NVB), and is normally attributed to Bänsch [23].

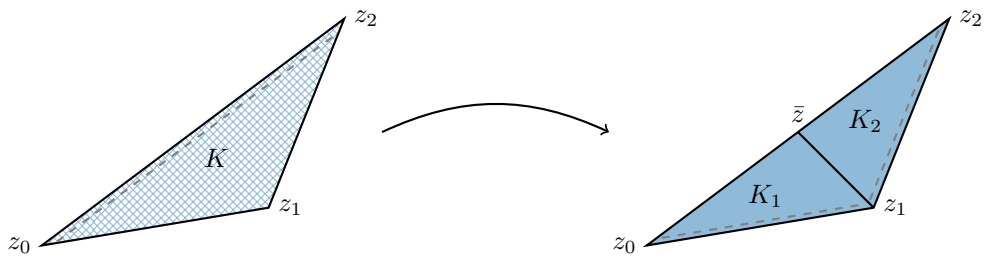


Figure 4.1: Bisection of a single triangle K into smaller triangles K_1 and K_2. Dashed lines indicate refinement edges.

Definition 4.1.4 (Newest-vertex bisection (NVB))
Let $\mathcal{T}_0$ be a given conforming initial triangulation, and let $\mathcal{M} \subseteq \mathcal{T}_0$ denote the set of marked triangles. We let each triangle $K \in \mathcal{T}_0$ be described by an *ordered* triplet (z_0, z_1, z_2), where $z_i \in \mathbb{R}^2$ denotes the coordinates of the local vertices. Then the following algorithm, $\mathcal{T} \leftarrow$ REFINE_MARKED$(\mathcal{T}_0, \mathcal{M})$, yields a conforming refinement of $\mathcal{T}_0$.

1. Let $\mathcal{T} = \mathcal{T}_0$.
2. For each $K = (z_0, z_1, z_2) \in \mathcal{M}$,
   (a) let $\bar{z} := \frac{1}{2}(z_0 + z_2)$ denote a new vertex in the triangulation.
   (b) let K_1 and K_2 denote the two new resulting triangles, where $K_1 = (z_0, \bar{z}, z_1)$, $K_2 = (z_1, \bar{z}, z_2)$.
   (c) remove K from $\mathcal{T}$.
   (d) add K_1 and K_2 to $\mathcal{T}$.
3. Let $\mathcal{M} = \{$triangles in $\mathcal{T}$ which contain nonconforming (hanging) nodes$\}$.
4. If $\mathcal{M} = \emptyset$, terminate. Otherwise, repeat from 2.

Remark. Note that the algorithm terminates in a finite number of steps [23].

By inspecting the above algorithm, we note that by refining only a subset of the triangles, we run the risk of leaving hanging nodes - vertices which are contained in triangles for which they are not corners - in the triangulation. Step 3 ensures that all such hanging nodes are eventually removed. The algorithm is probably best explained by some illustrations. See Figure 4.1 for an illustration of the bisection of a single triangle. Figure 4.2 demonstrates the treatment of hanging nodes. Here we observe that the first step of the algorithm produces a non-conforming triangulation, but subsequent iteration eventually yields a conforming triangulation.

To make the steps in the next section clearer, we present a simple algorithm which

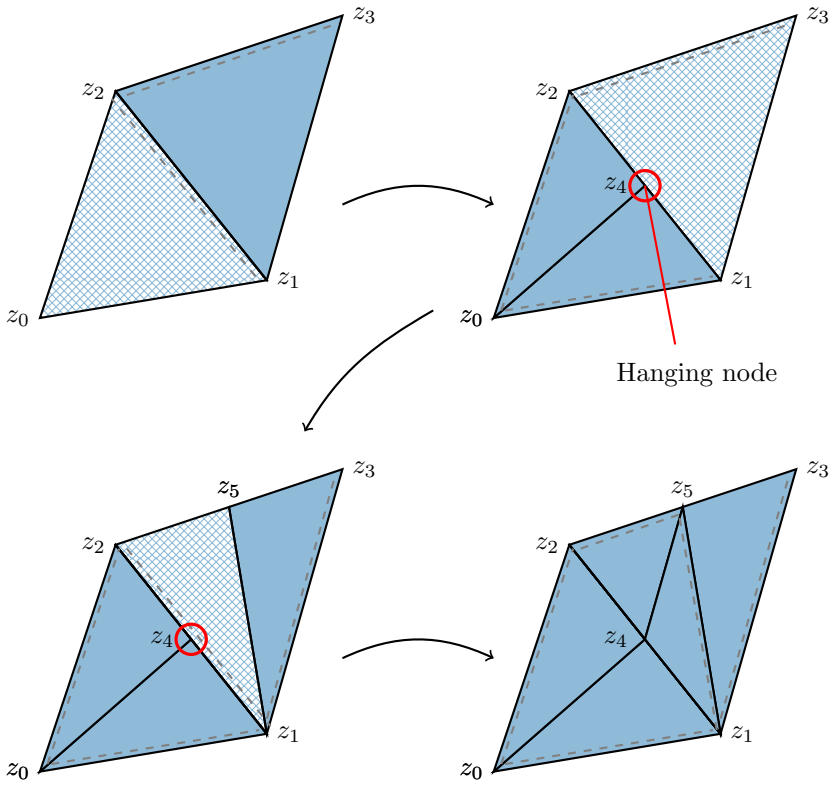


Figure 4.2: Bisection of a triangle in a two-element triangulation. An intermediate step leads to a nonconforming triangulation with a hanging node. Subsequent iterations produce a final, conforming triangulation. Marked triangles are indicated by a crosshatched (patterned) surface, and refinement edges for each triangle are indicated by dashed lines.

relies on the NVB algorithm to construct a mesh $\mathcal{T}_H$ such that all triangles in the mesh have diameter smaller or equal to H.

Lemma 4.1.5 (Refine mesh to given tolerance)
Let $\mathcal{T}_0$ be a conforming triangulation of the bounded, polygonal domain $\Omega \subseteq \mathbb{R}^2$. Then, for each $H > 0$, the following algorithm generates a conforming triangulation $\mathcal{T}_H$ of Ω with mesh size H, and the mesh family $\mathbb{T}_H = \{\mathcal{T}_H \mid H > 0\}$ is regular and quasi-uniform.

```
1: function REFINE_TO_TOLERANCE(T_0, H)
2:     M ← ∅
3:     T_H ← T_0
4:     repeat
5:         T_H ← REFINE_MARKED(T_H, M)
6:         M ← {T ∈ T_H | h_T > H}
7:     until M = ∅

8:     return T_H
9: end function
```

Proof. We omit the proof for brevity, but the properties of $\mathcal{T}_H$ are straightforward consequences of the properties of NVB. □

4.2 Corner singularities in non-convex domains

When we derived the error estimates for the Leapfrog method (Corollary 3.2.8) and the Crank-Nicolson method (Corollary 3.3.7) in piecewise polynomial spaces, we made some strong assumptions on the regularity of the exact solution u, namely that $u(t) \in H_0^s(\Omega)$ for $s \geq 2$ for all t. For sufficiently smooth initial conditions u_0 and v_0 and right-hand side f, this is a reasonable assumption in convex domains. However, the assumption of convexity is not practical for real-world applications, which indeed may require the solutions to problems in highly non-convex domains. In this case, even smooth problem data may lead to solutions u which are *not* contained in $H^2(\Omega)$, and so we cannot in general expect to attain the aforementioned convergence rates. In Chapter 8, we will study a model problem for which the standard polynomial space on a quasi-uniform mesh fails to attain the optimal convergence rates.

It turns out that the loss of regularity is due to the introduction of *corner singularities*, which is well-known also from the study of elliptic problems. Briefly explained, this means that the solution or its derivative blows up near re-entrant corners of the mesh. Re-entrant corners are corners whose interior angle exceeds π radians. We will rely on the results of Müller and Schwab [6], who detail the regularity of solutions to the wave equation given smooth problem data in non-convex domains. Moreover, they show how an algorithm originally introduced for elliptic problems by Gaspoz and Morin [5] can

be applied to recover the optimal convergence rates for the wave equation observed in convex domains.

In this section, we will present the most important results from the aforementioned papers which relate to our present work, but we will do so in a fashion which attempts to minimize the introduction of new terminology and language.

We begin by introducing the *Threshold* algorithm from [6], which we have adapted so that it is a more natural fit with the content that will be presented in Chapter 5. An example of the algorithm in action can be seen in Figure 4.3, where we have set $H = 0.5$. Note that the algorithm outputs *two* meshes - a quasi-uniform refinement $\mathcal{T}_H$ of the initial mesh $\mathcal{T}_0$ and a refinement $\mathcal{T}_h$ of $\mathcal{T}_H$.

Lemma 4.2.1 (The Threshold algorithm)
Let $\mathcal{T}_0$ be a conforming triangulation of the bounded, polygonal domain $\Omega \subseteq \mathbb{R}^2$, whose M corners are denoted $\mathbf{c}_i$ for $i = 1, \ldots, M$. $\theta_i \in (0, 2\pi)$ for $i = 1, \ldots, M$ denotes the interior angle of the corner $\mathbf{c}_i$. Furthermore, let $p \geq 1$ be the polynomial degree of the finite element space. Then the following algorithm generates conforming regular meshes $\mathcal{T}_H$ and $\mathcal{T}_h$ for any $H > 0$, with $\mathcal{T}_H$ quasi-uniform.

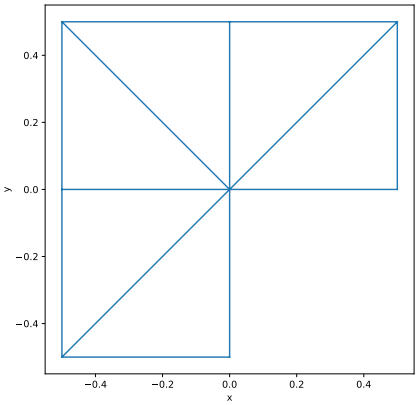
function THRESHOLD$(\mathcal{T}_0, H)$
  $\mathcal{M} \leftarrow \emptyset$
  $\lambda \leftarrow \frac{\pi}{2 \max_i \theta_i}$
  $\mathcal{T}_H \leftarrow$ REFINE_TO_TOLERANCE$(\mathcal{T}_0, H)$
  $\mathcal{T}_h \leftarrow \mathcal{T}_H$

  $K \leftarrow \lceil \frac{p+1}{\lambda} \log_2 H^{-1} \rceil$
  for all $1 \leq k < 2K$ ***do***
    $\mathcal{M} \leftarrow \left\{ T \in \mathcal{T}_h \mid h_T > H \cdot 2^{-\frac{k(p+1-\lambda)}{2(p+1)}} \text{ and } \min_i \operatorname{dist}(\mathbf{c}_i, \bar{T}) \leq 2^{-\frac{k}{2}} \right\}$
    $\mathcal{T}_h \leftarrow$ REFINE_MARKED$(\mathcal{T}_h, \mathcal{M})$
  end for

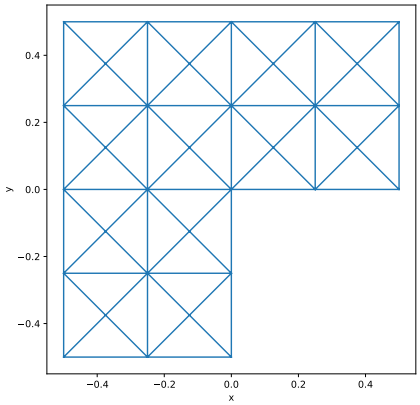
  return $\mathcal{T}_H$, $\mathcal{T}_h$
end function

In the above algorithm, $\operatorname{dist}(\mathbf{c}_i, \bar{T})$ *represents the distance from the corner* $\mathbf{c}_i$ *to the triangle* $\bar{T}$.

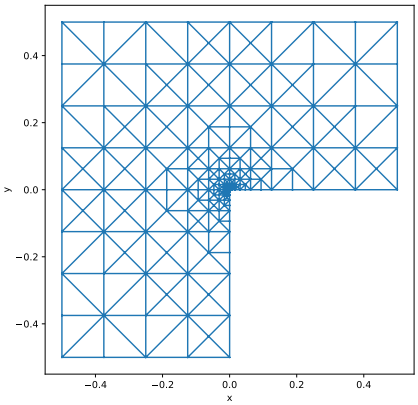
It turns out that the finite element spaces $S_0^1(\mathcal{T}_H)$ and $S_0^1(\mathcal{T}_h)$ associated with the two meshes generated by Threshold satisfy $S_0^1(\mathcal{T}_H) \subseteq S_0^1(\mathcal{T}_h)$. Because this property is very important for the content in Chapter 5, we emphasize it in the following lemma.



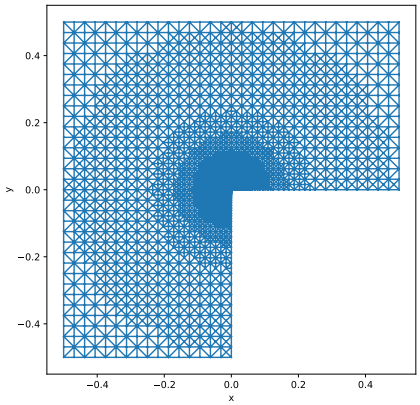
(a) An initial triangulation $\mathcal{T}_0$ of a classic L-shaped domain.



(b) The quasi-uniform triangulation $\mathcal{T}_H$ produced by running Threshold on $\mathcal{T}_0$ with $H = 0.3$.



(c) The locally refined triangulation $\mathcal{T}_h$ produced by running Threshold on $\mathcal{T}_0$ with $H = 0.3$.



(d) The locally refined triangulation $\mathcal{T}_h$ produced by running Threshold on $\mathcal{T}_0$ with $H = 0.075$.

Figure 4.3: Results $\mathcal{T}_H$ and $\mathcal{T}_h$ of the Threshold algorithm from Lemma 4.2.1 applied to an L-shaped domain with initial triangulation $\mathcal{T}_0$ for different H.

Lemma 4.2.2
Given $\mathcal{T}_H$ and $\mathcal{T}_h$ generated by the Threshold algorithm, the associated finite element spaces $S_0^1(\mathcal{T}_H)$ and $S_0^1(\mathcal{T}_h)$ satisfy $S_0^1(\mathcal{T}_H \subseteq S_0^1(\mathcal{T}_h)$.

Proof. This is due to properties of the newest-vertex bisection algorithm. See [6] for details. □

Another useful property of the mesh generated by the Threshold algorithm is that the number of vertices in the fine mesh $\mathcal{T}_h$ is bounded.

Lemma 4.2.3
Let $\mathcal{T}_H$ and $\mathcal{T}_h$ be meshes generated by Threshold. There exists a constant $C > 0$ independent of H such that the fine mesh $\mathcal{T}_h$ satisfies

$$\#\mathcal{N}(\mathcal{T}_h) \leq C\#\mathcal{N}(\mathcal{T}_H). \tag{4.4}$$

Proof. We use a complexity bound from [5], which states that

$$\#\mathcal{T}_h - \#\mathcal{T}_0 \leq C_0 H^{-2}$$

for some constant C_0. The result is obtained by combining this result with (4.2). □

For some of the following results, we will need to make some assumptions on the data (f, u_0, v_0) of the problem. We collect these assumptions in the following definition.

Definition 4.2.4 (Strong smoothness assumptions)
If u is the exact solution to the weak formulation in Definition 2.1.1 and the problem data satisfies

$$f \in C_c^\infty((0,T); C^\infty(\bar{\Omega})), \qquad u_0, v_0 \in C_c^\infty(\Omega), \tag{4.5}$$

we say that the data satisfies the *strong smoothness assumptions*.

Throughout the rest of this thesis, we will often refer to $\mathcal{T}_H$ as a *coarse* mesh, and $\mathcal{T}_h$ as a *fine* mesh. We will now present the main results of this chapter. First, we give error estimates for the elliptic projection onto the fine mesh $\mathcal{T}_h$ and show that this leads to the recovery of the optimal convergence rates for polynomial finite element spaces in convex domains. Then we finally give regularity results on the exact solution in the presence of corner singularities which will be essential to the theoretical discussion in Chapter 5. In the following results, we will assume that the domain Ω is *sufficiently regular*. The exact meaning of this is complicated and would require the introduction

of a large amount of theory to accurately describe, so we will again simply refer the reader to [6]. Suffice to say, however, that the assumptions include a large class of non-convex meshes.

Theorem 4.2.5 (Estimates for the elliptic projection onto $S_0^1(\mathcal{T}_h)$)
Let $\Omega \subseteq \mathbb{R}^2$ be a sufficiently regular, bounded polygonal domain, and let $\mathcal{T}_0$ be an initial triangulation of Ω. Furthermore, let H satisfy

$$0 < H < (\#\mathcal{N}(\mathcal{T}_0))^{-2},$$

and let $\mathcal{T}_h$ be the mesh produced by the algorithm from Lemma 4.2.1 with parameters H and $p \geq 1$. Moreover, assume that u satisfies the smoothness assumptions of Definition 4.2.4. Then there exists a constant $C = C(u, T) > 0$, such that for almost every $t \in [0, T]$,

$$\left\| \frac{\partial^k u}{\partial t^k} - \frac{\partial^k R_h u}{\partial t^k} \right\|_{H^1} \leq C N^{-\frac{p}{2}} \qquad k = 0, 1, 2 \tag{4.6}$$

where R_h denotes the elliptic projection onto $S_0^1(\mathcal{T}_h)$ and $N := \#\mathcal{N}(\mathcal{T}_h)$ denotes the number of vertices in the fine mesh $\mathcal{T}_h$.

Proof. This is clear from the discussion in the proofs of Theorem 5.3 and Theorem 5.5 in [6]. □

Recall from Theorem 3.2.7 and Theorem 3.3.6 that the error incurred by the Leapfrog and Crank-Nicolson methods was estimated in terms of the elliptic projection defined in Definition 2.2.5. Hence, if the H^1 error in the elliptic projection is $\mathcal{O}(H)$, the optimal H^1 convergence rate associated with the Leapfrog and Crank-Nicolson methods is restored. Combining Theorem 4.2.5 with (4.2) yields the following corollary.

Corollary 4.2.6
Let $\mathcal{T}_h$ and H be defined as in Theorem 4.2.5. Then there exists a constant $C > 0$ such that

$$\left\| \frac{\partial^k u}{\partial t^k} - \frac{\partial^k R_h u}{\partial t^k} \right\|_{H^1} \leq C H^p. \qquad k = 0, 1, 2 \tag{4.7}$$

We end this chapter with an important regularity result on the exact solution u in non-convex domains.

Lemma 4.2.7 (Regularity of solutions in non-convex domains)
Assume that the domain Ω is sufficiently regular, and that the exact solution u and the problem data satisfy the smoothness assumptions of Definition 4.2.4. *Then $\Delta u(t) \in L^2(\Omega)$ for all $t \in [0, T]$.*

Proof. From Corollary 3.10 in [6], we have that for each $k \in \mathbb{N}_0$ and each $t \in (0, T)$, the solution $u(\cdot, t)$ admits the following decomposition (using the notation in the article):

$$u(x,t) = u_r^k(x,t) + \sum_{i=1}^{M} \chi_i(r_i) \sum_{n=1}^{n^i_{\max,k}} d_n^i(x,t) S_{n,i}(r_i, \vartheta_i),$$

where $u_r^k(\cdot, t) \in H^k(\Omega)$, $d_n^i \in C^\infty([0,T]; C^\infty(\bar{\Omega}))$ and $\chi_i(r_i)$ is a smooth cutoff function. From Definition 3.4, Definition 3.5 and the proof of Theorem 5.3 in the same paper, we see that the definition of $S_{n,i}$ leads to $\Delta S_{n,i} = 0$ and with appropriate regularity assumptions we have $S_{n,i} \in H^1(\Omega)$, which together with the product rule for the Laplacian Δ applied to the above decomposition leads to $\|\Delta u(x,t)\| < \infty$. □

Chapter 5

CFL relaxation by spatial reduction

As we have seen in Chapter 4, local mesh refinement in the vicinity of re-entrant corners can potentially lead to drastic improvement of the accuracy of finite element solutions in non-convex domains by recovering the optimal convergence rate associated with smooth solutions in convex domains. However, it is observed that the refinement makes the Leapfrog method unstable for all but unreasonably small time steps, and in practice the method is rendered unusable. One can still use implicit methods like the Crank-Nicolson method presented in Chapter 3.3, but this is typically associated with much greater computational complexity.

Let $\mathcal{T}_H$ and $\mathcal{T}_h$ represent the meshes generated by the Threshold algorithm in Lemma 4.2.1, which means that $\mathcal{T}_H$ is a quasi-uniform triangulation and $\mathcal{T}_h$ a refinement of $\mathcal{T}_h$. We will only consider linear finite elements. To understand why the Leapfrog method becomes unstable in $S_0^1(\mathcal{T}_h)$, recall the statement of Theorem 3.2.6. An assumption for stability is that the finite element space has the *inverse property*. Because $\mathcal{T}_h$ is a *local* refinement of $\mathcal{T}_H$, the mesh size of $\mathcal{T}_h$ is in fact usually H, which means that the inverse property for $\mathcal{T}_h$ translates to

$$\|\nabla v_h\| \leq CH^{-1} \|v_h\| \qquad \forall v_h \in S_0^1(\mathcal{T}_h).$$

However, what is observed is that this does *not* hold for $S_0^1(\mathcal{T}_h)$, and instead the actual inverse inequality reads [12]

$$\|\nabla v_h\| \leq Ch_{\min}^{-1} \|v_h\| \qquad \forall v_h \in S_0^1(\mathcal{T}_h).$$

where $h_{\min}$ is the size of the *smallest* element in the mesh. Because of the strong local refinement in $\mathcal{T}_h$, we have that $h_{\min}$ is not bounded from below by H, and can be many orders of magnitude smaller than H, which leads to the requirement that the time step must be similarly small to regain stability.

However, because $\mathcal{T}_H$ is quasi-uniform, we know that the corresponding space $S_0^1(\mathcal{T}_H)$ satisfies the above inverse inequality proportional to H^{-1}, which we repeat here for clarity:

$$\|\nabla v_H\| \leq CH^{-1} \|v_H\| \qquad \forall v_H \in S_0^1(\mathcal{T}_H).$$

In other words, our current situation is the following:

- $S_0^1(\mathcal{T}_H)$ satisfies the inverse inequality proportional to H^{-1} which makes the Leapfrog method stable for reasonable timesteps, but it only admits a suboptimal convergence rate for the error.
- $S_0^1(\mathcal{T}_h)$ admits the optimal convergence rate for the error, but it does *not* satisfy the inverse inequality proportional to H^{-1}.

In this chapter, we will discuss a method introduced by Peterseim and Schedensack [12] which relaxes the CFL condition (3.10) associated with the Leapfrog method in the presence of strong local mesh refinement. The idea is essentially to combine the inverse inequality of $S_0^1(\mathcal{T}_H)$ with the convergence rate of $S_0^1(\mathcal{T}_h)$ by techniques known from multi-scale modeling and numerical homogenization.

In short, a new finite element space V_H is constructed, drawing on information from both spaces $S_0^1(\mathcal{T}_H)$ and $S_0^1(\mathcal{T}_h)$. The new space can be said to be a *reduced* space with respect to $S_0^1(\mathcal{T}_h)$, because the functions in V_H are typically expressed in terms of basis functions in $S_0^1(\mathcal{T}_h)$, yet the number of degrees of freedom in V_H is equal to that of the coarse space $S_0^1(\mathcal{T}_H)$. We will now outline the procedure in a very high-level fashion. The rest of the chapter will fill in the details we must leave out in this very brief description.

1. Recall that $S_0^1(\mathcal{T}_H) \subseteq S_0^1(\mathcal{T}_h)$.
2. Introduce a projection $I_H : H_0^1(\Omega) \to S_0^1(\mathcal{T}_H)$ which we can use to project functions from $S_0^1(\mathcal{T}_h)$ into $S_0^1(\mathcal{T}_H)$.
3. Define the space $W_h := \ker I_H\big|_{S_0^1(\mathcal{T}_h)}$. This is the space of functions $v_h \in S_0^1(\mathcal{T}_h)$ that vanish entirely when projected onto $S_0^1(\mathcal{T}_H)$. In other words, the coarse space $S_0^1(\mathcal{T}_H)$ does not "see" these functions at all.
4. Define V_H to be the orthogonal complement of W_h with respect to the $a(\cdot,\cdot)$ inner product. We will see that this particular choice leads to both the recovery of the inverse inequality of $S_0^1(\mathcal{T}_H)$ and the convergence rate of $S_0^1(\mathcal{T}_h)$.
5. To actually construct V_H, we can think of it as a "correction" of $S_0^1(\mathcal{T}_H)$. In fact, for any $v_H \in S_0^1(\mathcal{T}_H)$, we define its *corrector* $\mathcal{C}v_H$ as the elliptic projection onto W_h, which lets us define the corresponding function $\tilde{v}_H \in V_H$ by $\tilde{v}_H := (1-\mathcal{C})v_H$, which leads to $V_H = (1-\mathcal{C})S_0^1(\mathcal{T}_H)$.
6. We need a basis for V_H. Given the usual Lagrangian basis for $S_0^1(\mathcal{T}_H)$, we can form a basis for V_H by individually computing the corrector associated with each Lagrangian basis function $\lambda_{H,i} \in S_0^1(\mathcal{T}_H)$ and taking $\tilde{\lambda}_{H,i} = \lambda_{H,i} - \mathcal{C}\lambda_{H,i}$.

7. The basis for V_H constructed in this way has global support, which is a prohibitive restriction for practical computation. In Section 5.3.1, we will justify the approximation of the global basis by a localized approximation.

We must note that the claim in step 4 only holds if the projection I_H (which we will refer to as a *quasi-interpolator* in this chapter) satisfies certain stability and approximation properties. In Section 5.2, we will give an example of a quasi-interpolator which fulfills these properties. In Section 5.3.2, we will show that given some conditions on the quasi-interpolator I_H, correctors need only be computed in the neighborhood of locally refined portions of the mesh. In other words, correctors do not need to be computed in large areas in which $\mathcal{T}_H$ and $\mathcal{T}_h$ locally coincide.

In the rest of this chapter, we will present some of the most important theoretical results from [12], and in so doing we will justify each step of the aforementioned procedure. While we will cover the majority of the theory presented in [12] in detail, we will leave some long or involved proofs out. At the same time, we will further elaborate on some of the shorter or omitted proofs in the paper from which we hope may help the reader to better understand the method. Although enough of the theory is presented here as to be self-contained, there are certainly some useful insights to be learned from the original paper which has been left out in this exposition, and so we encourage the interested reader to study it.

In the rest of this thesis, we will make a habit of referring to the Lagrangian basis functions in $S_0^1(\mathcal{T}_H)$ as $\lambda_{H,z}$ (corresponding to vertex $z \in \mathcal{N}(\mathcal{T}_h)$) or $\lambda_{H,i}$ (corresponding to node labeled i), and similarly for the fine-scale Lagrangian basis function $\lambda_{h,z} \in S_0^1(\mathcal{T}_h)$.

5.1 Construction of a reduced finite element space

Our first step is to clearly define what is meant by the *quasi-interpolator* I_H. In the chapter introduction, we referred to it as a projection, but we will need to give it some additional properties. To this end, we recall the definition of a projection from Definition 2.2.6 and define *admissible quasi-interpolators* below.

Definition 5.1.1 (Admissible quasi-interpolators)
A surjective projection $I_H : H_0^1(\Omega) \to X_H$ is considered an *admissible quasi-interpolator* for a finite element space $X_H \subseteq H_0^1(\Omega)$ if there exists a constant $C_{I_H} > 0$ independent of H such that

$$\|v - I_H v\| \leq C_{I_H} H \|\nabla v\| \qquad \forall v \in H_0^1(\Omega), \tag{5.1}$$

in addition to the stability properties

$$\|\nabla I_H v\| \leq C_{I_H} \|\nabla v\| \qquad \forall v \in H_0^1(\Omega), \tag{5.2}$$

$$\|I_H v\| \leq C_{I_H} \|v\| \qquad \forall v \in H_0^1(\Omega). \tag{5.3}$$

We remark that the H^1 stability (5.2) and the L^2 approximation property (5.1) are standard properties of quasi-interpolators, but the L^2 stability (5.3) is less common. In order to demonstrate that the specific choice of I_H is immaterial for many of the main results in this chapter, we will keep the choice of I_H open for now, but we will give an example of an appropriate admissible quasi-interpolator in Section 5.2.

We will now define W_h, the kernel of the quasi-interpolator I_H when restricted to the fine space $S_0^1(\mathcal{T}_h)$.

Definition 5.1.2 (The space of fine-scale oscillations)
Given two finite element spaces $S_0^1(\mathcal{T}_H)$ and $S_0^1(\mathcal{T}_h)$ which satisfy $S_0^1(\mathcal{T}_H) \subseteq S_0^1(\mathcal{T}_h)$ and an admissible quasi-interpolator $I_H : H_0^1(\Omega) \to S_0^1(\mathcal{T}_H)$, we define the space

$$W_h := \ker(I_H\big|_{S_0^1(\mathcal{T}_h)}) \subseteq S_0^1(\mathcal{T}_h). \tag{5.4}$$

W_h is referred to as the *space of fine-scale oscillations* with respect to $S_0^1(\mathcal{T}_h)$ and $S_0^1(\mathcal{T}_H)$.

We see that W_h captures the functions in $S_0^1(\mathcal{T}_h)$ which vanish when projected onto $S_0^1(\mathcal{T}_H)$. If one thinks of the Lagrangian basis for $S_0^1(\mathcal{T}_H)$, the functions in W_h when projected onto $S_0^1(\mathcal{T}_H)$ vanish at each node, and so depending on the quasi-interpolator I_H, this excludes "large" functions at the coarse level. In other words, these functions can in some sense be thought of as oscillations at the fine-scale level, which justifies its name as the space of fine-scale oscillations.

The following lemma is not strictly necessary for the development of the subsequent theory, but it is nonetheless useful for developing an intuitive understanding of the procedure. Note that the result is merely a direct consequence of the kernel-range decomposition of a projection.

Lemma 5.1.3
Given two finite element spaces $S_0^1(\mathcal{T}_H)$ and $S_0^1(\mathcal{T}_h)$ which satisfy $S_0^1(\mathcal{T}_H) \subseteq S_0^1(\mathcal{T}_h)$ and an admissible quasi-interpolator $I_H : H_0^1(\Omega) \to S_0^1(\mathcal{T}_H)$ which generates W_h, we have that

$$S_0^1(\mathcal{T}_h) = S_0^1(\mathcal{T}_H) \oplus W_h.$$

Consequently, $\dim(S_0^1(\mathcal{T}_h)) = \dim(S_0^1(\mathcal{T}_H)) + \dim(W_h)$.

Proof. A direct consequence of the fact that I_H is a surjective projection is that if $I_H v_H = 0$ for any $v_H \in S_0^1(\mathcal{T}_H)$, then $v_H = 0$, and so $S_0^1(\mathcal{T}_H) \cap W_h = \{0\}$. Moreover, for any $v_h \in S_0^1(\mathcal{T}_h)$, we may write

$$\begin{aligned} v_h &= v_h + I_H v_h - I_H v_h \\ &= \underbrace{I_H v_h}_{\in S_0^1(\mathcal{T}_H)} + \underbrace{(1 - I_H) v_h}_{\in W_h}. \end{aligned}$$

The direct sum follows from the preceding argument. The sum of dimensions is a trivial consequence of all spaces being finite-dimensional vector spaces. □

The main idea in the construction of the reduced space is to maintain the direct sum property of Lemma 5.1.3, but orthogonalize $S_0^1(\mathcal{T}_H)$ against W_h with respect to the inner product $a(\cdot, \cdot)$. Our strategy for the orthogonalization procedure is to define *correctors* for each function in $S_0^1(\mathcal{T}_H)$. These correctors can be thought of as the adjustment necessary to make each function orthogonal to the whole of W_h. We now define the corrector problems, and go on to define the *corrected* space V_H.

Definition 5.1.4 (Corrector problem)
For any $v_H \in S_0^1(\mathcal{T}_H)$, we define its *corrector* $\mathcal{C} v_H \in W_h$ with respect to W_h as the solution to the *corrector problem*

$$a(\mathcal{C} v_H, w_h) = a(v_H, w_h) \qquad \forall w_h \in W_h. \tag{5.5}$$

Remark. The existence and uniqueness of solutions to the corrector problem is a direct consequence of the Lax-Milgram lemma.

We note that the corrector $\mathcal{C} v_H$ is nothing but the elliptic projection from $S_0^1(\mathcal{T}_h)$ onto W_h, but restricted to coarse functions $v_H \in S_0^1(\mathcal{T}_H) \subseteq S_0^1(\mathcal{T}_h)$. We now have the means to define the corrected space V_H, after which we will show that it is indeed the orthogonal complement of W_h.

Definition 5.1.5 (Corrected coarse space)
Given finite element spaces $S_0^1(\mathcal{T}_H)$ and $S_0^1(\mathcal{T}_h)$ which satisfy $S_0^1(\mathcal{T}_H) \subseteq S_0^1(\mathcal{T}_h)$, the *corrected coarse space* V_H is defined by

$$V_H := (1 - \mathcal{C})S_0^1(\mathcal{T}_H), \tag{5.6}$$

where $\mathcal{C} : S_0^1(\mathcal{T}_H) \to W_h$ is given by Definition 5.1.4.

Lemma 5.1.6
Given a fine-scale finite element space $S_0^1(\mathcal{T}_h)$ and V_H as defined in Definition 5.1.5, then

$$S_0^1(\mathcal{T}_h) = V_H \oplus W_h, \tag{5.7}$$

and the sum is orthogonal with respect to the inner product $a(\cdot, \cdot)$. Moreover,

$$\dim V_H = \dim S_0^1(\mathcal{T}_H). \tag{5.8}$$

Proof. We have that for any $v_H \in S_0^1(\mathcal{T}_H)$, $I_H(1-\mathcal{C})v_H = I_H v_H$, so if $I_H(1-\mathcal{C})v_H = 0$, then $v_H = 0$, and hence $V_H \cap W_h = \{0\}$. Moreover, for any $v_h \in S_0^1(\mathcal{T}_h)$,

$$\begin{aligned} v_h &= v_h + I_H v_h - I_H v_h + \mathcal{C} I_H v_h - \mathcal{C} I_H v_h \\ &= \underbrace{(1 - \mathcal{C})I_H v_h}_{\in S_0^1(\mathcal{T}_H)} + \underbrace{(1 - I_H)v_h + \mathcal{C} I_H v_h}_{\in W_h}. \end{aligned}$$

Hence, the direct sum follows, and the orthogonality of the sum is a simple consequence of (5.5). Thus, analogous to the case in Lemma 5.1.3, we have that

$$\dim(S_0^1(\mathcal{T}_h)) = \dim V_H + \dim W_h,$$

and (5.8) follows. □

Having constructed V_H, we must verify that it indeed achieves both the convergence rate of $S_0^1(\mathcal{T}_h)$ and an inverse inequality comparable to that of $S_0^1(\mathcal{T}_H)$.

Lemma 5.1.7 (Best approximation for the corrected space)
Let $\mathcal{T}_H$ denote a regular, conforming quasi-uniform mesh, and let $\mathcal{T}_h$ be a refinement of $\mathcal{T}_H$ such that $S_0^1(\mathcal{T}_H) \subseteq S_0^1(\mathcal{T}_h)$. Assume that $S_0^1(\mathcal{T}_h)$ for some $C_0 > 0$ satisfies

$$\inf_{v_h \in S_0^1(\mathcal{T}_h)} \|u - v_h\|_{H^1(\Omega)} \leq C_0 H \|\Delta u\|_{L^2(\Omega)} . \tag{5.9}$$

Then there exists a constant $C > 0$ such that for all $u \in H_0^1(\Omega)$ with $\Delta u \in L^2(\Omega)$, the corrected space V_H satisfies

$$\inf_{v_H \in V_H} \|u - v_H\|_{H^1(\Omega)} \leq C H \|\Delta u\|_{L^2(\Omega)} . \tag{5.10}$$

Proof. The first part of the proof follows the proof of Lemma 2.1 in [12]. Because the presentation is a little different, we will also provide a proof here.

Define $u_h \in S_0^1(\mathcal{T}_h)$ to be the projection of u onto $S_0^1(\mathcal{T}_h)$ with respect to the bilinear form $a(\cdot, \cdot)$,

$$a(u_h, v_h) = a(u, v_h) \qquad \forall v_h \in S_0^1(\mathcal{T}_h),$$

and similarly define $u_{h,H}$ to be the projection of u_h onto V_H. More precisely, $u_{h,H}$ satisfies

$$a(u_{h,H}, v_H) = a(u_h, v_H) \qquad \forall v_H \in V_H.$$

Define $e_{h,H} := u_{h,H} - u_h$. The Galerkin orthogonality $a(e_{h,H}, v_H) = 0$ for all $v_H \in V_H$ implies that $e_{h,H} \in V_H^{\perp} = W_h$, since W_h is the orthogonal complement of V_H. Hence, $I_H e_{h,H} = 0$, and we may employ the properties of the (implicit) admissible quasi-interpolator I_H, as defined in Definition 5.1.1, to write

$$\|e_{h,H}\|_{L^2} = \|e_{h,H} - I_H e_{h,H}\|_{L^2} \leq C_{I_H} H \|\nabla e_{h,H}\|_{L^2} .$$

Again using Galerkin orthogonality, integration by parts, and finally the above result, we may now write

$$\begin{aligned} \|\nabla e_{h,H}\|_{L^2}^2 &= (\nabla e_{h,H}, \nabla e_{h,H}) \\ &= (\nabla u_h, \nabla e_{h,H}) \\ &= (-\Delta u, e_{h,H}) \\ &\leq C_{I_H} H \|\Delta u\|_{L^2} \|\nabla e_{h,H}\|_{L^2} . \end{aligned}$$

Note that these results together imply that $\|e_{h,H}\|_{H^1} \leq C_1 H \|\Delta u\|_{L^2}$ for some constant $C_1 > 0$. Next, observe that Céa's lemma together with (5.9) implies that $\|u - u_h\|_{H^1} \leq$

$C_2 H \|\Delta u\|_{L^2}$ for some $C_2 > 0$, which permits us to write

$$\begin{aligned}\|u - u_{h,H}\|_{H^1} &= \|u - u_h + u_h - u_{h,H}\|_{H^1} \\ &\leq \|u - u_h\|_{H^1} + \|u_h - u_{h,H}\|_{H^1} \\ &\leq CH \|\Delta u\|_{L^2}\end{aligned}$$

for some constant $C > 0$, from which the final result trivially follows. □

The proof of Lemma 5.1.7 provides a useful insight as to why we defined V_H as the orthogonal complement to W_h. We see that for any function $v_h \in S_0^1(\mathcal{T}_h)$ and the error $\epsilon_h = v_h - R_H v_h$ where R_H is the elliptic projection onto V_H, we have that $\epsilon_h \in W_h$. Recall the direct sum property from Lemma 5.1.6, which states that for any $v_h \in S_0^1(\mathcal{T}_h)$, there exist unique $v_H \in V_H$ and $w_h \in W_h$ such that $v_h = v_H + w_h$. It follows that $R_H v_h = v_H$. An intuitive interpretation is that due to the orthogonality of V_H and W_h, the elliptic projection $R_H v_h$ captures as much information of v_h as possible. We now prove the inverse inequality.

Lemma 5.1.8 (Inverse inequality for the corrected space V_H)
Let I_H denote an admissible quasi-interpolator for the finite element space $S_0^1(\mathcal{T}_H)$, and let $S_0^1(\mathcal{T}_h)$ denote a finite element space for which $S_0^1(\mathcal{T}_H) \subseteq S_0^1(\mathcal{T}_h)$. Assume that $S_0^1(\mathcal{T}_H)$ satisfies the inverse inequality

$$\|\nabla v_H\| \leq C_0 H^{-1} \|v_H\| \qquad \forall v_H \in S_0^1(\mathcal{T}_H) \tag{5.11}$$

for some $C_0 > 0$ independent of H. Then the corrected space V_H satisfies

$$\|\nabla \tilde{v}_H\| \leq C_0 C_{I_H} H^{-1} \|\tilde{v}_H\| \qquad \forall \tilde{v}_H \in V_H, \tag{5.12}$$

where V_H denotes the corrected space with respect to $S_0^1(\mathcal{T}_H)$, I_H and $S_0^1(\mathcal{T}_h)$, and C_{I_H} is the constant from Definition 5.1.1.

Proof. The proof follows the second half of the proof for Lemma 2.1 in [12]. Recall that $(\nabla(1 - \mathcal{C})v_H, \nabla w_h) = 0$ for any $w_h \in W_H$ and $v_H \in S_0^1(\mathcal{T}_H)$. Furthermore, note that for any $\tilde{v}_H \in V_H$, we may write

$$\tilde{v}_H = I_H \tilde{v}_H - \mathcal{C} I_H \tilde{v}_H = (1 - \mathcal{C}) I_H \tilde{v}_H.$$

This further leads to

$$\begin{aligned}\|\nabla \tilde{v}_H\|^2 &= \|\nabla(1 - \mathcal{C}) I_H \tilde{v}_H\|^2 \\ &= (\nabla(1 - \mathcal{C}) I_H \tilde{v}_H, \nabla I_H \tilde{v}_H)) - \overbrace{(\nabla(1 - \mathcal{C}) I_H \tilde{v}_H, \nabla \mathcal{C} I_H \tilde{v}_H))}^{=0} \\ &= (\nabla \tilde{v}_H, \nabla I_H \tilde{v}_H) \\ &\leq \|\nabla \tilde{v}_H\| \, \|\nabla I_H \tilde{v}_H\| .\end{aligned}$$

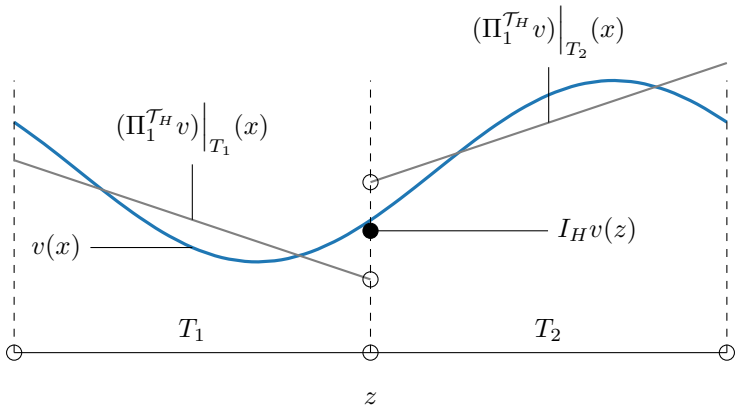


Figure 5.1: An illustration of the local admissible quasi-interpolator from Definition 5.2.1 in one dimension. At each node $z \in \mathcal{T}_H$, the values of the element-local L^2 projection $\Pi_1^{\mathcal{T}_H} v$ in each element T_1 and T_2 is averaged at $x = z$ to produce the final result $I_H v(z)$.

Using the inverse inequality assumption (5.11) and finally the quasi-interpolator L^2 stability (5.3), we obtain the inequality

$$\|\nabla \tilde{v}_H\| \leq \|\nabla I_H \tilde{v}_H\| \leq C_0 H^{-1} \|I_H \tilde{v}_H\| \leq C_0 C_{I_H} H^{-1} \|\tilde{v}_H\|,$$

which concludes the proof. □

We close this section by noting that while the space V_H does not preserve the inverse inequality of $S_0^1(\mathcal{T}_H)$ up to the same constant, the additional factor C_{I_H} is usually relatively small in practice, so that the stability region of the Leapfrog method is often largely preserved.

5.2 A local admissible quasi-interpolator

In Section 5.1, the definition of the corrected space V_H relied on the existence of an *admissible quasi-interpolator*. Here we will present one example of a quasi-interpolator that satisfies these properties. Moreover, it has the important property that the projection is local, in the sense that the projected values in a subset of the domain only depend on information from the geometrical vicinity of the subset. This is crucial for implementation efficiency, as we shall see later.

Definition 5.2.1 (A local quasi-interpolator)
Let $P_1(\mathcal{T}_H)$ denote the space of (possibly discontinuous) piecewise affine functions on $\mathcal{T}_H$, defined by

$$P_1(\mathcal{T}_H) := \{w \in L^2(\Omega) \mid w\big|_K \text{ is affine } \forall K \in \mathcal{T}_H\}, \tag{5.13}$$

and define the nodal averaging operator $J_1 : P_1(\mathcal{T}_H) \to S_0^1(\mathcal{T}_H)$ for any $w_H \in P_1(\mathcal{T}_H)$ by the property

$$J_1 w_H(z) := \frac{1}{\#\{K \in \mathcal{T}_H \mid z \in K\}} \sum_{K \in \mathcal{T}_H | z \in K} w_H\big|_K(z) \tag{5.14}$$

for any interior node z in $\mathcal{T}_H$. Let $\Pi_1^{\mathcal{T}_H} : H_0^1(\Omega) \to P^1(\mathcal{T}_H)$ denote the standard L^2 projection onto $P_1(\mathcal{T}_H)$ defined by the relation

$$\left(\Pi_1^{\mathcal{T}_H} v, p_H\right) = (v, p_H) \qquad \forall p_H \in P_1(\mathcal{T}_H) \tag{5.15}$$

for any $v \in H_0^1(\Omega)$. Then an operator $I_H : H_0^1(\Omega) \to S_0^1(\mathcal{T}_H)$ is defined by

$$I_H v := J_1(\Pi_1^{\mathcal{T}_H} v) \qquad \forall v \in H_0^1(\Omega). \tag{5.16}$$

To get an intuitive sense of how the operator I_H defined in Definition 5.2.1 works, an illustration is provided in Figure 5.1. Here we see that for any node $z \in \mathcal{T}_H$, the resulting value of $I_H v\,(z)$ for some $v \in H_0^1(\Omega)$ is obtained by averaging the element-local L^2 projections $\Pi_1^{\mathcal{T}_H} v\Big|_K$ at $x = z$ for each $K \in \mathcal{T}_H$. The illustration is one-dimensional, but the same idea generalizes to higher dimensions.

Lemma 5.2.2
The operator I_H defined in Definition 5.2.1 *is an admissible quasi-interpolator for the finite element space $S_0^1(\mathcal{T}_H)$ generated from any quasi-uniform mesh $\mathcal{T}_H$.*

Proof. The proof of the approximation and stability properties is somewhat lengthy, so we instead refer the reader to [12] for this proof. Here we will only briefly prove that I_H is a surjective projection as required by Definition 5.1.1.

Note first that $S_0^1(\mathcal{T}_H) \subseteq P_1(\mathcal{T}_H)$, and hence $\Pi_1^{\mathcal{T}_H} v_H = v_H$ for any $v_H \in S_0^1(\mathcal{T}_H)$. Hence, it follows that

$$\sum_{K \in \mathcal{T}_H | z \in K} \Pi_1^{\mathcal{T}_H} v_H\Big|_K(z) = \#\{K \in \mathcal{T}_H \mid z \in K\} \cdot v_H(z),$$

and it trivially follows that $J_1(\Pi_1^{\mathcal{T}_H} v_H) = v_H$. □

5.3 A basis for the corrected space

In this section, we will see how one can construct a basis for the corrected space V_H. The main idea relies on the following result.

Lemma 5.3.1 (Basis for V_H)
Let $\Lambda \subseteq S_0^1(\mathcal{T}_H)$ be a basis for $S_0^1(\mathcal{T}_H)$. Then $(1-\mathcal{C})\Lambda \subseteq V_H$ is a basis for V_H.

Proof. We wish to show that $(1-C)\Lambda$ is linearly independent. Let $\Lambda = \{\lambda_j \mid j = 1, \dots, N_H\}$. Let $\alpha_j \in \mathbb{R}$ be such that

$$\sum_{j=1}^{N_H} \alpha_j (1-\mathcal{C})\lambda_j = 0.$$

It follows that

$$v_H := \sum_{j=1}^{N_H} \alpha_j \lambda_j = \sum_{j=1}^{N_H} \alpha_j \mathcal{C}\lambda_j =: w_h,$$

where $v_H \in S_0^1(\mathcal{T}_H)$ and $w_h \in W_h$. But from Lemma 5.1.3, $S_0^1(\mathcal{T}_H) \cap W_h = \{0\}$, and so $\alpha_j = 0$ for all j. It follows that $(1-\mathcal{C})\Lambda$ is linearly independent. Since $\dim V_H = \dim S_0^1(\mathcal{T}_H) = \#\Lambda$, we conclude that $(1-\mathcal{C})\Lambda$ is a basis for V_H.

□

Hence, in order to form a basis of V_H, the general idea is to take the Lagrangian basis functions $\lambda_{H,z}$ of $S_0^1(\mathcal{T}_H)$ and for each basis function compute the corrector $\mathcal{C}\lambda_{H,z}$ by solving (5.5). However, this straightforward approach is not without issues. In order to take steps towards a practically feasible method we will need to make sure that we can obtain basis functions which have local support.

5.3.1 Localization

One major disadvantage of the approach we have presented so far is that (5.5) is a global problem, in the sense that for each corrector, one must take the entire domain into account, and that $\mathcal{C}v_H$ for any $v_H \in S_0^1(\mathcal{T}_H)$ has global support in the general case. This further implies that the stiffness and mass matrices associated with V_H are unlikely to have favorable sparsity patterns. As a result, the method, as presented thus far, is simply infeasible for actual computation.

Luckily, there exists a remedy. It has been observed that the correctors $\mathcal{C}\lambda_{H,z}$ decay at an exponential rate outside of the support of $\lambda_{H,z}$ [24]. This suggests that a local

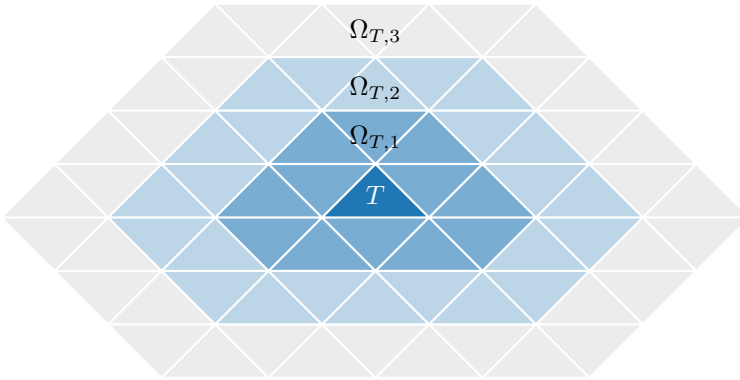


Figure 5.2: Illustration of local patches $\Omega_{T,m}$ around a triangle T for different values of m. Each increment of m adds another layer to the patch.

approximation might be possible, and indeed, that is the case. While the most straightforward approach would be to simply truncate correctors, it is instead advocated in [12] to use the localization procedure introduced in [25], which is based on *oversampling*. The idea is to solve a localized corrector problem in a local patch around each element $T \in \mathcal{T}_H$, where we replace W_h with a truncated subspace local to the patch. The patch size is determined by an oversampling parameter m, which is a new experimental parameter that must be chosen appropriately large to give sufficient accuracy. Summing up the individual solutions from each patch then constitute the localized correctors.

Definition 5.3.2 (Truncated kernel spaces)
For each $T \in \mathcal{T}_H$, we define the *truncated kernel space*

$$W_h(\Omega_{T,m}) := \{w_h \in W_h \mid \operatorname{supp}(w_h) \subseteq \Omega_{T,m}\}, \tag{5.17}$$

where $\Omega_{T,m}$ is the m-th order patch defined by

$$\Omega_{T,m} = \cup\left\{ K \in \mathcal{T}_H \mid \begin{array}{c} \exists K_0, \ldots, K_m \in \mathcal{T}_H \text{ with } K_0 = T, K_m = K \\ \text{and } K_j \cap K_{j+1} \neq \emptyset \text{ for all } j = 0, \ldots, m-1 \end{array} \right\}. \tag{5.18}$$

Remark. The definition of $\Omega_{T,m}$ looks more complicated than it really is. Figure 5.2 illustrates how the patch $\Omega_{T,m}$ may look like. Here it is observed that each increment of m merely adds another layer of neighboring triangles to the patch.

Definition 5.3.3 (Localized basis correctors for W_h)
Given any standard Lagrangian basis function $\lambda_{H,z} \in S_0^1(\mathcal{T}_H)$ associated with each internal node z, we define the m-th order localized approximation

$$\mathcal{C}_m \lambda_{H,z} := \sum_{T \in \mathcal{T}_H} \mathcal{C}_{T,m} \lambda_{H,z}, \tag{5.19}$$

where $\mathcal{C}_{T,m}\lambda_{H,z} \in W_h(\Omega_{T,m})$ solves the corrector problem

$$(\nabla \mathcal{C}_{T,m}\lambda_{H,z}, \nabla w_h) = \int_T \nabla \lambda_{H,z} \cdot \nabla w_h \, \mathrm{d}x \qquad \forall w_h \in W_h(\Omega_{T,m}). \tag{5.20}$$

While Definition 5.3.3 defines $\mathcal{C}_m\lambda_{H,z}$ as a sum over *all* elements in $\mathcal{T}_H$, note that if T is outside of the support of $\lambda_{H,z}$, the right-hand side of (5.20) is identically zero, and with the homogeneous Dirichlet conditions that are imposed on the problem, this implies that $\mathcal{C}_{T,m}\lambda_{H,z} = 0$ for any T which does not contain z. Hence, in practice, one only needs to sum the element correctors for elements for which z is contained in the element. When $\mathcal{T}_H$ is a (shape-)regular mesh like here, the number of such elements for any given z is bounded by a constant that only depends on the mesh family, and hence $\mathcal{O}(1)$. As a result, the number of corrector problems that need to be solved is $\mathcal{O}(N_H)$, where as before $N_H = \dim S_0^1(\mathcal{T}_H) = \dim(V_H)$.

The localization procedure brings up two important problems that need to be solved. Since the localization merely approximates the actual correctors $\mathcal{C}\lambda_{H,z}$, we need to determine the error associated with the procedure. Moreover, the oversampling parameter m is a new discretization parameter that needs to be chosen such that the localized basis well approximates the global basis of V_H. The next lemma addresses the first problem, and we will have more to say on the second.

Lemma 5.3.4 (Error of the localized basis functions)
Assume that $\mathcal{T}_h$ is (shape) regular and that I_H is the local admissible quasi-interpolator from Definition 5.2.1. *Then there exist constants $\beta > 0$ and $C > 0$ such that*

$$\|\nabla(\mathcal{C}\lambda_{H,z} - \mathcal{C}_m\lambda_{H,z})\| \leq Ce^{-\beta m} \|\nabla \lambda_{H,z}\| \tag{5.21}$$

for any oversampling parameter m and Lagrangian basis function $\lambda_{H,z} \in S_0^1(\mathcal{T}_H)$.

Proof. Proofs can be found in [26][25][24]. Here we present the formulation used in [12]. □

Lemma 5.3.4 demonstrates that by choosing a suitable oversampling parameter m, an arbitrarily accurate approximation of the global basis can be attained. However,

because larger m corresponds to increased computational complexity and density of the system matrices, we want to choose m as small as possible while still maintaining sufficient accuracy. To get an idea of how m needs to be chosen relative to H, we note that if $H < 1$ and we choose $m \geq \frac{1}{\beta}|\ln H|$, then

$$\|\nabla(\mathcal{C}\lambda_{H,z} - \mathcal{C}_m\lambda_{H,z})\| \leq CH\,\|\nabla\lambda_{H,z}\|\,.$$

This isn't a truly rigorous result, because $\lambda_{H,z}$ depends on H, but it does provide some intuition for the relationship between m and H. Unfortunately the minimal m corresponding to a given H which admits sufficient accuracy can not in general be determined in advance. Instead, it needs to be determined experimentally.

The above approximation bounds justifies using a local, oversampled approximation of V_H. Since this approximation is what we will work with when we discuss an efficient implementation, we provide the following definition.

Definition 5.3.5 (Localized corrected space)
The localized corrected space is defined by

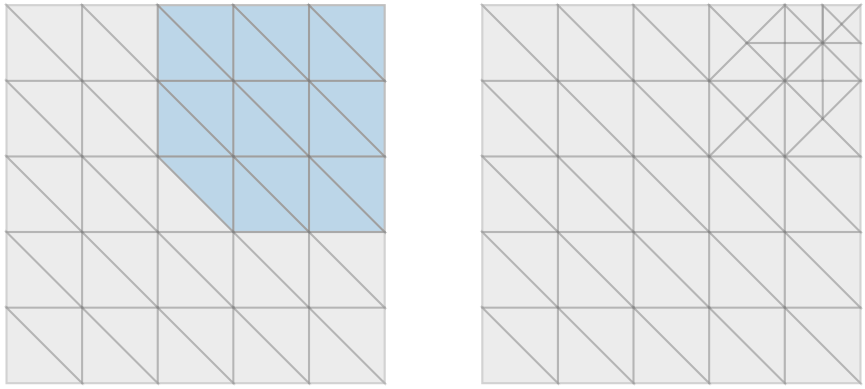
$$V_H^m := \text{span}\{\lambda_{H,z} - \mathcal{C}_m\lambda_{H,z} \mid z \text{ an interior node of } \mathcal{T}_H\}, \tag{5.22}$$

where $\mathcal{C}_m\lambda_{H,z}$ is the localized corrector from Definition 5.3.3.

We have so far not actually proved that the localization procedure actually maintains the desired convergence rate of $\mathcal{T}_h$, nor have we shown that the inverse inequality of $\mathcal{T}_H$ is maintained in V_H^m. While it looks as if this can be shown by piecing together results from papers on numerical homogenization methods, it is not done in [12], and we will also not do it here, as we would rather want to prioritize the study of the practical aspects of the method.

5.3.2 Support of basis correctors in locally refined meshes

The previous section demonstrated how a basis of a local approximation of V_H can be obtained. However, the original basis of V_H exhibits a property that can be exploited for a more efficient implementation in the case when $\mathcal{T}_h$ only *locally* refines $\mathcal{T}_H$, in the sense that only a comparatively small number of elements in $\mathcal{T}_H$ are actually refined. It turns out that the fine-scale functions in the kernel space W_h vanish within any non-refined coarse triangle for which all of its neighboring triangles in $\mathcal{T}_H$ are also not refined. The property is illustrated by Figure 5.3, and precisely defined by the following lemma.



(a) Quasi-uniform mesh $\mathcal{T}_H$. Shaded region denotes support of correctors.

(b) Local refinement $\mathcal{T}_h$ of $\mathcal{T}_H$.

Figure 5.3: Support of correctors in domain with limited local mesh refinement.

Lemma 5.3.6
Assume that I_H satisfies the local stability property

$$\|I_H v\|_{L^2(T)} \leq C \|v\|_{L^2(\Omega_T)} \qquad \forall v \in V \tag{5.23}$$

for all $T \in \mathcal{T}_H$ and $\Omega_T = \cup\{K \in \mathcal{T}_H \mid K \cap T \neq \emptyset\}$. Then, for any $w_h \in W_h$, we have that

$$w_h\big|_{\tilde{\Omega}} = 0, \tag{5.24}$$

where $\tilde{\Omega} = \cup\{T \in \mathcal{T}_H \mid K \in \mathcal{T}_H \cap \mathcal{T}_h \, \forall K \in \mathcal{T}_H$ for which $K \cap T \neq \emptyset\}$.

Proof. The proof closely follows the proof of Proposition 4.1 in [12], but the details are elaborated on for the benefit of the reader. Let $\mathcal{N} := \mathcal{N}(\mathcal{T}_h) \setminus \mathcal{N}(\mathcal{T}_H) \setminus \partial\Omega$ denote the set of interior nodes in $\mathcal{T}_h$ that are not in $\mathcal{T}_H$. We define $w_y := \lambda_{h,y} - I_H \lambda_{h,y}$ for any $y \in \mathcal{N}$. Here $\lambda_{h,y} \in S_0^1(\mathcal{T}_h)$ denotes the fine-scale Lagrangian basis function associated with node y in the fine mesh $\mathcal{T}_h$. Evidently, $w_y \in W_h$. We wish to show that all the w_y constitute a basis of W_h. To this end, we claim that $\{w_y\}_{y \in \mathcal{N}}$ is linear independent. We use an argument similar to that of the proof of Lemma 5.3.1, and introduce $\alpha_y \in \mathbb{R}$ such that

$$\sum_{y \in \mathcal{N}} \alpha_y w_y = 0.$$

From the definition of w_y, this lets us write

$$S_0^1(\mathcal{T}_h) \ni v_h := \sum_{y \in \mathcal{N}} \alpha_y \lambda_{h,y} = \sum_{y \in \mathcal{N}} \alpha_y I_H \lambda_{h,y} := v_H \in S_0^1(\mathcal{T}_H).$$

Since $v_h \in S_0^1(\mathcal{T}_H)$ and $\lambda_{h,y}(z_H) = 0$ for any coarse node $z_H \in \mathcal{T}_H$, we have that $v_h(z_H) = 0$ and consequently $v_h = 0$. Since $\{\lambda_{h,y} \mid y \in \mathcal{N}\}$ is linearly independent, it follows that $\alpha_y = 0$ for all $y \in \mathcal{N}$. Consequently, the functions w_y are linearly independent. Moreover, by a simple dimensional argument, it follows that the functions w_y constitute a basis of W_h.

Next, we first observe that if $T \in \tilde{\Omega}$, we have that $\lambda_{h,y}\big|_{\Omega_T} = 0$ for all $y \in \mathcal{N}$. To see this, it is enough to realize that since elements in $\tilde{\Omega}$ are not refined, $\tilde{\Omega} \cap \mathcal{N} = \emptyset$. This leads to $\operatorname{supp} \lambda_{h,y} \cap \tilde{\Omega} = \emptyset$ for any $y \in \mathcal{N}$.

Using this property, we may use (5.23) to write

$$\begin{aligned} \|w_y\|_{L^2(T)} = \|\lambda_{h,y} - I_H \lambda_{h,y}\|_{L^2(t)} &\leq \|\lambda_{h,y}\|_{L^2(T)} + \|I_H \lambda_{h,y}\|_{L^2(T)} \\ &\leq C_0 \|\lambda_{h,y}\|_{L^2(\Omega_T)} \\ &= 0, \end{aligned}$$

for any $T \in \tilde{\Omega}$. For any $w_h \in W_h$, we write $w_h = \sum_{y \in \mathcal{N}} \beta_y w_y$ for some $\beta_y \in \mathbb{R}$ for each $y \in \mathcal{N}$, and we finally obtain

$$\|w_h\|_{L^2(\tilde{\Omega})} = \sum_{T \in \tilde{\Omega}} \|w_h\|_{L^2(T)} = \sum_{T \in \tilde{\Omega}} \left\| \sum_{y \in \mathcal{N}} \beta_y w_y \right\|_{L^2(T)} \leq \sum_{T \in \tilde{\Omega}} \sum_{y \in \mathcal{N}} |\beta_y| \, \|w_y\|_{L^2(T)} = 0,$$

from which the result follows. □

Lemma 5.3.6 tells us that the correctors vanish in regions where the coarse mesh $\mathcal{T}_H$ is not refined. This is an important result for two reasons. First, depending on the application, it allows us to avoid corrector computation altogether for entire regions of the domain. This saves some time when computing the correctors. However, an arguably perhaps more important effect is that in practice it may significantly reduce the density of the mass- and stiffness matrices associated with V_H^m. When computing correctors in regions for which the correctors vanish in *exact* arithmetic, the insidious properties of floating-point arithmetic will lead to (barely) non-zero correctors in these regions. While some filtering of correctors depending on magnitude is possible, one cannot easily determine when a corrector is "small enough" to be considered zero, and so it is a much better approach to simply not compute these correctors - which one knows should be precisely zero in exact arithmetic - in the first place.

5.4 Application to the wave equation

We now summarize the results of the previous sections in the context of the wave equation.

The first corollary shows that the Leapfrog method is stable in V_H if it satisfies a CFL condition comparable to that of $S_0^1(\mathcal{T}_H)$. This is a straightforward consequence of Lemma 5.1.8 and Theorem 3.2.6.

Corollary 5.4.1 (CFL condition for V_H)
Assume that $S_0^1(\mathcal{T}_H)$ has the inverse property from Definition 2.2.3 *with constant C_I, and assume that Δt is chosen sufficiently small such that*

$$1 - \frac{C_{I_H}^2 C_I^2 (\Delta t)^2}{2H^2} \geq \lambda > 0 \tag{5.25}$$

for some $\lambda \in (0,1)$ and C_{I_H} from Definition 5.1.1. *Then the Leapfrog method is stable in the space V_H in the sense of Theorem* 3.2.6.

We are finally ready to reconcile the results of Chapter 4 with the results of this chapter. By combining the estimate for the elliptic projection onto $S_0^1(\mathcal{T}_h)$ from Theorem 4.2.5 with the regularity result on Δu from Lemma 4.2.7 and the best approximation error for V_H in Lemma 5.1.7 together with the error estimates for the Leapfrog method in Theorem 3.2.7, we see that under the (usually reasonable) assumptions of these results, the solution $u_h^n \in V_H$ by the Leapfrog method satisfies

$$\|u(t^n) - u_h^n\|_{H^1} \leq C(H + (\Delta t)^2) \tag{5.26}$$

for some constant $C > 0$ independent of H and Δt. We note also that the same applies to the Crank-Nicolson method, but with the estimate in the same style as that of Corollary 3.3.7.

Chapter 6

Efficient corrector computation

In Chapter 5, we introduced a method due to Peterseim and Schedensack which was shown to combine the favorable inverse inequality of a coarse space $S_0^1(\mathcal{T}_H)$ with the convergence rate of a fine space $S_0^1(\mathcal{T}_h)$ constructed from a local refinement $\mathcal{T}_h$. The method was introduced in [12]. However, many of the practical aspects of this method are not discussed in their paper. This chapter is largely dedicated to covering the remaining considerations that must be taken into account when implementing the method.

Our main objective will be to determine a suitable algebraic formulation of the localized corrector problem from Definition 5.3.3, and then discuss appropriate solvers for the resulting linear system.

The method in question originates from numerical homogenization techniques. In this domain, one usually deals with a coarse mesh $\mathcal{T}_H$ and a *global* refinement $\mathcal{T}_h$. Apart from the fact that $\mathcal{T}_h$ in our case is a local refinement, the method presented in Chapter 5 is otherwise almost identical to the Localized Orthogonal Decomposition (LOD) method for elliptic multi-scale problems in heterogeneous media. Engwer et al. [27] discuss an efficient implementation of the LOD, and show how to transform the corrector problems into suitable algebraic formulations, culminating in a linear saddle-point formulation of the corrector problem.

We will use a very similar approach as that described by Engwer et. al, but because $\mathcal{T}_h$ is only a local refinement in our case, the algebraic formulation used for the LOD may not in general be applied directly to the localized corrector problems from Definition 5.3.3. We propose a modified formulation of the linear saddle-point problem which overcomes this problem, enabling the fast computation of correctors.

We go on to discuss two different types of solvers for the saddle point problem. The first makes use of the *Schur complement* of the stiffness matrix associated with the

standard finite element space on the local patch. We conclude that for our problem, this approach is particularly appropriate when paired with a direct sparse solver for the stiffness matrix, such as Sparse Cholesky.

While direct solvers are very fast for problems of small to medium size, they tend to become too expensive for larger problems. With this in mind, we also propose a second method which relies on solving the saddle point problem directly with the iterative solver GMRES [13]. In order for GMRES to attain acceptable convergence rates, a suitable preconditioner is a necessity. We show how we can exploit the block structure of the saddle point problem to construct an appropriate (block) preconditioner for the problem.

We conclude the discussion with a high-level overview of the procedure to compute correctors for all basis functions and hence fully describe the space V_H^m as defined in Definition 5.3.5. Once all correctors have been obtained, we show that the computation of system matrices and load vectors is very straightforward.

Finally, as part of this thesis, an experimental prototype software library named `crest` is also released to the public. A brief description of this library is given at the end of the chapter.

Note that in this chapter, we will assume that $\Omega \subseteq \mathbb{R}^2$ for the sake of simplicity. The generalization to 3D is straightforward.

6.1 The local quasi-interpolator in matrix form

The interpolator presented in Definition 5.2.1 is a linear operator, and as such we can come up with a matrix representation for it if we restrict it to the domain $S_0^1(\mathcal{T}_h)$. This will be particularly useful when we wish to compute the basis correctors in later sections. From here on, I_H refers to this particular quasi-interpolator.

As we have seen in Definition 5.2.1, I_H is defined in terms of the composition of $\Pi_1^{\mathcal{T}_H}$ - the projection into the space of (possibly discontinuous) piecewise affine functions on $\mathcal{T}_H$ - and J_1, the nodal averaging operator. In order to construct a matrix that represents the action of applying I_H, we will first construct matrices that represent each of these linear operators when constrained to the domain $S_0^1(\mathcal{T}_h)$.

Definition 6.1.1 (Standard basis for $P_1(\mathcal{T}_H)$)
Assume that for every $T \in \mathcal{T}_H$, each local vertex in T is labeled by $l = 1, 2, 3$, and let $z_{T,l}$ refer to the l-th local vertex in T. Then $p_{T,l}$ for $T \in \mathcal{T}_H$ and $l = 1, 2, 3$ denotes the standard basis for $P_1(\mathcal{T}_H)$, defined by the properties

$$p_{T,l}\big|_K = 0 \text{ if } T \neq K, \tag{6.1}$$

$$p_{T,l}(z_{K,k}) = \begin{cases} 1 \text{ if } K = T \text{ and } k = l \\ 0 \text{ otherwise} \end{cases} . \tag{6.2}$$

Remark. The basis functions $p_{T,l}$ can be seen as the result of slicing the standard Lagrangian basis functions for the standard linear finite element space $S_0^1(\mathcal{T}_H)$ along element edges.

Lemma 6.1.2 (Matrix representation of the P_1 projection)
Let $n := 3\#\mathcal{T}_H$ where $\#\mathcal{T}_H$ denotes the number of elements in $\mathcal{T}_H$, and let $\lambda_{h,j}$ denote the standard Lagrangian basis function associated with the node labeled j in $S_0^1(\mathcal{T}_h)$. The matrix representation of $\Pi_1^{\mathcal{T}_H}\big|_{S_0^1(\mathcal{T}_h)}$ with respect to the standard (Lagrange) bases for $S_0^1(\mathcal{T}_h)$ and $P_1(\mathcal{T}_H)$ is given by $\hat{P} \in \mathbb{R}^{n \times N_h}$, defined by

$$\hat{P} := P_M^{-1} B, \tag{6.3}$$

where the matrices $P_M \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times N_h}$ are defined by

$$P_{M,(T,i),(K,j)} = (p_{K,j}, p_{T,i}) \text{ for } T, K \in \mathcal{T}_H \text{ and } i, j = 1, 2, 3, \tag{6.4}$$

$$B_{(T,i),j} = (\lambda_{h,j}, p_{T,i}) \text{ for } T \in \mathcal{T}_H \text{ and } i = 1, 2, 3 \text{ and } j = 1, \ldots, N_h. \tag{6.5}$$

Remark. Note that P_M has block diagonal structure if the coefficients for each element are grouped together, and so is very simple to invert. For example, in 2D, the diagonal blocks have the size 3×3.

Proof of Lemma 6.1.2. For any $v_h \in S_0^1(\mathcal{T}_h)$, we may write $v_h = \sum_{j=1}^{N_h} \xi_j \lambda_{h,j}$, and we similarly express $\Pi_1^{\mathcal{T}_H} v_h \in P_1(\mathcal{T}_H)$ as $\Pi_1^{\mathcal{T}_H} v_h = \sum_{K \in \mathcal{T}_H} \sum_{j=1}^{3} \mu_{K,j}\, p_{K,j}$. Inserting these expressions into (5.15) for $p_H = p_{T,i}$, we obtain

$$P_M \mu = B \xi,$$

from which the result follows. □

Lemma 6.1.3 (Matrix representation of the nodal averaging operator J_1)
The matrix representation of J_1 with respect to the standard (Lagrange) bases for $S_0(\mathcal{T}_H)$ and $P_1(\mathcal{T}_H)$ is given by $\hat{J} \in \mathbb{R}^{N_H \times 3\#\mathcal{T}_H}$, defined by

$$\hat{J}_{i,(T,j)} := \begin{cases} (\#\{K \in \mathcal{T}_H \mid z_i \in K\})^{-1} \text{ if } z_i = z_{T,j} \\ 0 \text{ otherwise} \end{cases}. \tag{6.6}$$

Proof. Let $p_H = \sum_{T\in\mathcal{T}_H} \sum_{j=1}^{3} \mu_{T,j} p_{T,j}$, and let $j_H \in S_0^1(\mathcal{T}_H)$ represent the result of applying the operator, written $j_H = \sum_{i=1}^{N_H} \gamma_i \lambda_{H,i}$ for basis functions $\lambda_{H,i}$. Then, for any interior node $z_i \in \mathcal{T}_H$,

$$\begin{aligned} \gamma_i = Jw_H(z_i) &= (\#\{K \in \mathcal{T}_H \mid z_i \in K\})^{-1} \sum_{K\in\mathcal{T}_H | z_i \in K} w_H(z_i) \\ &= (\#\{\dots\})^{-1} \sum_{K\in\mathcal{T}_H | z_i\in K} \sum_{T\in\mathcal{T}_H} \sum_{j=1}^{3} \mu_{T,j} p_{T,j}(z_i) \\ &= \sum_{T\in\mathcal{T}_H | z_i \in T} \sum_{j=1}^{3} \rho_{T,j} \underbrace{\frac{p_{T,j}(z_i)}{\#\{\dots\}}}_{\hat{J}_{i,(T,j)}}. \end{aligned}$$

The realization that $p_{T,j}(z_i) = 1$ if and only if $z_i = z_{T,j}$ and otherwise 0 concludes the proof.

□

Lemma 6.1.4 (Matrix representation of the local admissible quasi-interpolator)
The matrix representation of $I_H\big|_{S_0^1(\mathcal{T}_h)}$ with respect to the standard Lagrange bases for $S_0^1(\mathcal{T}_h)$ and $S_0^1(\mathcal{T}_H)$ is given by $\hat{I}_H \in \mathbb{R}^{N_H \times N_h}$, defined by

$$\hat{I}_H := \hat{J}\hat{P}, \tag{6.7}$$

with $\hat{J}$ as defined in Lemma 6.1.3 and $\hat{P}$ as defined in Lemma 6.1.2. Moreover,

$$\operatorname{rank}(\hat{I}_H) = N_H = \dim S_0^1(\mathcal{T}_H). \tag{6.8}$$

Proof. (6.7) is a simple consequence of the fact that I_H is defined as the composition of J_1 and $\Pi_1^{\mathcal{T}_H}$. That $\hat{I}_H$ has full row rank is due to the fact that I_H is surjective by definition of the admissible quasi-interpolator in Definition 5.1.1. □

6.2 An algebraic formulation for the corrector problem

Because the localization procedure discussed in 5.3.1 introduces additional complexities to the corrector computation procedure, we will first work with the global corrector problem defined in Definition 5.1.4. Having presented a fully algebraic formulation for the global problem, we will show how to make some modifications to obtain a similar algebraic formulation for the localized problem defined in Definition 5.3.3. The method presented here for the global problem is essentially the same as the one advocated in [27], but this approach does not consider issues of rank-deficiency related to localization that may occur if the mesh $\mathcal{T}_h$ is only a local (i.e. not global) refinement of $\mathcal{T}_H$, which we will discuss later in this section.

As was already explained in previous chapters, we have no *a priori* knowledge of a basis for W_h, the kernel of $I_H\big|_{S_0^1(\mathcal{T}_h)}$. To cope with this situation, we will instead constrain the corrector such that it lies in W_h. We will first present the main result which we will need for a practical implementation. The accompanying proof should hopefully provide some of the intuition.

Lemma 6.2.1 (Constrained corrector problem)
Given any Lagrangian basis function $\lambda_{H,z} \in S_0^1(\mathcal{T}_H)$ and its associated corrector $\mathcal{C}\lambda_{H,z} \in W_h \subseteq S_0^1(\mathcal{T}_h)$, let $\xi \in \mathbb{R}^{N_h}$ and $\eta \in \mathbb{R}^{N_h}$ satisfy

$$\lambda_{H,z} = \sum_{j=1}^{N_h} \xi_j \lambda_{h,j}, \qquad \mathcal{C}\lambda_{H,z} = \sum_{j=1}^{N_h} \eta_j \lambda_{h,j}, \tag{6.9}$$

where $\lambda_{h,j} \in S_0^1(\mathcal{T}_h)$ denotes the standard fine-scale Lagrangian basis function associated with node j. Then there exists a unique $\kappa \in \mathbb{R}^{N_H}$ such that

$$\mathcal{A}_{h,H} \begin{pmatrix} \eta \\ \kappa \end{pmatrix} = \begin{pmatrix} A_h & \hat{I}_H^T \\ \hat{I}_H & 0 \end{pmatrix} \begin{pmatrix} \eta \\ \kappa \end{pmatrix} = \begin{pmatrix} A_h \xi \\ 0 \end{pmatrix}. \tag{6.10}$$

Here A_h denotes the stiffness matrix of $a(\cdot,\cdot)$ associated with the standard basis for $S_0^1(\mathcal{T}_h)$.

Remark. The vector κ is simply a Lagrange multiplier, and has no real use other than as an auxiliary variable.

Proof of Lemma 6.2.1. We first wish to show that the block matrix $\mathcal{A}_{h,H}$ is invertible. We note that the matrix $S := I_H A_h^{-1} I_H^T$ is symmetric positive definite because A_h is s.p.d. and $\hat{I}_H$ has full row rank. We denote the identity matrix of dimension N_H as

E_{N_H}, and similarly for E_{N_h}. We have the block triangular decomposition

$$\begin{pmatrix} A_h & \hat{I}_H^T \\ \hat{I}_H & 0 \end{pmatrix} = \begin{pmatrix} A_h & 0 \\ \hat{I}_H & E_{N_H} \end{pmatrix} \begin{pmatrix} E_{N_h} & A_h^{-1} I_H^T \\ 0 & -S \end{pmatrix},$$

from which we deduce that

$$\det(\mathcal{A}_{h,H}) = \det(A_h)\det(-S) \neq 0,$$

and hence $\mathcal{A}_{h,H}$ is invertible. It follows that there exist unique η, κ that solve (6.10). Next, we wish to show that η is also the unique solution to the corrector problem in Definition 5.1.4. Let $w \in R^{N_h}$ denote the weights of an arbitrary function $w_h = \sum_i w_i \lambda_{h,i} \in W_h$, so that $\hat{I}_H w = 0$. Writing out parts of the linear system in (6.10), we have

$$w^T(A_h\eta - A_h\xi) = -w^T(\hat{I}_H^T\kappa) = -\kappa(\hat{I}_H^T w) = 0 \implies w^T A_h \eta = w^T A_h \xi.$$

Rewriting each side of the resulting expression, we have

$$w^T A_h \eta = \sum_i \sum_j w_i (A_h)_{ij} \eta_j = \sum_i \sum_j a(\eta_j \lambda_{h,j}, w_i \lambda_{h,i}) = a(\mathcal{C}\lambda_{H,z}, w_h),$$
$$w^T A_h \xi = \sum_i \sum_j w_i (A_h)_{ij} \xi_j = \sum_i \sum_j a(\eta_j \lambda_{h,j}, w_i \lambda_{h,i}) = a(\lambda_{H,z}, w_h).$$

Together with the constraint $\hat{I}_H \eta = 0 \implies I_H \mathcal{C} \lambda_{H,z} = 0$, we have that η satisfies

$$\begin{aligned} a(\mathcal{C}\lambda_{H,z}, w_h) &= a(\lambda_{H,z}, w_h) \qquad \forall w_h \in W_h, \\ I_H \mathcal{C} \lambda_{H,z} &= 0, \end{aligned}$$

which is exactly the corrector problem from Definition 5.1.4. Since the corrector problem has a unique solution, we know that η coincides with the unique solution, and we can conclude the proof. □

An algebraic formulation for the localized problem

Lemma 6.2.1 transforms the abstract (global) corrector problem from Definition 5.1.4 into a linear system which can be solved numerically. In order to define a practically feasible method, however, we will need to determine a similar approach for the localized corrector problem as defined by Definition 5.3.3. We first introduce some notation.

Definition 6.2.2 (Patch-local meshes)
Given a patch $\Omega_{T,m} \subseteq \Omega$ and associated coarse and fine meshes $\mathcal{T}_H$ and $\mathcal{T}_h$, we define the patch-local meshes $\mathcal{T}_{H,T,m}$ and $\mathcal{T}_{h,T,m}$ by

$$\mathcal{T}_{H,T,m} := \{K \in \mathcal{T}_H \mid K \subseteq \bar{\Omega}_{T,m}\}, \tag{6.11}$$
$$\mathcal{T}_{h,T,m} := \{K \in \mathcal{T}_h \mid K \subseteq \bar{\Omega}_{T,m}\}. \tag{6.12}$$

The meshes $\mathcal{T}_{H,T,m}$ and $\mathcal{T}_{h,T,m}$ defined in Definition 6.2.2 are just the meshes of $\Omega_{T,m}$ that result from taking the elements in $\mathcal{T}_H$ and $\mathcal{T}_h$ that fit inside of $\Omega_{T,m}$. We see then that the truncated kernel space $W_h(\Omega_{T,m})$ is simply a subspace of the finite-element space $S_0^1(\mathcal{T}_{h,T,m})$. We now want to relate the fine-scale stiffness matrix $A_{h,T,m}$ associated with the fine-scale space $S_0^1(\mathcal{T}_{h,T,m})$ to the stiffness matrix A_h associated with the global fine-scale space $S_0^1(\mathcal{T}_h)$. We see that if we interpret the space $S_0^1(\mathcal{T}_{h,T,m})$ as a subspace of the larger space $S_0^1(\mathcal{T}_h)$ by extending each function $v_h \in S_0^1(\mathcal{T}_{h,T,m})$ to the whole of Ω by $v_h = 0$ outside of $\Omega_{T,m}$, we have that for any $v_h, v_h' \in S_0^1(\mathcal{T}_{h,T,m}) \subseteq S_0^1(\mathcal{T}_h)$, we may write

$$\begin{aligned} a(v_h, v_h') &= \sum_{z,z' \in \mathcal{N}(\mathcal{T}_h)} \xi_z \xi_{z'}' a(\lambda_{h,z}, \lambda_{h,z'}) \\ &= \sum_{z,z' \in \mathcal{N}(\mathcal{T}_{h,T,m})} \xi_z \xi_{z'}' a(\lambda_{h,z}, \lambda_{h,z'}) + \sum_{z,z' \notin \mathcal{N}(\mathcal{T}_{h,T,m})} \underbrace{\xi_z \xi_{z'}'}_{=0} a(\lambda_{h,z}, \lambda_{h,z'}) \\ &= \sum_{z,z' \in \mathcal{N}(\mathcal{T}_{h,T,m})} \xi_z \xi_{z'}' a(\lambda_{h,z}, \lambda_{h,z'}), \end{aligned}$$

which means that the local stiffness matrix $A_{h,T,m}$ is a submatrix of the global submatrix A_h, and can hence be assembled simply by taking the appropriate rows and columns corresponding to nodes that reside in $\mathcal{T}_{h,T,m}$. This is a convenient property, but not a strictly necessary one: the local matrix $A_{h,T,m}$ can be assembled for each corrector problem. This has both advantages and disadvantages. For one, one does not need to keep the entire global matrix A_h in memory. On the other hand, it may prove detrimental to performance. Matrix assembly is however usually a memory-bound process, so the cost of fetching a submatrix from A_h might very well be comparable to that of simply assembling it from scratch. The right choice depends on the application. For this thesis the submatrix approach was used.

In order to formulate the problem in terms of a saddle point problem as for the global problem in Lemma 6.2.1, we need to formulate the kernel constraint $I_H v_h = 0$ in terms of an appropriate matrix representation of I_H. We cannot use the global matrix $\hat{I}_H$ for two reasons: for one, it would be woefully inefficient, and second, the number of columns must be equal to the dimension of $S_0^1(\mathcal{T}_{h,T,m})$. In Chapter 5.2 we saw that the quasi-interpolator I_H is *local* in nature. We will now exploit this property.

Lemma 6.2.3
Assume that $\operatorname{supp} v_h \subseteq \Omega_{T,m}$ *for* $v_h \in S_0^1(\mathcal{T}_h)$. *Then*

$$I_H v_h(z) = 0$$

for all vertices $z \in \mathcal{N}(\mathcal{T}_H)$ *such that* $z \notin \bar{\Omega}_{T,m}$.

Remark. Recall that the value of $I_H v_h(z)$ corresponds exactly to the basis weight associated with the node z.

Proof of Lemma 6.2.3. Recall that $I_H = J_1 \circ \Pi_1^{\mathcal{T}_H}$. Since the projection $\Pi_1^{\mathcal{T}_H}$ is local for each element, we have that for all $K \in \mathcal{T}_H$ with $K \not\subset \bar{\Omega}_{T,m}$, $\Pi_1^{\mathcal{T}_H} v_h$ satisfies

$$\left(\Pi_1^{\mathcal{T}_H} v_h\Big|_K, p_H\right)_{L^2(K)} = (v_h, p_H)_{L^2(K)} = 0 \qquad \forall p_H \in P_1(\mathcal{T}_H),$$

and hence $\Pi_1^{\mathcal{T}_H} v_h\Big|_K = 0$ for all $K \in \mathcal{T}_H$ outside of the patch $\Omega_{T,m}$. Thus $\operatorname{supp} \Pi_1^{\mathcal{T}_H} v_h \subseteq \Omega_{T,m}$. The result now follows directly from the definition of J_1 applied to any vertex z outside of the local patch $\Omega_{T,m}$.

□

Since $v_h(z) = 0$ for $v_h \in S_0^1(\mathcal{T}_{h,T,m})$ and any vertex z not in the interior of $\Omega_{T,m}$, Lemma 6.2.3 tells us that we only need to consider the columns of $\hat{I}_H$ which correspond to interior vertices of $\mathcal{T}_{h,T,m}$ and rows that correspond to all vertices $z \in \mathcal{T}_{H,T,m}$ (including the boundary vertices). More precisely, this lets us define a localized version $\hat{I}_{H,T,m}$ of $\hat{I}_H$ by

$$\hat{I}_{H,T,m} = \hat{I}_H[\mathcal{I}, \mathcal{J}],$$

where we have used MATLAB-like index notation to denote the submatrix with respect to the index sets $\mathcal{I}$ and $\mathcal{J}$ defined by

$$\begin{aligned} \mathcal{I} &:= \{i \mid z_i \in \bar{\Omega}_{T,m} \text{ for } z_i \in \mathcal{N}(\mathcal{T}_H)\}, \\ \mathcal{J} &:= \{j \mid z_j \in \operatorname{int}(\Omega_{T,m}) \text{ for } z_j \in \mathcal{N}(\mathcal{T}_h)\}. \end{aligned}$$

Putting the above pieces together and proceeding as in the proof of Lemma 6.2.1, we eventually arrive at a saddle point problem of the form

$$\begin{pmatrix} A_{h,T,m} & \hat{I}_{H,T,m}^T \\ \hat{I}_{H,T,m} & 0 \end{pmatrix} \begin{pmatrix} \eta \\ \kappa \end{pmatrix} = \begin{pmatrix} b_{T,m} \\ 0 \end{pmatrix}.$$

Unfortunately, this is not quite correct. Recall that in the derivation of the algebraic formulation of the global problem, we relied on the fact that $\hat{I}_H$ has full row rank to show that the saddle point matrix is invertible, which we further used to justify the existence of the Lagrange multiplier κ. We claim now that the matrix $\hat{I}_{H,T,m}$ does not generally have full row rank. To see this, let us consider a simple counterexample. If $\mathcal{T}_{H,T,m} = \mathcal{T}_{h,T,m}$, we have from the above definitions that $\#\mathcal{I} > \#\mathcal{J}$. This implies that $\hat{I}_{H,T,m}$ has more rows than columns, and it follows that it cannot have full row rank. The reason that $\hat{I}_{H,T,m}$ may be rank-deficient whereas $\hat{I}_H$ has full row rank comes from the fact that $I_H v(x)$ is defined to be identically zero at the boundary of Ω, but in the localized case, there are no similar constraints on the value of $I_H v(x)$ at the boundary of $\Omega_{T,m}$.

The rank-deficiency of $\hat{I}_{H,T,m}$ presents a challenge for numerical computation of the corrector. One approach is to replace $\hat{I}_{H,T,m}$ by a related matrix with full row rank

which shares the null space of $\hat{I}_{H,T,m}$, but because $\hat{I}_{H,T,m}$ is sparse, this is generally quite difficult and possibly very expensive. At the outset of this thesis, the primary goal was the fast computation of correctors. With that in mind, a very simple approximation turned out to work remarkably well. The idea is essentially to remove the kernel constraint $I_H v_h$ at the boundary of $\Omega_{T,m}$ altogether, and instead only enforce the kernel constraint for the *interior* coarse nodes in $\mathcal{T}_{H,T,m}$. This way the new (perturbed) localized quasi-interpolator matrix $\tilde{I}_{H,T,m}$ always has full rank by virtue of Lemma 6.2.3. Note that this *perturbs* the system: the corrector $\mathcal{C}\lambda_{H,z}$ is no longer contained exactly in W_h. However, in practice this still seems to work very well, and it is likely that the error introduced can be mitigated by for example an additional layer of oversampling (incrementing m by 1), but the perturbation remains theoretically unjustified.

We note that $\tilde{I}_{H,T,m}$ corresponds exactly to the matrix representation of $I_H : H_0^1(\Omega_{T,m}) \to S_0^1(\mathcal{T}_{H,T,m})$ (note the local domain and codomain). We use this to state the final perturbed, localized corrector problem in algebraic form.

Definition 6.2.4 (Perturbed local corrector problem in algebraic form)
Given any Lagrangian basis function $\lambda_{H,z} \in S_0^1(\mathcal{T}_{H,T,m})$, denote by $\lambda_{h,j} \in S_0^1(\mathcal{T}_{h,T,m})$ the standard fine-scale Lagrangian basis function associated with node j and define $b_{T,m} \in \mathbb{R}^{\dim S_0^1(\mathcal{T}_{h,T,m})}$ by

$$(b_{T,m})_k := \int_T \nabla\lambda_{H,z}\nabla\lambda_{h,k} \qquad k = 1, \dots, \dim S_0^1(\mathcal{T}_{h,T,m}). \tag{6.13}$$

Let $\eta \in \mathbb{R}^{\dim S_0^1(\mathcal{T}_{h,T,m})}$ satisfy

$$\mathcal{A}_{h,H,T,m}\begin{pmatrix}\eta\\ \kappa\end{pmatrix} = \begin{pmatrix}A_{h,T,m} & \tilde{I}_{H,T,m}^T\\ \tilde{I}_{H,T,m} & 0\end{pmatrix}\begin{pmatrix}\eta\\ \kappa\end{pmatrix} = \begin{pmatrix}b_{T,m}\\ 0\end{pmatrix}. \tag{6.14}$$

for some $\kappa \in \mathbb{R}^{\dim S_0^1(\mathcal{T}_{H,T,m})}$. We then define the associated perturbed corrector $\tilde{\mathcal{C}}_{T,m}\lambda_{H,z} \in S_0^1(\mathcal{T}_{h,T,m})$ by

$$\tilde{\mathcal{C}}_{T,m}\lambda_{H,z} := \sum_{j=1}^{N_h} \eta_j \lambda_{h,j}, \tag{6.15}$$

In the above, $A_{h,T,m}$ denotes the stiffness matrix associated with the standard basis for $S_0^1(\mathcal{T}_{h,T,m})$ and $\tilde{I}_{H,T,m}$ is the matrix representation of the local operator $I_H : H_0^1(\Omega_{T,m}) \to S_0^1(\mathcal{T}_{H,T,m})$.

6.3 Linear solvers for the corrector problem

In this section, we will review several techniques for solving the linear systems that arise from the (perturbed) localized corrector problems, as defined by Definition 6.2.4. The techniques are generally applicable also for the global problem in Lemma 6.2.1, provided that the right-hand side is changed accordingly. We note that the problem is a classical symmetric saddle point system, on which there exists a vast amount of literature. For more background on the methods presented here, the reader is referred to [28], which is a survey of solution methods for linear saddle point problems.

For the sake of readability, we will refer to the matrices and vectors in (6.14) by their global equivalents. That is, we will denote by A_h the stiffness matrix $A_{h,T,m}$ for the *local* problem, and similarly $\tilde{I}_{H,T,m}$ and $\mathcal{A}_{h,H,T,m}$ will be denoted simply $\hat{I}_H$ and $\mathcal{A}_{h,H}$, respectively. Moreover, we will (re)define $N_H = \dim S_0^1(\mathcal{T}_{H,T,m})$ and $N_h = \dim S_0^1(\mathcal{T}_{h,T,m})$.

6.3.1 Schur complement reduction and sparse direct solvers

We first note that the localized system can be solved quite simply by a direct method such as Sparse LU. This actually works relatively well for small to medium size problems. We can however do better. To do so, we first require the notion of the *Schur complement*.

Definition 6.3.1 (The Schur complement of A_h)
The *Schur complement* S of A_h with respect to the linear system (6.14) is defined by

$$S = \hat{I}_H A_h^{-1} \hat{I}_H^T. \tag{6.16}$$

Remark. The reader may notice that this definition is a specialized definition of the more general notion of a Schur complement in linear algebra.

It is straightforward to see that S is symmetric positive definite because A_h is.

Lemma 6.3.2 (Schur complement reduction)
η, ξ and κ from (6.14) *satisfy*

$$S\kappa = \hat{I}_H A_h^{-1} b_{T,m}, \tag{6.17}$$

$$A_h \eta = b_{T,m} - \hat{I}_H^T \kappa. \tag{6.18}$$

Proof. Rewriting $\mathcal{A}_{h,H}$ as a system of equations, we have

$$A_h\eta + \hat{I}_H^T\kappa = b_{T,m},$$
$$\hat{I}_H\eta = 0.$$

Left-multiplying the first equation by $\hat{I}_H A_h^{-1}$ and inserting the result of the second gives us

$$\underbrace{\hat{I}_H A_h^{-1}\hat{I}_H^T}_{S}\kappa = \hat{I}_H A_h^{-1} b_{T,m}.$$

This gives us the first equation in the result, and the second is merely a reorder of terms. □

Lemma 6.3.2 enables us to formulate a simple method for solving (6.14). First, solve (6.17) to obtain κ, and then insert it into (6.18) and solve for η. Because A_h is simply the standard stiffness matrix for the Poisson problem, there is a large body of literature available for the efficient solution of such systems. The more interesting problem here, however, is arguably how to solve (6.17) for κ.

There seem to be two main ideas. Both are based on the fact that while S is not initially available, we may perform matrix-products Sx in the following fashion:

- Solve $A_h z = \hat{I}_H^T x$ for z, which means that $z = A_h^{-1}\hat{I}_H^T x$.
- Compute $Sx = \hat{I}_H z$.

First, one can make the assumption that N_H is small, in which case it's feasible to explicitly form S by computing N_H matrix-vector products Se_i where e_i is the unit vector for $i = 1, \ldots, N_H$. This is advocated in [27]. Second, one may use the conjugate gradient method in a matrix-free context to solve for κ without explicitly forming S. We will consider the second method, because it scales much better as N_H increases, and can often be faster even for small N_H.

The idea is to leverage the fact that iterative linear solvers such as the Conjugate Gradient method do not depend on having an explicit matrix representation available. Rather, it is sufficient to be able to perform matrix-vector multiplication with the matrix. In other words, we can use the conjugate gradient method to solve (6.17) simply by computing a series of matrix-vector products Sx_k for $k = 1, ...$ Moreover, this feature is commonly available in linear algebra software libraries, and as such is usually straightforward to implement.

One question that arises when dealing with CG applied to S is what - if anything - to use as a preconditioner. Here we may exploit our knowledge of the origins of S to come up with a preconditioner based on some heuristic ideas. In particular, note that the action of S on some vector x essentially maps a coarse-scale vector into the fine-scale space, applies A_h^{-1} and then maps it back to the coarse-scale space. It is thus not unreasonable to expect A_H^{-1} - the inverse of the stiffness matrix for the local coarse space - to be a fair approximation for S, and consequently the preconditioner A_H

would approximate S^{-1}. In practice this does indeed seem to accelerate convergence somewhat.

From the above discussion, it is clear that the Schur complement reduction procedure means that one might have to solve a system $A_h x = b$ for different right-hand sides b a significant number of times. Consequently, if obtaining the solution to this system is a very expensive process, this particular method may not be so attractive. On the other hand, linear solvers based on factorization - such as Sparse LU or Sparse Cholesky - typically perform an expensive factorization step *once*, after which subsequent solves are very inexpensive in comparison. As a result, they are very well suited when paired with Schur complement reduction. Sparse direct solvers typically outperform iterative methods for small to medium systems, after which they quickly become unusable due to high storage and asymptotic runtime costs. The exact equilibrium point between direct and iterative solvers is naturally very heavily problem dependent, but as a rule of thumb, direct solvers are often competitive up to the order of 100 000 unknowns. For the specific problem at hand, we remind the reader that A_h is a symmetric positive definite matrix, and as such, Sparse Cholesky is both applicable and a very good choice for small to medium systems. Its factorization step can often be roughly twice as fast that of Sparse LU.

6.3.2 Algebraic Multigrid and block-preconditioned GMRES

In 6.3.1 we concluded that Schur complement reduction in combination with a sparse direct solver can provide an efficient way to solve the corrector problem (6.14) for systems of small to medium size. However, as was noted, direct solvers become practically unusable for sufficiently large systems. Recall that A_h is simply the stiffness matrix associated with the standard FEM Poisson problem, and as such there is ample choice in applicable linear solvers. An important feature of A_h is that its condition number grows with the size of the system, and so simple preconditioning techniques for iterative solvers are usually not efficient. A well-known efficient solver for elliptic problems is Algebraic Multigrid (AMG)[29][30]. It can be used a linear solver on its own, but it is arguably more commonly used as a preconditioner for iterative methods such as CG or GMRES.

For a given coefficient matrix, AMG first constructs a suitable hierarchy associated with the matrix. We refer to this process as the *setup phase*. This only needs to be done *once* for a given matrix, after which solving $A_h x = b$ can be done with reasonably high efficiency. It is therefore possible to just plug in AMG in the Schur complement method that was described in 6.3.1. However, unlike direct factorization-based methods, the relative cost of solving the system $A_h x = b$ compared to the cost of the setup phase of AMG is much higher. Moreover, it is very unfortunate that when using the Schur complement method, we must accurately solve systems involving A_h many times just to obtain κ, which is a Lagrange multiplier that can be discarded afterwards. Instead, it would be ideal to solve for κ and η *simultaneously*, so that for each step of an iterative method, we get closer to the solution of the corrector η.

One way to accomplish this is to use GMRES [13] on the saddle point problem (6.14) combined with a suitable preconditioner. The main difficulty then is to construct such a preconditioner. Since (6.14) is represented in the form of a block matrix, it makes sense to look for a preconditioner which has a favorable block pattern. The following lemma, inspired by [31], provides us with a useful starting point.

Lemma 6.3.3 (Ideal preconditioner for the corrector problem)
Let $\mathcal{P}_{h,H}$ be the matrix defined by

$$\mathcal{P}_{h,H} := \begin{pmatrix} A_h^{-1} & A_h^{-1}\hat{I}_H^T S^{-1} \\ 0 & -S^{-1} \end{pmatrix}, \tag{6.19}$$

where S is the Schur complement from Definition 6.3.1. *Furthermore, let $\mathcal{A}_{h,H}$ be the coefficient matrix from* (6.14). *Then $\mathcal{P}_{h,H}$ is an ideal* **right** *preconditioner for $\mathcal{A}_{h,H}$ in the sense that the eigenvalues of the preconditioned system matrix $\mathcal{A}_{h,H}\mathcal{P}_{h,H}$ are all* 1.

Proof. Performing the product and denoting by E_{N_h} and E_{N_H} the identity matrix of respective size N_h and N_H, we have that

$$\begin{aligned} \mathcal{A}_{h,H}\mathcal{P}_{h,H} &= \begin{pmatrix} A_h & \hat{I}_H^T \\ \hat{I}_H & 0 \end{pmatrix} \begin{pmatrix} A_h^{-1} & A_h^{-1}\hat{I}_H^T S^{-1} \\ 0 & -S^{-1} \end{pmatrix} \\ &= \begin{pmatrix} E_{N_h} & \hat{I}_H^T S^{-1} - I_H^T S^{-1} \\ \hat{I}_H A_h^{-1} & \underbrace{\hat{I}_H A_h^{-1}\hat{I}_H^T}_{S} S^{-1} \end{pmatrix} = \begin{pmatrix} E_{N_h} & 0 \\ \hat{I}_H A^{-1} & E_{N_H} \end{pmatrix}, \end{aligned}$$

which is a triangular matrix with ones on the diagonal, from which the result follows. □

Although we could in fact apply the preconditioner defined in Lemma 6.3.3 directly by solving multiple systems involving A_h and S, this would demand at least as much work as just solving the whole system by using the Schur complement method in the first place. Instead, we realize that it is typically sufficient to use a good approximation of $\mathcal{P}_{h,H}$. Moreover, we can once again use the fact that Krylov subspace methods only require a linear operator and not a full matrix. The idea is then to construct approximate operators $\tilde{A}_h^{-1} \approx A_h^{-1}$ and $\tilde{S}^{-1} \approx S^{-1}$ such that the preconditioned system is still (hopefully) well conditioned. If these approximations are available, an application of the approximate preconditioner to a vector $\begin{pmatrix} x & y \end{pmatrix}^T \in \mathbb{R}^{N_h+N_H}$ can be written

$$\mathcal{P}_{h,H}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} A_h^{-1}(x + \hat{I}_H^T S^{-1} y) \\ -S^{-1} y \end{pmatrix} \approx \begin{pmatrix} \tilde{A}_h^{-1}(x + \hat{I}_H^T \tilde{S}^{-1} y) \\ -\tilde{S}^{-1} y \end{pmatrix}.$$

If one saves the result of $\tilde{S}^{-1}y$, each application of the approximate preconditioner requires only one application of $\tilde{S}^{-1}$ and one application of $\tilde{A}_h^{-1}$.

The last remaining piece of the puzzle is to determine which approximations to use for $\tilde{A}_h^{-1}$ and $\tilde{S}^{-1}$. Drawing on our earlier discussion, we can expect that we can replace $\tilde{A}_h^{-1}$ by an AMG-based preconditioner (for example a single V-cycle). A suitable approximation of S^{-1} is less obvious, but from the discussion on Schur complement methods in 6.3.1, $\tilde{S}^{-1} \approx A_H$ may provide a starting point.

We close this section by noting that a block-preconditioned GMRES with an AMG-based preconditioner is a heavy piece of advanced machinery. As a result, one could reasonably expect the method not to be competitive for sufficiently small problems when compared to the Schur complement reduction method. On the other hand, as the size of the system increases, one could hope to achieve much greater efficiency than what is possible to achieve with direct solvers. Given that distinct corrector problems may come in many different sizes, it may be worthwhile to conditionally decide on which type of solver to use based on the size of each corrector problem.

6.4 A high-level algorithm for corrector computation

Having determined how to solve the individual corrector problems, we are ready to summarize the process of computing all (localized) correctors for the basis functions of $S_0^1(\mathcal{T}_H)$. We will see that as we compute the correctors for each patch $\Omega_{T,m} \subseteq \Omega$, we will need to store the weights of the corrected basis functions in a (sparse) matrix that we will refer to as the *correction matrix*.

Definition 6.4.1 (Correction matrix)
The *correction matrix* $Q_{\text{corr}} \in \mathbb{R}^{N_H \times N_h}$ is defined such that for the *corrected* basis function $\tilde{\lambda}_{H,i} \in V_H$ associated with node i in V_H, we have that

$$\tilde{\lambda}_{H,i} = \sum_{j=1}^{N_h} (Q_{\text{corr}})_{ij} \lambda_{h,j}, \tag{6.20}$$

where $\lambda_{h,j} \in S_0^1(\mathcal{T}_h)$ corresponds to the fine-scale Lagrangian basis function associated with node j in $S_0^1(\mathcal{T}_h)$.

In general, the correction matrix Q_{corr} provides a way to map weights of functions in V_H with respect to the basis of the fine-scale space $S_0^1(\mathcal{T}_h)$ into weights with respect to the basis of V_H.

The following lemma explains how to obtain the stiffness and mass matrices associated with the corrected space V_H once the correction matrix has been obtained.

Lemma 6.4.2
The mass and stiffness matrices $\tilde{M}_H$ and $\tilde{A}_H$ associated with the basis for the space V_H and the mass and stiffness matrices M_h and A_h associated with the standard basis for the space $S_0^1(\mathcal{T}_h)$ satisfy

$$\tilde{M}_H = Q_{corr} M_h Q_{corr}^T, \tag{6.21}$$
$$\tilde{A}_H = Q_{corr} A_h Q_{corr}^T. \tag{6.22}$$

Proof. This is simply a straightforward consequence of inserting (6.20) into the definition of the mass and stiffness matrices $\tilde{M}$ and $\tilde{A}$. □

When solving the wave equation with the finite element method, we must compute load vectors whose individual elements take the form $b_i(t) = (f(t), \lambda_i)$ for a basis function λ_i. The next lemma demonstrates a very simple relation between the load vectors associated with each space V_H and $S_0^1(\mathcal{T}_h)$.

Lemma 6.4.3 (Load vectors for V_H)
Let $b_h(t) \in \mathbb{R}^{N_h}$ denote the load vector associated with the fine space $S_0^1(\mathcal{T}_h)$. Then the load vector $\tilde{b}_H(t) \in \mathbb{R}^{N_H}$ associated with the corrected space V_H satisfies

$$\tilde{b}_H(t) = Q_{corr}\, b_h(t). \tag{6.23}$$

Proof. Given a basis function $\tilde{\lambda}_{H,i} \in V_H$, we may write for each element $i = 1, \ldots, N_H$,

$$\begin{aligned}\tilde{b}_{H,i}(t) = (f(t), \tilde{\lambda}_{H,i}) &= (f(t), \sum_j (Q_{\text{corr}})_{ij} \lambda_{h,j}) \\ &= \sum_j (Q_{\text{corr}})_{ij} (f(t), \lambda_{h,j}) = \sum_j (Q_{\text{corr}})_{ij}\, b_{h,j}(t),\end{aligned}$$

from which the result immediately follows. □

We are now ready to provide a step-by-step high-level algorithm for how to compute the correctors. Given a quasi-uniform mesh $\mathcal{T}_H$ and a (possibly non-quasi-uniform) refinement of $\mathcal{T}_h$ and the oversampling parameter $m \geq 1$, the following is a summary of the steps necessary to compute the correction matrix Q_{corr}:

1. Let $C_{\text{corr}} \in \mathbb{R}^{N_H \times N_h} = 0$. This matrix will hold the actual correctors as we compute them. Specifically, row i corresponds to the corrector $\mathcal{C}\lambda_{H,i}$ of the coarse basis function $\lambda_{H,i}$.
2. For every coarse element $T \in \mathcal{T}_H$, determine all $K \in \mathcal{T}_h$ that are descendants of T and store the results in an appropriate data structure.

3. Assemble A_h and $\hat{I}_H$.
4. For every coarse element $T \in \mathcal{T}_H$:
   (a) Compute the patch $\Omega_{T,m}$.
   (b) Determine the fine-scale vertices that are in the interior of the (coarse) patch by performing a lookup in the data structure that was constructed in step 2.
   (c) Assemble $A_{h,T,m}$ and $\tilde{I}_{H,T,m}$ by taking submatrices of A_h and $\hat{I}_H$ corresponding to the appropriate degrees of freedom.
   (d) For each vertex z_i in T:
      i. Compute the right-hand side vector in (6.14).
      ii. Solve the system in (6.14) (see 6.3 for solution methods) to obtain corrector weights η.
      iii. Set any weights in η that are below a certain tolerance (i.e. close to machine epsilon) to 0, in order to avoid accidental fill-in due to rounding errors.
      iv. Map the weights stored in η into corresponding weights for $S_0^1(\mathcal{T}_h)$ and add them to the (sparse) row i in C_{corr}.
5. Form the (sparse) matrix $B \in \mathbb{R}^{N_H \times N_h}$, which is defined by the relation $\lambda_{H,i} = \sum_{j=1}^{N_h} B_{ij}\lambda_{h,j}$. Here, $\lambda_{H,i}$ corresponds to the standard Lagrange basis function for node i associated with the *coarse* space $S_0^1(\mathcal{T}_H)$, and $\lambda_{h,j}$ corresponds to the standard Lagrange basis function for node j associated with the *fine* space $S_0^1(\mathcal{T}_h)$.
6. Form the correction matrix $Q_{\text{corr}} = B - C_{\text{corr}}$.

6.5 crest: An open-source implementation

The method proposed in this thesis is quite involved, and considerable work has gone into building an experimental prototype software library which powers the numerical experiments.

In the event that anyone should want to build on this research, some information about this library, called *crest*, will be summarized below. While the library bears several of the hallmarks of experimental code - such as leaky abstractions at the wrong level, sparse documentation and the occasional hack - a large amount of its functionality is thoroughly tested by an automatic test suite, and so it is straightforward to experiment with modifications while still ensuring correctness of the implementation. It is the author's belief and hope that the library can provide a useful starting point for any further work on this subject.

`crest` is a software library written in `C++14`, and at the time of writing can be obtained at `github.com/Andlon/crest`. In the event that the provided URL should become invalid, the reader is invited to contact the author upon interest. The library relies heavily on the open-source `Eigen` [32] library for linear algebra, and provides the following features:

- A header-only template library that abstracts over the scalar type (i.e. `double`, `float`).
- Geometry:
  - A data structure for 2D triangle meshes.
  - A fast and memory-efficient implementation of the *newest vertex bisection* (`NVB`) algorithm, as well as a fast implementation of the *Threshold* algorithm from Lemma 4.2.1.
  - A `BiscaleMesh` data structure which serves as an abstraction over a mesh $\mathcal{T}_H$ and its refinement $\mathcal{T}_h$. Among other things, the data structure provides a convenient way to quickly determine fine triangles in $\mathcal{T}_h$ that are descendants of specific triangles in $\mathcal{T}_H$.
- Corrector computation:
  - An abstraction over different ways to compute correctors, which simplifies experimenting with new ways to compute correctors.
  - A default implementation based on direct solution with Sparse LU from the `Eigen` library.
  - An implementation which leverages the Sparse Cholesky decomposition from `Eigen` in conjunction with the Schur complement reduction method based on the Conjugate Gradient method described in 6.3.1.
  - An implementation based on AMG block-preconditioned GMRES as described in 6.3.2. The AMG functionality as well as the LGMRES [33] solver used are powered by the open-source library `amgcl` [34].
- A comprehensive test suite consisting of unit tests, automatic convergence tests and property-based tests with randomized input ensure correctness of many of the algorithms that have been implemented. In particular, the output from the different implementations for corrector computation is verified to satisfy fundamental properties of the method for randomized input meshes.
- Fast quadrature computation based on quadrature points from [35] for polynomial orders ranging from 1 to 20, using compile-time selection of quadrature strength.
- Temporal discretization schemes: Leapfrog, mass-lumped Leapfrog, Crank-Nicolson.
- A limited abstraction for solving the wave equation for various input data, including functionality to estimate the error with respect to some reference solution.

Chapter 7

Augmented Leapfrog

Recall that the method presented in chapter 5 has its origins in numerical homogenization of elliptic problems. In this context, it is known that one can often achieve similar accuracy by replacing the load vector associated with the corrected space V_H with the load vector associated with $S_0^1(\mathcal{T}_H)$. This is desirable because in those kind of problems, one often has $\dim S_0^1(\mathcal{T}_h) \gg \dim S_0^1(\mathcal{T}_H)$, which means that the load vector computation for $\dim S_0^1(\mathcal{T}_h)$ (and consequently V_H through Lemma 6.4.3) may be prohibitively expensive. In the kind of problems we consider in this thesis, the difference in the number of vertices between the two meshes is not quite as drastic, but as we will see in Chapter 8, it may still be significant.

During the course of the development of the implementation for the method discussed in chapters 5 and 6, it was observed that using the load vectors from $S_0^1(\mathcal{T}_H)$ also seemed to work very well for the model problem. Moreover, it was observed that replacing the mass matrix associated with V_H with the mass matrix from $S_0^1(\mathcal{T}_H)$ worked similarly very well. This is interesting for several reasons:

- The mass matrix associated with $S_0^1(\mathcal{T}_H)$ may be considerable sparser than the matrix associated with V_H.
- If it can be proved that the act of replacing the mass matrix does not reduce the favorable convergence rate associated with V_H, then this may theoretically justify mass lumping of the mass matrix.

Since the construction of the corrected space V_H relies on orthogonalization with the inner product $a(\cdot, \cdot)$, the intuitive explanation for why this might work is that this property is encoded perhaps primarily in the stiffness matrix.

We will now formalize these modifications to the Leapfrog method in terms of a functional-analytic description. For lack of a better name, we will call the resulting method *augmented Leapfrog*.

Definition 7.0.1 (Augmented Leapfrog)
Let V_H be the corrected space defined in Definition 5.1.5, and let I_H be its associated admissible quasi-interpolator. Then the weak formulation of the augmented Leapfrog method is for $t = n\Delta t$ and integer $n \geq 1$ is given by

$$\left(I_H\Big(\frac{u_h^{n+1} - 2u_h^n + u_h^{n-1}}{(\Delta t)^2}\Big), I_H v_H\right) + a(u_h^n, v_H) = (f^n, I_H v_H) \quad \forall v_h \in V_H. \tag{7.1}$$

We will now briefly show stability. The idea is the same as for the standard Leapfrog method. First, we define an appropriate discrete energy, show that this energy is non-negative under the CFL condition, demonstrate how this leads to energy conservation and finally stability of the method. However, because the derivation of these results is virtually identical to that of the standard Leapfrog method, we will omit these intermediate results and instead only show that the energy indeed is non-negative. The only difference is the definition of the energy and an additional occurrence of I_H on the right-hand side. For example, instead of

$$2(\hat{\mathcal{E}}_h^{n+1/2} - \hat{\mathcal{E}}_h^{n-1/2}) = (f^n, u_H^{n+1} - u_H^{n-1}),$$

one has

$$2(\hat{\mathcal{E}}_{H,h}^{n+1/2} - \hat{\mathcal{E}}_{H,h}^{n-1/2}) = (f^n, I_H(u_H^{n+1} - u_H^{n-1})).$$

The rest of the derivation follows naturally with these modifications.

Definition 7.0.2 (Augmented energy)
We define the discrete energy for the augmented Leapfrog method by

$$\hat{\mathcal{E}}_{H,h}^{n+1/2} := \frac{1}{2}\left\|I_H \delta u_H^{n+1/2}\right\|_{L^2(\Omega)}^2 + \frac{1}{2}a(u_H^n, u_H^{n+1}), \tag{7.2}$$

where $u_H^n \in V_H$ is the augmented Leapfrog solution of the wave equation at time $t = t^n$.

Lemma 7.0.3 (Non-negativity of augmented energy)
Assume that $S_0^1(\mathcal{T}_H)$ has the inverse property, and assume that Δt is chosen sufficiently small such that it satisfies the CFL condition (3.10) *for the standard coarse space $S_0^1(\mathcal{T}_H)$. Then the augmented energy for the space V_H satisfies*

$$\hat{\mathcal{E}}_{H,h}^{n+1/2} \geq \frac{\lambda}{2}\left\|I_H \delta u_H^{n+1/2}\right\|_{L^2(\Omega)}^2 + \frac{1}{4}\left[a(u_H^{n+1}, u_H^{n+1}) + a(u_H^n, u_H^n)\right] \geq 0. \tag{7.3}$$

Proof. The key property that we need for the proof is the following observation. Let $v_H \in V_H$, and recall that we may write $v_H = I_H v_H - \mathcal{C} I_H v_H$. This leads to

$$a(v_H, v_H) = a(I_H v_H, I_H v_H) - a(\mathcal{C} I_H v_H, \mathcal{C} I_H v_H).$$

Using this property, we do as in the proof of Lemma 3.2.4 and write

$$\begin{aligned}
2a(u_H^n, u_H^{n+1}) &= a(u_H^{n+1}, u_H^{n+1}) + a(u_H^n, u_H^n) - (\Delta t)^2 a(\delta u_H^{n+1/2}, \delta u_H^{n+1/2}) \\
&= a(u_H^{n+1}, u_H^{n+1}) + a(u_H^n, u_H^n) \\
&\quad - (\Delta t)^2 a(I_H \delta u_H^{n+1/2}, I_H \delta u_H^{n+1/2}) + (\Delta t)^2 a(\mathcal{C} I_H \delta u_H^{n+1/2}, \mathcal{C} I_H \delta u_H^{n+1/2}) \\
&\geq a(u_H^{n+1}, u_H^{n+1}) + a(u_H^n, u_H^n) - (\Delta t)^2 a(I_H \delta u_H^{n+1/2}, I_H \delta u_H^{n+1/2}) \\
&\geq a(u_H^{n+1}, u_H^{n+1}) + a(u_H^n, u_H^n) - \frac{C_{\text{inv}} (\Delta t)^2}{H^2} \left\| I_H \delta u_H^{n+1/2} \right\|_{L^2(\Omega)}.
\end{aligned}$$

In the above, C_{inv} is the inverse constant from Lemma 2.2.4 associated with $S_0^1(\mathcal{T}_H)$. From this point, we proceed in exactly the same manner as in Lemma 3.2.4 to arrive at the result. □

Theorem 7.0.4 (Stability of the augmented Leapfrog method)
Assume that the conditions in Lemma 7.0.3 *hold. Then the augmented Leapfrog method is stable in the sense that there exists some $C > 0$ independent of H and Δt such that*

$$\begin{aligned}
\left\| I_H \delta u_H^{n+1/2} \right\|_{L^2} + \left\| u_H^{n+1} \right\|_{H^1} \leq C \Big(& \left\| I_H \delta u_H^{1/2} \right\|_{L^2} + \left\| u_H^0 \right\|_{H^1} + \left\| u_H^1 \right\|_{H^1} \\
& + \sum_{k=1}^{n} \Delta t \left\| f^k \right\|_{L^2} \Big).
\end{aligned} \tag{7.4}$$

Proof. Omitted because it is almost identical to the derivation of the standard Leapfrog stability result presented in Chapter 3.2. □

An interesting feature of the augmented Leapfrog method made clear by Lemma 7.0.3 is that the CFL condition that is necessary for stability is the one associated with the *coarse* space $S_0^1(\mathcal{T}_H)$, which means that the stability region of the augmented Leapfrog method is at least as large as that of the standard Leapfrog method for $S_0^1(\mathcal{T}_H)$, which is a very desirable property. On the other hand, the stability result is somewhat weaker than for the standard Leapfrog method applied to V_H, as it does not bound $\left\| \delta u_H^{n+1/2} \right\|$, but only $\left\| I_H \delta u_H^{n+1/2} \right\|$. However, the H^1-stability of the solution is still maintained, which is what we are primarily concerned with.

The fact that the method is stable is not interesting unless it converges faster than the standard Leapfrog method in $S_0^1(\mathcal{T}_H)$. Ideally we would be able to show an improved

convergence rate, but a successful proof remains elusive. Because the proof of stability only needed some very minor modifications compared to the standard Leapfrog method, it is natural to think that perhaps this would also be the case for the convergence analysis. It does not look like this is the case. However, as we will see in Chapter 8, the numerical experiments show quite conclusively that the method works very well in the case of our model problem. This raises the question of whether the optimal convergence rate observed can be proved with no added assumptions compared to the standard Leapfrog method, or if there are special circumstances surrounding our model problem that make the augmented Leapfrog attain the optimal convergence rate. For now, this remains an open question.

Chapter 8

Numerical experiments

In this chapter, we will study the approximation properties and performance of the methods presented in the preceding chapters.

By comparing the errors incurred and the performance of the Leapfrog method applied to the corrected space V_H^m with the more straightforward application of the Leapfrog method applied to $S_0^1(\mathcal{T}_H)$ and the Crank-Nicolson method applied to $S_0^1(\mathcal{T}_h)$, we will attempt to answer the following questions:

- Are the approximation properties of the corrected space V_H^m favorable compared to $S_0^1(\mathcal{T}_H)$ and competitive with the space $S_0^1(\mathcal{T}_h)$?
- How do the approximation properties of the augmented Leapfrog method presented in Chapter 7 compare with the standard Leapfrog method?
- Can mass lumping of the Leapfrog method be successfully applied in conjunction with the space V_H^m?
- Does the space V_H^m allow the model problem to be solved in a significantly shorter amount of time than the standard finite element spaces $S_0^1(\mathcal{T}_H)$ and $S_0^1(\mathcal{T}_h)$ do?
- Can the correctors that are necessary to form V_H^m be computed in reasonable time, or is the corrector computation a prohibitively expensive operation?

In order to try to answer these questions, we will study a model problem in a 2D prototypical non-convex domain which exhibits the typical loss of the optimal convergence rate for polynomial finite element spaces.

It is necessary to make a distinction between *offline* computation and *online* computation. The offline computations are comprised of the steps that can be performed independently of the problem data, which corresponds to choices of f, u_0 and v_0, and primarily includes corrector computation and matrix assembly. The online computations include the computation of load vectors, time integration and initialization of the integrators.

Because one typically has for sufficiently small mesh resolution that $N_H = \dim S_0^1(\mathcal{T}_H) \approx \#\mathcal{N}(\mathcal{T}_H))$, we will — in order to reduce the notational burden, particularly in graphs — in this chapter somewhat lazily repurpose N_H such that it refers to the number of vertices $\#\mathcal{N}(\mathcal{T}_H))$ in $\mathcal{T}_H$ and analogously for N_h.

8.1 Experimental setup

This section describes the details of the experimental setup, which includes the definition of the model problem as well as the various technical choices and parameters that were used in the computations.

8.1.1 Model problem

Due to time constraints and the effort required to implement the methods presented, as well as the sheer computational complexity of running the experiment at a sufficiently large scale as to be able to discuss behavior, only a single example problem is examined. It has been chosen such that it exhibits significant local refinement throughout almost the entire mesh, and so it can in some sense be considered a "worst case" example for the method in terms of computational effort for the offline computation as well as the number of added non-zeros in the corrected system matrices when compared to the standard coarse finite element method.

Definition 8.1.1 (Singular model problem)
The model problem is defined in polar coordinates by the exact solution

$$u(t, r, \theta) = \cos(2\pi t)\sin(2\theta/3)r^{2/3}$$

for all $t \in [0, 0.5]$ and the domain $\Omega = [-0.5, 0.5]^2 \backslash \big([0, 0.5] \times [-0.5, 0]\big)$, an L-shaped domain centered at the origin.

Remark. The domain and the mesh family associated with the problem coincide with the examples for the Threshold algorithm presented in Figure 4.3.

The model problem is singular because its (spatial) gradient blows up as $r \to 0$. An important thing to note is that the model problem does not admit *homogeneous* Dirichlet boundary conditions. One way to construct a model problem which is identically zero on the boundary is to multiply the above model problem by a bump function, but the resulting closed form of the right-hand side function turned out to be absurdly complicated. Instead, the problem is solved numerically by applying inhomogeneous Dirichlet conditions, using the value of the exact solution at the boundary as the condition to be satisfied. Since the practical implementation of the inhomogeneous Dirichlet conditions involve solving a *homogeneous* problem, all the theory we have derived for

N_H	65	225	833	3201	12545	49665	197633	788481
N_h	322	1404	5970	24690	100592	405984	1630510	6537150

Table 8.1: Relationship between number of vertices N_H in $\mathcal{T}_H$ and N_h in $\mathcal{T}_h$.

homogeneous Dirichlet conditions still hold, provided that the approximation of the boundary terms is sufficiently accurate.

It must also be noted that the right-hand side that arises from the model problem is *not* continuously differentiable in space, and so it does not fully conform to Theorem 4.2.5. However, based on numerical experiments, it turns out that the above model problem does indeed exhibit the properties which we wish to study. Namely, that the standard finite element method applied to a quasi-uniform triangulation does not achieve the optimal convergence rate, but refining the mesh with the algorithm from Lemma 4.2.1 recovers the optimal convergence rate as predicted by Theorem 4.2.5.

Experiments were run for a range of different mesh sizes. Table 8.1 demonstrates the relationship between the number of vertices N_H in $\mathcal{T}_H$ and N_h in the refined mesh $\mathcal{T}_h$. In the rest of this chapter, we will only concern ourselves with N_H.

8.1.2 Offline computation

Two implementations for the offline computation of the correctors were considered.

The first is an implementation of the Schur complement reduction method discussed in Section 6.3.1, using an implementation of Sparse Cholesky from the Eigen [32] library to solve systems involving A_h, and the preconditioned Conjugate Gradient method with A_H as a preconditioner for the implicit S matrix. For clarity, the parameters are summarized in Table 8.2.

Parameter	**Value**
CG tolerance	10^{-10}
S preconditioner	A_H
A solver	Sparse Cholesky

Table 8.2: Schur complement reduction parameters.

The second is an implementation of the GMRES scheme with AMG-based preconditioning as discussed in Section 6.3.2. However, because the `amgcl` library which provided the implementations for (restarted) GMRES and AMG did not allow the use of right preconditioners for its GMRES solver, an implementation of LGMRES [33] from the same library was used instead. This is likely of little practical consequence, and in fact LGMRES has been shown to outperform GMRES in many situations (or at least rarely perform worse). Moreover, there are a number of available choices for parameters. The

author of this thesis is admittedly an amateur when it comes to multigrid methods, and although experimenting with different choices of parameters seemed to yield little difference in performance for small problems, it is quite possible that a practitioner with expert knowledge can tune the preconditioner in such a way that performance can be (possibly significantly) increased. In the end, the parameters chosen coincided with the default settings of the library. Table 8.3 attempts to list some of the most important experimental parameters used, but it is not complete. There are a multitude of additional parameters that can be tweaked in the library.

Parameter	**Value**
Coarsening scheme	Smoothed aggregation
Relaxation scheme	`SPAI0`
Cycles	Single V-cycle
LGMRES tolerance	10^{-10}
S preconditioner	A_H

Table 8.3: LGMRES-AMG parameters.

As is usual with iterative methods, one can obtain the solution of a linear system in a shorter amount of time if one can accept a lower tolerance for the stopping criterion used. At the same time, the tolerance needs to be chosen such that the obtained solution is still sufficiently accurate. For both methods of offline computation considered here, a somewhat arbitrary tolerance of 10^{-10} was chosen. Keep in mind that not only do the two methods (CG and LGMRES) prescribe somewhat different meanings to their respective stopping criterion, but they are also solving completely different systems. The fact that the tolerance is chosen to be the same for both schemes does not imply that it is necessarily a fair comparison. However, informal numerical experiments suggested that lowering the iterative tolerance seemed to only produce roughly 40% faster computations in the best case, and at a significant loss of accuracy. Hence, the somewhat conservative choice of 10^{-10} was made as a judicious trade-off between reliability and performance.

8.1.3 Online computation

From previous discussion, it is expected that the Leapfrog method is not stable for refined meshes constructed by the algorithm in Lemma 4.2.1. For this reason, the Crank-Nicolson scheme is used as a comparable baseline, because it offers the same asymptotic error rates, yet it is unconditionally stable. In addition, the "Lumped Leapfrog" method - Leapfrog with mass lumping - is considered.

For the Leapfrog and Crank-Nicolson methods, which involve solving a linear system at every step, the coefficient matrices here (M and $M + \frac{(\Delta t)^2}{4}A$) are positive definite matrices, and the Conjugate Gradient method is applied to solve the system at each step. An important feature of iterative methods is that they allow an "initial guess"

to be specified. For time-dependent problems discretized by finite differences in time, it is very natural to choose the current solution x_n as the initial guess for the solution at the next time step x_{n+1}. If the solution does not change too abruptly, this typically leads to very fast convergence.

Diagonal preconditioning was used for both the Leapfrog and the Crank-Nicolson methods. More precisely, the inverse of M is approximated by $\text{diag}(M)^{-1}$, which is very cheap to apply. This is well-justified for the Leapfrog method, because it is well known that the mass matrix M associated with $S_0^1(\mathcal{T}_H)$ can be efficiently pre-conditioned this way. Moreover, it is shown in [12] that diagonal preconditioning is also an efficient preconditioner for the mass matrix associated with the corrected space V_H. From experience, diagonal preconditioning also works very well for the Crank-Nicolson in the context of quasi-uniform meshes if time steps are sufficiently small to be within the region of stability for the Leapfrog method, but its performance quickly degrades with larger time steps. Because of the much stricter inverse inequality of the locally refined space, it is likely that diagonal preconditioning is insufficient for the Crank-Nicolson method on $S_0^1(\mathcal{T}_h)$, but it's not clear at this point what exactly would be a more well-suited preconditioner.

As is the case with offline computation, an iterative tolerance needs to be chosen for solving the linear system at every step with CG. For all experiments, the tolerance was set to 10^{-12}. At first, this may seem to demand excessive accuracy, but the later discussion will demonstrate the reasoning for this choice.

At every step t^n, one needs to compute the load vector $b(t^n)$, which involves the solution of an integral. This integral is approximated in each element by a 3-point quadrature rule with *quadrature strength* 2, which means that it can integrate polynomials of order 2 exactly. See [35] for the quadrature rules used.

For initialization — computation of u_h^0 and u_h^1 — nodal interpolation of u_0 and the approximation $u_1 \approx u_0 + \Delta t v_0 + \frac{1}{2}(\Delta t)^2 \ddot{u}(0)$ was observed to yield sufficiently accurate results for all experiments considered. Note that here we have used explicit knowledge of $\ddot{u}(0)$ to get a more accurate approximation, because we are not overly concerned with initialization and thus wanted to make sure it did not have an adverse effect on the results.

The parameters used for online computation are summarized in Table 8.4.

The error $\|u(t^n) - u_h^n\|$ in both L^2 and H^1 at each step is estimated by a 4-strength 6-point quadrature rule. These samples are further used in conjunction with the Composite Simpson's rule to estimate the space-time error $\|u - u_h\|_{L^2(0,T;V)}$ for $V = H^1(\Omega)$ and $V = L^2(\Omega)$. An important thing to note is that the error in V_H is computed in terms of its weights in the fine space $S_0^1(\mathcal{T}_h)$.

Each experiment was run in isolation on its own node on the Atacama cluster at the Institute of Numerical Simulation in Bonn. This minimizes errors from e.g. other applications using the same hardware. Because of the sheer number of combinations of parameters that had to be run for the experiments, as well as the immense computational effort required to obtain results for the larger problems, each parameter

Parameter	Value
Integrators	Crank Nicolson, Leapfrog, Lumped Leapfrog
Oversampling (m)	$0, 1, 2, 3, 4, 5$ (where applicable)
CG tolerance	10^{-12}
Preconditioning	Diagonal preconditioning (where applicable)
Load quadrature strength	2 (3 points)
Initialization	Nodal interpolation of u_0 and $u_1 \approx u_0 + \Delta t v_0 + \frac{1}{2}(\Delta t)^2 \ddot{u}(0)$

Table 8.4: Parameters for the online computations.

combination was only run once. Ideally, one would run them several times in order to get a more accurate average time. However, because the implementation is entirely deterministic, it was observed that random errors in the runtime tend to be comparatively small, and so the result presented here should still give a very good indication of the expected runtimes for the implementation.

Finally, we mention that we will present both error estimates and runtimes as functions of the number of vertices in the coarse mesh $\#\mathcal{N}(\mathcal{T}_H)$. Until now, we have presented errors as a function of H, but from Chapter 4, we have seen that we can expect that if the error is $\mathcal{O}(H^q)$ for some $q > 0$, then it is $\mathcal{O}(\#\mathcal{N}(\mathcal{T}_H)^{-q/2})$ for both $S_0^1(\mathcal{T}_H)$ and $S_0^1(\mathcal{T}_h)$. The primary reason is that while the number of vertices is a useful measure of complexity, the diameter of the mesh is less intuitive to deal with. This is particularly true for runtime measurements, which clearly scale with the number of degrees of freedom.

8.2 Error measurements for the online computations

In this section, the estimates for the errors involved when solving the model problem will be discussed. In particular, we verify that the corrected space V_H^m admits the optimal convergence rates associated with $S_0^1(\mathcal{T}_h)$ for sufficiently large m, and it is also of interest to study the impact of the choice of the oversampling parameter m. Moreover, we will see that the *augmented* Leapfrog method in which the mass matrix and right-hand side for V_H are replaced with the ones associated with $S_0^1(\mathcal{T}_H)$ perform very well in practice, and that mass lumping is applicable to the mass matrix associated with the corrected space V_H, as well as the augmented method.

We will also see that the strong local refinement associated with the mesh refinement algorithm from Lemma 4.2.1 leads to problems with the H^1 accuracy of the solution for the Crank-Nicolson method when applied to the fine space $S_0^1(\mathcal{T}_h)$, and that the method when applied to the corrected space V_H does not have similar issues.

Although we have computed the correctors that generate V_H^m in two different ways, it was observed that the resulting error estimates were virtually independent of the

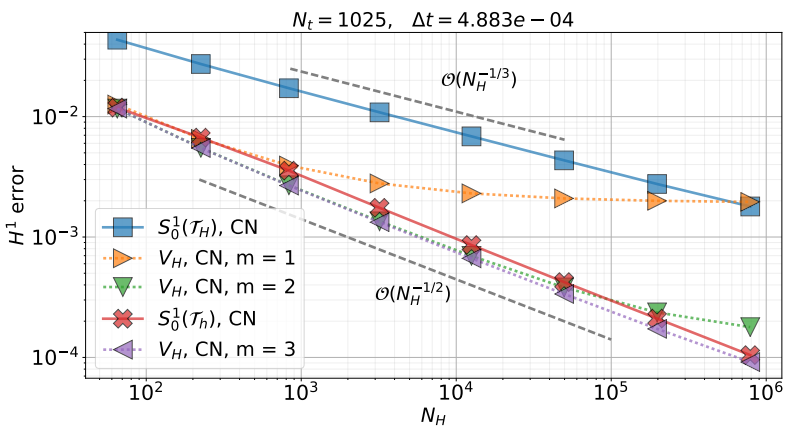


Figure 8.1: Estimated error $\|u - u_h\|_{L^2(0,T;H^1)}$ as a function of number of vertices N_H in the coarse mesh $\mathcal{T}_H$ for the Crank-Nicolson method applied to the coarse space $S_0^1(\mathcal{T}_H)$, the fine space $S_0^1(\mathcal{T}_h)$ and the corrected space V_H.

way in which the correctors were computed, which merely suggests that the correctors were computed with higher accuracy than other sources of errors. In the following study of convergence behavior, we will simply consider the space V_H^m to be identical to the space spanned by the corrected basis functions as computed by the Schur-based method. Note that this choice is arbitrary, and we might as well have chosen the correctors from the GMRES-based computation.

Convergence behavior of the Crank-Nicolson method

We begin our discussion by studying the convergence rates associated with each space when using the Crank-Nicolson method to solve the model problem. Figure 8.1 demonstrates how the coarse space $S_0^1(\mathcal{T}_H)$ only attains the suboptimal convergence rate $\mathcal{O}(N_H^{-1/3})$, whereas $S_0^1(\mathcal{T}_h)$ admits the optimal rate $\mathcal{O}(N_H^{-1/2})$. For oversampling parameter $m = 1$, we see that V_H attains the optimal rate for $N_H \lesssim 1000$, after which it fails to yield better error estimates. For $m = 2$, the optimal rate is attained for $N_H \lesssim 200000$, and it is observed that $m = 3$ admits the optimal rate for all mesh resolutions considered.

It is important to note that the time step here is chosen deliberately large. While it is just barely small enough to make the error term proportional to $(\Delta t)^2$ negligible in comparison with the term proportional to H, it would usually be desirable to use a somewhat smaller time step in order to minimize errors due to the size of the chosen time step. However, it turns out that the strong refinement of T_h causes issues related to numerical accuracy for the Crank-Nicolson method. Figure 8.2 shows that as the

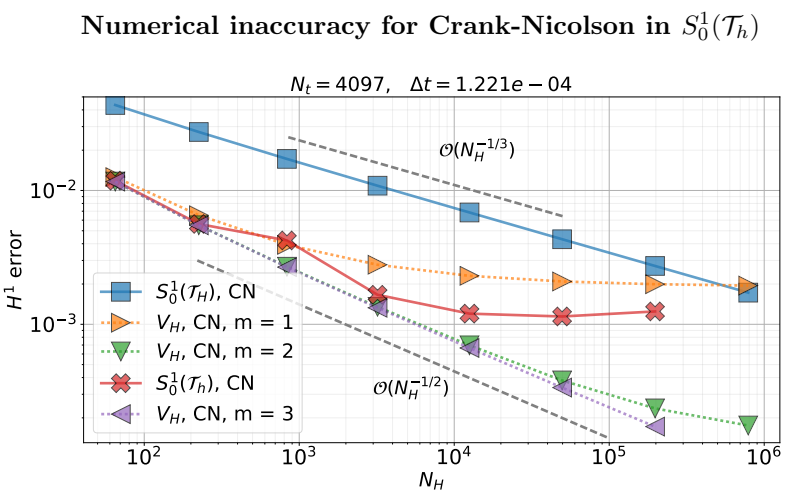


Figure 8.2: Estimated error $\|u - u_h\|_{L^2(0,T;H^1)}$ as a function of number of vertices N_H in the coarse mesh $\mathcal{T}_H$ for the Crank-Nicolson method applied to the coarse space $S_0^1(\mathcal{T}_H)$, the fine space $S_0^1(\mathcal{T}_h)$ and the corrected space V_H.

time step is made smaller compared to that used in Figure 8.1, the Crank-Nicolson method *fails* to attain the optimal convergence rate - contrary to what is expected. The reason seems to be related to accuracy. From some informal experiments whose results we will not reproduce here, it is clear that it is related to the accuracy of the solution to the system $(M+(\Delta t)^2 A/4)x = b$. In particular, decreasing the tolerance for the residual when applying the Conjugate Gradient method to the system makes the solutions more accurate, but only up to a point. If the time step is made smaller or the spatial resolution is made finer, the problems reappear. As we have noted earlier, the tolerance used in these experiments was set to 10^{-12}. It is clear that one cannot make this very much lower due to lack of precision in floating point arithmetic. From Figure 8.2, it is also apparent that the corrected space V_H^m does not suffer from this particular problem. Throughout all experiments, including the large number of experiments which we have not presented here, it was observed that the methods applied to V_H^m seemed to be equally robust as when applied to the coarse, quasi-uniform space $S_0^1(\mathcal{T}_H)$.

While we are primarily concerned with the H^1 errors in this thesis, it also of interest to study the L^2 errors associated with the method. Figure 8.3 showcases the convergence behavior in the spatial L^2 norm. First, it is observed that the Crank-Nicolson method approximates the solution in $S_0^1(\mathcal{T}_H)$ with the suboptimal rate $\mathcal{O}(N_H^{-2/3})$ for the L^2 error, and that in $S_0^1(\mathcal{T}_h)$ it attains the optimal rate $\mathcal{O}(N_H^{-1})$ for the L^2 error. Unlike the case of H^1 errors previously discussed, there seem to be no issues with accuracy. We also observe that the corrected space V_H^m yields competitive error estimates when compared to $S_0^1(\mathcal{T}_h)$, although the data suggests that it doesn't *quite* achieve the optimal rate. As before, we see that the oversampling parameter m must be chosen judiciously in order to attain the correct convergence behavior.

L^2 convergence behavior of the Crank-Nicolson method

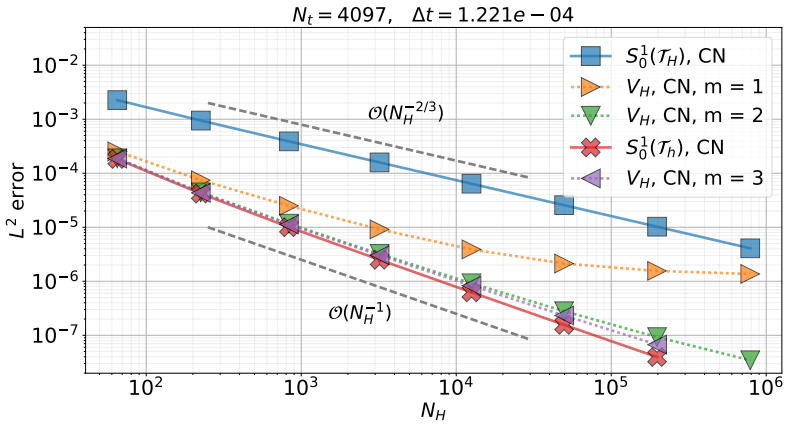


Figure 8.3: Estimated error $\|u - u_h\|_{L^2(0,T;L^2)}$ as a function of number of vertices N_H in the coarse mesh $\mathcal{T}_H$ for the Crank-Nicolson method applied to the coarse space $S_0^1(\mathcal{T}_H)$, the fine space $S_0^1(\mathcal{T}_h)$ and the corrected space V_H.

Convergence behavior of the Leapfrog method

The premise of this thesis is to study a method which relaxes the CFL condition for the Leapfrog method in the presence of locally refined meshes. With this in mind, it was observed that the Leapfrog method diverged even for the coarsest mesh resolution when applied to $S_0^1(\mathcal{T}_h)$. However, as shown in Figure 8.4, it is seen that the Leapfrog method applied to V_H is stable in roughly the same region as when applied to $S_0^1(\mathcal{T}_H)$. Moreover, it attains the optimal convergence rate $\mathcal{O}(N_H^{-1/2})$ in the H^1 norm, and from Figure 8.5 we see that the convergence behavior in the L^2 norm is similar to that when applying the Crank-Nicolson method. In other words, it attains something close to the optimal convergence rate also in the L^2 norm. We stress that the divergence observed at the largest step is merely a consequence of choosing a somewhat too large time step in order to use the Crank-Nicolson scheme for $S_0^1(\mathcal{T}_h)$ as a reference.

Next, we wish to study the case of mass lumping. This involves replacing the mass matrix M with a diagonalized approximation as described by Lemma 3.4.1. From Figure 8.6, we observe that mass lumping for the corrected space V_H^m seems to behave similarly to that of the coarse space $S_0^1(\mathcal{T}_H)$, with the exception that the optimal convergence rate is attained for V_H^m, and consequently significantly better approximations to the exact solution are attained. It is interesting to note that there is a significant (but acceptable) error involved when applying mass lumping to the coarsest spaces, but that this discrepancy virtually disappears for higher mesh resolution. For clarity of presentation, we have only included the case of $m = 3$ here, but the behavior was seen to be analogous for the other values of m, in that the error very closely resembles that of the non-lumped Leapfrog method.

H^1 convergence behavior of the Leapfrog method (no lumping)

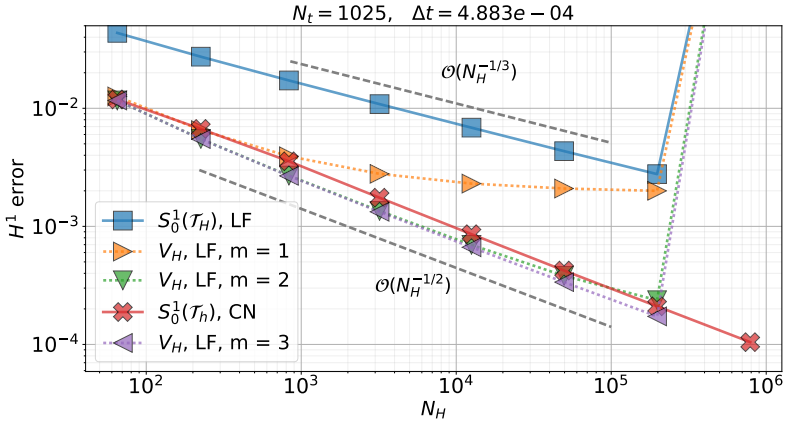


Figure 8.4: Estimated error $\|u - u_h\|_{L^2(0,T;H^1)}$ as a function of number of vertices N_H in the coarse mesh $\mathcal{T}_H$ for the Leapfrog method (LF) applied to the coarse space $S_0^1(\mathcal{T}_H)$ and the corrected space V_H.

L^2 convergence behavior of the Leapfrog method (no lumping)

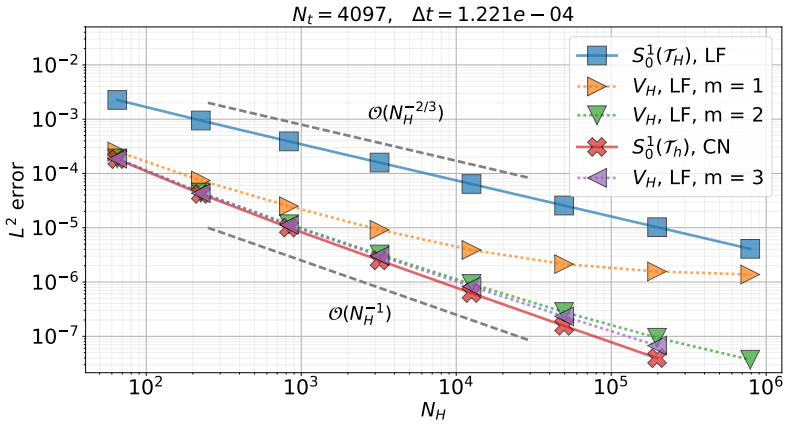


Figure 8.5: Estimated error $\|u - u_h\|_{L^2(0,T;L^2)}$ as a function of number of vertices N_H in the coarse mesh $\mathcal{T}_H$ for the Leapfrog method (LF) applied to the coarse space $S_0^1(\mathcal{T}_H)$ and the corrected space V_H.

H^1 convergence behavior of the Leapfrog method with mass lumping

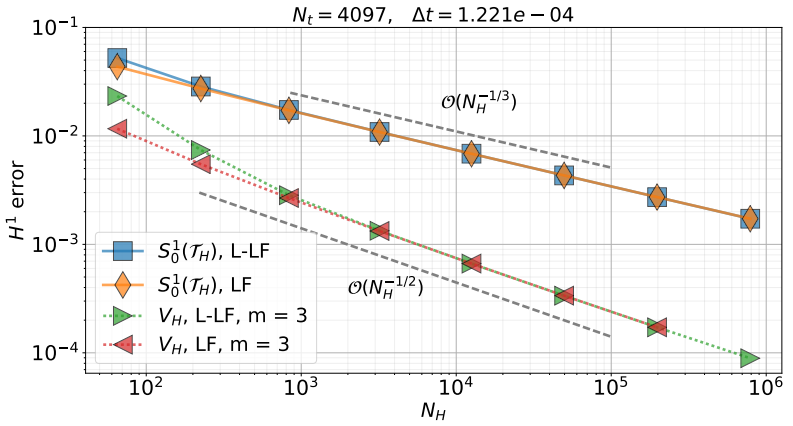


Figure 8.6: Estimated error $\|u - u_h\|_{L^2(0,T;L^2)}$ as a function of number of vertices N_H in the coarse mesh $\mathcal{T}_H$ for the mass-lumped Leapfrog method (L-LF) applied to the coarse space $S_0^1(\mathcal{T}_H)$ and the corrected space V_H, as well as the Leapfrog method (LF) applied to both spaces.

We now turn to studying the augmented Leapfrog method from Chapter 7. It turns out that - at least for this particular model problem - the errors introduced by replacing the corrected mass matrix and load vectors with their coarse space equivalents are almost negligible. Because this is impossible to discern in a logarithmic plot, the numerical values for the errors in the H^1 norm are available for inspection in Table 8.5. In order to save some space we have excluded the experiments for the coarsest meshes. It must also be noted that data was not available for some experiments at the finest mesh resolutions.

Next, we look at what happens when we apply mass lumping to the augmented Leapfrog method and compare the errors to the standard mass lumped Leapfrog method applied to V_H^m. We know from prior discussion that the mass lumping procedure seems to cause some additional errors at low mesh resolutions, and unsurprisingly this means that there's a slight deviation between the mass-lumped augmented Leapfrog method and the usual mass-lumped Leapfrog method. However, as in the case of augmented Leapfrog without mass-lumping, the errors are almost identical to the usual mass-lumped Leapfrog method as the mesh resolution increases. The results are presented in Table 8.6. Note that for brevity we only present the case $m = 3$.

N_H	m	Error (non-augmented)	Error (augmented)
3201	1	2.78519e-03	2.78533e-03
	2	1.35095e-03	1.35009e-03
	3	1.32771e-03	1.32680e-03
12545	1	2.30199e-03	2.30204e-03
	2	6.96820e-04	6.96661e-04
	3	6.65340e-04	6.65165e-04
49665	1	2.08943e-03	2.08944e-03
	2	3.80683e-04	3.80657e-04
	3	3.36544e-04	3.36513e-04
197633	1	1.99348e-03	1.99349e-03
	2	2.35217e-04	2.35219e-04
	3	1.71977e-04	1.71974e-04
788481	1	1.94662e-03	1.94666e-03
	2	1.75244e-04	1.75260e-04
	3	8.92752e-05	

Table 8.5: Estimated error $\|u - u_h\|_{L^2(0,T;H^1)}$ for the non-augmented Leapfrog and augmented Leapfrog method applied to the space V_H^m.

N_H	m	Error (non-augmented)	Error (augmented)
65	3	2.33179e-02	2.53831e-02
225	3	7.40836e-03	7.57917e-03
833	3	2.83990e-03	2.84497e-03
3201	3	1.33472e-03	1.33409e-03
12545	3	6.64906e-04	6.64716e-04
49665	3	3.36345e-04	3.36302e-04
197633	3	1.71908e-04	1.71901e-04
788481	3	8.91769e-05	8.91774e-05

Table 8.6: Estimated error $\|u - u_h\|_{L^2(0,T;H^1)}$ for the non-augmented Leapfrog and augmented Leapfrog method with mass lumping applied to the space V_H^m.

8.3 Performance

A central question that arises when considering the practical efficacy of the corrected basis V_H^m is whether or not it allows for sufficiently fast solutions to the problem. More precisely, does the Leapfrog method applied to V_H^m accelerate accurate solutions to the wave equation compared to standard methods? Alternatively, is it possible to achieve an accurate solution in a shorter amount of time by applying the Leapfrog method to $S_0^1(\mathcal{T}_H)$ for a higher mesh resolution or by using the Crank-Nicolson method applied to $S_0^1(\mathcal{T}_h)$?

The results from Section 8.2 suggest that mass lumping is a viable procedure even for the corrected space V_H^m. This is very desirable, because we do not need to employ an iterative solver at every time step, instead only having to apply the inverse of a diagonal matrix. Since $\dim V_H^m = \dim S_0^1(\mathcal{T}_H)$, it is tempting to expect that the mass-lumped Leapfrog method will take the same amount of time time when applied to V_H^m as when applied to $S_0^1(\mathcal{T}_H)$. However, recall that the right-hand side of the system in Definition 3.4.2 requires the application of the stiffness matrix A to a vector. Because the stiffness matrix associated with V_H^m is generally denser than for $S_0^1(\mathcal{T}_H)$, we can not expect this to be the case.

In the following, we will compare actual runtimes for the following computational methods:

- The mass-lumped Leapfrog method applied to $S_0^1(\mathcal{T}_H)$.
- The mass-lumped Leapfrog method applied to V_H^m.
- The augmented mass-lumped Leapfrog method applied to V_H^m.
- The Crank-Nicolson method applied to $S_0^1(\mathcal{T}_h)$.

In order to simplify the presentation, we neglect to show benchmarks for all values of m. Instead, for each value of N_H, we fix a single m, which is chosen such that it is the lowest value of m which attains the optimal convergence properties at that particular mesh resolution. More precisely, we have made the choice

$$m(N_H) := \begin{cases} 2 \text{ if } N_H < 13000 \\ 3 \text{ otherwise .} \end{cases}$$

To evaluate the performance of the (mass-lumped) Leapfrog method applied to the corrected space V_H^m, we must make a distinction between *offline* and *online* computation. It is important to realize that the basis only depends on the meshes $\mathcal{T}_H$ and $\mathcal{T}_h$, and so the system matrices can be assembled once and reused for different right-hand side f and initial conditions u_0 and v_0. It thus makes sense to evaluate the offline and online performance of the method separately.

The most important aspect is arguably the *online performance*, because if the method fails to provide benefits over the standard finite element approaches even when not taking into account the offline costs, it is of little practical value. However, if the

method can be shown to perform better than the standard methods in the online phase, it must still be shown that the offline costs are not prohibitive. In other words, it is crucial that the basis can be computed in reasonable time.

When measuring performance, there are many aspects to take into account. Here we have made an effort to study the most important characteristics, as well as their combined effects. To clarify what is measured, we make the following distinctions for different runtime measurements. For *offline* performance measures, we define:

- Corrector computation time: Time spent computing correctors in order to form the correction matrix Q_{corr} from Definition 6.4.1.
- Assembly time: Time spent assembling the mass- and stiffness matrices. Note that for V_H^m, this does *not* include the time for computing the correctors. It only includes the time necessary to assemble the matrices using the relations defined in Lemma 6.4.2 after the correction matrix Q_{corr} has already been obtained.

For *online* performance measures, we define:

- Integration time: Total time across all time steps exclusively for setting up and solving the linear system at each time step. Does *not* include the time to construct load vectors.
- Load computation time: Total time across all time steps that was necessary to compute the load vectors associated with the method. In this context, this depends only on the dimension of the finite element space.
- Step time: Integration time + Load computation time.

Note that while the Crank-Nicolson method requires access to load vectors for three distinct time steps, only a single *new* load vector must be computed for each iteration, and so the costs of computing load vectors for the Leapfrog method and the Crank-Nicolson method remain the same.

Before we begin to study the results, we must make the obligatory comment that while parts of the implementation have been heavily profiled and optimized, other parts have not undergone the same treatment. For example, the assembly procedure for the augmented Leapfrog method still assembles the full corrected mass matrix associated with V_H^m even though it is not needed, and it has otherwise not received any study as to whether it can be made faster. On the other hand, the load computation has been extensively profiled and rewritten to be fairly efficient, and the integration at each step relies on fairly optimized machinery from the `Eigen` [32] library.

Online performance

We first consider the *step time*. This is essentially a representation of the total online performance, because it describes the cost of the two most dominant factors of the online phase - the time integration and load computation. From Figure 8.7, we see that the two mass-lumped Leapfrog methods applied to V_H^m clearly outperform the

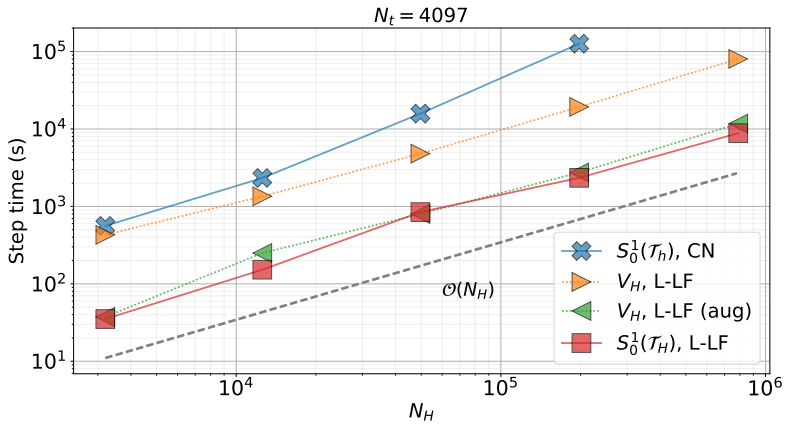


Figure 8.7: Selected step times for the model problem for the mass-lumped Leapfrog method (L-LF), the augmented mass-lumped Leapfrog method (L-LF (aug)) and the Crank-Nicolson method (CN).

Crank-Nicolson method applied to $S_0^1(\mathcal{T}_h)$. In fact, it looks as if the augmented method performs similarly to that of $S_0^1(\mathcal{T}_H)$. These results tell us something important, but may ultimately be deceptive. It turns out that the load vectors for the model problem are *very* expensive to compute, involving vast numbers of trigonometric calculations, and clearly dominate the runtime for anything but the fine space $S_0^1(\mathcal{T}_h)$. We also see that the augmented method significantly outperforms the non-augmented method by a constant factor. From the preceding discussion, it should come as no surprise that the difference is due to the fact that the augmented method only needs to compute the coarse load vectors, which for this particular problem means roughly 1/8 of the effort for load computation.

What if $f = 0$, or what if f is constant? In this case, load computation is virtually free, and the results that were just presented are not indicative of actual performance. In this case, a much better performance metric is the *integration time*, which is presented by Figure 8.8. The same data is also presented in Table 8.7. Unsurprisingly, the coarse space $S_0^1(\mathcal{T}_H)$ admits much faster integration, but it's important to keep in mind that it does so with substantially less accuracy.

It is clear from Figure 8.8 that the corrected space V_H^m admits much more efficient computation than what the Crank-Nicolson method is able to achieve in this case, since we have seen that V_H^m yields approximations of the solution that are at least as accurate as that of $S_0^1(\mathcal{T}_h)$. However, having just seen how fast the mass-lumped Leapfrog method is applied to the coarse space $S_0^1(\mathcal{T}_H)$, it begs the question if one is able to come up with equally accurate approximations in a shorter amount of time by running the standard mass-lumped Leapfrog method on $S_0^1(\mathcal{T}_H)$ to much higher mesh resolutions.

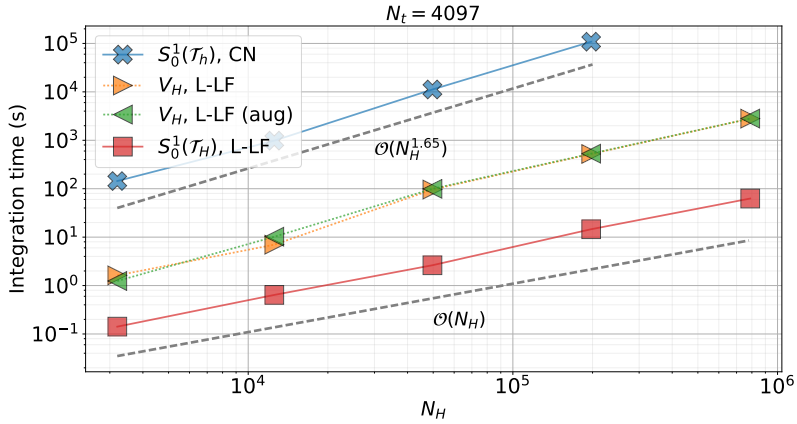


Figure 8.8: Selected integration times for the model problem for the mass-lumped Leapfrog method (L-LF), the augmented mass-lumped Leapfrog method (L-LF (aug)) and the Crank-Nicolson method (CN). See Table 8.7 for the same data in a tabular format.

To test this hypothesis, consider $N_H = 788481$. From Figure 8.6, we have seen that the error produced by $S_0^1(\mathcal{T}_H)$ is slightly higher than the error produced by V_H^m for $m = 3$ and $N_H = 3201$. Moreover, this is also the case for $m = 2$. Comparing the runtimes from Table 8.7, we see that the integration time for $S_0^1(\mathcal{T}_H)$ for $N_H = 788481$ is about 63 seconds. For comparison, the integration time for V_H^m for the same number of time steps is about 1 second. Here we could also have used a much smaller number of time steps to achieve roughly the same error, which would have further reduced the integration time. At this point, it is fair to say that the method based on the corrected space V_H^m represents a significant improvement over the coarse space $S_0^1(\mathcal{T}_H)$ in terms of online performance.

We have now established that in our experiments, the mass-lumped Leapfrog method applied to V_H^m (augmented or not) clearly outperforms both the mass-lumped Leapfrog method applied to $S_0^1(\mathcal{T}_H)$ and the Crank-Nicolson method applied to $S_0^1(\mathcal{T}_h)$ with a diagonal preconditioner. It is then natural to ask how the results would look like if we were to use an effective preconditioner for the Crank-Nicolson method. We can not make any definitive assessment, but we can actually make a prediction. The integration time of the mass-lumped Leapfrog method is essentially completely dominated by the matrix-vector product Ax, since only a diagonal matrix needs to be inverted. However, the Crank-Nicolson method must perform matrix-vector multiplication with both M and A, and then solve an ill-conditioned linear system. This lets us make the following claim: the mass-lumped Leapfrog method in V_H^m is *always* faster than the Crank-Nicolson method in $S_0^1(\mathcal{T}_h)$ if the number of non-zeros in the stiffness matrix A_H associated with V_H^m is less than the combined number of non-zeros in the stiffness matrix A_h and M_h associated with $S_0^1(\mathcal{T}_h)$.

Some casual observations and results presented in [12] seems to suggest that the density

N_H	V_H^m L-LF	V_H^m L-LF (aug)	$S_0^1(\mathcal{T}_H)$ L-LF	$S_0^1(\mathcal{T}_h)$ CN
3201	1.62	1.24	0.14	143.98
12545	6.98	10.19	0.64	982.73
49665	96.01	99.29	2.63	11191.94
197633	521.17	529.12	14.57	108529.08
788481	2800.36	2806.56	62.67	

Table 8.7: Selected integration times for the model problem for the mass-lumped Leapfrog method (L-LF), the augmented mass-lumped Leapfrog method (L-LF (aug)) and the Crank-Nicolson method (CN). This is the same data as presented by Figure 8.8.

of A_H tends to be greater than A_h for these kind of problems, so the above claim is not directly very useful. However, consider now that the solution of the ill-conditioned system typically dominates the integration time for the Crank-Nicolson method. This means that there is a great deal of leniency in the number of additional non-zero elements that can be accepted in A_H. However, due to the lack of results with an appropriate preconditioner for the Crank-Nicolson method, we are not able to quantify this claim.

Offline performance

One of the main goals of this thesis is to determine if correctors can be computed in reasonable time. We have seen in the previous section that the corrected space V_H^m may provide significantly better online performance than the alternatives we have studied here. A natural follow-up question is how much of an overhead the offline phase incurs.

Moreover, we have proposed two alternative ways to compute correctors: the Schur-complement based method discussed in 6.3.1, and the GMRES/AMG-based method discussed in 6.3.2.

The corrector computation times are presented in Figure 8.9 and Table 8.8.

The most important observation from Figure 8.9 is that the corrector computation is not typically prohibitively expensive. In fact, we see that the timings are of the same order of magnitude as the integration times we studied in the previous section. In other words, this heavily suggests that the spatial reduction method is in fact a practically feasible method for numerical computation.

Another thing to note from Figure 8.9 is that runtimes seem to increase at a steady rate for problems of size $N_H \leq 197633$, but then suddenly make a sharp increase for the largest problems of size $N_H = 788481$. Although it cannot be verified with certainty at this point, the cause seems to be partly due to technical issues rather than exclusively the mathematical properties of the method, and so it's hard to make inferences based on

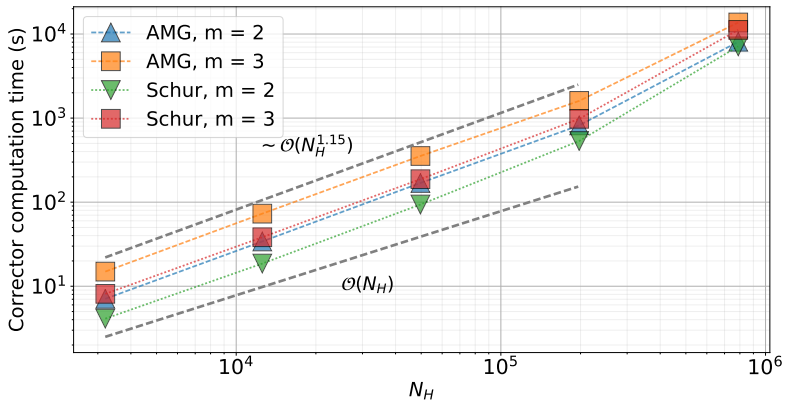


Figure 8.9: Corrector computation times for the GMRES/AMG and Schur-based methods.

N_H	GMRES/AMG m = 2	GMRES/AMG m = 3	Schur m = 2	Schur m = 3
3201	7.08	14.93	4.12	8.11
12545	34.10	73.02	18.60	38.27
49665	169.06	354.09	93.74	187.64
197633	818.90	1601.28	533.68	979.51
788481	8083.41	13707.65	7130.53	11192.46

Table 8.8: Corrector computation times in seconds for the GMRES/AMG and Schur-based methods.

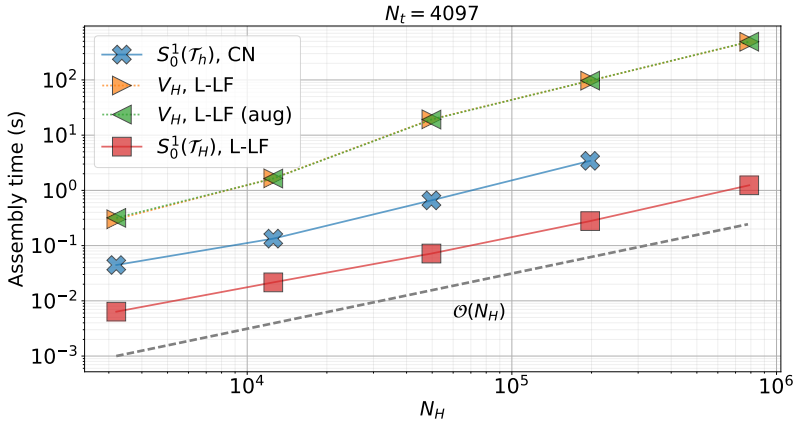


Figure 8.10: Assembly computation times.

N_H	V_H^m L-LF	V_H^m L-LF (aug)	$S_0^1(\mathcal{T}_H)$ L-LF	$S_0^1(\mathcal{T}_h)$ CN
3201	0.30	0.32	0.01	0.04
12545	1.65	1.64	0.02	0.13
49665	19.36	19.14	0.07	0.66
197633	96.91	97.14	0.28	3.43
788481	490.30	492.07	1.24	

Table 8.9: Assembly computation times.

this particular behavior. In particular, the computation exhausted available memory for $m = 4$ and higher for the finest mesh, and so it is natural to think that these computations are running at a very high memory load, which is a well-known cause of slowdowns for memory-bound code that frequently allocates memory. While the code was profiled rather extensively in terms of runtime for smaller problems, almost no consideration was taken with respect to memory efficiency, and so it is likely that the computations expend much more memory than what is truly needed.

The final piece of the puzzle, so to speak, is the cost of assembly. These numbers are presented by Figure 8.10 and Table 8.9. We see that these timings are much smaller than the basis construction times, and so does not have an impact on the conclusions we have made so far.

We will conclude this chapter with a brief discussion of the efficiency of each of the two methods for computing the correctors. We see from 8.9 that the runtimes for the experiments we consider here only differ by a constant factor. In fact, the GMRES/AMG-based method never exceeds twice the runtime of the Schur-based method for all experiments. Moreover, we see that the Schur-based method is competitive even for

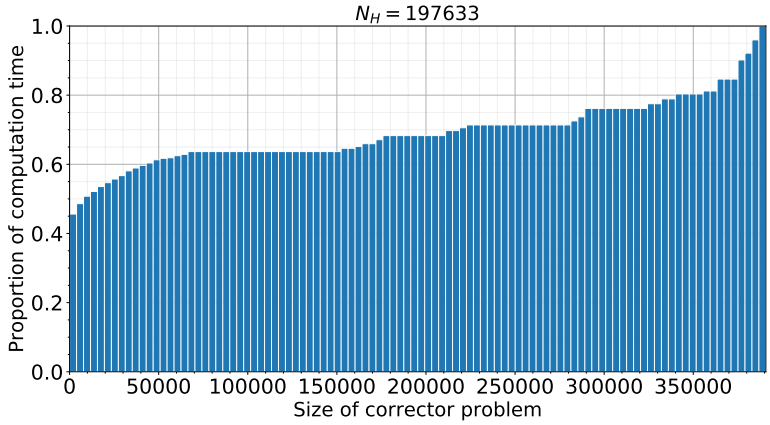


Figure 8.11: Approximate cumulative distribution of the proportion of time spent by the Schur-based solver as a function of the size of the corrector problems. For any value x on the x-axis, the corresponding value on the y-axis represents the proportion of time relative to the total corrector computation time spent on solving corrector problems of size smaller or equal to x. The size of a corrector problem is defined to be $\#\mathcal{N}(\mathcal{T}_{h,T,m})$, the number of vertices in the local fine-scale mesh.

moderately large problems, although due to the memory-related technical issues mentioned earlier, it's not clear if the GMRES/AMG-based method will scale better as the problems grow in size, although the numbers may seem to suggest that this is the case.

It is tempting to think that the bottleneck of the corrector computation is the time it takes to solve the largest corrector problems. In that case, you would be in for a surprise. Consider the results in Figure 8.11. Here we present an approximate discrete cumulative distribution of the proportion of time spent in solving corrector problems of different sizes. The corrector solver used here is the Schur-based one, but results would be similar for the GMRES-based solver. That is, for any value x on the x-axis, the corresponding value on the y-axis represents the proportion of the corrector computation time that was spent in solving corrector problems of size smaller or equal to x. As an example, we see that more than 60 % of the computation time is spent in solving corrector problems of size 70000 or less, which corresponds to the smallest 20 % of corrector problems. It was observed that as N_H grows, the smaller problems seem to demand an increasingly larger proportion of the computation time. Note that in this context we define the size of a corrector problem to be measured by $\#\mathcal{N}(\mathcal{T}_{h,T,m})$, which is the number of vertices in the local fine-scale mesh.

Chapter 9

Concluding remarks

We will conclude this thesis with a summary of what is believed to be the most important conclusions to be made from what has been presented.

9.1 Main takeaways

The main goal of this thesis was to evaluate the method for CFL relaxation proposed by Peterseim and Schedensack [12], and try to determine if it has merit for practical computation. Based on the numerical experiments, we can at least partially answer this question. With regards to online computation - that is, actually solving the problem after corrector computation and matrix assembly, we conclude that:

- The proposed method is a significant improvement over the standard Leapfrog method applied to the *coarse* space $S_0^1(\mathcal{T}_H)$, which suffers from a reduction in convergence rate.
- In the experiments considered, the method was also a significant improvement over the Crank-Nicolson method applied to the fine space $S_0^1(\mathcal{T}_h)$, which was also observed to have accuracy issues, but this comparison is not indicative of the real relationship between the two methods had a more appropriate preconditioner been used for the Crank-Nicolson method.
- Although not currently theoretically justified, mass lumping seems to work very well for the Leapfrog method applied to the corrected space V_H, at least for the model problem considered. It was observed to attain virtually the same error as the Leapfrog method without mass lumping.
- The oversampling parameter m must be chosen judiciously to get the best trade-off between accuracy and performance. Even for the largest experiment considered, with roughly 200 000 vertices in the coarse mesh $\mathcal{T}_H$ and about 6 500 000

vertices in the fine mesh $\mathcal{T}_h$, $m = 3$ was sufficient to attain optimal convergence rates.

- The augmented Leapfrog method proposed in Chapter 7 was observed to give virtually identical errors as that of the standard Leapfrog method.
- The augmented Leapfrog method was seen to yield drastic cost savings in the case when the right-hand side f is very expensive to compute.
- Stability of the augmented Leapfrog method was proved, but optimal convergence rates have still not been proved.
- Even though mass lumping works for V_H and $\dim V_H = \dim S_0^1(\mathcal{T}_H)$, the cost of integration with the mass-lumped Leapfrog method may be drastically higher for V_H due to the increased density of the stiffness matrix, which is used in a matrix-vector product at each time step.

In this thesis, we also proposed two methods for solving the corrector problems that appear in the offline computation associated with forming the space V_H^m. With regards to the offline computation, we make the following conclusions:

- In general, the runtime cost of computing the correctors is *not prohibitive*, in the sense that it can be on the same order of magnitude as solving even a single problem, depending on how many time steps are taken.
- The Schur-based method generally outperforms the GMRES/AMG-based method for the mesh sizes studied here, but only by a factor of at most 2. This might suggest that the preconditioner suggested for the GMRES/AMG-based method is quite appropriate, and that it might be effective also for larger problems.
- The bottleneck of offline computation seems to be the number of small problems to be solved, and not the time to solve the largest problems. It seems that the number of small problems increases faster than the comparative change in size of the large problems. This suggests that using a different corrector solver depending on the size of the corrector problem may be a good idea.

While the author believes the method certainly has merit, it must be pointed out that the implementation is very involved. For many purposes it may be more cost-effective to focus on using an off-the-shelf suitable preconditioner for the Crank-Nicolson method. We would venture a guess that an AMG-based preconditioner would work well.

To ease the implementation difficulties for anyone wishing to study this method, a prototype `C++` library named `crest` is published alongside this thesis. Details are available in Chapter 6.5.

9.2 Possible improvements

We will here briefly summarize some of the possible improvements that — given enough time to implement — the thesis would have benefited from.

- The preconditioner used for the Crank-Nicolson method in the fine space $S_0^1(\mathcal{T}_h)$ in the numerical experiments was very ineffective, and so the results for this method are not indicative of what is possible to achieve with an appropriate preconditioner. With more time, it would perhaps have been wise to attempt to use an AMG-based preconditioner, such as the one we used for the GMRES/AMG-based corrector solver.
- Only a single, artificial model problem was considered. So far the method has not been evaluated on real-world workloads.
- The perturbation of the corrector problems defined in Definition 6.2.4 is not theoretically justified. However, at least for the model problem considered, it works flawlessly in practice. That said, it is possible that it inflates the required value of the oversampling parameter m slightly.
- The corrector computations did not leverage the property which ensures that correctors vanish in non-refined areas of the mesh, as shown in Lemma 5.3.6. For the model problem in question, this probably did not make so much of a difference for the offline runtime, but it may have had some limited effect on the online performance due to accidental fill-in of stiffness matrices (though at the most fill-in would only be reduced by approximately 20%).
- Offline runtime results for the very largest problem may have been affected by memory limitations which may have slowed down computations.

9.3 Applications

We will now briefly suggest some domains for which the method is believed to be particularly useful, and we will also point out some scenarios in which the method is expected to perform poorly.

The method may be particularly useful in the following settings:

- **Real-time simulation**: In this domain one will often readily allow long precomputation times to achieve faster online performance. Moreover, this method may be particularly suited for this domain because Leapfrog with mass lumping avoids any kind of iterative solvers. For real-time computation, one is often equally concerned with *predictable* performance as the speed of the computation. The performance of the mass-lumped Leapfrog method performs exactly the same number of operations at each time step, and so is completely predictable.
- **Optimization**: Because the correctors only need to be computed once for different initial conditions u_0, v_0, right-hand side f and final time T, the method may be well suited for optimization problems in which these quantities vary. In these scenarios one must often solve problems many, many times for different parameters.

- **Long-running simulations**: Since the offline costs are fixed, the relative cost of corrector computation decreases very quickly as the number of time steps increases.
- **Large domains with few re-entrant corners**: In problems where only a relatively small part of the domain needs refinement, the online computations of the Leapfrog method in V_H can be expected to perform at almost the same level as $S_0^1(\mathcal{T}_H)$ in terms of online runtime.

On the other hand, we can expect the method not to be effective or applicable in the following contexts:

- **Domains with complicated boundaries**: If the boundary of the domain does not fit in an appropriately "coarse" quasi-uniform mesh, the method cannot be applied with the goal of recovering Leapfrog stability. For example, if the boundary has edges of size much less than H, a quasi-uniform mesh with mesh size H can not be fit to the domain.
- **Problems with a large portion of mesh refinement**: If most of the area or volume of the domain needs local mesh refinement, the number of non-zeros in the system matrices can be much larger than for $S_0^1(\mathcal{T}_h)$. However, this was actually the case in our model problem, and it still performed relatively well. The fill-in is roughly $\sim Cm^d$ times the number of non-zeros for each refined triangle in $\mathcal{T}_H$, where d is the dimension of the domain and m is the oversampling parameter, so in 3D the density of the system matrices may be prohibitive in these cases.

9.4 Future work

Natural follow-up work to this thesis would be:

- Study the augmented Leapfrog method to see if it can be proved that the optimal convergence rates are recovered with the same assumptions as the ones made for V_H, or if additional assumptions must be made.
- Test more appropriate preconditioners for the Crank-Nicolson method and see how it performs relative to the Leapfrog method applied to V_H^m.
- In this thesis, we only compared the method against standard finite element methods, and not any alternative approaches for overcoming a restrictive CFL condition. This would be a natural next step after comparing with a more appropriately preconditioned Crank-Nicolson method.
- Come up with an efficient way to solve the *exact* localized corrector problem, in the sense that we avoid the perturbation introduced in Definition 6.2.4.
- Evaluate the corrected space V_H for real-world applications.
- Evaluate the performance of the corrected space V_H in 3D.

Bibliography

[1] S. H. Christiansen. "Applied Wave Mathematics: Selected Topics in Solids, Fluids, and Mathematical Methods". In: Springer Berlin Heidelberg, 2009. Chap. Foundations of Finite Element Methods for Wave Equations of Maxwell Type, pp. 335–393.

[2] P. Joly. "Variational Methods for Time-Dependent Wave Propagation Problems". In: *Topics in Computational Wave Propagation: Direct and Inverse Problems*. Springer Berlin Heidelberg, 2003, pp. 201–264.

[3] J.T. Oden and J.N. Reddy. *An Introduction to the Mathematical Theory of Finite Elements*. Dover Publications, 1976.

[4] V. Mazoya and J. Rossmann. *Elliptic equations in polyhedral domains*. 2010.

[5] Fernando D Gaspoz and Pedro Morin. "Convergence rates for adaptive finite elements". In: *IMA journal of numerical analysis* 29.4 (2009), pp. 917–936.

[6] F. L. Müller and C. Schwab. "Finite Elements with mesh refinement for wave equations in polygons". In: *Journal of Computational and Applied Mathematics* 283 (2015), pp. 163–181.

[7] J. Crank and P. Nicolson. "A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type". In: *Mathematical Proceedings of the Cambridge Philosophical Society*. Vol. 43. 01. 1947, pp. 50–67.

[8] J. Diaz and M. J. Grote. "Energy conserving explicit local time-stepping for second-order wave equations". In: *SIAM J. Sci. Comput* (2009), pp. 1985–2014.

[9] J. Diaz and M. J. Grote. "Multi-level explicit local time-stepping methods for second-order wave equations". In: *Computer Methods in Applied Mechanics and Engineering* 291 (2015), pp. 240–265.

[10] M. Hochbruck and A. Sturm. "Error Analysis of a Second-Order Locally Implicit Method for Linear Maxwell's Equations". In: *SIAM Journal on Numerical Analysis* 54.5 (2016), pp. 3167–3191.

[11] P. Ciarlet and J. He. "The Singular Complement Method for 2d scalar problems". In: *Comptes Rendus Mathematique* 336.4 (2003), pp. 353–358.

[12] D. Peterseim and M. Schedensack. "Relaxing the CFL Condition for the Wave Equation on Adaptive Meshes". In: *Journal of Scientific Computing* (2017), pp. 1–18.

[13] Y. Saad and M. H. Schultz. "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems". In: *SIAM Journal on scientific and statistical computing* 7.3 (1986), pp. 856–869.

[14] L. C. Evans. *Partial Differential Equations: Second Edition (Graduate Studies in Mathematics)*. 2nd ed. American Mathematical Society, 2010.

[15] A. Quarteroni. *Numerical Models for Differential Problems*. 2nd ed. Springer, 2014.

[16] S. Brenner and Scott L.R. *The Mathematical Theory of Finite Element Methods*. Springer New York, 2008.

[17] S. Larsson and V. Thomée. *Partial Differential Equations with Numerical Methods*. Springer Berlin Heidelberg, 2003.

[18] T. Dupont. "L^2-estimates for Galerkin methods for second order hyperbolic equations". In: *SIAM journal on numerical analysis* 10.5 (1973), pp. 880–889.

[19] G. A. Baker. "Error estimates for finite element methods for second order hyperbolic equations". In: *SIAM journal on numerical analysis* 13.4 (1976), pp. 564–576.

[20] A. Abdulle and P. Henning. "Localized orthogonal decomposition method for the wave equation with a continuum of scales". In: *Mathematics of Computation* 86.304 (2017), pp. 549–587.

[21] A. Bamberger, G. Chavent, and P. Lailly. *Etude de schémas numériques pour les équations de l'élastodynamique linéaire*. INRIA, 1980.

[22] G. Cohen et al. "Higher Order Triangular Finite Elements with Mass Lumping for the Wave Equation". In: *SIAM Journal on Numerical Analysis* 38.6 (2001).

[23] E. Bänsch. "Local mesh refinement in 2 and 3 dimensions". In: *IMPACT of Computing in Science and Engineering* 3.3 (1991), pp. 181–191.

[24] A. Målqvist and D. Peterseim. "Localization of elliptic multiscale problems". In: *Mathematics of Computation* 83.290 (2014), pp. 2583–2603.

[25] P. Henning and D. Peterseim. "Oversampling for the Multiscale Finite Element Method". In: *Multiscale Model. Simul.* 11.4 (2013), pp. 1149–1175.

[26] P. Henning, P. Morgenstern, and D. Peterseim. "Multiscale partition of unity". In: *Meshfree Methods for Partial Differential Equations VII*. Springer, 2015, pp. 185–204.

[27] C. Engwer et al. "Efficient implementation of the localized orthogonal decomposition method". In: *arXiv preprint arXiv:1602.01658* (2016).

[28] M. Benzi, G. Golub, and J. Liesen. "Numerical solution of saddle point problems". In: *Acta numerica* 14 (2005), pp. 1–137.

[29] J. W. Ruge and K. Stüben. "Algebraic Multigrid". In: *Multigrid Methods*. Chap. 4, pp. 73–130. DOI: 10.1137/1.9781611971057.ch4.

[30] K. Stüben. "A review of algebraic multigrid". In: *Journal of Computational and Applied Mathematics* 128.1–2 (2001), pp. 281–309. DOI: 10.1016/S0377-0427(00)00516-1.

[31] M. F. Murphy, G. Golub, and A. J. Wathen. "A Note on Preconditioning for Indefinite Linear Systems". In: *SIAM Journal on Scientific Computing* 21.6 (2000), pp. 1969–1972. DOI: 10.1137/S1064827599355153.

[32] G. Guennebaud, B. Jacob, et al. *Eigen v3*. http://eigen.tuxfamily.org. 2017.

[33] A. H. Baker, E. R. Jessup, and T. Manteuffel. "A Technique for Accelerating the Convergence of Restarted GMRES". In: *SIAM Journal on Matrix Analysis and Applications* 26.4 (2005), pp. 962–984. DOI: 10.1137/S0895479803422014.

[34] D. Demidov. *AMGCL*. https://github.com/ddemidov/amgcl. Last accessed March 11 2017.

[35] F. D. Witherden and P. E. Vincent. "On the identification of symmetric quadrature rules for finite element methods". In: *Computers & Mathematics with Applications* 69.10 (2015).