Rice University

# Inexact Hierarchical Scale Separation for Linear Systems in Modal Discontinuous Galerkin Discretizations

by

**Christopher Thiele**

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

**Master of Arts**

Approved, Thesis Committee:

Dr. Béatrice M. Rivière,
*Chair*
Noah G. Harding Chair and Professor of Computational and Applied Mathematics

Dr. Matthias Heinkenschloss
Noah G. Harding Chair and Professor of Computational and Applied Mathematics

Dr. Jesse Chan
Assistant Professor of Computational and Applied Mathematics

Dr. Mauricio Araya-Polo
Senior Researcher Computer Science, Shell International Exploration and Production, Inc.

Houston, Texas
April 2018

Rice University

# Inexact Hierarchical Scale Separation for Linear Systems in Modal Discontinuous Galerkin Discretizations

by

**Christopher Thiele**

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

**Master of Arts**

Approved, Thesis Committee:

---

Dr. Béatrice M. Rivière,
*Chair*
Noah G. Harding Chair and Professor of Computational and Applied Mathematics

---

Dr. Matthias Heinkenschloss
Noah G. Harding Chair and Professor of Computational and Applied Mathematics

---

Dr. Jesse Chan
Assistant Professor of Computational and Applied Mathematics

---

Dr. Mauricio Araya-Polo
Senior Researcher Computer Science, Shell International Exploration and Production, Inc.

Houston, Texas
April 2018

# Abstract

Inexact Hierarchical Scale Separation for Linear Systems in Modal Discontinuous Galerkin Discretizations

by

Christopher Thiele

This thesis proposes the inexact hierarchical scale separation (IHSS) method for the solution of linear systems in modal discontinuous Galerkin (DG) discretizations. Like $p$-multigrid methods, IHSS alternates between discretizations of different polynomial order to improve the computational performance of solving linear systems. IHSS uses two discretizations, which are obtained from a hierarchical splitting of the modal DG basis, resulting in two weakly coupled problems for the low-order and high-order components of the solution (coarse and fine scale). While a global linear system of reduced size is solved for the coarse-scale problem, the fine-scale components are updated locally. IHSS extends the original hierarchical scale separation method, using an iterative solver to *approximate* the coarse-scale problem and shifting more computations to the highly parallel local fine-scale updates.

Convergence and computational performance of IHSS are evaluated using an example problem with applications in the oil and gas industry, the simulation of the phase separation of binary fluid mixtures in three spatial dimensions. The problem is modeled by the Cahn–Hilliard equation, a fourth-order, nonlinear partial differential equation, which is discretized using the nonsymmetric interior penalty DG method. Numerical experiments demonstrate the

applicability of IHSS to the linear systems arising in this problem. The results show that the solution of these linear systems can be accelerated significantly when common iterative methods are used as coarse-scale solvers within IHSS instead of being applied directly. All parameters of IHSS are discussed in detail, and the method is easily calibrated for the test problems.

# Acknowledgments

I would like to thank my advisor Dr. Béatrice Rivière for her guidance and support during the preparation of this thesis and for providing helpful insight into discontinuous Galerkin methods and the wider context of my work. Furthermore, I would like to thank Dr. Florian Frank for introducing me to the original hierarchical scale separation method and for his contributions to the development of inexact hierarchical scale separation. Finally, I would like to thank Dr. Mauricio Araya-Polo and Dr. Faruk Omer Alpak from the Shell Technology Center Houston for supporting my research within the Digital Rock project and for providing the computing resources for the numerical experiments in this thesis.

# Contents

# List of figures

# List of tables

Chapter 1

# Introduction

This introductory chapter motivates the need for and construction of linear solvers that are specifically designed for linear systems of equations arising in discontinuous Galerkin approximations. It is divided into two sections. In the first section I discuss the general framework of discontinuous Galerkin methods on a high level of abstraction. I then employ this abstract procedure to derive a class of discontinuous Galerkin methods, the interior penalty discontinuous Galerkin methods, using the well-known example of Poisson's equation. This example will illustrate some of the main features of discontinuous Galerkin methods and will help explain the interest in linear solvers that are suited to these methods.

## 1.1 Discontinuous Galerkin methods

Discontinuous Galerkin methods are a family of methods for the discretization of partial differential equations. They can be viewed as a special class of finite element methods that are based on a local formulation of the problem at hand and a coupling of these local problems using fluxes between neighboring elements. In this sense, discontinuous Galerkin methods build upon the main ideas of finite volume methods and extend them to the fi-

nite element framework. Before I turn to the construction of a discontinuous Galerkin discretization of an example problem in section 1.2, I will describe the structure of these methods on a more abstract level. I begin with a general discussion of nonconforming Galerkin methods, since discontinuous Galerkin methods are typically nonconforming. This presentation is inspired by Di Pietro and Ern [1, p. 7ff.] and Rivière [2, p. 25ff.].

The derivation starts with a Hilbert space $V$ over the field of real numbers $\mathbb{R}$. Furthermore, let $a : V \times V \to \mathbb{R}$ be a bilinear form, and let $\ell : V \to \mathbb{R}$ be a linear form on $V$. Now consider the following variational problem: Find $u \in V$ such that

$$\forall v \in V : a(u, v) = \ell(v). \tag{1.1}$$

I will refer to problem (1.1) as the *weak* problem, as it resembles the weak formulation of a linear partial differential equation. The classical Galerkin approach is to approximate the solution $u \in V$ with an element of a *finite-dimensional* subspace $V_h \subset V$, where the subscript $h$ indicates that the subspace hosts the solution to a discretized version of the weak problem. However, it is also possible to choose a more general finite-dimensional space $W_h \not\subset V$ for the approximation, which leads to so-called *nonconforming* Galerkin methods. In this case, the task is to find $u_h \in W_h$ such that

$$\forall w_h \in W_h : \tilde{a}(u_h, w_h) = \tilde{\ell}(w_h), \tag{1.2}$$

where $\tilde{a}$ and $\tilde{\ell}$ are (bi)linear forms on the space $W_h$ that are related to the (bi)linear forms $a$ and $\ell$ of the weak problem (1.1) in a way that I will soon define more rigorously. From now on, I will refer to problem (1.2) as the *discrete* problem.

Clearly, there must be a connection between the weak and the discrete problem in order for the approximation $u_h$ to be meaningful. It is unreasonable to expect that $u \in V \cap W_h$, as this means that $u$ itself is a member of the approximation space. Thus, we

cannot simply insert the weak solution $u$ into the bilinear form $\tilde{a}$. Instead, suppose that there exists another Hilbert space $W$ that contains $W_h$ as a subspace, and that the (bi)linear forms $\tilde{a}$ and $\tilde{\ell}$ can be extended to all of $W$. This extension allows us to consider the variational problem of finding $\tilde{u} \in W$ such that

$$\forall w \in W : \tilde{a}(\tilde{u}, w) = \tilde{\ell}(w), \tag{1.3}$$

to which I will refer as the *broken weak* problem, because $W$ is usually a broken Sobolev space with respect to some mesh of the domain on which the original problem (1.1) is posed. That is, functions in $W$ satisfy certain regularity requirements when restricted to each element of the mesh. I will properly define and discuss broken Sobolev spaces in the next section. For now it suffices to note that the spaces $W_h$ and $W$ *both* depend on a particular mesh of the problem domain. This is an important observation, because it means that the solution to the broken weak problem (1.3) depends on the mesh as well. As any convergence analysis necessitates the use of multiple meshes, one must resort to the mesh-independent space $V$ to pose the problem whose solution is to be approximated.

The desired relations between the weak, the broken weak, and the discrete problem can now be formulated: I will call the broken weak problem *consistent* if $u \in V \cap W$ solves (1.1) if and only if it also solves (1.3). Thus, consistency states that both problems have the same solution, provided that the solution is contained in both spaces $V$ and $W$. This equivalence of the two problems warrants the search for an approximation in the space $W_h$, even though it is not a subspace of $V$. However, it is important to realize that unlike in conforming methods, essential properties of the bilinear form $a$, like boundedness and coercivity, do not automatically apply to the bilinear form $\tilde{a}$ in the discrete problem, because it is defined on a different space. One must therefore verify the well-posedness of the weak and the broken weak problem separately. In fact, it is a nontrivial task to construct a bilinear form $\tilde{a}$ in a way that ensures coercivity and boundedness while

maintaining consistency with the weak problem. This gives rise to a variety of discontinuous Galerkin methods, as we will see in the next section.

## 1.2 The interior penalty discontinuous Galerkin method: an example

In this section I derive a particular class of discontinuous Galerkin methods, the interior penalty discontinuous Galerkin (IPG) methods, using Poisson's equation as an example. This derivation will illustrate the main features of discontinuous Galerkin methods, and it will allow me to introduce some notation that is used in subsequent chapters. The construction of the method follows Rivière [2, sec. 2.2], and I adhere to the abstract approach from section 1.1 as much as possible.

Let $\Omega \subset \mathbb{R}^d$ be an open polytope, where $d \in \{2, 3\}$. Consider Poisson's problem with homogeneous Dirichlet boundary conditions,

$$-\Delta u = f \text{ in } \Omega, \tag{1.4}$$

$$u = 0 \text{ on } \partial\Omega, \tag{1.5}$$

where I assume sufficient regularity of all involved functions for now. If we multiply with a test function $v$ that satisfies the boundary conditions and integrate by parts, we obtain

$$\underbrace{\int_\Omega \nabla u \cdot \nabla v}_{=:a(u,v)} - \underbrace{\int_{\partial\Omega} \nabla u \cdot \eta_{\partial\Omega} v}_{=0} = \underbrace{\int_\Omega f v}_{=:\ell(v)}, \tag{1.6}$$

where $\eta_{\partial\Omega}$ is the outward-pointing unit normal vector of $\partial\Omega$. This is the well-known variational formulation of (1.4)–(1.5). According to the approach from section 1.1 one should now identify an appropriate Hilbert space $V$ in which the weak problem is posed.

A natural choice is the Sobolev space $V = H_0^1(\Omega)$, as it provides enough regularity for equation (1.6) to be well-defined. Furthermore, the bilinear form

$$a(u, v) = \int_\Omega \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} v$$

is a common inner product on $H_0^1(\Omega)$, and hence existence and uniqueness of the solution $u$ are an immediate consequence of the Riesz representation theorem.

In order to discretize the weak problem, let us introduce a conforming mesh $\mathcal{E}_h$ on $\Omega$ whose elements have a maximum diameter of $h$. The mesh $\mathcal{E}_h$ will typically be a triangular or quadrilateral mesh if $d = 2$, or a tetrahedral or hexahedral mesh if $d = 3$. Let $\Gamma_h$ denote the set of edges or faces of $\mathcal{E}_h$, and let $\mathring{\Gamma}_h$ contain only the interior edges or faces.

Following the approach from section 1.1, one should now define a Hilbert space $W$ based on $\mathcal{E}_h$ in which to pose the broken weak problem. Naturally, only piecewise regularity is enforced for the derivation of a discontinuous Galerkin method, so the broken Sobolev space

$$W = H^s(\mathcal{E}_h) = \left\{ w \in L^2(\Omega) : w|_E \in H^s(E) \text{ for all } E \in \mathcal{E}_h \right\}$$

is a good choice. Choosing $s > 3/2$ will provide sufficient regularity for the following derivation, which requires directional derivatives and their traces on edges or faces of the mesh $\mathcal{E}_h$ [2, theorem 2.5].

Now consider the restriction of problem (1.4) to an element $E \in \mathcal{E}_h$. If we multiply with a test function and use integration by parts, we obtain the local variational problem

$$\int_E \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} w - \int_{\partial E} \boldsymbol{\nabla} u \cdot \boldsymbol{\eta}_{\partial \boldsymbol{E}} w = \int_E f w,$$

but this time the boundary integral does not vanish. Summation over all elements $E \in \mathcal{E}_h$

yields

$$\sum_{E\in\mathcal{E}_h}\int_E \nabla u\cdot\nabla w - \sum_{e\in\mathring{\Gamma}_h}\int_e \left(\nabla u|_{E^-(e)}\cdot\boldsymbol{\eta_e} w|_{E^-(e)} - \nabla u|_{E^+(e)}\cdot\boldsymbol{\eta_e} w|_{E^+(e)}\right) - \sum_{e\in\Gamma_h\setminus\mathring{\Gamma}_h}\int_e \nabla u|_e\cdot\boldsymbol{\eta_e} w|_e = \int_\Omega f w, \tag{1.7}$$

where $E^-(e)$ and $E^+(e)$ denote the two elements that share the edge or face $e$. The assignment is arbitrary, but it must be uniquely determined for each interior edge $e\in\mathring{\Gamma}_h$. For exterior edges $e\in\Gamma_h\setminus\mathring{\Gamma}_h$, I will also use $E^-(e)$ to refer to the adjacent element. By $\boldsymbol{\eta_e}$ I denote the unit normal vector to $e$ that points from $E^-(e)$ to $E^+(e)$ if $e\in\mathring{\Gamma}_h$, and the outward-pointing unit normal vector if $e\in\Gamma_h\setminus\mathring{\Gamma}_h$.

At this point, we can simplify equation (1.7) by introducing some additional notation. Let us define the *jump* of $w\in H^s(\mathcal{E}_h)$ across an edge or face $e\in\mathring{\Gamma}_h$ as the difference

$$[\![w]\!] := w|_{E^-(e)} - w|_{E^+(e)},$$

and the *average* of $w$ on $e$ as

$$\{\!\{w\}\!\} := \frac{1}{2}w|_{E^-(e)} + \frac{1}{2}w|_{E^+(e)}.$$

Furthermore, let us extend these definitions to exterior edges or faces $e\in\Gamma_h\setminus\mathring{\Gamma}_h$ by letting

$$[\![w]\!] = \{\!\{w\}\!\} = w|_{E^-(e)}.$$

Notice that both $[\![w]\!]$ and $\{\!\{w\}\!\}$ are not just scalar quantities, but functions defined on $e$. Observe that

$$[\![\nabla u\cdot\boldsymbol{\eta_e} w]\!] = [\![\nabla u\cdot\boldsymbol{\eta_e}]\!]\{\!\{w\}\!\} + \left\{\!\!\left\{\nabla u\cdot\boldsymbol{\eta_e}\right\}\!\!\right\}[\![w]\!]$$

holds for all interior edges or faces $e \in \mathring{\Gamma}_h$, while we have

$$[\![\nabla u \cdot \eta_e w]\!] = \{\!\{\nabla u \cdot \eta_e\}\!\}\, [\![w]\!]$$

for all exterior edges or faces $e \in \Gamma_h \backslash \mathring{\Gamma}_h$. Both identities are easily verified by direct computation.

Using the new notation and identities, (1.7) can be written as

$$\begin{aligned}
&\sum_{E\in\mathcal{E}_h} \int_E \nabla u \cdot \nabla w - \sum_{e\in\mathring{\Gamma}_h} \int_e [\![\nabla u \cdot \eta_e w]\!] - \sum_{e\in\Gamma_h\backslash\mathring{\Gamma}_h} \int_e [\![\nabla u \cdot \eta_e w]\!] \\
&= \sum_{E\in\mathcal{E}_h} \int_E \nabla u \cdot \nabla w - \sum_{e\in\mathring{\Gamma}_h} \int_e \Big( [\![\nabla u \cdot \eta_e]\!]\, \{\!\{w\}\!\} + \{\!\{\nabla u \cdot \eta_e\}\!\}\, [\![w]\!] \Big) - \sum_{e\in\Gamma_h\backslash\mathring{\Gamma}_h} \int_e \{\!\{\nabla u \cdot \eta_e\}\!\}\, [\![w]\!] \\
&= \int_\Omega f w.
\end{aligned}$$

Now observe that

$$[\![\nabla u \cdot \eta_e]\!] = 0,$$

provided that $u$ is sufficiently smooth (see [1, lemma 1.23]). Since $u$ is the solution of the partial differential equation (1.4), it is reasonable to assume that $u \in H_0^2(\Omega)$. We can therefore drop the term $[\![\nabla u \cdot \eta_e]\!]$ in the above equation to obtain

$$\sum_{E\in\mathcal{E}_h} \int_E \nabla u \cdot \nabla w - \sum_{e\in\Gamma_h} \int_e \{\!\{\nabla u \cdot \eta_e\}\!\}\, [\![w]\!] = \int_\Omega f w. \tag{1.8}$$

The final step in our derivation is to choose (bi)linear forms $\tilde{a}$ and $\tilde{\ell}$ on $W = H^s(\mathcal{E}_h)$ for the broken weak problem. The choice of $\tilde{\ell}(w) = \int_\Omega f w$ is natural, but the left-hand side of (1.8) is not a coercive bilinear form, so it must be further modified to ensure that $\tilde{a}$ has all desired properties while maintaining consistency.

Hence, let us define the bilinear form for the broken weak problem as

$$\begin{aligned}\tilde{a}(w_1, w_2) = &\sum_{E\in\mathcal{E}_h}\int_E \nabla w_1 \cdot \nabla w_2 - \sum_{e\in\Gamma_h}\int_e \left\{\!\!\left\{ \nabla w_1 \cdot \boldsymbol{\eta_e} \right\}\!\!\right\} [\![ w_2 ]\!] \\ &\underbrace{+ \varepsilon \sum_{e\in\Gamma_h}\int_e \left\{\!\!\left\{ \nabla w_2 \cdot \boldsymbol{\eta_e} \right\}\!\!\right\} [\![ w_1 ]\!]}_{=:\mathcal{S}(w_1,w_2)} + \underbrace{\sum_{e\in\Gamma_h} \frac{\sigma}{|e|^\beta}\int_e [\![ w_1 ]\!] [\![ w_2 ]\!]}_{=:\mathcal{P}(w_1,w_2)}. \end{aligned} \tag{1.9}$$

The two additional terms $\mathcal{S}(w_1, w_2)$ and $\mathcal{P}(w_1, w_2)$ each have an intuitive purpose: The term $\mathcal{P}(w_1, w_2)$ penalizes jumps between elements, which helps to make $\tilde{a}$ coercive. I will call $\sigma > 0$ the *penalty parameter*, and $\beta > 0$ is a constant that depends on the spatial dimension $d$ of $\Omega$ [2]. On the other hand, $\mathcal{S}(w_1, w_2)$ makes the bilinear form symmetric if $\varepsilon = -1$, and the resulting method is called the symmetric interior penalty discontinuous Galerkin (SIPG) method. However, other choices of $\varepsilon$ are also possible. For $\varepsilon = 1$ we obtain the nonsymmetric interior penalty discontinuous Galerkin (NIPG) method, whereas $\varepsilon = 0$ results in the incomplete interior penalty discontinuous Galerkin (IIPG) method. The properties and analysis of the different methods are discussed in greater detail in the book by Rivière [2]. Notice that both $\mathcal{S}(w_1, w_2)$ and $\mathcal{P}(w_1, w_2)$ vanish if $w_1 = u \in H_0^1(\Omega) \cap H^s(\mathcal{E}_h)$, so they do not affect consistency.

With the broken weak variational problem in place, it remains to identify a finite-dimensional subspace $W_h \subset H^s(\mathcal{E}_h)$ for the approximation of its solution. Since $s > 3/2$, elements of $H^s(\mathcal{E}_h)$ can be represented by piecewise continuous functions [2, theorem 2.4]. Thus, the broken polynomial space

$$\mathbb{P}_p(\mathcal{E}_h) = \left\{ w \in L^2(\Omega) : w|_E \in \mathbb{P}_p(E) \text{ for all } E \in \mathcal{E}_h \right\}$$

is a natural choice, but other choices are possible and may be preferred in some instances [2, p. 35]. Here, $\mathbb{P}_p(E)$ denotes the space of polynomials of degree less than or

equal to $p$ on the element $E$. With this choice of $W_h$, the discrete problem is to find $u_h \in \mathbb{P}_p(\mathcal{E}_h)$ such that

$$\tilde{a}(u_h, w_h) = \tilde{\ell}(w_h) \tag{1.10}$$

for all $w_h \in \mathbb{P}_p(\mathcal{E}_h)$.

It is interesting to note that while the bilinear form $\tilde{a}$ was constructed to accommodate the choice of $W_h = \mathbb{P}_p(\mathcal{E}_h)$, one could still choose a finite-dimensional subspace $W_h \subset H_0^1(\Omega) \cap H^s(\mathcal{E}_h)$ of *continuous*, piecewise polynomial functions instead. In this case, all jump terms in (1.9) vanish, recovering a continuous Galerkin method.

I will conclude this section by transforming the discrete problem (1.10) into a system of linear equations. Let $M = \dim(\mathbb{P}_p(E))$ and let us denote the number of elements in the mesh $\mathcal{E}_h$ by $N$, i.e., $\mathcal{E}_h = \{E_0, \ldots, E_{N-1}\}$. For each $k = 0, \ldots, N-1$ there exist basis functions $\psi_{k,0}, \ldots, \psi_{k,M-1}$ of $\mathbb{P}_p(E_k)$. If these are extended by zero to all of $\Omega$, then the collection $\bigcup_{k=0}^{N-1} \{\psi_{k,0}, \ldots, \psi_{k,M-1}\}$ is a basis of $\mathbb{P}_p(\mathcal{E}_h)$. Suppose that

$$u_h = \sum_{k=0}^{N-1} \sum_{j=0}^{M-1} u_{k,j} \psi_{k,j} \tag{1.11}$$

is the representation of the discrete solution $u_h$ with respect to this basis. Using the finite-dimensional nature of the space $\mathbb{P}_p(\mathcal{E}_h)$ and the bilinearity of $\tilde{a}$, problem (1.10) can be further simplified. The task is then to find coefficients $u_{k,j}$ such that

$$\sum_{k'=0}^{N-1} \sum_{j'=0}^{M-1} \tilde{a}(\psi_{k',j'}, \psi_{k,j}) u_{k',j'} = \tilde{\ell}(\psi_{k,j}) \tag{1.12}$$

for $k = 0, \dots, N-1$ and $j = 0, \dots, M-1$. For all $k, k' \in \{0, \dots, N-1\}$, let

$$\boldsymbol{u_k} = (u_{k,0}, \dots, u_{k,M-1})^T \in \mathbb{R}^M, \tag{1.13}$$

$$\boldsymbol{\ell_k} = (\tilde{\ell}(\psi_{k,0}), \dots, \tilde{\ell}(\psi_{k,M-1}))^T \in \mathbb{R}^M, \tag{1.14}$$

$$\text{and } \boldsymbol{A_{k,k'}} = \begin{pmatrix} \tilde{a}(\psi_{k',0}, \psi_{k,0}) & \cdots & \tilde{a}(\psi_{k',M-1}, \psi_{k,0}) \\ \vdots & \ddots & \vdots \\ \tilde{a}(\psi_{k',0}, \psi_{k,M-1}) & \cdots & \tilde{a}(\psi_{k',M-1}, \psi_{k,M-1}) \end{pmatrix} \in \mathbb{R}^{M \times M}. \tag{1.15}$$

The linear system (1.12) can now be written as

$$\begin{pmatrix} \boldsymbol{A_{0,0}} & \cdots & \boldsymbol{A_{0,N-1}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{A_{N-1,0}} & \cdots & \boldsymbol{A_{N-1,N-1}} \end{pmatrix} \begin{pmatrix} \boldsymbol{u_0} \\ \vdots \\ \boldsymbol{u_{N-1}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\ell_0} \\ \vdots \\ \boldsymbol{\ell_{N-1}} \end{pmatrix}. \tag{1.16}$$

To solve this linear system efficiently, we must understand some of its essential properties.

It is evident from equation (1.16) that the system matrix is composed of $N \times N$ blocks, each of size $M \times M$. Recall that each basis function $\psi_{k,j}$ is supported on just a single element $E_k \in \mathcal{E}_h$. Hence, $\tilde{a}(\psi_{k',j'}, \psi_{k,j})$ can be nonzero only if the elements $E_{k'}$ and $E_k$ are either identical or adjacent, because each integrand in (1.9) is a product of gradients, jumps, and averages, and at least one of the factors in each integrand is zero if the involved functions are supported on elements that are not identical or adjacent. We conclude that the system matrix in (1.16) is very sparse, provided that the number $N$ of elements in the mesh is sufficiently large.

Another important property of the linear system (1.16) is that the unknowns $u_{k,j}$ are not shared between the elements of the mesh $\mathcal{E}_h$. Instead, each element $E_k \in \mathcal{E}_h$ corresponds to a block $\boldsymbol{u_k}$ in the vector of unknowns, and all of these blocks are of equal size $M$. This property will be one of the key ingredients for the construction of the hierarchical scale separation (HSS) method in chapter 3.

I conclude this introduction with an overview of the remaining chapters: In chapter 2, I discuss the origins of HSS and review the related literature. The HSS algorithm is derived in chapter 3. This chapter also motivates several modifications to the algorithm, resulting in the inexact hierarchical scale separation (IHSS) method, which is the focus of this thesis. Chapter 3 also discusses the efficient implementation of the IHSS algorithm. The convergence and computational performance of the method are evaluated in chapter 4. The chapter begins with the introduction of the test problem, the simulation of the phase separation of a binary fluid mixture, which is governed by the Cahn–Hilliard equation. Section 4.1 discusses the discretization of the test problem. In the remaining sections of chapter 4, I evaluate the computational performance of the IHSS method when applied to the test problem and how performance is affected by the method's parameters. The final chapter summarizes the main results and suggests ideas for future work.

Chapter 2

# Literature review

The inexact hierarchical scale separation (IHSS) method, which is derived and analyzed in later chapters of this thesis, was recently proposed by Araya-Polo, Alpak, Rivière, Frank, and myself as a way to accelerate the solution of linear systems in modal discontinuous Galerkin (DG) discretizations [3]. It is a modification and extension of the original hierachical scale separation (HSS) method, which was introduced by Kuzmin in 2014 [4]. Before discussing the origins of the HSS method itself, let us begin with an overview of iterative linear solvers in general and solvers for linear systems arising in the discretization of partial differential equations in particular. I will start with a discussion of stationary iterative methods and Krylov subspace methods in section 2.1. After that, I will review the main ideas of multigrid, $p$-multigrid, and hierarchical scale separation methods in sections 2.2 and 2.3. This overview is not intended to be a comprehensive review of the vast body of literature on iterative solvers. Instead, I will highlight some of the important contributions in this field that led to the development of HSS, IHSS, and related methods. This context will also help to better understand the key ideas in the derivation of the IHSS algorithm in chapter 3.

## 2.1 Iterative linear solvers

Systems of linear equations are at the core of many problems in applied mathematics, and efficient linear solvers are highly relevant for a multitude of applications in science and engineering [5, preface]. While the development of linear solvers and preconditioners for specific applications is an active field of research, it is unique in the sense that the problem of solving a system of linear equations seems almost trivial from a purely theoretical perspective. An explicit formula for the solution of linear systems was published by Cramer in 1750 [6], and an algorithm for numerical calculations, Gaussian elimination, is known to most students at the early undergraduate or even high school level. Hence, advances in the development of linear solvers are not primarily driven by a desire to expand mathematical theory, but by the restrictions that available computing resources impose on speed, numerical stability, and scalability of existing algorithms. These restrictions, the specific properties of the linear systems at hand, and the required accuracy of the numerical solution give rise to numerous linear solvers.

When the exact solution of a linear system is not needed within machine precision, iterative linear solvers can be used to approximate the solution step by step until a prescribed tolerance is met. Stationary iterative methods such as the Richardson, Jacobi, Gauss-Seidel, and SOR method are among the early iterative linear solvers, but they remain relevant to this day, as we will see in the next section (see also [5, chapter 4]). These methods split the matrix $\boldsymbol{A}$ of a linear system

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

into

$$\boldsymbol{A} = \boldsymbol{B} + \boldsymbol{C}.$$

The resulting linear system

$$\boldsymbol{B}\boldsymbol{x} + \boldsymbol{C}\boldsymbol{x} = \boldsymbol{b}$$

then leads to the iteration

$$\boldsymbol{B}\boldsymbol{x}^{(k+1)} = \boldsymbol{b} - \boldsymbol{C}\boldsymbol{x}^{(k)},$$

starting from an initial guess $\boldsymbol{x}^{(0)}$. The splitting is chosen such that the resulting linear systems with system matrix $\boldsymbol{B}$ are easier to solve than the original system with matrix $\boldsymbol{A}$. Natural choices for $\boldsymbol{B}$ are the diagonal part of $\boldsymbol{A}$ (Jacobi method) or one of its triangular components (Gauss-Seidel method). A more detailed overview of stationary iterative methods can be found in chapter 4 of the book by Saad [5].

Another popular class of iterative linear solvers are Krylov subspace methods. If we assume for simplicity that the initial guess is $\boldsymbol{x}^{(0)} = \boldsymbol{0}$, these methods approximate the solution $\boldsymbol{x}$ in the subspaces

$$\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{b}) = \text{span}\{\boldsymbol{b}, \boldsymbol{A}\boldsymbol{b}, \ldots, \boldsymbol{A}^{(k-1)}\boldsymbol{b}\}.$$

An intuition for the choice of these particular subspaces can be gained by considering the minimal polynomial $\chi(t) = \alpha_0 + \alpha_1 t + \ldots + \alpha_d t^d$ of the matrix $\boldsymbol{A}$, which is the polynomial of least degree that satisfies $\chi(\boldsymbol{A}) = \boldsymbol{0}$. Provided that $\alpha_0 \neq 0$, the solution of the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ can then be expressed as

$$\boldsymbol{A}^{-1}\boldsymbol{b} = -\frac{1}{\alpha_0}\left(\alpha_1 \boldsymbol{b} + \ldots + \alpha_d \boldsymbol{A}^d \boldsymbol{b}\right) \in \mathcal{K}_{d+1}(\boldsymbol{A}, \boldsymbol{b}).$$

The way by which $\boldsymbol{x}$ is approximated in $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{b})$ depends on the particular Krylov subspace method. The well-known conjugate gradient (CG) method for symmetric positive

definite matrices $\boldsymbol{A}$ finds the minimizer of the functional

$$f(\boldsymbol{s}) = \frac{1}{2}\boldsymbol{s}^T\boldsymbol{A}\boldsymbol{s} - \boldsymbol{s}^T\boldsymbol{b}$$

in $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{b})$ to approximate the solution of the linear system, which satisfies

$$\nabla f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} = \boldsymbol{0}$$

and is therefore the unique global minimizer of $f$ [5, section 6.7].

The Generalized Minimum Residual (GMRES) method instead minimizes the residual

$$f(\boldsymbol{s}) = \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{s}\|_2$$

in $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{b})$ to approximate the solution of the linear system [5, section 6.5].

## 2.2 Multigrid and $p$-multigrid

While certain stationary iterative methods or Krylov subspace methods may perform better than others for a given problem, the algorithms themselves are not designed for any specific application. The main ideas of these methods were derived in the previous section using only the tools of linear algebra, and these iterative solvers are guaranteed to converge for any linear system whose coefficient matrix satisfies certain algebraic properties like symmetry and positive definiteness in case of the conjugate gradient method [5, section 6.7]. In this sense, the methods discussed so far can be viewed as general-purpose iterative solvers.

A different strategy is to incorporate the specifics of the application into the design of the linear solver. While the resulting algorithm is less general, this approach can significantly accelerate the linear solves in the targeted application [5, preface]. A popular

class of methods that follow this approach are multigrid methods. These methods solve linear systems arising in the discretization of partial differential equations, using a hierarchy of discretizations with decreasing spatial resolution, i.e., multiple computational grids. A linear system is solved to high accuracy only for the coarsest grid. On all other grids, the corresponding linear systems are approximated roughly (smoothing), often using a fixed number of iterations with a stationary iterative method (see previous section). These approximations are used to construct the right-hand side of the linear systems for coarser grids (restriction) until the coarsest grid is reached. After the linear system for the coarsest grid has been solved, the coarse-scale information is propagated back to finer grids (prolongation or interpolation). The multigrid method then cycles between fine and coarse grids until the residual on the finest scale is sufficiently small. The order in which the grids are visited may vary. For an overview of different cycling strategies, common smoothers, restriction and interpolation techniques, as well as a general introduction to multigrid methods, I refer the reader to the book by Wesseling [7].

A more detailed discussion of the convergence behavior and performance of multigrid methods is beyond the scope of this overview. However, the central idea of multigrid is to view the discretization of a partial differential equation as a *hierarchy* of linear systems on different scales and to solve or approximate the linear systems on each scale in the most efficient way using different linear solvers. I will revisit this idea when deriving the IHSS algorithm in chapter 3, but there is another method that is based on this approach and that links multigrid with IHSS: the $p$-multigrid methods.

Instead of discretizing a partial differential equation with different spatial resolutions, $p$-multigrid methods are based on a hierarchy of discretizations with different polynomial order. The fine and coarse grids in the original multigrid method correspond to approximations of higher and lower order in $p$-multigrid. The method was first introduced by Fidkowski et al. in 2005 [8], building on ideas published by Rønquist and Patera for spec-

tral element methods in 1987 [9].

Finally, the IHSS method shares a central idea with the hierarchical basis multigrid method, which was introduced by Bank et al. in 1988 [10]. Both methods use a hierarchy of bases for the different scales of approximation, and the unknowns associated with each level of the hierarchy are updated *independently*. However, unlike IHSS and the $p$-multigrid method, hierachical basis multigrid does not use a hierarchy of basis functions with different polynomial degree. Instead, the hierarchy is constructed by selecting nodal basis functions of the same degree from grids with different refinement levels.

## 2.3 Hierarchical scale separation

The HSS method was introduced by Kuzmin in 2014 as a technique to solve the linear systems in discontinuous Galerkin (DG) discretizations with hierarchical basis functions [4]. Although the algorithm proposed in this first publication already resembles what I will present as the HSS method in chapter 3 (see algorithm 1), Kuzmin did not focus on the computational performance of the method. Instead, HSS was presented as a way to extend hierarchical slope limiting strategies to time-implicit discretizations. In fact, the original HSS algorithm explicitly includes a slope limiting step [4, p. 1149].

The computational performance of the method was first analyzed in a subsequent publication by Aizinger, Kuzmin, and Korous in 2015 [11]. Their paper compares HSS to the $p$-multigrid method, both from an algorithmic perspective and in numerical experiments. Like $p$-multigrid methods, HSS cycles between multiple scales that correspond to polynomial approximations of different order, and a linear system is solved for each scale. However, the HSS method uses only two scales, coarse and fine, which in turn are less coupled than in the case of $p$-multigrid [11, p. 2]. The paper of Aizinger et al. also presents HSS as a more general algorithm in a linear algebra framework. It no longer includes an explicit slope limiting step, but the algorithm is still closely connected to the discretization

of a partial differential equation, as it is presented as a (pseudo-)timestepping method [11, algorithm 1]. The paper compares the computational performance of HSS to that of $p$-multigrid using a convection-diffusion problem in 2D. In these experiments, the authors demonstrate a performance improvement with HSS in most test cases, with reductions in CPU time by 50% and more in many cases [11, table 1].

Motivated by the paper of Aizinger et al., I developed the IHSS method with Araya-Polo, Alpak, Rivière, and Frank. Our first publication presented HSS as an algebraic approach to the solution of linear systems in modal discontinuous Galerkin discretizations that does not include (pseudo-)timestepping or other specifics of the application [12]. The first key idea of the IHSS method is to approximate the linear system on the coarse scale using an iterative linear solver to save CPU time. The second modification is to shift more work to the highly parallel fine-scale solver using a different cycling between the scales. While this initial version of the IHSS method showed promising computational performance, it is difficult to calibrate it in the sense that the convergence of the coarse-scale and fine-scale components is often unbalanced. We addressed this issue in a second publication, in which we modified the IHSS algorithm to automatically maintain this balance while preserving good computational performance [3]. I will use this modified IHSS algorithm for the numerical experiments in this thesis, after deriving it step by step in chapter 3.

Another recent development was the application of the original HSS algorithm to hybridized DG discretizations by Schütz and Aizinger in 2017 [13]. Using a convection-diffusion problem in 2D, the authors verify the convergence of the method for approximation orders of up to $p = 7$. They also compare the convergence of HSS when applied to hybridized and non-hybridized DG discretizations of the same problem, showing that for many test problems fewer HSS iterations are required in the hybridized case [13, figure 10]. Jaust, Schütz, and Aizinger extended this work to the hybridized DG discretization

of Burgers' equation, i.e., to a nonlinear problem, again verifying convergence of the HSS method [14].

Chapter 3

# Inexact hierarchical scale separation

In this chapter, I first motivate and derive the original hierarchical scale separation (HSS) scheme. After a discussion of its main properties and potential improvements, I modify the algorithm to obtain the *inexact* hierarchical scale separation (IHSS) scheme, which I first published with Araya-Polo, Alpak, Rivière, and Frank in 2017 [3, 12]. For a discussion of the history and development of these methods, please refer to the literature review in chapter 2. In the final section of this chapter, I discuss the efficient parallel implementation of the IHSS method.

## 3.1 The hierarchical scale separation scheme

The starting point for the derivation of the HSS method is the linear system (1.16), which results from the discontinuous Galerkin discretization outlined in section 1.2. Suppose that the basis $\{\psi_{k,0}, \ldots, \psi_{k,M-1}\}$ on each element $E_k \in \mathcal{E}_h$ admits a splitting

$$\{\psi_{k,0}, \ldots, \psi_{k,M-1}\} = \left\{\overline{\psi}_{k,0}, \ldots, \overline{\psi}_{k,\overline{M}-1}\right\} \cup \left\{\widehat{\psi}_{k,0}, \ldots, \widehat{\psi}_{k,\widehat{M}-1}\right\}, \; M = \overline{M} + \widehat{M},$$

such that $\left\{\overline{\psi}_{k,0},\dots,\overline{\psi}_{k,\overline{M}-1}\right\}$ forms a basis of $\mathbb{P}_q(E_k)$ for some $q < p$. That is, the set $\bigcup_{k=0}^{N-1}\left\{\overline{\psi}_{k,0},\dots,\overline{\psi}_{k,\overline{M}-1}\right\}$ forms a basis of the broken polynomial space $\mathbb{P}_q(\mathcal{E}_h)$ with a polynomial order less than that of $\mathbb{P}_p(\mathcal{E}_h)$, the space which contains the discrete solution $u_h$. I will therefore call this basis *hierarchical*, as it gives rise to a hierarchy of broken polynomial spaces $\mathbb{P}_q(\mathcal{E}_h) \subset \mathbb{P}_p(\mathcal{E}_h)$. As before, the functions $\psi_{k,j}, \overline{\psi}_{k,j}$, and $\widehat{\psi}_{k,j}$ should be interpreted either as functions on the element $E_k$ or as their extensions by zero to all of $\Omega$, depending on the context.

Henceforth, I will refer to $\mathbb{P}_q(\mathcal{E}_h)$ as the *coarse scale*, whereas $\mathbb{P}_p(\mathcal{E}_h)\backslash\mathbb{P}_q(\mathcal{E}_h)$ will be referred to as the *fine scale*. Any functions, quantities, etc., related to the coarse scale will be marked with a line, e.g., $\overline{\psi}_{k,j}$, and functions, quantities, etc., related to the fine scale will be marked with a hat, e.g., $\widehat{\psi}_{k,j}$.

With this notation, the discrete solution (1.11) can be written as

$$u_h = \sum_{k=0}^{N-1}\left(\sum_{j=0}^{\overline{M}-1}\overline{u}_{k,j}\overline{\psi}_{k,j} + \sum_{j=0}^{\widehat{M}-1}\widehat{u}_{k,j}\widehat{\psi}_{k,j}\right),$$

and the linear system (1.16) can be rearranged into

$$\underbrace{\begin{pmatrix}\overline{\boldsymbol{A}} & \overline{\boldsymbol{C}}\\ \widehat{\boldsymbol{C}} & \widehat{\boldsymbol{A}}\end{pmatrix}}_{=:\boldsymbol{A}_h}\underbrace{\begin{pmatrix}\overline{\boldsymbol{u}}\\ \widehat{\boldsymbol{u}}\end{pmatrix}}_{=:\boldsymbol{u}_h} = \underbrace{\begin{pmatrix}\overline{\boldsymbol{\ell}}\\ \widehat{\boldsymbol{\ell}}\end{pmatrix}}_{=:\boldsymbol{\ell}_h} \tag{3.1}$$

where I simply accumulated all coarse-scale quantities in the upper left-hand block $\overline{\boldsymbol{A}}$.

More specifically, let us define

$$\begin{aligned}
\overline{\boldsymbol{u}} &= (\overline{u}_{0,0},\dots,\overline{u}_{0,\overline{M}-1},\overline{u}_{1,0},\dots,\overline{u}_{1,\overline{M}-1}\dots,\overline{u}_{N-1,0},\dots,\overline{u}_{N-1,\overline{M}-1})^T \in \mathbb{R}^{N\overline{M}},\\
\widehat{\boldsymbol{u}} &= (\widehat{u}_{0,0},\dots,\widehat{u}_{0,\widehat{M}-1},\widehat{u}_{1,0},\dots,\widehat{u}_{1,\widehat{M}-1}\dots,\widehat{u}_{N-1,0},\dots,\widehat{u}_{N-1,\widehat{M}-1})^T \in \mathbb{R}^{N\widehat{M}},\\
\overline{\boldsymbol{\ell}} &= (\overline{\ell}_{0,0},\dots,\overline{\ell}_{0,\overline{M}-1},\overline{\ell}_{1,0},\dots,\overline{\ell}_{1,\overline{M}-1}\dots,\overline{\ell}_{N-1,0},\dots,\overline{\ell}_{N-1,\overline{M}-1})^T \in \mathbb{R}^{N\overline{M}},\\
\widehat{\boldsymbol{\ell}} &= (\widehat{\ell}_{0,0},\dots,\widehat{\ell}_{0,\widehat{M}-1},\widehat{\ell}_{1,0},\dots,\widehat{\ell}_{1,\widehat{M}-1}\dots,\widehat{\ell}_{N-1,0},\dots,\widehat{\ell}_{N-1,\widehat{M}-1})^T \in \mathbb{R}^{N\widehat{M}}.
\end{aligned}$$

The matrix $\boldsymbol{A_h}$ in (3.1) is then obtained by permuting the rows and columns of the system matrix in (1.16) accordingly.

While it is instructive to consider the original system matrix with a structure as shown in (1.13)–(1.16), which is essentially the Gram matrix of the bilinear form $\tilde{a}$, it should be noted that the HSS scheme can also be applied to more general matrices, e.g., matrices that are the result of a Schur complement reduction (see chapter 4 and [3]). In this sense, the above rearrangement of the linear system is a purely algebraic manipulation that depends only on the numbers $\overline{M}$ and $\widehat{M}$ in the splitting of the basis functions.

The final step of preparation is to split the rearranged system (3.1) into two coupled linear systems

$$\begin{cases} \overline{\boldsymbol{A}}\overline{\boldsymbol{u}} + \overline{\boldsymbol{C}}\widehat{\boldsymbol{u}} &= \overline{\boldsymbol{\ell}} \\ \widehat{\boldsymbol{C}}\overline{\boldsymbol{u}} + \widehat{\boldsymbol{A}}\widehat{\boldsymbol{u}} &= \widehat{\boldsymbol{\ell}} \end{cases}. \tag{3.2}$$

If we furthermore split $\widehat{\boldsymbol{A}}$ into $\widehat{\boldsymbol{A}} =: \widehat{\boldsymbol{A}}_{\mathbf{diag}} + \widehat{\boldsymbol{A}}_{\mathbf{off}}$, where $\widehat{\boldsymbol{A}}_{\mathbf{diag}}$ is the $\widehat{M}\times\widehat{M}$ block diagonal of $\widehat{\boldsymbol{A}}$, the system (3.2) can be written as

$$\begin{cases} \overline{\boldsymbol{A}}\overline{\boldsymbol{u}} &= \overline{\boldsymbol{\ell}} - \overline{\boldsymbol{C}}\widehat{\boldsymbol{u}} \\ \widehat{\boldsymbol{A}}_{\mathbf{diag}}\widehat{\boldsymbol{u}} &= \widehat{\boldsymbol{\ell}} - \widehat{\boldsymbol{C}}\overline{\boldsymbol{u}} - \widehat{\boldsymbol{A}}_{\mathbf{off}}\widehat{\boldsymbol{u}} \end{cases}, \tag{3.3}$$

which is still equivalent to (3.1).

The main idea of HSS is now to start with initial guesses $\overline{\boldsymbol{u}}^{(0)}$ and $\widehat{\boldsymbol{u}}^{(0)}$ and to alternate

between the two linear systems in (3.3) until both components converge, using the most recent approximations to $\overline{\boldsymbol{u}}$ and $\widehat{\boldsymbol{u}}$ on the right-hand side of both systems to solve for a new approximation [4, 11]. The full method is described in algorithm 1, which takes as inputs the matrix $\boldsymbol{A_h}$, right-hand side $\boldsymbol{\ell_h}$, and initial guess $\boldsymbol{u_h}^{(0)}$, as well as a tolerance $\eta$ for the relative residual.

**Algorithm 1:** Hierarchical scale separation (cf. [4, 11])

$\boldsymbol{u_h} = \mathsf{HSS}(\boldsymbol{A_h},\ \boldsymbol{\ell_h},\ \boldsymbol{u_h}^{(0)},\ \eta)$

*1* $i := 0$

*2 While* $\|\boldsymbol{\ell_h} - \boldsymbol{A_h}\boldsymbol{u_h}^{(i)}\| / \|\boldsymbol{\ell_h} - \boldsymbol{A_h}\boldsymbol{u_h}^{(0)}\| \geq \eta$*:*

*2.1* $i \leftarrow i + 1$

*2.2 Solve coarse-scale system* $\overline{\boldsymbol{A}}\overline{\boldsymbol{u}}^{(i)} = \overline{\boldsymbol{\ell}} - \overline{\boldsymbol{C}}\widehat{\boldsymbol{u}}^{(i-1)}$.

*2.3 Solve fine-scale system* $\widehat{\boldsymbol{A}}_{\mathbf{diag}}\widehat{\boldsymbol{u}}^{(i)} = \widehat{\boldsymbol{\ell}} - \widehat{\boldsymbol{C}}\overline{\boldsymbol{u}}^{(i)} - \widehat{\boldsymbol{A}}_{\mathbf{off}}\widehat{\boldsymbol{u}}^{(i-1)}$.

*3 Accept solution* $\boldsymbol{u_h} := \boldsymbol{u_h}^{(i)}$.

There are two key observations: Due to the block-diagonal structure of $\widehat{\boldsymbol{A}}_{\mathbf{diag}}$, the fine-scale solve in step 2.3 of algorithm 1 is equivalent to solving $N$ independent linear systems, each of size $\widehat{M} \times \widehat{M}$. As $\widehat{M}$ is usually very small compared to $N$, these linear solves can be performed efficiently and in parallel, as I will discuss in sections 3.2 and 3.5. The second observation is that the linear system in step 2.2 of algorithm 1 is of size $N\overline{M} \times N\overline{M}$ and smaller than the original system (3.1), which has size $NM \times NM$. Thus, one might expect that solving the coarse-scale system is computationally less expensive than solving the original system. However, the tradeoff is that the coarse-scale update is repeated in each iteration of the HSS algorithm. The numerical experiments in chapter 4 will show that this tradeoff can indeed be beneficial in applications, but first I motivate some further modifications to algorithm 1 in the next section.

## 3.2 The inexact hierarchical scale separation scheme

The inexact hierarchical scale separation (IHSS) scheme modifies and extends the HSS method in order to use the highly parallel fine-scale update more extensively. These modifications, which will be discussed in more detail soon, are based on two key ideas: Since HSS updates the coarse-scale and fine-scale components of the solution in an iterative process, it may not be necessary to solve the coarse-scale system to machine precision in each iteration. Instead, an iterative linear solver can be used to approximate the coarse-scale component $\overline{\boldsymbol{u}}^{(i)}$ in each HSS iteration. The second key idea of the IHSS method is based on the observation that one of the fine-scale approximations $\widehat{\boldsymbol{u}}^{(i)}$ and $\widehat{\boldsymbol{u}}^{(i-1)}$ appears on either side of the linear system in step 2.3 of algorithm 1. Thus, the fine-scale update resembles a fixed-point iteration, and it can be repeated multiple times before returning to the coarse scale.

Combining the approximation of the coarse-scale system with the repeated fine-scale updates results in a basic version of the IHSS method, which I first published with Araya-Polo, Alpak, Rivière, and Frank in 2017 [12]. It is presented in algorithm 2. The function `CSS` is simply an iterative solver for the coarse-scale system. Its arguments are the system matrix, right-hand side, an initial guess, and a tolerance. The function `FSS` on the other hand solves the fine-scale system to machine precision. The method also introduces a parameter $\delta > 0$ that governs the approximation of $\overline{\boldsymbol{u}}^{(i)}$ (see algorithm 2, step 2.2) and a parameter $\nu \in \mathbb{N}$ that determines the number of fine-scale updates in each IHSS iteration (see algorithm 2, step 2.4). For now, the tolerance $\delta$ can be thought of as either an absolute or relative tolerance. Notice that the choice of $\nu = 1$ and sufficiently small $\delta$ essentially recovers the HSS method in algorithm 1.

While algorithm 2 already illustrates the workings of IHSS, it is of limited use for actual computations, because its computational performance depends strongly on an appropriate

**Algorithm 2:** Basic inexact hierarchical scale separation (as first published by Thiele et al. in 2017 [12])

$\boldsymbol{u_h}$ = IHSS_basic($\boldsymbol{A_h}$, $\boldsymbol{\ell_h}$, $\boldsymbol{u_h}^{(0)}$, $\eta$, $\delta$, $\nu$)

*1* $i := 0$

*2 While* $\|\boldsymbol{\ell_h} - \boldsymbol{A_h}\boldsymbol{u_h}^{(i)}\| / \|\boldsymbol{\ell_h} - \boldsymbol{A_h}\boldsymbol{u_h}^{(0)}\| \geq \eta$*:*

*2.1* $i \leftarrow i + 1$

*2.2* $\overline{\boldsymbol{u}}^{(i)} := \mathrm{CSS}\left(\overline{\boldsymbol{A}},\ \overline{\boldsymbol{\ell}} - \overline{\boldsymbol{C}}\widehat{\boldsymbol{u}}^{(i-1)},\ \overline{\boldsymbol{u}}^{(i-1)},\ \delta\right)$ *(approximate coarse-scale system)*

*2.3* $\widehat{\boldsymbol{u}}_0^{(i)} := \widehat{\boldsymbol{u}}^{(i-1)}$

*2.4 For* $k = 1, \ldots, \nu$*:*

*2.4.1* $\widehat{\boldsymbol{u}}_k^{(i)} := \mathrm{FSS}\left(\widehat{\boldsymbol{A}}_{\mathbf{diag}},\ \widehat{\boldsymbol{\ell}} - \widehat{\boldsymbol{C}}\overline{\boldsymbol{u}}^{(i)} - \widehat{\boldsymbol{A}}_{\mathbf{off}}\widehat{\boldsymbol{u}}_{k-1}^{(i)}\right)$ *(solve fine-scale system)*

*2.5* $\widehat{\boldsymbol{u}}^{(i)} := \widehat{\boldsymbol{u}}_\nu^{(i)}$

*3 Accept solution* $\boldsymbol{u_h} := \boldsymbol{u_h}^{(i)}$.

choice of the parameters $\delta$ and $\nu$ (see chapter 4 and [12]). Moreover, it is not obvious that the repeated fine-scale solves in step 2.4 are indeed a fixed point iteration in all cases. Thus, the IHSS algorithm is further modified in sections 3.3 and 3.4 to resolve these issues.

## 3.3 Adaptive choice of coarse-scale solver tolerances

The approximation of the coarse-scale system is one of the features that distinguish the IHSS method from the original HSS method. Intuitively, one might expect the coarse-scale solver tolerance $\delta$ in algorithm 2 to have a major impact on the behavior of the IHSS method: If $\delta$ is small, the coarse-scale systems are solved to an accuracy that may not be needed for sufficiently fast convergence of the method, incurring unnecessary computational cost. On the other hand, if $\delta$ is chosen too large, the coarse-scale approximation $\overline{\boldsymbol{u}}^{(i)}$ may converge slowly, or it may not converge at all.

A simple way of choosing $\delta$ is to use a fixed relative tolerance in all IHSS iterations.

Our first experiments with the method showed that this approach can result in good computational performance. However, the experiments also revealed that $\delta$ and $\nu$, the number of fine-scale updates per IHSS iteration, cannot be chosen independently (see section 4.3 and [12]).

While this interplay of $\delta$ and $\nu$ makes it difficult to calibrate the IHSS method, it can also be used to choose one of the parameters automatically. My collaborators and I proposed an approach to effectively eliminate the coarse-scale tolerance $\delta$ from the calibration process in our second publication on IHSS [3]. This approach, which is presented below, is based on the intuition that the convergence of the coarse-scale and fine-scale components should be balanced: It seems unreasonable to solve the coarse-scale system to great accuracy while the error in the fine-scale approximation $\widehat{\boldsymbol{u}}^{(i)}$ is still large and vice versa. To formalize this idea, consider the absolute residual $\|\boldsymbol{\ell}_h - \boldsymbol{A}_h {\boldsymbol{u}_h}^{(i-1)}\|$ of the linear system 3.1 after $i-1$ IHSS iterations. If we choose the Euclidean norm, the block structure of the system allows us to write

$$\|\boldsymbol{\ell}_h - \boldsymbol{A}_h {\boldsymbol{u}_h}^{(i-1)}\|_2^2 = \left\| \begin{pmatrix} \overline{\boldsymbol{\ell}} \\ \widehat{\boldsymbol{\ell}} \end{pmatrix} - \begin{pmatrix} \overline{\boldsymbol{A}} & \overline{\boldsymbol{C}} \\ \widehat{\boldsymbol{C}} & \widehat{\boldsymbol{A}} \end{pmatrix} \begin{pmatrix} \overline{\boldsymbol{u}}^{(i-1)} \\ \widehat{\boldsymbol{u}}^{(i-1)} \end{pmatrix} \right\|_2^2 \tag{3.4}$$

$$= \|\overline{\boldsymbol{\ell}} - \overline{\boldsymbol{A}}\overline{\boldsymbol{u}}^{(i-1)} - \overline{\boldsymbol{C}}\widehat{\boldsymbol{u}}^{(i-1)}\|_2^2 + \|\widehat{\boldsymbol{\ell}} - \widehat{\boldsymbol{C}}\overline{\boldsymbol{u}}^{(i-1)} - \widehat{\boldsymbol{A}}\widehat{\boldsymbol{u}}^{(i-1)}\|_2^2. \tag{3.5}$$

Since the convergence of the coarse-scale and fine-scale components should be balanced, both terms in equation (3.5) should be of similar magnitude, i.e.,

$$\|\overline{\boldsymbol{\ell}} - \overline{\boldsymbol{A}}\overline{\boldsymbol{u}}^{(i-1)} - \overline{\boldsymbol{C}}\widehat{\boldsymbol{u}}^{(i-1)}\|_2^2 \approx \|\widehat{\boldsymbol{\ell}} - \widehat{\boldsymbol{C}}\overline{\boldsymbol{u}}^{(i-1)} - \widehat{\boldsymbol{A}}\widehat{\boldsymbol{u}}^{(i-1)}\|_2^2. \tag{3.6}$$

Note that the first term in (3.5) resembles the squared absolute residual

$$\|\overline{\boldsymbol{\ell}} - \overline{\boldsymbol{A}}\overline{\boldsymbol{u}}^{(i)} - \overline{\boldsymbol{C}}\widehat{\boldsymbol{u}}^{(i-1)}\|_2^2$$

of the coarse-scale solve in step 2.2 of algorithm 2, except that the coarse-scale approximation has a different index. Assuming that the change from $\overline{\boldsymbol{u}}^{(i-1)}$ to $\overline{\boldsymbol{u}}^{(i)}$ is small, one can make the approximation

$$\|\overline{\boldsymbol{\ell}} - \overline{\boldsymbol{A}}\overline{\boldsymbol{u}}^{(i)} - \overline{\boldsymbol{C}}\widehat{\boldsymbol{u}}^{(i-1)}\|_2^2 \approx \|\overline{\boldsymbol{\ell}} - \overline{\boldsymbol{A}}\overline{\boldsymbol{u}}^{(i-1)} - \overline{\boldsymbol{C}}\widehat{\boldsymbol{u}}^{(i-1)}\|_2^2. \tag{3.7}$$

This approximation can be combined with (3.6) to obtain the requirement that

$$\|\overline{\boldsymbol{\ell}} - \overline{\boldsymbol{A}}\overline{\boldsymbol{u}}^{(i)} - \overline{\boldsymbol{C}}\widehat{\boldsymbol{u}}^{(i-1)}\|_2^2 \approx \|\widehat{\boldsymbol{\ell}} - \widehat{\boldsymbol{C}}\overline{\boldsymbol{u}}^{(i-1)} - \widehat{\boldsymbol{A}}\widehat{\boldsymbol{u}}^{(i-1)}\|_2^2.$$

Hence, using an *absolute* tolerance of

$$\delta_{\text{abs}} := \|\widehat{\boldsymbol{\ell}} - \widehat{\boldsymbol{C}}\overline{\boldsymbol{u}}^{(i-1)} - \widehat{\boldsymbol{A}}\widehat{\boldsymbol{u}}^{(i-1)}\|_2$$

for the coarse-scale update in the $i$th IHSS iteration should balance the convergence of the coarse-scale and fine-scale components. Equivalently, a *relative* tolerance of

$$\delta := \|\widehat{\boldsymbol{\ell}} - \widehat{\boldsymbol{C}}\overline{\boldsymbol{u}}^{(i-1)} - \widehat{\boldsymbol{A}}\widehat{\boldsymbol{u}}^{(i-1)}\|_2 / \|\overline{\boldsymbol{\ell}} - \overline{\boldsymbol{C}}\widehat{\boldsymbol{u}}^{(i-1)} - \overline{\boldsymbol{A}}\overline{\boldsymbol{u}}^{(i-1)}\|_2$$

can be used as well, where $\delta_{\text{abs}}$ was simply scaled by the initial residual of the coarse-scale system. The argument for this choice of $\delta$ is heuristic, but it is supported by good computational performance in numerical experiments (see chapter 4 and [3]).

Algorithm 3 describes the modified IHSS method with the adaptive choice of tolerances for the coarse-scale systems as discussed above. The Euclidean norm is now used throughout the entire algorithm, as the computation of the tolerance $\delta^{(i)}$ depends on this particular norm. Notice that the iterative solver used in step 2.2 of algorithm 3 must also be configured to compute residuals in this norm.

While an initial tolerance $\delta^{(1)}$ still has to be provided, its choice has a limited impact

on the behavior of the method, as the tolerance is adjusted automatically in subsequent iterations (see subsection 4.3.2).

**Algorithm 3:** Modified inexact hierarchical scale separation (based on [3])

$\boldsymbol{u_h} = \texttt{IHSS\_modified}(\boldsymbol{A_h},\ \boldsymbol{\ell_h},\ \boldsymbol{u_h}^{(0)},\ \eta,\ \delta^{(1)},\ \nu)$

*1* $i := 0$

*2 While* $\|\boldsymbol{\ell_h} - \boldsymbol{A_h}\boldsymbol{u_h}^{(i)}\|_2 / \|\boldsymbol{\ell_h} - \boldsymbol{A_h}\boldsymbol{u_h}^{(0)}\|_2 \geq \eta$*:*

*2.1* $i \leftarrow i + 1$

*2.2* $\overline{\boldsymbol{u}}^{(i)} := \texttt{CSS}\left(\overline{\boldsymbol{A}},\ \overline{\boldsymbol{\ell}} - \overline{\boldsymbol{C}}\widehat{\boldsymbol{u}}^{(i-1)},\ \overline{\boldsymbol{u}}^{(i-1)},\ \delta^{(i)}\right)$

*2.3* $\widehat{\boldsymbol{u}}_0^{(i)} := \widehat{\boldsymbol{u}}^{(i-1)}$

*2.4 For* $k = 1, \ldots, \nu$*:*

*2.4.1* $\widehat{\boldsymbol{u}}_k^{(i)} := \texttt{FSS}\left(\widehat{\boldsymbol{A}}_{\mathbf{diag}},\ \widehat{\boldsymbol{\ell}} - \widehat{\boldsymbol{C}}\overline{\boldsymbol{u}}^{(i)} - \widehat{\boldsymbol{A}}_{\mathbf{off}}\widehat{\boldsymbol{u}}_{k-1}^{(i)}\right)$

*2.5* $\widehat{\boldsymbol{u}}^{(i)} := \widehat{\boldsymbol{u}}_\nu^{(i)}$

*2.6* $\delta^{(i+1)} := \|\widehat{\boldsymbol{\ell}} - \widehat{\boldsymbol{C}}\overline{\boldsymbol{u}}^{(i)} - \widehat{\boldsymbol{A}}\widehat{\boldsymbol{u}}^{(i)}\|_2 / \|\overline{\boldsymbol{\ell}} - \overline{\boldsymbol{C}}\widehat{\boldsymbol{u}}^{(i)} - \overline{\boldsymbol{A}}\overline{\boldsymbol{u}}^{(i)}\|_2$

*3 Accept solution* $\boldsymbol{u_h} := \boldsymbol{u_h}^{(i)}$.

## 3.4 Stabilization of the repeated fine-scale update

The final modification to the IHSS algorithm concerns the fine-scale updates in step 2.4 of algorithm 3. These repeated fine-scale updates resemble a fixed point iteration. However, it is not guaranteed that the corresponding mapping

$$g : \widehat{\boldsymbol{u}} \mapsto \widehat{\boldsymbol{A}}_{\mathbf{diag}}^{-1}\left(\widehat{\boldsymbol{\ell}} - \widehat{\boldsymbol{C}}\overline{\boldsymbol{u}}^{(i)} - \widehat{\boldsymbol{A}}_{\mathbf{off}}\widehat{\boldsymbol{u}}\right)$$

is indeed a contraction or that a fixed point even exists for general $\widehat{\boldsymbol{A}}_{\mathbf{diag}}$ and $\widehat{\boldsymbol{A}}_{\mathbf{off}}$. While it is certainly possible to derive conditions on these matrices or the original system matrix $\boldsymbol{A_h}$ that ensure that the mapping $\boldsymbol{g}$ has the desired properties, I will instead augment the

repeated fine-scale updates with an *Anderson acceleration*. My collaborators and I first proposed this idea in our second publication on IHSS [3].

The Anderson acceleration improves the convergence of a fixed point iteration

$$\boldsymbol{x}_{k+1} := \boldsymbol{g}(\boldsymbol{x}_k)$$

by incorporating not only the evaluation of $\boldsymbol{g}$ at the current iterate $\boldsymbol{x}_k$, but also evaluations at previous iterates. The new iterate is then chosen as an affine combination of $\boldsymbol{g}(\boldsymbol{x}_k)$, $\boldsymbol{g}(\boldsymbol{x}_{k-1})$, $\boldsymbol{g}(\boldsymbol{x}_{k-2})$, etc. The full method is described in algorithm 4. The presentation follows Walker and Ni [15].

**Algorithm 4:** Anderson acceleration with fixed number of iterations (see Walker and Ni [15])

$\boldsymbol{x}_\nu = \texttt{Anderson\_acceleration}(\boldsymbol{g},\ \boldsymbol{x}_0,\ m,\ \nu)$

*1* $\boldsymbol{x}_1 := \boldsymbol{g}(\boldsymbol{x}_0)$

*2 For* $k = 1, \dots, \nu - 1$*:*

*2.1* $m_k := \min\{m, k\}$

*2.2* $\boldsymbol{F}_k := (\boldsymbol{g}(\boldsymbol{x}_{k-m_k}) - \boldsymbol{x}_{k-m_k}, \dots, \boldsymbol{g}(\boldsymbol{x}_k) - \boldsymbol{x}_k)$

*2.3 Find* $\boldsymbol{\alpha}$ *that minimizes* $\|\boldsymbol{F}_k \boldsymbol{\alpha}\|$ *s.t.* $\sum_{j=0}^{m_k} \alpha_j = 1$

*2.4* $\boldsymbol{x}_{k+1} := \sum_{j=0}^{m_k} \alpha_j \boldsymbol{g}(\boldsymbol{x}_{k-m_k+j})$

The Anderson acceleration can now be included into the IHSS method. The result is algorithm 5, to which I will simply refer as the IHSS method from now on.

**Algorithm 5:** Inexact hierarchical scale separation (as first published by Thiele et al. in 2017 [3])

$\boldsymbol{u_h} = \texttt{IHSS}(\boldsymbol{A_h},\ \boldsymbol{\ell_h},\ \boldsymbol{u_h}^{(0)},\ \eta,\ m,\ \delta^{(1)},\ \nu)$

*1* $i := 0$

*2* *While* $\|\boldsymbol{\ell_h} - \boldsymbol{A_h}\boldsymbol{u_h}^{(i)}\|_2 / \|\boldsymbol{\ell_h} - \boldsymbol{A_h}\boldsymbol{u_h}^{(0)}\|_2 \geq \eta$*:*

*2.1* $i \leftarrow i + 1$

*2.2* $\overline{\boldsymbol{u}}^{(i)} := \texttt{CSS}\left(\overline{\boldsymbol{A}},\ \overline{\boldsymbol{\ell}} - \overline{\boldsymbol{C}}\widehat{\boldsymbol{u}}^{(i-1)},\ \overline{\boldsymbol{u}}^{(i-1)},\ \delta^{(i)}\right)$

*2.3* *Define* $\boldsymbol{g} : \widehat{\boldsymbol{u}} \mapsto \texttt{FSS}\left(\widehat{\boldsymbol{A}}_{\mathbf{diag}},\ \widehat{\boldsymbol{\ell}} - \widehat{\boldsymbol{C}}\overline{\boldsymbol{u}}^{(i)} - \widehat{\boldsymbol{A}}_{\mathbf{off}}\widehat{\boldsymbol{u}}\right)$

*2.4* $\widehat{\boldsymbol{u}}^{(i)} := \texttt{Anderson\_acceleration}(\boldsymbol{g},\ \widehat{\boldsymbol{u}}^{(i-1)},\ m,\ \nu)$

*2.5* $\delta^{(i+1)} := \|\widehat{\boldsymbol{\ell}} - \widehat{\boldsymbol{C}}\overline{\boldsymbol{u}}^{(i)} - \widehat{\boldsymbol{A}}\widehat{\boldsymbol{u}}^{(i)}\|_2 / \|\overline{\boldsymbol{\ell}} - \overline{\boldsymbol{C}}\widehat{\boldsymbol{u}}^{(i)} - \overline{\boldsymbol{A}}\overline{\boldsymbol{u}}^{(i)}\|_2$

*3* *Accept solution* $\boldsymbol{u_h} := \boldsymbol{u_h}^{(i)}$.

## 3.5 Implementation details

With the IHSS algorithm in place, it is time to discuss some aspects of its implementation before proceeding with numerical examples and an evaluation of the computational performance in the next chapter. The remarks made in this section are general, but I will also mention the specific programming language, tools, and design choices for the IHSS implementation that I used for my experiments.

The key ingredients for the IHSS method are the linear solvers for the coarse-scale and fine-scale problems. The coarse-scale solver (function CSS in algorithm 5) can be any iterative linear solver that is suitable for the matrix $\overline{\boldsymbol{A}}$. Note that the previous coarse-scale approximation $\overline{\boldsymbol{u}}^{(i-1)}$ is used as an initial guess in order to accelerate the solution of the coarse-scale system. If a preconditioner is used, it can likely be reused for all IHSS iterations, as the matrix of the coarse-scale system remains unchanged. Finally, it is worth reiterating that the IHSS method as shown in algorithm 5 requires the coarse-scale solver

to terminate based on the *relative* residual norm measured in the *Euclidean norm*.

The matrix $\widehat{\boldsymbol{A}}_{\mathbf{diag}}$ of the fine-scale system also remains the same throughout all IHSS iterations. This fact can be used to speed up the fine-scale solves (function FSS in algorithm 5). Since the fine-scale system is block-diagonal, an LU or QR decomposition can be computed for each individual block. The fine-scale updates then reduce to a backward and forward substitution for each block in case of an LU decomposition, or a matrix-vector product and a backward substitution for each block in case of a QR decomposition. Furthermore, solving an individual block is computationally inexpensive, as the blocks are typically very small. For example, piecewise linear discretizations result in blocks with just four rows and columns (see subsection 4.2.1). The block structure of $\widehat{\boldsymbol{A}}_{\mathbf{diag}}$ and the low computational cost for solving each individual block make it very easy to parallelize the fine-scale solve with good load balancing.

At this point, it should be emphasized that solving the coarse-scale and fine-scale systems are not the only steps in the IHSS algorithm that incur significant computational cost. For instance, the cost of the matrix-vector product $\widehat{\boldsymbol{A}}_{\mathbf{off}}\widehat{\boldsymbol{u}}$ in step 2.3 of algorithm 5 is not negligible, because even though the matrix $\widehat{\boldsymbol{A}}_{\mathbf{off}}$ is sparse, it is still large. The computation of residuals in steps 2 and 2.5 is also computationally expensive, again due to the size of $\widehat{\boldsymbol{A}}$. The high cost of residual evaluations to some extent precludes the use of residual-based stopping criteria such as relative or absolute tolerances for the Anderson-accelerated fine-scale updates. Computing a residual in each iteration of the Anderson acceleration would increase its computational cost significantly. Thus, a fixed number $\nu$ of iterations is used instead.

The Anderson acceleration itself can be implemented efficiently by reusing the columns of the matrix $\boldsymbol{F}_k$ in step 2.2 of algorithm 4. Only the most recent error $\boldsymbol{g}(\boldsymbol{x}_k) - \boldsymbol{x}_k$ needs to be computed in each iteration, while the error with the lowest index can be dropped from the matrix $\boldsymbol{F}_k$. The minimization problem in step 2.3 can be solved using a QR decompo-

sition of $\boldsymbol{F}_k$ [15].

The IHSS implementation used for the numerical experiments in chapter 4 is written in C++, and it is based on the Trilinos framework [16]. It uses Trilinos' Tpetra package for hybrid parallel linear algebra computations. More specifically, it combines distributed memory parallelism using MPI and shared memory parallelism using OpenMP. Krylov solvers from Trilinos' Belos package are used to solve the coarse-scale systems. The fine-scale systems are solved using precomputed QR decompositions for each block of $\widehat{\boldsymbol{A}}_{\mathbf{diag}}$. The blocks are stored in a single contiguous array to improve cache efficiency. Both the computation of the QR decompositions and their application during the fine-scale update again use MPI and OpenMP for parallelism by simply assigning a subset of the diagonal blocks of $\widehat{\boldsymbol{A}}_{\mathbf{diag}}$ to each MPI rank and OpenMP thread.

Based on this implementation, I evaluate the computational performance of the IHSS method in the next chapter.

Chapter 4

# Numerical experiments

In this chapter I evaluate the computational performance of the IHSS method. I begin with a description of the test problem, a two-component fluid simulation based on the Cahn–Hilliard equation. Using this test problem, I then report results from a number of experiments to determine the effect of the parameters $\nu$,$\delta$, and $m$ (see algorithm 5) on the computational performance. Finally, I analyze how the performance characteristics change with increasing problem size.

## 4.1 The Cahn–Hilliard problem

The test problem for the performance evaluation of IHSS will be the Cahn–Hilliard equation [17, 18]. This fourth-order, nonlinear partial differential equation models the separation of an immiscible binary fluid mixture into areas that are dominated by one of the two fluids. Figure 4.1 shows an example of this process, which is of interest in the simulation of fluid flow in porous rocks in the oil and gas industry [12].

My presentation of the Cahn–Hilliard model follows that in a recent publication by Frank, Liu, Alpak, and Riviere [19]. Indeed, the discretization and implementation used for the numerical experiments in this thesis are the same as those discussed in their paper.

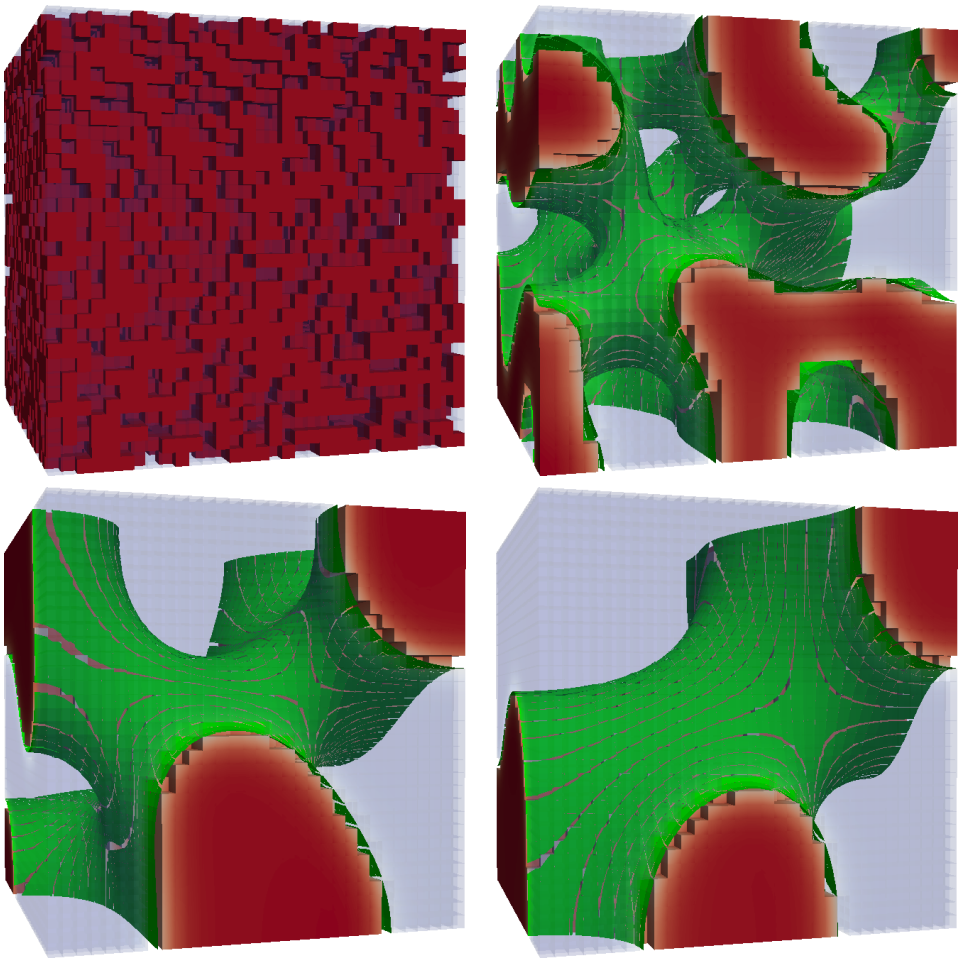

Figure 4.1: The spinodal decomposition scenario. Pseudo-random initial data (after 0 steps, upper left), early interface structure (after 10 steps, upper right), and progressing phase separation (after 100 and 300 steps respectively, bottom row). The two fluid components are shown in red ($c = 1$, dark in grayscale) and blue ($c = -1$, light in grayscale). The center of the interface between the components ($c = 0$) is shown in green.

The Cahn–Hilliard equation reads

$$\partial_t c - \Delta \mu = 0 \qquad \text{in } (0,T)\times\Omega, \tag{4.1}$$

$$\mu = \Phi'(c) - \kappa \Delta c \qquad \text{in } (0,T)\times\Omega, \tag{4.2}$$

$$\nabla c \cdot \eta = 0 \qquad \text{on } (0,T)\times\partial\Omega, \tag{4.3}$$

$$\nabla \mu \cdot \eta = 0 \qquad \text{on } (0,T)\times\partial\Omega, \tag{4.4}$$

$$c = c^0 \qquad \text{on } \{0\}\times\Omega, \tag{4.5}$$

where $\Omega \subset \mathbb{R}^3$ is the spatial domain, and $(0,T)$ is some time interval. The vector $\eta$ denotes the outward-pointing normal vector on the boundary of $\Omega$. The two unknowns of the equation are the *order parameter* $c$ and the *chemical potential* $\mu$. The order parameter $c(t,\boldsymbol{x})$ is closely related to the volume fraction of the two fluids that are present at location $\boldsymbol{x}$ and at time $t$, and it assumes values in the interval $[-1,1]$. A value of $c(t,\boldsymbol{x}) = \pm 1$ indicates that only one of the fluids is present, whereas a value of $c(t,\boldsymbol{x}) = 0$ means that the relative amount of both fluids is equal [19]. The initial fluid distribution is prescribed by $c^0$. The chemical potential $\mu$ is an auxiliary unknown that is introduced by writing the Cahn–Hilliard equation as a system of two second-order equations.

If the order parameter $c$ satisfies equations (4.1) to (4.5), then the *Helmholtz free energy*

$$F(c) := \int_\Omega \left( \Phi(c) + \frac{\kappa}{2} \|\nabla c\|_2^2 \right)$$

decreases over time [19, section 2.1]. The role of the two terms in the integrand of this energy functional is intuitive: The nonlinear function $\Phi(c) = \frac{1}{4}(c^2-1)^2$ assumes its minima at $c = \pm 1$ (see figure 4.2), so its contribution to the energy $F(c)$ is minimized when the order parameter is close to $\pm 1$ in large parts of $\Omega$, i.e., when the two fluids in the domain are mostly separated.

The gradient term $\kappa\|\nabla c\|_2^2$ is related to the tension of the interface between the fluids.

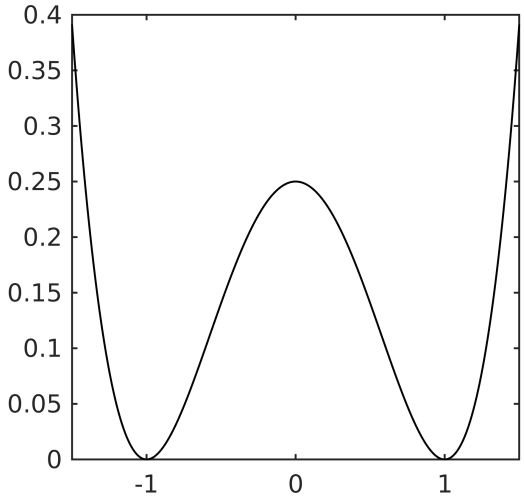


Figure 4.2: The potential function $\Phi(c) = \frac{1}{4}(c^2 - 1)^2$. It assumes its minima at $c = \pm 1$, thereby penalizing order parameters other than $c = \pm 1$ in the free energy functional $F(c)$.

The constant $\kappa > 0$ determines the thickness of this diffuse interface [19]. Note that $\|\nabla c\|_2^2$ denotes the square of the *Euclidean* norm of $\nabla c$ at a point, i.e.,

$$\|\nabla c(t, \boldsymbol{x})\|_2^2 = \nabla c(t, \boldsymbol{x})^T \nabla c(t, \boldsymbol{x}).$$

### 4.1.1 Time discretization

In order to discretize equations (4.1) to (4.5) in time, the nonlinear function $\Phi$ is split into a convex part $\Phi_+$ and a concave part $\Phi_-$ such that

$$\Phi = \Phi_+ + \Phi_-.$$

In the $n$th time step, the task is then to find $c^n$ and $\mu^n$ such that

$$c^n - c^{n-1} - \tau \Delta \mu^n = 0, \tag{4.6}$$

$$\mu^n = \Phi_+'(c^n) + \Phi_-'(c^{n-1}) - \kappa \Delta c^n, \tag{4.7}$$

where $\tau > 0$ is the time step size. The convex part $\Phi_+$ is treated implicitly, while the concave part $\Phi_-$ is treated explicitly. This is done to ensure unique solvability of the equation as well as energy dissipation, i.e., $F(c^n) \leq F(c^{n-1})$ [19, section 3.2].

### 4.1.2 Space discretization

The fully discrete formulation of the numerical scheme in equations (4.6) to (4.7) can now be obtained using the interior penalty discontinuous Galerkin (IPG) approach for the Laplace operators (see section 1.2). Let $a$ denote the resulting DG bilinear form. The problem is then to find $c_h^n$ and $\mu_h^n$ such that

$$(c_h^n, w_h) - (c_h^{n-1}, w_h) + \tau a(\mu_h^n, w_n) = 0, \tag{4.8}$$

$$(\mu_h^n, w_h) = (\Phi_+'(c_h^n), w_h) + (\Phi_-'(c_h^{n-1}), w_h) + \kappa a(c_h^n, w_h), \tag{4.9}$$

$$(c_h^0, w_h) = (c^0, w_h) \tag{4.10}$$

for all $w_h \in \mathbb{P}_p(\mathcal{E}_h)$ [3, section 2.3]. Here, $\mathbb{P}_p(\mathcal{E}_h)$ is the broken polynomial space as defined in section 1.2, and $(\cdot, \cdot)$ denotes the standard inner product on $L^2(\Omega)$. Using a uniform hexahedral mesh $\mathcal{E}_h$ of elements with an edge length of $h$, the bilinear form $a$ reads

$$\begin{aligned} a(c, w) = &\sum_{E \in \mathcal{E}_h} \int_E \nabla c \cdot \nabla w - \sum_{e \in \mathring{\Gamma}_h} \int_e \left\{\!\!\left\{ \nabla c \cdot \boldsymbol{\eta_e} \right\}\!\!\right\} [\![ w ]\!] \\ &+ \varepsilon \sum_{e \in \mathring{\Gamma}_h} \int_e \left\{\!\!\left\{ \nabla w \cdot \boldsymbol{\eta_e} \right\}\!\!\right\} [\![ c ]\!] + \sum_{e \in \mathring{\Gamma}_h} \frac{\sigma}{h} \int_e [\![ c ]\!] [\![ w ]\!] \end{aligned} \tag{4.11}$$

(cf. [3, section 2.3] and [19, section 3.4.1]). This bilinear form strongly resembles the one derived for Poisson's equation in section 1.2, but this time the integrals over the exterior faces $e \in \Gamma_h \backslash \mathring{\Gamma}_h$ vanish due to the homogeneous Neumann boundary conditions (4.3) and (4.4).

For the numerical experiments in subsequent sections, I used the nonsymmetric IPG

discretization, i.e., $\varepsilon = 1$ with a penalty of $\sigma = 1$.

### 4.1.3 Derivation of the linear systems

Let the functions $\{\psi_{k,j}\}$ form a Legendre basis for the space $\mathbb{P}_p(\mathcal{E}_h)$. Using the notation from section 1.2, let us define matrices $\boldsymbol{M}$, $\boldsymbol{A}$, and the vector $\boldsymbol{E}_{\boldsymbol{\Phi}'_\pm}(\boldsymbol{X}_c^{\boldsymbol{n}})$ by

$$(\boldsymbol{M})_{kM+j,k'M+j'} := (\psi_{k',j'}, \psi_{k,j}), \tag{4.12}$$

$$(\boldsymbol{A})_{kM+j,k'M+j'} := a(\psi_{k',j'}, \psi_{k,j}), \tag{4.13}$$

$$(\boldsymbol{E}_{\boldsymbol{\Phi}'_\pm}(\boldsymbol{X}_c^{\boldsymbol{n}}))_{kM+j} := (\Phi'_\pm(c_h^n), \psi_{k,j}), \tag{4.14}$$

where the vector $\boldsymbol{X}_c^{\boldsymbol{n}}$ contains the coefficients of $c_h^n$ with respect to the basis functions $\{\psi_{k,j}\}$. Note that the Legendre basis functions can be scaled to make them *orthonormal* on the reference element. Then the matrix $\boldsymbol{M}$ is a scalar multiple of the identity matrix, i.e.,

$$\boldsymbol{M} = \gamma \boldsymbol{I}$$

for some constant $\gamma > 0$.

The fully discrete scheme (4.8) to (4.9) can now be written in matrix form as

$$\begin{pmatrix} \kappa\boldsymbol{A} & -\boldsymbol{M} \\ \boldsymbol{M} & \tau\boldsymbol{A} \end{pmatrix} \begin{pmatrix} \boldsymbol{X}_c^{\boldsymbol{n}} \\ \boldsymbol{X}_\mu^{\boldsymbol{n}} \end{pmatrix} + \begin{pmatrix} \boldsymbol{E}_{\boldsymbol{\Phi}'_+}(\boldsymbol{X}_c^{\boldsymbol{n}}) \\ \boldsymbol{0} \end{pmatrix} = \begin{pmatrix} -\boldsymbol{E}_{\boldsymbol{\Phi}'_-}(\boldsymbol{X}_c^{\boldsymbol{n-1}}) \\ \boldsymbol{M}\boldsymbol{X}_c^{\boldsymbol{n-1}} \end{pmatrix}.$$

This nonlinear system can be further simplified using a Schur complement reduction (cf. [3, section 2.3] and [19, section 3.4.1]). The upper block of the equation yields

$$\begin{aligned} &\kappa\boldsymbol{A}\boldsymbol{X}_c^{\boldsymbol{n}} - \boldsymbol{M}\boldsymbol{X}_\mu^{\boldsymbol{n}} + \boldsymbol{E}_{\boldsymbol{\Phi}'_+}(\boldsymbol{X}_c^{\boldsymbol{n}}) = -\boldsymbol{E}_{\boldsymbol{\Phi}'_-}(\boldsymbol{X}_c^{\boldsymbol{n-1}}) \\ \Leftrightarrow &\boldsymbol{X}_\mu^{\boldsymbol{n}} = \boldsymbol{M}^{-1}\left(\kappa\boldsymbol{A}\boldsymbol{X}_c^{\boldsymbol{n}} + \boldsymbol{E}_{\boldsymbol{\Phi}'_+}(\boldsymbol{X}_c^{\boldsymbol{n}}) + \boldsymbol{E}_{\boldsymbol{\Phi}'_-}(\boldsymbol{X}_c^{\boldsymbol{n-1}})\right), \end{aligned}$$

which is inserted into the lower block to obtain

$$f(X_c^n) := \underbrace{\left(\frac{\gamma^2}{\tau}I + \kappa A^2\right)X_c^n}_{\text{linear}} + \underbrace{AE_{\Phi'_+}(X_c^n)}_{\text{nonlinear}} + \underbrace{AE_{\Phi'_-}(X_c^{n-1}) - \frac{\gamma^2}{\tau}X_c^{n-1}}_{\text{constant}} = 0. \tag{4.15}$$

Finally, this root-finding problem for $X_c^n$ is linearized using an inexact Newton method, which is described in algorithm 6. Like the classical Newton method, the inexact Newton method finds a root of $f$ using the function itself, its Jacobian $f^{-1}$, and an initial guess $z^{(0)}$. However, the linear system in each Newton step is not solved to high accuracy, but it is approximated up to a certain tolerance in order to save computational cost (see steps 2.2 and 2.3 of algorithm 6). This idea is similar to that of IHSS, and indeed it influenced the design of the IHSS method. For more detail on how the tolerance $\eta^{(i)}$ and the step length $\alpha^{(i)}$ are chosen in each Newton step, I refer the reader to the book by Kelley [20, chapters 6 and 8]. Kelley also suggests the stopping criterion in step 2 of algorithm 6 [20, section 5.2].

**Algorithm 6:** Inexact Newton method (see [20])

$z$ = `inexact_newton`($f$, $f'$, $z^{(0)}$, $\varepsilon_{\text{abs}}$, $\varepsilon_{\text{rel}}$)

*1* $i := 0$

*2 While* $\|f(z^{(i)})\| \geq \varepsilon_{\text{abs}} + \varepsilon_{\text{rel}}\|f(z^{(0)})\|$*:*

*2.1* $i \leftarrow i + 1$

*2.2 Determine a relative tolerance* $\eta^{(i)}$*.*

*2.3 Approximate the solution of* $f'(z^{(i-1)})\Delta z^{(i)} = f(z^{(i-1)})$ *to a tolerance of* $\eta^{(i)}$*.*

*2.4 Determine a step length* $\alpha^{(i)}$*.*

*2.5* $z^{(i)} := z^{(i-1)} - \alpha^{(i)}\Delta z^{(i)}$

*3 Accept solution* $z := z^{(i)}$*.*

In order to apply the inexact Newton method to the root-finding problem (4.15), the Jacobian $f'$ must be available numerically. The implementation used for the experiments

in this thesis computes $f'(X_c^n)$ using the matrix in the linear part of (4.15) and an analytic formula for the Jacobian of the nonlinear part (see [19, section 3.4.5]). Hence, the implementation does not use automatic differentiation.

## 4.2 Experimental setup

This section describes the experimental setup for the analysis of the convergence behavior and computational performance of the IHSS method in subsequent sections. After the introduction of the two test scenarios in subsection 4.2.1, I discuss the hardware and software platform in subsection 4.2.2

### 4.2.1 Test scenarios

Two test scenarios are used for the performance evaluation and analysis of the convergence of the IHSS method. In both scenarios, 10 time steps are computed for the Cahn–Hilliard equation on the domain $\Omega = [0, 1]^3$. This number of time steps results in a sufficient number of linear solves, since multiple Newton steps are required in each time steps. A greater number of linear solves also helps to limit the impact of cases in which the number of Newton steps is different with and without IHSS. This behavior can sometimes be observed because the inexact Newton solver often prescribes rather high relative tolerances on the order of $10^{-3}$ to $10^{-1}$ for the linear solvers. While the solutions obtained with and without IHSS both satisfy these tolerances, they may result in different nonlinear residuals, causing the Newton solver to stop after different numbers of iterations (cf. algorithm 6).

The first test scenario uses the initial data

$$c^0(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \left\| \boldsymbol{x} - \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)^T \right\|_1 < \frac{1}{2} \\ -1, & \text{otherwise} \end{cases},$$

which is shown in figure 4.3. This data contains large areas in which only one of the two fluids is present. It is therefore representative of the later stages of Cahn–Hilliard simulations when the fluids are mostly separated. Due to its structure, I will refer to it as the *droplet* scenario.

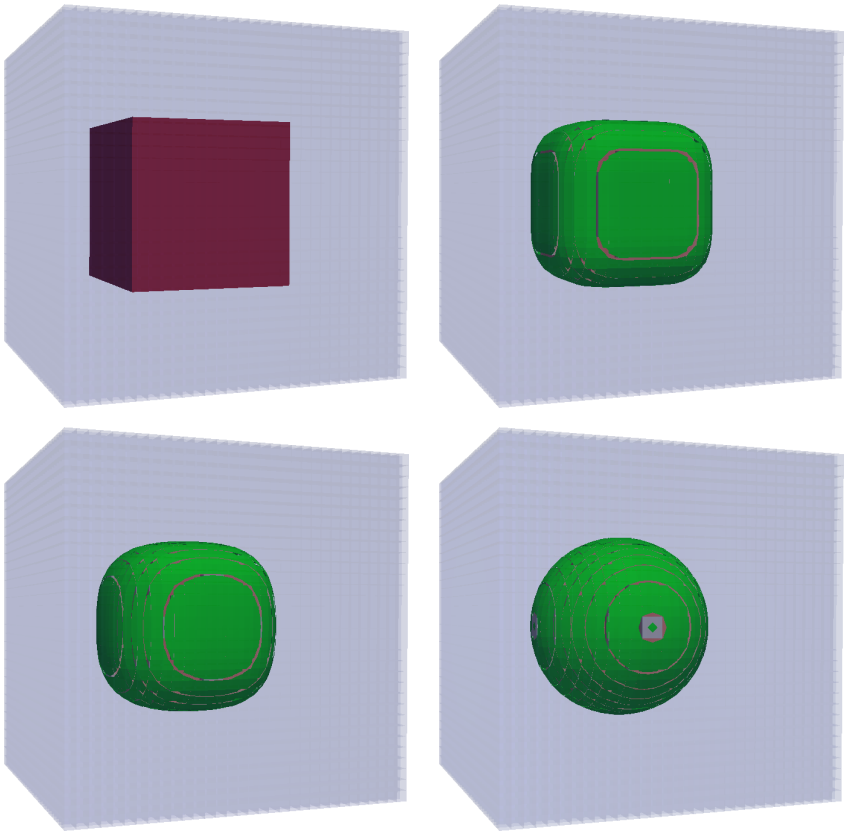

Figure 4.3: The droplet scenario. Over time, an initially cubical droplet assumes a more stable spherical shape (after time steps 0, 1, 3, and 10, from left to right, top to bottom). The two fluid components are shown in red ($c = 1$, dark in grayscale) and blue ($c = -1$, light in grayscale). The center of the interface between the components ($c = 0$) is shown in green.

The second test scenario uses pseudo-random, piecewise constant initial data with values in $\{-1, 1\}$ (see figure 4.1, upper left image). It is more representative of the early stages of Cahn–Hilliard simulations when the system is far from its steady state and stable interfaces between the fluids have not yet developed. In fact, this initial data can be seen as the most complicated of all possible initial fluid distributions. I will refer to this test scenario as the *spinodal decomposition* scenario (see [19, section 4.2]). While it appears entirely random, the same initial data is used for all experiments on the same computational grid in order to ensure comparability of the results.

In subsequent sections, I show numerical results for both the droplet scenario and the spinodal decomposition scenario using piecewise linear DG basis functions and problem sizes $N_1 \in \{32, 64, 128\}$, where $N_1$ is the number of elements in each spatial dimension. Hence, the total number of elements is $N = N_1^3$, and it ranges from $32^3 \approx 3 \cdot 10^4$ to $128^3 \approx 2 \cdot 10^6$. The size of the resulting linear systems is $MN = 4N_1^3$, because $M = \dim \mathbb{P}_1(E_k) = 4$ is the number of basis functions on each element $E_k \in \mathcal{E}_h$. The basis is split into piecewise constant components for the coarse scale ($\overline{M} = 1$) and piecewise linear components for the fine scale ($\widehat{M} = 3$).

Note that smaller time steps must be used as $N_1$ increases. More specifically, a time step size of $\tau = 1/N_1^2$ is used, which is consistent with the choice made by Frank et al. for the same discretization of the Cahn–Hilliard equation [19, section 4.1]. Due to the different time step sizes, the simulation results after 10 time steps are not directly comparable for different problem sizes even if the same initial data is used. However, my experiments will focus on the convergence and performance of IHSS with different parameters for a fixed problem size. Thus, the simulation results are not important so long as they are the same for all parameter choices. To ensure convergence of the IHSS method, the final residuals after each linear solve were computed again using the original, non-rearranged linear system.

### 4.2.2 Hardware and software platform

All numerical experiments were performed on computing nodes with the hardware configuration shown in table 4.1 using one MPI process per CPU socket and 12 threads per process. Although the code supports distributed memory parallelism (see section 3.5), only a single computing node was used for each experiment in order to avoid any influence of the network connection on the results. A future comparison of the scalability of IHSS and other iterative solvers could provide further insight into the performance of IHSS, but here I focus on the convergence behavior and single-node performance of the method.

Table 4.1: Harware used for the numerical experiments

| | |
|---|---|
| Processor(s) | 2× Xeon E5-2680 v3 (Haswell) |
| Cores | 24 (2× 12) |
| Frequency | 2.5 GHz |
| L3 cache | 30 MB |
| Instruction set | AVX2 |
| Memory | 256 GB (8× 32 GB) |

The Trilinos version used for the experiments was obtained from the Git master branch on May 31, 2017 [21]. The Intel C++ compiler in version 16.0.3 was used to compile Trilinos, the Cahn–Hilliard solver, and the IHSS implementation. For parallelism, the Intel OpenMP runtime was used, as well as Intel MPI in version 5.1.3.

## 4.3 Results

In this section I present a number of numerical experiments that will help to understand the convergence behavior and computational performance of the IHSS method. Each subsection starts with a description of the experiment, followed by the results. First, I establish a baseline for the convergence behavior and computational performance of IHSS for

both test scenarios from subsection 4.2.1 and all problem sizes using different numbers of fine-scale updates per IHSS iteration. I then analyze the influence of the initial coarse-scale solver tolerance $\delta^{(1)}$ on convergence and performance, and I compare the adaptive choice of tolerances to using a fixed tolerance. Next, I investigate the impact of the Anderson acceleration for the repeated fine-scale updates. Finally, I repeat some experiments using a different coarse-scale solver to ensure that the observed performance of IHSS does not depend on a particular choice.

Whenever elapsed times are reported and compared, the corresponding experiments were repeated three times, and the results were averaged. The measurements represent the accumulated time for the solution of linear systems in all time and Newton steps. The measurements do not include the overhead for rearranging the linear system into the form (3.1) for IHSS, as this step could easily be avoided by assembling the system in this way in the first place.

### 4.3.1 Effect of the number of fine-scale updates per iteration

For the first experiment, 10 time steps of both test scenarios are computed, using the restarted GMRES method with a restart length of 30 and IHSS to solve the linear systems in all time and Newton steps. The same GMRES implementation from Trilinos' Belos package is also used as the coarse-scale solver within IHSS. No preconditioners are used, since they might have a different effect when GMRES is used on its own and as a coarse-scale solver, thereby complicating the performance analysis. I briefly discuss ideas for preconditioning IHSS in section 5.2. The initial tolerance for the coarse-scale solver is chosen as $\delta^{(1)} = 10^{-1}$, and $m = 2$ is used for the Anderson-accelerated fine-scale updates (see algorithm 4). Values of $\nu \in \{2, 4, 8, 16\}$ are tested for the number of fine-scale updates per IHSS iteration.

The setup for this experiment is similar to the one my collaborators and I conducted

for our previous publication on IHSS [3]. I repeat the experiment here to ensure that all experiments use the same set of parameters and the same version of the Cahn–Hilliard code. It will be used as a baseline and reference for the additional experiments in subsections 4.3.2 to 4.3.5, which will provide further insight into the convergence behavior and computational performance of the IHSS method.

The results of this first experiment are presented in tables 4.2 and 4.3. They show the accumulated number of IHSS iterations and coarse-scale solver iterations from all time and Newton steps, as well as the accumulated elapsed time for the linear solves using either GMRES or IHSS. The time for the assembly of the linear systems is not taken into account, since it does not depend on the selected linear solver. The last column shows the achieved speedups, which are the ratio of the elapsed time using GMRES and the elapsed time using IHSS. These speedups are shown for two reasons: First, they demonstrate that for this application, using the GMRES method as a coarse-scale solver within IHSS is indeed more efficient than using the same GMRES implementation to solve the linear systems directly. Second, the speedups help to compare the effectiveness of IHSS for different choices of $\nu$. However, the obtainable speedups depend on the simulation parameters and time step size, so not too much emphasis should be put on the exact speedups achieved for this test problem. Instead, the important result is that significant speedups are observed for both test scenarios and all problem sizes and values of $\nu$.

As $\nu$ increases, the number of IHSS iterations decreases, resulting in improved performance up to a value of $\nu = 8$. When $\nu$ is increased further, the number of coarse-scale solver iterations rises, and computational performance begins to deteriorate. The choice of $\nu = 8$ resulted in optimal or near-optimal performance in all cases. Hence, the IHSS method is easily calibrated for the test scenarios.

Even though the optimal speedups are similar for all problem sizes, I would like to remind the reader that results for different problem sizes cannot be compared directly,

Table 4.2: Computational performance of IHSS for the droplet scenario (10 time steps $\delta^{(1)} = 10^{-1}$, $m = 2$). CSS stands for coarse-scale solver.

| $N_1$ | $\nu$ | IHSS iter. | CSS iter. | Time GMRES | Time IHSS | Speedup |
|---|---|---|---|---|---|---|
| 32 | 2 | 348 | 985 | 5.0 s | 2.0 s | 2.5 × |
| –»– | 4 | 131 | 983 | –»– | 1.3 s | 3.8 × |
| –»– | 8 | 74 | 976 | –»– | 1.2 s | 4.2 × |
| –»– | 16 | 65 | 1414 | –»– | 1.6 s | 3.1 × |
| 64 | 2 | 266 | 752 | 21.9 s | 9.3 s | 2.4 × |
| –»– | 4 | 104 | 793 | –»– | 5.9 s | 3.7 × |
| –»– | 8 | 60 | 792 | –»– | 5.6 s | 3.9 × |
| –»– | 16 | 50 | 1066 | –»– | 8.3 s | 2.6 × |
| 128 | 2 | 165 | 486 | 132.9 s | 44.9 s | 3.0 × |
| –»– | 4 | 69 | 525 | –»– | 32.1 s | 4.1 × |
| –»– | 8 | 43 | 557 | –»– | 33.9 s | 3.9 × |
| –»– | 16 | 40 | 837 | –»– | 56.2 s | 2.4 × |

Table 4.3: Computational performance of IHSS for the spinodal decomposition scenario (10 time steps, $\delta^{(1)} = 10^{-1}$, $m = 2$). CSS stands for coarse-scale solver.

| $N_1$ | $\nu$ | IHSS iter. | CSS iter. | Time GMRES | Time IHSS | Speedup |
|---|---|---|---|---|---|---|
| 32 | 2 | 312 | 922 | 5.1 s | 1.9 s | 2.7 × |
| –»– | 4 | 133 | 918 | –»– | 1.3 s | 3.9 × |
| –»– | 8 | 75 | 954 | –»– | 1.2 s | 4.3 × |
| –»– | 16 | 55 | 1071 | –»– | 1.4 s | 3.6 × |
| 64 | 2 | 272 | 794 | 24.3 s | 10.0 s | 2.4 × |
| –»– | 4 | 112 | 776 | –»– | 6.6 s | 3.7 × |
| –»– | 8 | 63 | 789 | –»– | 6.1 s | 4.0 × |
| –»– | 16 | 48 | 878 | –»– | 8.1 s | 3.0 × |
| 128 | 2 | 206 | 601 | 175.0 s | 58.9 s | 3.0 × |
| –»– | 4 | 90 | 618 | –»– | 42.6 s | 4.1 × |
| –»– | 8 | 49 | 597 | –»– | 40.3 s | 4.3 × |
| –»– | 16 | 39 | 730 | –»– | 57.2 s | 3.1 × |

because a different time step size was used for each of them. This also explains why both the number of IHSS iterations and the number of coarse-scale solver iterations decrease with increasing problem size, which would be a very surprising observation if the same parameters were used for all problem sizes.

### 4.3.2 Effect of the initial tolerance for the coarse-scale solver

For the intermediate problem size of $N_1 = 64$, different choices for the initial tolerance $\delta^{(1)}$ for the coarse-scale solver are compared using the spinodal decomposition test scenario. The convergence behavior and performance of IHSS during the computation of 10 time steps are shown in table 4.4.

Table 4.4: Influence of the initial tolerance $\delta^{(1)}$ on convergence and computational performance of IHSS (spinodal decomposition scenario, 10 time steps, $N_1 = 64$, $\nu = 8$, $m = 2$). CSS stands for coarse-scale solver.

| $\delta^{(1)}$ | IHSS iter. | CSS iter. | Time GMRES | Time IHSS | Speedup |
|---|---|---|---|---|---|
| $10^{-1}$ | 63 | 789 | 24.3 s | 6.1 s | 4.0 × |
| $10^{-3}$ | 45 | 1072 | –»– | 5.5 s | 4.4 × |
| $10^{-6}$ | 45 | 2057 | –»– | 7.4 s | 3.3 × |

We see that reducing the initial tolerance $\delta^{(1)}$ results in fewer IHSS iterations, while each iteration becomes more expensive due to the increased number of coarse-scale solver iterations. Even though decreasing $\delta^{(1)}$ from $10^{-1}$ to $10^{-3}$ reduces the elapsed time, it does so by less than 10%. When the initial tolerance is further reduced to $10^{-6}$, no additional reduction in the number of IHSS iterations is achieved, but the elapsed time increases, because the coarse-scale systems are solved to an accuracy that is not needed. These results indicate that the choice of $\delta^{(1)} = 10^{-1}$ in previous experiment was reasonable. The results also underline the effectiveness of a central idea of IHSS, namely, to reduce computational cost by relaxing the tolerances for the coarse-scale solver: Reducing the initial tolerance $\delta^{(1)}$ from $10^{-3}$ to $10^{-6}$ does not change the number of IHSS iterations in this test case, but it results in a significant increase in the number of coarse-scale solver iterations and thus in degraded performance.

### 4.3.3 Non-adaptive choice of coarse-scale solver tolerances

In order to verify that the adaptive choice of tolerances proposed in section 3.3 helps to balance the coarse-scale and fine-scale solvers, the spinodal decomposition experiment is repeated using a fixed relative tolerance $\delta$ in each IHSS iteration. The results of this experiment are presented in table 4.5.

Table 4.5: Computational performance using fixed coarse-scale solver tolerances (spinodal decomposition scenario, 10 time steps, $N_1 = 64$, $\nu = 8$, $m = 2$). CSS stands for coarse-scale solver.

| Type | $\delta^{(1)}$ | IHSS iter. | CSS iter. | Time GMRES | Time IHSS | Speedup |
|---|---|---|---|---|---|---|
| adaptive | 0.9 | 72 | 823 | 24.3 s | 6.9 s | 3.5 × |
| –»– | 0.5 | 72 | 815 | –»– | 6.8 s | 3.6 × |
| –»– | $10^{-1}$ | 63 | 789 | –»– | 6.1 s | 4.0 × |
| –»– | $10^{-3}$ | 45 | 1072 | –»– | 5.5 s | 4.4 × |
| –»– | $10^{-6}$ | 45 | 2057 | –»– | 7.4 s | 3.3 × |
| fixed | 0.9 | 647 | 1513 | –»– | 50.0 s | 0.5 × |
| –»– | 0.5 | 150 | 825 | –»– | 12.4 s | 2.0 × |
| –»– | $10^{-1}$ | 59 | 802 | –»– | 5.8 s | 4.2 × |
| –»– | $10^{-3}$ | 35 | 1260 | –»– | 5.2 s | 4.7 × |
| –»– | $10^{-6}$ | 35 | 2725 | –»– | 7.9 s | 3.1 × |

We see that fixed tolerances of $10^{-1}$, $10^{-3}$, and $10^{-6}$ yield speedups similar to those observed with the adaptive choice of tolerances when starting with the same initial tolerances. However, speedups of roughly 3.5 × are preserved when the adaptive choice of tolerances is used with large initial values $\delta^{(1)}$, whereas performance quickly deteriorates when large fixed tolerances of 0.5 or 0.9 are used. While this indicates that the adaptive choice of tolerances had little effect on computational performance in previous experiments, where $\delta^{(1)} = 10^{-1}$ was used, it does simplify the calibration of IHSS when appropriate tolerances $\delta$ are unknown. This simplification is particularly helpful when IHSS is first applied to a new problem. Indeed, my collaborators and I initially used large fixed tolerances for the coarse-scale solves (see [12]), and hence the introduction of adaptive tolerances greatly simplified our early experiments with IHSS.

### 4.3.4 Effect of the Anderson acceleration on convergence and computational performance

In order to analyze the effect of the Anderson acceleration on the repeated fine-scale updates in each IHSS iteration, different choices of the parameter $m$ are compared using the spinodal decomposition scenario of size $N_1 = 64$. Recall that $m$ is the number of previous function evaluations taken into account in the Anderson acceleration to compute the next iterate

$$\boldsymbol{x}_{k+1} := \sum_{j=0}^{m} \alpha_j \boldsymbol{g}(\boldsymbol{x}_{k-m+j}).$$

Hence, the choice of $m = 0$ effectively disables the Anderson acceleration, falling back to a regular fixed point iteration

$$\boldsymbol{x}_{k+1} := \boldsymbol{g}(\boldsymbol{x}_k).$$

The results for the spinodal decomposition scenario with different choices of $m$ are shown in table 4.6. While the parameter has no significant impact on computational performance, we see that the Anderson acceleration indeed stabilizes the IHSS method. When disabled ($m = 0$), IHSS diverges for all tested values of $\nu$, whereas the method converged in all cases when the Anderson acceleration is enabled ($m \geq 1$). This observation was the main motivation for my collaborators and me to augment the repeated fine-scale updates with an Anderson acceleration (see [3, section 3.3.1]).

Table 4.6: Influence of the Anderson acceleration on convergence and computational performance (spinodal decomposition scenario, $N_1 = 64$, 10 time steps, $\delta^{(1)} = 10^{-1}$). The choice of $m = 0$ disables the Anderson acceleration. CSS stands for coarse-scale solver.

| $m$ | $v$ | IHSS iter. | CSS iter. | Time GMRES | Time IHSS | Speedup |
|---|---|---|---|---|---|---|
| 0 | 2 | div.[1] | n/a | 24.3 s | n/a | n/a |
| –»– | 4 | div.[1] | n/a | –»– | n/a | n/a |
| –»– | 8 | div.[1] | n/a | –»– | n/a | n/a |
| –»– | 16 | div.[1] | n/a | –»– | n/a | n/a |
| 1 | 2 | 272 | 794 | –»– | 9.8 s | 2.5 × |
| –»– | 4 | 123 | 765 | –»– | 6.8 s | 3.6 × |
| –»– | 8 | 67 | 739 | –»– | 6.1 s | 4.0 × |
| –»– | 16 | 49 | 781 | –»– | 7.6 s | 3.2 × |
| 2 | 2 | 272 | 794 | –»– | 10.0 s | 2.4 × |
| –»– | 4 | 112 | 776 | –»– | 6.6 s | 3.7 × |
| –»– | 8 | 63 | 789 | –»– | 6.1 s | 4.0 × |
| –»– | 16 | 48 | 878 | –»– | 8.1 s | 3.0 × |
| 3 | 2 | 272 | 794 | –»– | 10.3 s | 2.4 × |
| –»– | 4 | 107 | 813 | –»– | 6.4 s | 3.8 × |
| –»– | 8 | 61 | 811 | –»– | 6.2 s | 3.9 × |
| –»– | 16 | 48 | 1001 | –»– | 8.7 s | 2.8 × |

[1] IHSS diverged

### 4.3.5 Using a different coarse-scale solver

For the final experiment, GMRES(30) is replaced with BiCGStab as the coarse-scale solver, and the experiment from subsection 4.3.1 is repeated for the spinodal decomposition scenario of size $N_1 = 64$. Again, the BiCGStab implementation is taken from Trilinos'

Belos package. This experiment is intended to verify that the performance of the IHSS method and the adaptive choice of tolerances for the coarse-scale solver do not depend on specific features of the GMRES(30) algorithm. Note that on average, the coarse-scale solver performed fewer than 30 iterations in each IHSS iteration in previous experiments (cf., e.g., the third and fourth column of table 4.3). Hence, the GMRES algorithm was rarely restarted when used as a coarse-scale solver within IHSS, whereas the restart did come into play when the linear systems were solved directly with GMRES(30). Since the BiCGStab method does not require the choice of a restart length or other parameters, it can be used to eliminate these concerns.

The results of this experiment are shown in table 4.7. We see that the speedups obtained with BiCGStab as the coarse-scale solver are similar to those previously observed (cf. table 4.3). In particular, the optimal choice of $\nu$ does not change. These results confirm that the observed speedups are indeed due to the IHSS algorithm and that they do not depend on specific properties of the restarted GMRES method.

Table 4.7: Computational performance using BiCGStab as coarse-scale solver (spinodal decomposition scenario, $N_1 = 64$, 10 time steps, $\delta^{(1)} = 10^{-1}$, $m = 2$). CSS stands for coarse-scale solver.

| $\nu$ | IHSS iter. | CSS iter. | Time BiCGStab | Time IHSS | Speedup |
|---|---|---|---|---|---|
| 2 | 286 | 465 | 23.6 s | 9.9 s | 2.4 × |
| 4 | 111 | 482 | –»– | 6.3 s | 3.7 × |
| 8 | 62 | 512 | –»– | 5.9 s | 4.0 × |
| 16 | 49 | 643 | –»– | 8.1 s | 2.9 × |

The experimental results reported and analyzed in this chapter show that the linear solves in the test problems are accelerated significantly when common iterative solvers are used as coarse-scale solvers within IHSS rather than being applied directly. The IHSS method is easily calibrated for the test problem. The relaxation of the coarse-scale tolerances and the repeated fine-scale updates both reduce computational cost, which underlines the effectiveness of the two central features of the IHSS method.

Chapter 5

# Conclusions

The final two sections of this thesis summarize the main features of the IHSS method and important results from the numerical experiments detailed in the previous chapter. Section 5.2 discusses possible future work and suggests further improvements to IHSS.

## 5.1 Summary of the results

The IHSS method extends the original hierarchical scale separation method for the solution of linear systems in modal discontinuous Galerkin discretizations. The key modifications to the algorithm are the use of an iterative solver for the approximation of the coarse-scale problems and the repetition of the parallel fine-scale update in each iteration. The repeated fine-scale updates are augmented with an Anderson acceleration, and an adaptive choice of the coarse-scale solver tolerances is proposed to balance the coarse-scale and fine-scale updates for improved convergence and performance.

Guidelines for the parallel implementation of the method were provided, and a series of numerical experiments was conducted using a three-dimensional, fourth-order, nonlinear model problem with applications in the oil and gas industry. For these test problems, speedups of up to 4.7 $\times$ are observed when a GMRES or BiCGStab solver is used for the

coarse-scale updates within IHSS instead of being applied to the linear systems directly. The calibration of IHSS for the test problems is easy, with an initial coarse-scale tolerance of $\delta^{(1)} = 10^{-1}$ and $\nu = 8$ fine-scale updates per iteration resulting in speedups of $3.9\times$ and more for all tested problem sizes. While augmenting the repeated fine-scale updates with an Anderson acceleration has little impact on computational performance, it stabilizes the method and ensures convergence in all test cases. The adaptive choice of tolerances for the coarse-scale solver simplifies the calibration of the IHSS algorithm, preserving computational performance even for initial tolerances close to one. Furthermore, the experiments show that the relaxation of the coarse-scale tolerances can save computational cost and that the repeated fine-scale updates improve convergence of the method for the test problems, thereby confirming the effectiveness of the two central features of the IHSS method.

## 5.2 Future work

Future work could include an investigation of the applicability of IHSS to other problems in fluid dynamics such as the Navier–Stokes equations, the evaluation of preconditioned coarse-scale solvers, an analysis of the scalability of IHSS, and the use of GPUs for its implementation.

While preconditioners could be applied to the coarse-scale solver in the IHSS algorithm presented in this thesis, doing so may change the balance of the coarse-scale and fine-scale updates. Thus, the calibration of the method and the adaptive choice of coarse-scale solver tolerances would have to be further analyzed in order to use preconditioners efficiently. Moreover, the use of IHSS itself as a preconditioner for other iterative methods could be investigated.

The scalability of parallel IHSS implementations is another important aspect that could be analyzed. Existing iterative solvers may scale better when applied to the smaller coarse-

scale systems within IHSS, and the decoupled fine-scale updates should further improve parallel efficiency.

Finally, GPUs could be used to solve the decoupled fine-scale systems, possibly even in conjunction with CPUs. Due to their limited memory, GPUs might also be more suitable to solve the smaller coarse-scale systems in IHSS than to solve the full linear system.

# Bibliography


[1] D. A. Di Pietro and A. Ern, *Mathematical aspects of discontinuous Galerkin methods*. Springer Science & Business Media, 2011.

[2] B. Rivière, *Discontinuous Galerkin Methods for Solving Elliptic and Parabolic Equations: Theory and Implementation*. Society for Industrial and Applied Mathematics, 2008.

[3] C. Thiele, M. Araya-Polo, F. O. Alpak, B. Riviere, and F. Frank, "Inexact hierarchical scale separation: A two-scale approach for linear systems from discontinuous Galerkin discretizations," *Computers and Mathematics with Applications*, vol. 74, no. 8, pp. 1769–1778, 2017.

[4] D. Kuzmin, "Hierarchical slope limiting in explicit and implicit discontinuous Galerkin methods," *Journal of Computational Physics*, vol. 257, Part B, pp. 1140–1162, 2014.

[5] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. Society for Industrial and Applied Mathematics, 2003.

[6] A. A. Kosinski, "Cramer's Rule Is Due to Cramer," *Mathematics Magazine*, vol. 74, no. 4, pp. 310–312, 2001.

[7] P. Wesseling, *An introduction to multigrid methods*, ser. Pure and applied mathematics. John Wiley & Sons Australia, Limited, 1992.

[8] K. J. Fidkowski, T. A. Oliver, J. Lu, and D. L. Darmofal, "p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations," *Journal of Computational Physics*, vol. 207, no. 1, pp. 92–113, 2005.

[9] E. M. Rønquist and A. T. Patera, "Spectral element multigrid. I. Formulation and numerical results," *Journal of Scientific Computing*, vol. 2, no. 4, pp. 389–406, 1987.

[10] R. E. Bank, T. F. Dupont, and H. Yserentant, "The hierarchical basis multigrid method," *Numerische Mathematik*, vol. 52, no. 4, pp. 427–458, 1988.

[11] V. Aizinger, D. Kuzmin, and L. Korous, "Scale separation in fast hierarchical solvers for discontinuous Galerkin methods," *Applied Mathematics and Computation*, vol. 266, pp. 838–849, 2015.

[12] C. Thiele, M. Araya-polo, F. O. Alpak, B. Riviere, and F. Frank, "Inexact Hierarchical Scale Separation: An Efficient Linear Solver for Discontinuous Galerkin Discretizations," in *SPE Reservoir Simulation Conference*, 2017.

[13] J. Schütz and V. Aizinger, "A hierarchical scale separation approach for the hybridized discontinuous Galerkin method," *Journal of Computational and Applied Mathematics*, vol. 317, pp. 500–509, 2017.

[14] A. Jaust, J. Schütz, and V. Aizinger, "An efficient linear solver for the hybridized discontinuous Galerkin method," *PAMM*, vol. 16, no. 1, pp. 845–846, 2016.

[15] H. F. Walker and P. Ni, "Anderson acceleration for fixed-point iterations," *SIAM Journal on Numerical Analysis*, vol. 49, no. 4, pp. 1715–1735, 2011.

[16] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley, "An overview of the Trilinos project," *ACM Trans. Math. Softw.*, vol. 31, no. 3, pp. 397–423, 2005.

[17] J. W. Cahn and J. E. Hilliard, "Free energy of a nonuniform system. I. Interfacial free energy," *The Journal of Chemical Physics*, vol. 28, pp. 258–267, 1958.

[18] J. W. Cahn, "On spinodal decomposition," *Acta Metallurgica*, vol. 9, pp. 795–801, 1961.

[19] F. Frank, C. Liu, F. O. Alpak, and B. Riviere, "A finite volume / discontinuous Galerkin method for the advective Cahn–Hilliard equation with degenerate mobility on porous domains stemming from micro-CT imaging," *Computational Geosciences*, vol. 22, no. 2, pp. 543–563, 2018.

[20] C. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, ser. Frontiers in Applied Mathematics. Society for Industrial and Applied Mathematics, 1995.

[21] "Git repository for the Trilinos project," https://github.com/trilinos/Trilinos, accessed: 03/19/2018.