Variational Domain Decomposition for Parallel Image Processing

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von

Dipl.-Inf. Timo Kohlberger
aus Heidelberg

Mannheim, 2007

Dekan:	Professor Dr. Matthias Krause, Universität Mannheim
Referent:	Professor Dr. Christoph Schnörr, Universität Mannheim
Korreferent:	Professor Dr. Joachim Weickert, Universität des Saarlandes

Tag der mündlichen Prüfung: 11. Juni 2007

Abstract

Many important techniques in image processing rely on partial differential equation (PDE) problems, which exhibit spatial couplings between the unknowns throughout the whole image plane. Therefore, a straightforward spatial splitting into independent subproblems and subsequent parallel solving aimed at diminishing the total computation time does not lead to the solution of the original problem. Typically, significant errors at the local boundaries between the subproblems occur. For that reason, most of the PDE-based image processing algorithms are not directly amenable to coarse-grained parallel computing, but only to fine-grained parallelism, e.g. on the level of the particular arithmetic operations involved with the specific solving procedure. In contrast, Domain Decomposition (DD) methods provide several different approaches to decompose PDE problems spatially so that the merged local solutions converge to the original, global one. Thus, such methods distinguish between the two main classes of overlapping and non-overlapping methods, referring to the overlap between the adjacent subdomains on which the local problems are defined. Furthermore, the classical DD methods — studied intensively in the past thirty years — are primarily applied to linear PDE problems, whereas some of the current important image processing approaches involve solving of nonlinear problems, e.g. Total Variation (TV)-based approaches.

Among the linear DD methods, non-overlapping methods are favored, since in general they require significanty fewer data exchanges between the particular processing nodes during the parallel computation and therefore reach a higher scalability. For that reason, the theoretical and empirical focus of this work lies primarily on non-overlapping methods, whereas for the overlapping methods we mainly stay with presenting the most important algorithms.

With the *linear* non-overlapping DD methods, we first concentrate on the theoretical foundation, which serves as basis for gradually deriving the different algorithms thereafter. Although we make a connection between the very early methods on two subdomains and the current two-level methods on arbitrary numbers of subdomains, the experimental studies focus on two prototypical methods being applied to the model problem of estimating the optic flow, at which point different numerical aspects, such as the influence of the number of subdomains on the convergence rate, are explored. In particular, we present results of experiments conducted on a PC-cluster (a distributed memory parallel computer based on low-cost PC hardware for up to 144 processing nodes) which show a very good scalability of non-overlapping DD methods.

With respect to *nonlinear* non-overlapping DD methods, we pursue two distinct

approaches, both applied to nonlinear, PDE-based image denoising. The first approach draws upon the theory of optimal control, and has been successfully employed for the domain decomposition of Navier-Stokes equations. The second nonlinear DD approach, on the other hand, relies on convex programming and relies on the decomposition of the corresponding minimization problems.

Besides the main subject of parallelization by DD methods, we also investigate the linear model problem of motion estimation itself, namely by proposing and empirically studying a new variational approach for the estimation of turbulent flows in the area of fluid mechanics.

Zusammenfassung

Viele Bildverarbeitungsverfahren basieren auf linearen und nicht-linearen partiellen Differenzialgleichungen (PDG), welche räumliche Abhängigkeiten zwischen den Unbekannten über den gesamten Bereich der Bildebene aufweisen. Eine direkte räumliche Zerlegung in seperate Teilprobleme und daran anschließende parallele Berechnung führt nicht zur Lösung des ursprünglichen Problems. Typischerweise treten starke Fehler an den lokalen Grenzen zwischen den Teilgebieten auf. Folglich ermögichen die meisten PDG-basierten Bildverarbeitungsalgorithmen keine direkten grobkörnigen Parallelisierungen, sondern nur solche für fein-körnige, z.B. auf der Ebene der einzelnen Rechenoperationen des jeweiligen Lösungsverfahrens. Im Gegensatz dazu stellen Gebietszerlegungsmethoden (GZ) verschiedene Ansätze zur räumlichen Zerlegung von PDG-Problemen zur Verfügung, so daß die vereinigte Lösung der lokalen Teilprobleme zur ursprünglichen Lösung konvergiert. Hierbei wird zwischen den beiden Klassen der überlappenden und nicht-überlappenden Methoden – in Bezug auf die Überdeckung von benachbarten Teilgebieten – unterschieden. Zudem werden klassische Gebietszerlegungsmethoden – selbst Gegenstand intensiver Forschung in den vergangenen dreißig Jahren – primär auf lineare PDG-Probleme angewandt, wohingegen viele der heutigen Bildverarbeitungsverfahren, wie z.B. solche basierend auf der total Ableitung (TA), nicht-lineare Probleme mit sich bringen.

Unter den linearen GZ-Methoden sind die Nicht-Überlappenden generell von Vorteil, da sie im allgemeinen einen wesentlich geringen Datenaustausch zwischen den einzelnen Rechenknoten während der parallelen Berechnung erfordern und hierdurch eine höhere Skalierbarkeit erreichen. Daher liegt der theoretische und empirische Schwerpunkt dieser Arbeit hauptsächlich auf nicht-überlappenden Methoden, wohingegen wir in bezug auf die überlappenden Methoden im Großen und Ganzen bei der Erklärung der verschiedenen Berechnungsverfahren verbleiben.

Bei den linearen nicht-überlappenden GZ-Methoden konzentrieren wir uns zu Beginn auf die theoretischen Grundlagen, welche hernach als Basis für die Herleitung der verschiedenen Algorithmen dient. Obwohl wir einen großen Bogen von den relativen simplen Methoden auf zwei Teilgebieten bis zu den heutigen Zwei-Gitter-Methoden auf einer beliebigen Anzahl von Teilgebieten spannen, beschränken sich die experimentellen Studien auf zwei prototypische Verfahren, anhand deren, in Anwendung auf das lineare Modellproblem der Bewegungschätzung, verschiedene numerische Aspekte, wie z.B. der Einfluß der Anzahl der Teilgebiete auf die Konvergenzgeschwindigkeit, untersucht werden. Im besonderen präsentieren wir experimentelle Ergebnisse, die auf einem PC-Cluster (einem auf kostengünstiger PC-Hardware basierenden Parallelcomputer mit verteiltem Speicher) mit bis zu 144

Prozeßknoten durchgeführt erzielt, die ihrerseits die sehr guten Skalierungseigenschaften von nicht-überlappenden GZ-Methoden aufzeigen.

Im Hinblick auf nicht-lineare nicht-überlappende GZ-Methoden verfolgen wir zwei unterschiedliche Ansätze, beide jedoch in Anwendung auf nicht-lineare, PDG-basierte Bildentrauschung. Die erste Methode basiert auf einem kontrolltheoretischen Ansatz, welcher bereits erfolgreich zur Parallelisierung von Navier-Stokes-Gleichungen eingesetzt wurde. Der zweite nicht-lineare Gebietszerlegungsansatz hingegen fußt auf konvexer Programmierung und basiert auf einer Zerlegung der korrespondierenden Minimierungsprobleme.

Neben des Hauptthemas der Parallelisierung durch GZ-Methoden untersuchen wir auch das lineare Modellproblem der Bewegungsschätzung selbst. Hierbei stellen wir einen neuen variationellen Ansatz zur Schätzung von turbulenten Flußfeldern auf dem Gebiet der Flußmechanik vor und untersuchen ihn experimentell.

Acknowledgments

First and foremost, I would like to express my gratitude to Prof. Christoph Schnörr for supervising my doctorate, introducing me into the research areas of computer vision and pattern recognition, and for guiding me along the way of scientific thinking and argumentation. Secondly, I am grateful to Prof. Joachim Weickert for serving as an external referee.

Moreover, I would like to thank several people which contributed to this work in a direct or indirect manner. In particular, there is Prof. Daniel Cremers with whom I enjoyed and enjoy many fruitful scientific debates, be it on the conceptual design or the correct mathematical modeling of new and existing approaches. Also, I am most grateful to the countless in-depth discussions with Matthias Heiler on almost all major topics in the field, as well as those in the area of flow estimation and convex optimization with Yuan Jing and Paul Ruhnau, all of which being members of the CVGPR research group at that time. Speaking of the latter, in addition, the broad variety of scientific topics being worked on, the level of expertise and enthusiasm with respect to each of those, and the willingness to discuss them at any time, gave me great support. Moreover, there is Andres Bruhn of the MIA group at the Saarland University, with whom I enjoyed the close cooperation and additional insights into the latest optic flow approaches as well as their implementation.

Also, I would like to thank the group of Prof. Patrick Bouthemy at the INRIA in Rennes, France, for their hospitality in hosting me for three weeks in 2002. In particular, I am grateful to Etienne Mémin both for the organization of the stay, as well as the scientific discussions on high-order motion estimation approaches. In terms of the latter, also I would like to thank Frederic Cao and Elise Arnaud for their comments.

I greatly acknowledge the financial support of the taxpayers of Federal Republic of Germany, in particular those of the State of Baden-Württemberg, which was given to me through the Deutsche Forschungsgemeinschaft (DFG) (grant Schn457/4).

Furthermore, I am most grateful to my parents and my brother for giving me great support as well as the opportunity to pursuing my doctorate.

Last but not least, I want to thank Roy Hovey for proofreading and commenting on the textual parts of this work.

Contents

1 Introduction **1**

1.1 Motivation and Overview 1
1.1.1 Parallel Computing 1
1.1.2 Domain Decomposition 2
1.1.2.1 Overlapping Methods 3
1.1.3 Non-overlapping Methods 5
1.1.4 Domain Decomposition for Nonlinear Problems 6
1.1.4.1 A Control Approach 7
1.1.4.2 A Convex Programming Approach 7
1.1.5 Motion Estimation with High-order Regularization 8
1.2 Contribution and Organization 9
1.3 Related Work 11
1.3.1 Domain Decomposition 11
1.3.2 Nonlinear Domain Decomposition 12
1.3.3 Motion Estimation with High-order Regularization 12
1.4 Mathematical Preliminaries and Notation 12
1.4.1 Sets and Function Spaces and other Notations 13
1.4.2 Finite Element Discretization 14

2 Variational Motion Estimation Methods **15**

2.1 Problem Statement 15
2.2 The Approach by Lucas and Kanade 16
2.3 The Approach by Horn and Schunck 17
2.4 The Combined Local-Global Approach 18
2.4.1 The Approach 18
2.4.2 Discretization by Finite Elements 19
2.4.3 The Solving 20

3 Non-overlapping Domain Decomposition Methods 21
3.1 The Mathematical Basis of Substructuring 25
3.1.1 The Steklov-Poincaré Operator 25
3.1.1.1 The Model Problem in Two-Domain Formulation . 25
3.1.1.2 The Action of S 26
3.1.1.3 The Action of S^{-1} 27
3.1.2 The Schur Complement System 28
3.1.2.1 Two Case of Two Subdomains 28
3.1.2.2 The Multiple Subdomain Case 29
3.2 Iterative Substructuring Methods 30
3.2.1 One-level Methods on Two Subdomains 31
3.2.1.1 The Dirichlet-Neumann Method 32
3.2.1.2 The Neumann-Neumann Method 34
3.2.1.3 Other Methods 35
3.2.2 One-level Methods on Multiple Subdomains 35
3.2.2.1 The Dirichlet-Neumann Preconditioner 36
3.2.2.2 The Neumann-Neumann Preconditioner 37
3.2.2.3 The Block-Jacobi Preconditioner 38
3.2.3 Two-Level Preconditioners 40
3.2.3.1 The Bramble-Pasciak-Schatz Preconditioner 41
3.2.3.2 The Vertex Space Preconditioner 41
3.2.3.3 The Balancing Neumann-Neumann Method 42
3.2.4 Finite Element Tearing and Interconnection Methods 46
3.2.4.1 The One-Level FETI Method 46
3.2.4.2 The Dual-primal FETI Method 52
3.3 Experimental Studies 57
3.3.1 Parameter Selection and Input Data 58
3.3.2 Algorithms and Implementation Details 59
3.3.3 The Impact of Interface Preconditioning 62
3.3.4 Convergence in Dependence on the Number of Subdomains . 62
3.3.5 Convergence in Dependence of the Local Solver's Precision . 63
3.3.6 Scalability Study on a Parallel Computer 64
3.4 Conclusion 66

4 Overlapping Domain Decomposition Methods 73
4.1 One-level Methods 74
4.1.1 The Case of Two Subdomains 74
4.1.1.1 The Alternating Schwarz Method 74
4.1.1.2 The Multiplicative Schwarz Method 76
4.1.1.3 The Additive Schwarz Method 78

4.1.2 The Case of Multiple Subdomains . . . 78
4.1.2.1 The Multiplicative Schwarz Method . . . 79
4.1.2.2 The Additive Schwarz Method . . . 80
4.1.2.3 Schwarz Methods as Parallel Preconditioners . . . 80
4.1.2.4 Links to Gauss-Seidel and Jacobi Iteration . . . 81
4.1.2.5 Scalability Characteristics . . . 82
4.2 Multi-level Algorithms . . . 83
4.2.1 Two-Level Methods . . . 83
4.2.2 Multiplicative Multi-level Methods . . . 85
4.2.3 Additive Multi-level Methods . . . 87
4.2.4 Multi-level Methods as Parallel Preconditioners . . . 87
4.2.5 Links to Multigrid Methods . . . 88
4.2.6 Scalability and Comparison to Iterative Substructuring . . . 90
4.3 Summary . . . 91

5 Motion Estimation with High-order Regularization 93
5.1 The Helmholtz Decomposition . . . 94
5.2 Direct Estimation of the Potential Functions . . . 95
5.3 A Structure-preserving Regularization . . . 96
5.3.1 The Approach . . . 96
5.3.2 Discretization and Solving . . . 98
5.3.3 Embedding into a Multi-resolution Framework . . . 98
5.4 Experimental Studies . . . 99
5.4.1 Parameter Studies . . . 100
5.4.2 Comparison with Existing Approaches . . . 101
5.4.3 Reconstructing the Vortexes of a Landing Air Plane . . . 102
5.5 Conclusion . . . 103

6 TV-based Variational Image Restoration 109
6.1 Regularization Based on the TV-norm . . . 110
6.1.1 Problem Statement . . . 110
6.1.2 Euler-Lagrange Equation . . . 112
6.2 Solving Methods . . . 112
6.2.1 Steepest Descent . . . 113
6.2.2 Fixed Point Iteration . . . 113
6.2.3 Newton's Method . . . 114
6.2.4 Primal-dual Newton's Method . . . 115
6.2.4.1 Mitigating the Nonlinearity . . . 115
6.2.4.2 The Algorithm . . . 116
6.2.5 Experimental Studies . . . 118

6.2.5.1 Input Data, Parameter Values and Error Measures . 118
6.2.5.2 Results 119
6.3 Conclusion 120

7 A Control Approach to Nonlinear Domain Decomposition 127
7.1 The Case of Two Subdomains 128
7.1.1 Problem Statement 128
7.1.2 Lagrange Relaxation and the Optimality System 130
7.1.3 Gradient-based Solving 132
7.1.3.1 Calculating the Gradient 133
7.1.3.2 Application to the Model Problem 135
7.1.3.3 The Solving Algorithm 136
7.1.3.4 Experimental Studies 136
7.2 The Case of Many Subdomains 140
7.2.1 Problem Statement 140
7.2.2 The Optimality System 141
7.2.3 Calculation of the Gradient 142
7.2.4 The Solving Algorithm 144
7.2.5 Experimental Studies 145
7.2.6 Complexity Considerations 145
7.3 Conclusion 146

8 A Convex Programming Approach to Nonlinear DD 151
8.1 Primal-dual Domain Decomposition 152
8.1.1 The Approach 152
8.1.2 Solving 153
8.1.3 Application to the Model Problem 153
8.1.4 Experimental Studies 155
8.2 The Case of Many Subdomains 158
8.2.1 Experimental Results 159
8.2.2 Comparison to Control-based Decomposition 163
8.3 Conclusion 163

9 Conclusion 165
9.1 Summary 165
9.1.1 Linear Domain Decomposition 165
9.1.1.1 Non-overlapping Methods 165
9.1.1.2 Overlapping Methods 167
9.1.2 Nonlinear Domain Decomposition 167
9.1.2.1 Control Approach 167

9.1.2.2 Convex Programming Approach 168
9.1.2.3 Motion Estimation with High-order Regularization . 168
9.2 Future Work 168

List of Figures

1.1 Examples for non-overlapping and overlapping decompositions of Ω . 4

3.1 An ad-hoc decomposition of a variational motion estimation problem 22
3.2 Exemplary partitions used with the different substructuring methods 38
3.3 Examples of torn meshes 46
3.4 Input data for experiments with the marble data set 59
3.5 Input data for experiments with the particles data set 60
3.6 Optimized initialization initialization scheme for the coarse S_0. . . . 61
3.7 Measured convergence rates and theoretical upper limits. 68
3.8 Per-pixel L^2-errors 69
3.9 Measured run and communication times on a PC-cluster 70
3.10 Measured relative and absolute speed-up factors 71
3.11 Measured communication volume for a varying number of subdomains 72

4.1 Examples for overlapping coverings 75
4.2 Example of an overlapping decomposition with coloring 81
4.3 Illustration of the different restriction operators 84

5.1 Input data for the parameter studies 101
5.2 Quantitative parameter studies. 102
5.3 Qualitative parameter study for γ 103
5.4 Qualitative parameters study for λ 104
5.5 Input data for comparison experiment 1 and 2 104
5.6 Direct versus indirect approach results on data set 1. 106
5.7 Direct versus indirect approach results on data set 2. 107
5.8 Results of the real-world experiment 108

6.1 L^2-norm versus TV-norm regularization in 1D denoising 111
6.2 L^2-norm and TV-norm regularization in comparison 122
6.3 L^2-norm and TV-norm regularization in comparison 123

6.4 Convergence diagrams for the different solving methods 124
6.5 Depiction of the dual variable w for the synthetic data set 125

7.1 Illustration of the two model partitions used 128
7.2 Ground truth and noised input image 138
7.3 Total and per-pixel L^2-error for a two-subdomain decomposition . . 139
7.4 Total and per-pixel L^2-error for a 2×2 decomposition 149

8.1 Ground truth and input image 156
8.2 Result and per-pixel L^2-error for a two-subdomain decomposition . . 156
8.3 Development of the L^2-error for different step sizes 157
8.4 Result and per-pixel L^2-error for a 2×2 decomposition 161
8.5 Development of the L^2-error for different step sizes 161
8.6 Development of the L^2-error for different regularization strengths . . 162

List of Algorithms

1	The Dirichlet-Neumann method on two subdomains	33
2	The Neumann-Neumann method on two subdomains	35
3	PCG iteration with Balancing Neumann-Neumann preconditioning	45
4	PCG iteration applied to the FETI problem	50
5	Multiplicative Schwarz iteration in the continuous case	74
6	Alternating Schwarz iteration on non-matching grids	75
7	Multiplicative Schwarz iteration on matching grids	76
8	Additive Schwarz iteration	78
9	Multiplicative multi-level Richardson iteration	86
10	The multiplicative multi-level method as preconditioner	88
11	V/W-cycle multigrid with Schwarz smoothing	89
12	Full multigrid with Schwarz smoothing	90
13	Gradient descent w.r.t. to the control g on two subdomains	137
14	Nonlinear DD by convex programming on a two-subdomain partition	154
15	Nonlinear DD by convex programming on a 2×2 partition	160

Chapter 1

Introduction

1.1 Motivation and Overview

In this chapter, we provide an introduction to the scientific subjects and questions being dealt with in this work.

1.1.1 Parallel Computing

Parallelization denotes a set approaches for the distribution of either the computational requirements or the memory requirements associated with a computational task from one to several computation nodes (one or more processors sharing the same memory physically or virtually). In the first case, the goal is to decrease the computation time, which is also denoted as run-time, whereas in the second case it is to enable feasible computation for numerical problems whose memory requirements are too high for one machine.

In order to reach either goal, the first step in terms of separating the solving algorithm is to identify data independencies either along the work flow or within data being processed at the same time. The first class of approaches is denoted by *functional parallelism*, an example of which could be an image processing pipeline, where the pre-processing, main processing, and post-processing of the video stream is carried out on three different computers at the same time. The second class of approaches, on the other hand, is referred to by *data parallelism*, which would, in the previous example, amount to running several sequential image processing pipelines on different subregions of the image domain.

Despite the existence of shared memory machines requiring almost no communication time for a small number of computation nodes, with every splitting approach the total number of data communications as well as its volume needs to be minimized in order to reach a sufficient *scalability*, i.e. the computation time will decrease when

increasing the computation nodes to even large numbers. In particular, the parameters *latency*, i.e. the initial amount of time to transfer a message from one node to another, and *bandwidth*, i.e. the amount of time to transfer a certain communication volume, need to be taken into account. Ideally, the total communication time does not increase with the number of sub-procedures; however in reality it increases up to a point where the gain by distributing the computation time to more nodes is over-compensated by the increase in communication time. Especially with data parallelism, various kinds of techniques for the optimal spatial splitting, denoted as *data partitioning*, have emerged.

In terms of reaching low communication time, the so-called *granularity* of the algorithm splitting plays an important role. Almost all numerical procedures provide clues for parallel computing on an operationally local level — a matrix-vector multiplication for example where the per-component multiply-and-add operations can be carried out group-wise on different machines concurrently and only the resultant partial sums need to be added sequentially. This so-called *fine-grained* parallelization approaches typically require a high communication volume, i.e. the distribution and collection of the sub-vectors/-matrices by a central node in the aforementioned example. By contrast, *coarse-grained* parallelization approaches aim at splitting the computational problem into independent subproblems at a coarse scale, such that the amount of parallel computation time is significantly larger than the time spend for data exchange.

1.1.2 Domain Decomposition

Although classical image processing techniques, such as linear filtering techniques, are directly amenable to data parallelism, many modern PDE-based methods exhibit couplings between the variables throughout the image plane, which thereby prohibits a direct coarse-grained decomposition. This holds especially true for the class of variational problems, which play a significant role in areas like image segmentation, motion estimation, image enhancement and restoration.

In order to clarify this aspect, let us consider an example from variational motion estimation. The Euler-Lagrange equations of the well-known Horn and Schunck approach read

$$\begin{aligned} \left(\partial_x^2 I \; u_1 + \partial_x I \; \partial_y I \; u_2 + \partial_x I \; \partial_t I\right) - \alpha \Delta u_1 = 0 \\ \left(\partial_x I \; \partial_y I \; u_1 + \partial_y^2 I \; u_2 + \partial_y I \; \partial_t I\right) - \alpha \Delta u_2 = 0 \end{aligned} \tag{1.1}$$

with $I : \Omega \times [0,T] \rightarrow \mathbb{R}$ denoting a sequence of images over the image plane section $\Omega \in \mathbb{R}^2$ and the time period $[0,T]$, α being a scalar parameter, and in conjunction with the boundary conditions $\partial_n u_1 = 0$ and $\partial_n u_2 = 0$ on $\partial\Omega$. Obviously,

because of the presence of the Laplacian terms Δu_1 and Δu_2, after discretization, all unknows in u_1 and u_2 are more or less tightly coupled, such that this problem cannot be directly broken into independent subproblems. For example, partitioning Ω into several subregions, and independent solving of (1.1) on each of those, would yield strong discontinuities across partition boundaries (see also Fig. 3.1 in Chapter (3)). Especially in the case of subregions with very little image information, i.e. texture-less regions, strong artifacts occur when spatial couplings are neglected. On the other hand, applying fine-grained parallelization to the (typically iterative) solving method for problem (1.1), is insufficient to reach significant speed-ups for the reasons mentioned above.

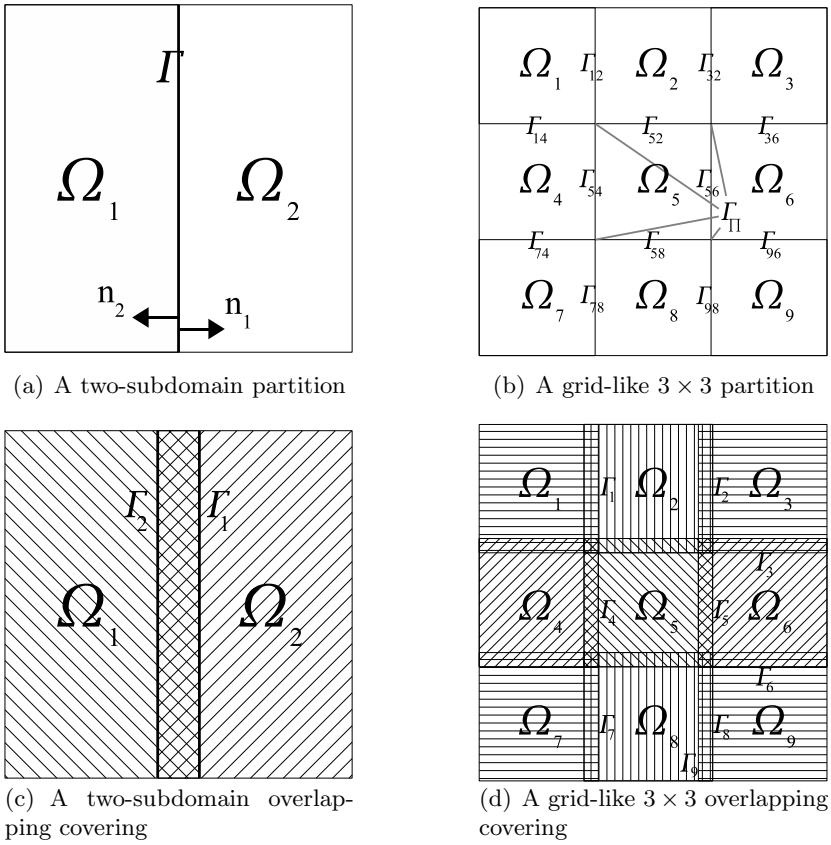
Fortunately, so-called *domain decomposition* (DD) methods provide approaches to spatially decompose PDE problems, thereby making them amenable to coarse-grained parallelization. DD needs to be distinguished from data parallelization in parallel computing, which deals with optimizing the data partitioning and communication patterns of algorithms which already provide clues for coarse-grained parallelization. By contrast, DD methods apply on a higher level, providing means to partition PDE problems on a mathematical level, which can then be optimized for communication patterns by methods of parallel computing.

In general, DD methods are classified with respect to their underlying spatial decomposition. With the first class, the overlapping methods, an *overlapping covering* of the image plane section Ω is assumed, see Figure 1.1(a) and (b), for example. With non-overlapping methods on the other hand, is assumed a *partition* of Ω (in the mathematical sense), (see Figure 1.1(c) and (d), for example). Since the two classes rely on quite different approaches, throughout this work we focus on each of them separately. In the following we provide short introductions for the discrete case of each class. Thereby, we assume a generic linear system of equations $A\,u = f$ on Ω, which realizes a discretization of (1.1), for example, with $u = (u_1, u_2)^\top$, $f = (f_1, f_2)^\top$ and the operator A structured accordingly.

1.1.2.1 Overlapping Methods

Overlapping methods are usually employed to parallelize the *preconditioning step* of a Krylov subspace iteration, e.g. of GMRES or PCG type, *on the original problem*, which involves the independent solving of restrictions of the original problem to each of the subdomains.

For example, let us assume an overlapping covering of Ω by M subdomains $\{\Omega_i,\ i = 1, \ldots, M\}$, and let $R_i,\ i = 1, \ldots, M$ denote operators which restrict a vector on $\overline{\Omega}$ onto the subdomain $\overline{\Omega}_i$, and where $R_i^\top$ refers to its adjoint. Then, by $A_i := R_i^\top A R_i,\ i = 1, \ldots, M$ it is given the restriction of the operator matrix A to each of the subdomain. Then, an overlapping preconditioning step within a Krylov



(a) A two-subdomain partition

(b) A grid-like 3×3 partition

(c) A two-subdomain overlapping covering

(d) A grid-like 3×3 overlapping covering

Figure 1.1: **Examples for non-overlapping and overlapping decompositions of Ω.** (a) and (b): Partitions of Ω into open subsets $\{\Omega_i\}$ sharing boundaries Γ or $\{\Gamma_{ij}, \Gamma_\Pi\}$, respectively. (c) and (d): Overlapping coverings of Ω consisting of overlapping subdomains $\{\Omega_i\}$.

subspace iteration is of the form

$$p^{(k)} = \sum_i R_i^\top A_i^{-1} R_i \, r^{(k)} \ , \tag{1.2}$$

while k denotes the iteration count. Practically, i.e. in terms of parallel implementation, the R_i require a distribution ('scattering' in parallel programming terminology) of the residual vector $r^{(k)}$ from the master process to N slaves processes. The N slave pocesses carry out the action of A_i^{-1} in parallel, and return their local results to the master process, where they are merged to a global $p^{(k)}$ ('gathering'). The remaining operations of the Krylov iteration, such as the operator application step, are left unchanged by overlapping methods, and thus are usually carried out

sequentially in a master process.

Although the preconditioner in (1.2) gives a maximum of data independence, and therefore clues for coarse-grained parallel computing, it is a poor preconditioner in terms the condition number and thus the overall convergence speed of the Krylov iteration. This arises from the fact that (1.2) propagates information only from one subdomain to the other, at each application. As a remedy, several variants of (1.2) arising from different discretizations on coarser grids, in connection with different partitions of Ω, are employed. Thereby, at each iteration, the coarser preconditioners yield a propagation of (aggregated) update information over larger parts of Ω.

1.1.3 Non-overlapping Methods

By contrast, with non-overlapping methods or so-called *substructuring methods*, a Krylov subspace iteration is applied to a reduced variant of the model problem $A\,u = f$, which is defined on the *inner partition boundaries*, i.e. a one-dimensional subset Γ of Ω.

In the case of partitioning Ω into two (non-overlapping) subdomains Ω_1 and Ω_2 for example, the first step is to consider the local restrictions $A^{(i)}u^{(i)} = f^{(i)}, i = 1, 2$ of the original problem to each subdomain. Then, each subproblem is further split into unknowns on the common boundary $\Gamma = \overline{\Omega}_1 \cap \overline{\Omega}_2$ and those on $\Omega \setminus \Gamma$, which reads

$$\begin{pmatrix} A_{II}^{(i)} & A_{I\Gamma}^{(i)} \\ A_{\Gamma I}^{(i)} & A_{\Gamma\Gamma}^{(i)} \end{pmatrix} \begin{pmatrix} u_I^{(i)} \\ u_\Gamma^{(i)} \end{pmatrix} = \begin{pmatrix} f_I^{(i)} \\ f_\Gamma^{(i)} \end{pmatrix} , \quad i = 1, 2 \, . \tag{1.3}$$

Combining them into one global problem on Ω yields

$$\begin{pmatrix} A_{II}^{(1)} & 0 & A_{I\Gamma}^{(1)} \\ 0 & A_{II}^{(2)} & A_{I\Gamma}^{(2)} \\ A_{\Gamma I}^{(1)} & A_{\Gamma I}^{(2)} & A_{\Gamma\Gamma}^{(1)} + A_{\Gamma\Gamma}^{(2)} \end{pmatrix} \begin{pmatrix} u_I^{(1)} \\ u_I^{(2)} \\ u_\Gamma \end{pmatrix} = \begin{pmatrix} f_I^{(1)} \\ f_I^{(2)} \\ f_\Gamma \end{pmatrix} , \tag{1.4}$$

where $f_\Gamma = f_\Gamma^{(1)} + f_\Gamma^{(2)}$ and $u_\Gamma = u_\Gamma^{(1)} = u_\Gamma^{(2)}$. Thirdly, by solving the first two equations for $u_I^{(1)}, u_I^{(2)}$ and substitution into the third equation one obtains a problem, which is defined on the common boundary Γ only:

$$A_{\Gamma I}^{(1)}\big(A_{II}^{(1)}\big)^{-1}(f_I^{(1)} - A_{I\Gamma}^{(1)}u_\Gamma) + A_{\Gamma I}^{(2)}\big(A_{II}^{(2)}\big)^{-1}(f_I^{(2)} - A_{I\Gamma}^{(2)}u_\Gamma) + \big(A_{\Gamma\Gamma}^{(1)} + A_{\Gamma\Gamma}^{(2)}\big)u_\Gamma = f_\Gamma \, . \tag{1.5}$$

This can be written in a more readable form by means of the so called *Steklov-Poincaré* operators (*Schur complement operators* in the discrete case), defined as

$$S^{(i)} = A_{\Gamma\Gamma}^{(i)} - A_{\Gamma I}^{(i)}\big(A_{II}^{(i)}\big)^{-1}A_{I\Gamma}^{(i)}, \quad i = 1, \ldots, N \, , \tag{1.6}$$

by which (1.5) reads:

$$\big(S^{(1)} + S^{(2)}\big)u_\Gamma = f_\Gamma - A_{\Gamma I}^{(1)}\big(A_{II}^{(1)}\big)^{-1} f_I^{(1)} - A_{\Gamma I}^{(2)}\big(A_{II}^{(2)}\big)^{-1} f_I^{(2)} \,. \tag{1.7}$$

In case of a partitioning into an arbitrary number $N > 1$ of subdomains, (1.7) would be of the form

$$\underbrace{\Big(\sum_{i=1}^{N} R_i^\top S^{(i)} R_i\Big)}_{=:S}\; u_\Gamma = \underbrace{\sum_{i=1}^{N} R_i^\top b_\Gamma^{(i)} - \sum_{i=1}^{N} R_i^\top A_{\Gamma I}^{(i)}\big(A_{II}^{(i)}\big)^{-1} b_I^{(i)}}_{=:\chi} \,, \tag{1.8}$$

while the $R_i,\ i = 1, \ldots, N$ denote restrictions from Γ to $\Gamma \setminus \partial\Omega_i$.

Now, the main idea of substructuring is to solve Equation (1.8) by a Krylov subspace iteration, while computing the actions of the S_i independently in parallel. In addition, various kinds of preconditioners exist, which are parallelizable in a similar manner, all of which we will addressed in detail in Chapter 3.

In general, non-overlapping methods have a systematic advantage over overlapping methods in terms of scalability, since the only common unknowns between the subdomain problems are located on a one-dimensional set. That is, the total number of variables to be exchanged between per-subdomain problem processes grows much slower with the number of subdomain problems than it does with the overlapping methods — where overlapping widths of two to five nodes are typical (given a grid-like partition). Especially on parallel computers with shared memory this aspect is of importance, since there the amount of data to be transfered in each iteration can have a significant impact on the total computation time. On the other hand, non-overlapping methods are, in general, more complex both to apply to PDE problems as well as in terms of implementation, as it is with their overlapping counterparts.

1.1.4 Domain Decomposition for Nonlinear Problems

Domain decomposition were primarily applied to linear problems. Nonlinear problems, such as apparent in the are of computational fluid dynamics for example, were ususlly solved by repetitive local linearizations, while parallelization came into play with solving the resulting linear problems at each linearization step. Other approaches pursue the decomposition of the nonlinear problem itself, which then yields nonlinear subproblems, being solved independently. Two approaches of the latter type we will outline in the following sections. Thereby, we assume a generic nonlinear model problem

$$A(u) = f \tag{1.9}$$

on Ω.

1.1.4.1 A Control Approach

Let us consider two non-overlapping subdomains Ω_1 and Ω_2 as defined above, and the restriction of (1.9) to each of those, being of the form $A(u_i) = f_i,\ i = 1, 2$. As in the linear case, an independent solving for each subdomain yields significant errors for all model problems with spatial couplings.

However, these couplings can be re-established by enforcing the function values of u_1 and u_2 as well as their normal derivatives to be the same. In the control approach pursued in here, this is realized through artificial Neumann boundary conditions at Γ:

$$\begin{aligned} A(u_1) = f_1 \quad \text{and} \quad \frac{\partial u_1}{\partial n_1} = g \text{ on } \Gamma, \quad \frac{\partial u_1}{\partial n} = 0 \text{ on } \partial\Omega_1 \setminus \Gamma \\ A(u_2) = f_2 \quad \text{and} \quad \frac{\partial u_2}{\partial n_2} = -g \text{ on } \Gamma, \quad \frac{\partial u_2}{\partial n} = 0 \text{ on } \partial\Omega_2 \setminus \Gamma \end{aligned} , \tag{1.10}$$

while the function g is chosen such that the value of u_1 and u_2 is equal on Γ. The latter condition is stated as a minimization problem of the form[1]

$$\min_{u_1,u_2,g} \frac{1}{2} \int_\Gamma (u_1 - u_2)^2 \, d\chi + \frac{\gamma}{2} \int_\Gamma g^2 \, d\chi \tag{1.11}$$

$$\text{subject to (1.10)} , \tag{1.12}$$

whith (1.10) giving additional equality constraints.

Thereby, it is gives a control problem with the control g, the state variables u_1 and u_2 and the objective functional in (1.11). Standard iterative solving methods from optimal control theory can be applied, whose main computational efforts concentrate on the *independent* solving of the local problems (1.10), thereby making the parallelization of nonlinear problems feasible.

1.1.4.2 A Convex Programming Approach

Another approach for the decomposition of a nonlinear problem can be pursued on the level corresponding energy minimizations. Let the unique solution to the minimization problem

$$\min_{u \in V(\Omega)} J(u) \tag{1.13}$$

with $J(u) : V(\Omega) \to \mathbb{R}$ being a convex functional, be the solution to the nonlinear model problem (1.9). Subsequently, we consider the restriction of $J(u)$ to each of

[1] The presence of the second integral is to exclude arbitrarily large solutions for g.

the subdomains, i.e. $J_{\Omega_1}(\cdot) := J_{|\Omega_1}(\cdot)$ and $J_{\Omega_2}(\cdot) := J_{|\Omega_2}(\cdot)$ in the two-subdomain case, and to minimize the sum of those restrictions while requiring the local solutions $u_1 := u_{|\Omega_1}$ and $u_2 := u_{|\Omega_2}$ to be equal on the shared boundaries:

$$\min_{u_1,u_2} J_{|\Omega_1}(u_1) + J_{|\Omega_2}(u_2) \tag{1.14}$$

$$\text{subject to:} \quad u_{1|\Gamma} = u_{2|\Gamma} \; . \tag{1.15}$$

Thereby, it is known the global solution, given by merging the local ones to (1.14), to converge to the one of the original, non-decomposed problem.

Technically, the equality constraints in (1.14) are relaxed by introducing a Lagrangian multiplier function λ, by which the following minimization-maximization problem is reached:

$$\sup_{\lambda} \left\{ \min_{u_1} \{ J_{\Omega_1}(u_1) + \langle \lambda, u_1 \rangle_\Gamma \} + \min_{u_2} \{ J_{\Omega_2}(u_2) - \langle \lambda, u_2 \rangle_\Gamma \} \right\} \; . \tag{1.16}$$

Obviously, the minimization problems are independent with respect to the variables being minimized for. This fact is exploited within an iterative convex programming method for solving (1.16), where the main computational burden then lies on the parallel solving of the local minimization problems, whereas the sequential operations with respect λ are negligible within that respect.

1.1.5 Motion Estimation with High-order Regularization

Besides the main focus of the parallelization of PDE-based model problems in image processing, the improvement of the model problems and their underlying approaches is dealt with in this work too.

With respect to the Horn and Schunck motion estimation approach for example, given by the energy minimization

$$\inf_{u_1,u_2} \int_\Omega (\partial_x I u_1 + \partial I u_2 + \partial_t I)^2 + \alpha \big(|\nabla u_1|^2 + |\nabla u_2|^2 \big) \, dx \, dy \; , \tag{1.17}$$

the two last terms implement a prior smoothness assumption on the optical flow field (u_1, u_2), in order to overcome the aperture problem. In some application areas, such as that of computational fluid mechanics, however, this general assumption is not appropriate for reconstructing motion patterns which are typical in such scenarios, among which are vortices, or sink- and source-like patterns. Instead, our approach is to penalize changes in the divergence and curl of the vector field to be estimated. That gives energies of the form

$$\inf_{u_1,u_2} \int_\Omega (\partial_x I u_1 + \partial_y I u_2 + \partial_t I)^2 + \alpha \big(|\nabla \mathrm{div}\, u|^2 + |\nabla \mathrm{curl}\, u|^2 \big) \, dx \, dy \; , \tag{1.18}$$

where $u = (u_1, u_2)^\top$.

Furthermore, u can be represented by scalar-valued potential functions $\phi \in V(\Omega)$ and $\psi \in V(\Omega)$ according to

$$u = \nabla\phi + \nabla^\top\psi \,, \tag{1.19}$$

while $\nabla\phi$ generates sink and source-like motions, and $\nabla^\top\psi$ vortices of either direction. The representation through potential functions has the advantage of an inherent separability into these two classes of motion patterns, as well as easier localization of extrema, which becomes important for subsequent processing, e.g. tracking. However, substituting u in (1.18) by its representation given in (1.19) results in third-order derivatives in the regularization terms, which means sixth-order derivatives in the corresponding Euler-Lagrange equations, thereby rendering a discretization very involved. Thus, it makes sense to search for alternative formulations of the same idea in order to decrease the degree of involved derivatives.

1.2 Contribution and Organization

Having provided a motivating outline of the topics being dealt with in this work, in the following we outline our own contribution, as well as the organizational structure of this work.

In Chapter 2 we provide a short introduction into optical flow estimation by explaining the three most well-known approaches, one of which then will serve as the *linear* model problem in the succeeding chapters on classical domain decomposition methods.

Chapter 3 is dedicated exclusively to classical *non-overlapping* domain decomposition methods. First, we introduce and discuss the mathematical foundation, which will lead to the *Steklov-Poincaré interface equation* in the continuous and the *Schur complement equation* in the discrete case. Subsequently, we derive and explain most of the known *iterative substructuring methods* in the order of their appearance. Thus, we always start with the simple case of two subdomains in order to stress out the link to the aforementioned theory and then move on to the case of an arbitrary number of subdomains. Besides the technical aspects of the various methods, we address and discuss in particular the problem of limited information propagation with single-level methods and show how this problem is overcome by presenting different kind of two-level approaches.

Thereafter, we focus on *Finite Element Tearing and Interconnecting (FETI)* methods, which can be seen as dual two-level substructuring methods. Here too, starting with the theoretical foundation, we gradually derive the main algorithm followed by a comparison to primal iterative two-level methods. The section is

followed by an elaboration on a combination of the primal and the dual approach, which finally results in the recently emerged *primal-dual FETI* approach.

In the subsequent *experimental section* we present empirical results for two prominent primal approaches being applied to a standard variational motion estimation method. In particular, measurements of the convergence rate in dependence on the number of subdomains for two different data sets, as well as its comparison to theoretical upper limits, will be shown. In addition, measurement results of the sensitivity of the convergence rate to the precision of the local solvers are given. The experimental section concludes with results of run-time measurements on a dedicated PC-cluster for up to 144 processors, including a scalability and speed-up analysis in comparison to a non-parallel multi-grid implementation.

Chapter 4 is dedicated to *overlapping* methods. However, because of their adversarial communication requirements in comparison to non-overlapping methods, we leave out empirical studies and remain with the derivation and explanation of the established methods, again in the order of their appearance. In particular, we show the well-known multigrid methods to be special cases in the multi-level framework.

As already addressed in Section 1.1.5, the improvement of the model problem itself, i.e. here the variational motion estimation, is pursued also. Thereby, a new method for the *direct* estimation of the potential fields ϕ and ψ from a given image sequence of turbulent flows is derived and explained in Chapter 5. After details on a multi-resolution solving, the chapter concludes with an elaborate experimental section showing results for synthetic and real input data.

In the remaining chapters we focus on the domain decomposition of *nonlinear* PDE-based problems. As a prototypical exemplar of the latter type, we have chosen an variational image denoising method based on a *Total Variation* (TV) regularization. Because of its superior filtering results in comparison to L^2-norm-based methods while requiring special numerical schemes dealing with the nonlinearity, it has been the subject of intensive research in the past 15 years, which we detail on both theoretically and empirically in Chapter 6.

Subsequently, in Chapter 7 we propose a new decomposition approach stated as a control problem, as outlined in Section 1.1.4.1. Starting with the formulation as energy minimization on a two-subdomain partition, we derive step-by-step the corresponding solving method consisting of several linear PDE problems. Experimental results for the denoising model problem at the end of the chapter show its feasibility for two and more subdomains.

Alternatively, in Chapter 8 we propose another approach for decomposing nonlinear problems, this time based on a Lagrangian relaxation. As outlined in Section 1.1.4.2, we formulate the decomposed problem as a constrained convex optimization problem, relax it by the means of Lagrangian multiplier functions, and solve the corresponding primal-dual problem by a subgradient iteration. Also here,

besides the theoretical derivation, results of an experimental feasibility study are presented at the end of the chapter.

The results in the aforementioned chapters have been published in journals [82] and conferences [79, 80, 81], respectively.

1.3 Related Work

Since most of the relevant references are given along with their elaborations in the following chapters, in this section we mention only the most important monographs and survey papers.

1.3.1 Domain Decomposition

The extensive work which has been done and findings which have been published on these topics can be found in the annual conferences on domain decomposition, whose proceedings, beginning in the late eighties [60, 28, 74, 61], reflect the evolvement of DD methods, their underlying theory as well as their application to various areas (see, e.g., the proceedings of the latest conferences [71, 83]). In addition, the monographs by Quarteroni and Valli [101] and Toselli and Widlund [120] provide deep insights into the mathematical foundations of Schwarz and substructuring methods and detailed overviews of the multitude of existing methods as well as many application examples. In particular, in [120] also the more recently emerged dual and primal-dual FETI methods are discussed. On the other hand, the monograph by Smith, Bjørstad and Gropp [112] pursues a more algebraic approach of explaining most of the primal overlapping and non-overlapping methods, while focusing more on implementation issues in terms of parallel computation as well as complexity, though providing a substantial chapter on the abstract convergence analysis of Schwarz methods. Similarly, the survey articles by Chan and Mathew [34] and LeTallec [87] give detailed algebraic and algorithmic explanations of the existing primal procedures, while the publications by Xu and Zou [128] and Xu [127] provide explanations by well-founded continuous formulations then thoroughly focusing on their discretization by finite elements. Especially in terms of dual substructuring methods (FETI), the survey article by Farhat and Roux [55] provides a strong algebraic insight amidst the application-context of computational mechanics.

An important application area of DD methods can be found in computational fluid dynamics and computational mechanics, e.g. by parallelizing Stokes problems, advection-diffusion problems, or the linear elasticity problem, see, e.g., [101]

and [120]. More recently, vector-valued problems of the form

$$-\nabla \cdot (\alpha \operatorname{div} u) + Bu = f \quad \text{on} \quad \Omega \tag{1.20}$$

$$u \cdot n = 0 \quad \text{on} \quad \partial\Omega\,, \tag{1.21}$$

with B denoting a symmetric positive coefficient matrix, α a positive scalar and n the outer unit normal on $\partial\Omega$, which are similar to our model problem (1.1), were the subject of extensive studies in connection with Domain Decomposition, [126, 125] and Chapter 10 of [120].

Regarding the fields of image processing and computer vision, we are not aware of any published investigation of Domain Decomposition of variational problems.

1.3.2 Nonlinear Domain Decomposition

In general, the approaches for spatially decomposing nonlinear PDE problems can be grouped into three different classes: With the oldest approach, a Newton is applied as outer iteration in connection with a standard overlapping DD technique to parallelize the involved linear problems (see [25], [45]). The Schwarz alternating method, a well-known overlapping method, is applied directly to the nonlinear problem (see, e.g., [91, 92]). Besides employing classical DD methods in the nonlinear case, by a third group of approaches, a non-overlapping decomposition is reached by means of control theory, which results in two sets of nonlinear subdomain problems (see [86, 65, 67, 66]).

1.3.3 Motion Estimation with High-order Regularization

Building upon the div-curl regularization proposed by Suter [116], Corpetti, Mémin et al. [39, 40] proposed the splitting of the estimated flow field into an irrotational and solenoidal component in Fourier space, followed by an estimation of the potential functions ϕ and ψ by means of path integration. Subsequently, they show how to track the local extrema once those estimates have been computed. In addition, they present a technique to lower the degree of involved derivatives by relaxing the regularization onto auxiliary functions.

The topics addressed in Chapter 5 have been recently extended end elaborated by Yuan et al. [129, 130].

1.4 Mathematical Preliminaries and Notation

In this section we note the basic notation being used throughout this work.

1.4.1 Sets and Function Spaces and other Notations

Let $\Omega_1, \Omega_2, \ldots, \Omega_N \subset \Omega \subset \mathbb{R}^2$ be open and bounded domains with Lipschitz continuous boundaries $\partial\Omega, \partial\Omega_1, \partial\Omega_2, \ldots \partial\Omega_N$, while the $\{\Omega_i\}$ are referred to as *subdomains* of Ω.

Moreover, we make use of the usual Sobolev space for second order elliptic boundary value problems:

$$V(\Omega) = H^1(\Omega) = W^{1,2}(\Omega) = \{v \in L^2(\Omega) \,:\, \partial^\alpha v \in L^2(\Omega)\,,\; 0 \leq |\alpha| \leq 1\}\,, \quad (1.22)$$

with α denoting a multi-index. The corresponding scalar product is defined as

$$(u, v)_1 = \sum_{|\alpha|\leq 1} (\partial^\alpha u, \partial^\alpha v)\,, \quad (1.23)$$

while the scalar product of $L^2(\Omega)$ is given by:

$$(u, v) = \int_\Omega u(x)\, v(x)\, dx\,, \quad (1.24)$$

or by

$$\big(u, v\big)_\Gamma := \int_\Gamma u(x) v(x)\, dx, \qquad \text{for } \Gamma \subset \Omega. \quad (1.25)$$

when being restricted to some subset $\Gamma \subset \Omega$, respectively. Often, we will make use of subspaces $V_i := V(\Omega_i),\ i = 1, \ldots, N$ which are restricted to a certain subdomain Ω_i.

We do not need the notion of 'traces' and 'trace spaces' in the following. Hence we loosely speak of functions with vanishing boundary values:

$$V_0 = H_0^1(\Omega) \subset H^1(\Omega)\,. \quad (1.26)$$

Likewise, we use the symbolic notation

$$\int_\Gamma \frac{\partial u}{\partial n}\, v\, ds = \langle \partial_n u, v\rangle_\Gamma \quad (1.27)$$

for the duality pairing with respect to the trace space $H^{1/2}(\Gamma)$ and its dual, and call $\partial_n u$, the outer normal derivative of u, a function as well.

For $u, v \in H^1(\Omega)$ and $\Delta u \in L^2(\Omega)$, the following version of Green's formula holds:

$$\int_\Omega \nabla u \cdot \nabla v\, dx = (-\Delta u, v) + \langle \partial_n u, v\rangle_{\partial\Omega}\,. \quad (1.28)$$

1.4.2 Finite Element Discretization

If not specified otherwise, all discretizations in this work are based on standard conforming piecewise linear finite elements.

Therefore, we restrict our considerations to those (sub)domains $\Omega, \Omega_1, \ldots, \Omega_N$ which allow for conforming, shape-regular and quasi uniform triangulations $\mathcal{T}_i^h$ of the maximum diameters h_i, $i = 1, \ldots, N$. Furthermore, we will distinguish between the sub-cases of *matching* and *non-matching* $\mathcal{T}_i^h$ triangulations. In the matching case, all overlapping triangulations are compatible, i.e. all nodes in the overlap region are element of all triangulations involved. In the non-matching case the opposite holds.

To simplify notation, we use the *same symbols* for some function $v = v(x)$ and the coefficient vector $v \in \mathbb{R}^N$ representing the approximation of $v(x)$ in the subspace spanned by the piecewise linear basis functions $\{\phi(x)\}_{i=1,\ldots,N}$:

$$v(x) \in H^1(\Omega) \quad \leftrightarrow \quad v \in \mathbb{R}^N \quad \leftrightarrow \quad \sum_{i=1}^{N} v_i \phi_i(x) \; . \tag{1.29}$$

Furthermore, we use the *same symbol* for the vector obtained by discretizing the action of some linear functional on some function $v(x)$. For example, we simply write f and w for the discretized versions of the linear functionals (f, v) and $a(w, v)$ (with $a(\cdot, \cdot)$ being a bilinear form and $w(x)$ fixed). We refer to standard textbooks like, e.g. [5], [37], [107], for the discretization of boundary value problems with finite elements.

Chapter 2

Variational Motion Estimation Methods

In this chapter we introduce the main model problem for the linear domain decomposition methods, which will be the so-called *combined local-global* (CLG) motion estimation approach. Therefore, we start with motivating the problem of estimating the optical flow from an image sequence, introduce the global Horn and Schunck approach, followed by the local approach by Lucas and Kanade and conclude with the CLG approach. Thereby, we always address the aspect of parallel computation.

2.1 Problem Statement

Let $I : \Omega \times 0, T \rightarrow \mathbb{R}$ denote a sequence of images taken over the time period $[1, T]$. Assuming that a change of intensities over time arises from the movement of the recorded objects only, the problem of optic flow is to determine the vector field $(u_1, u_2) \in V(\Omega \times \{1, T\}) \times V(\Omega \times \{1, T\})$ denoted as *optic flow*, which fulfills the equation

$$I(x + u_1, y + u_2, t + 1) = I(x, y, t) \; , \tag{2.1}$$

and are commonly referred to as *brightness constancy* assumption. Thereby, (u_1, u_2) give a mapping of each intensity I at (x, y) from time point t to $t + 1$.

Obviously, (2.1) not only is a nonlinear, but also an under-determined problem. Unlike image registration approaches, nonlinearity here is avoided by a first-order Taylor series expansion of the left term with respect to u_1, u_2 and t, respectively, giving the so-called *optic flow constraint*

$$\partial_x I \, u_1 + \partial_y I \, u_2 + \partial_t I = 0 \; , \tag{2.2}$$

a solution (u_1, u_2) which is a sufficiently close to that of (2.1) if the magnitudes are relatively small, e.g. not more than two pixels.

Still the problem of under-determination remains. In particular, having only (2.1) or (2.2) only the so-called *normal flow*

$$u_n = -\frac{\partial_t I}{|\nabla I|}\frac{\nabla I}{|\nabla I|}\,, \tag{2.3}$$

that is, the displacement vectors projected onto the image gradients can be determined, which is commonly denoted as *aperture problem*. However, since we are interested in the original displacement vectors, additional assumptions besides brightness constancy, are necessary, which give rise to several different approaches. We will focus on three of the most important of these approaches in the following section. Thereby, we will make use of the following notation:

$$\begin{aligned} u &:= (u_1, u_2)^\top & I_\xi &:= \partial_\xi I, \quad \xi \in \{x, y, t\} \\ |\nabla u|^2 &:= |\nabla u_1|^2 + |\nabla u_2|^2 & \nabla I &:= (\partial_x I,\ \partial_y I)^\top\,. \end{aligned}$$

2.2 The Approach by Lucas and Kanade

Common to all motion estimation approaches is the presence of additional *prior* assumptions on the vector field (u_1, u_2). In the class of local approaches, for example, Lucas and Kanade [90] in 1981 assumed (u_1, u_2) to be *constant in a local neighborhood* of size ρ. In that case for each location (x, y, t) the flow components u_1 and u_2 are determined by minimizing the functional

$$J(\hat{u}_1, \hat{u}_2) = \inf_{u_1, u_2 \in V} K_\rho * \left((I_x u_1 + I_y u_2 + I_t)^2\right), \tag{2.4}$$

with $K_\rho * \cdot$ denoting convolution by a Gaussian kernel $K_\sigma(x, y)$ of standard deviation σ, which is equivalent to a weighted least squares fit.

Deriving J_{LK} for u_1 and u_2, respectively, and setting equal to zero, yields the linear system

$$\begin{pmatrix} K_\rho * (I_x)^2 & K_\rho * (I_x I_y) \\ K_\rho * (I_x I_y) & K_\rho * (I_y)^2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = - \begin{pmatrix} K_\rho * (I_t I_x) \\ K_\rho * (I_t I_y) \end{pmatrix}\,. \tag{2.5}$$

Important with respect to parallelization here is the fact that (2.5) can be solved for each image location $(x, y) \in \Omega$ *separately*. Thus, the solving of (2.5) can be parallelized directly so that the total execution time can be decreased by distributing the computational effort to more than one processing node.

However, in image regions lacking sufficient intensity gradients, the matrix in (2.5) becomes degenerate, that is, its second eigenvalue gets close to 0, which corresponds to having the aperture problem again. Consequently, this approach yields only *sparse* flow fields for locations where both eigenvalues are significantly present, which is a major disadvantage with respect to subsequent image analysis. On the other hand, the magnitude of the second eigenvalue gives a good confidence measure of the flow vectors at each location.

2.3 The Approach by Horn and Schunck

As opposed to the local approach by Lucas-Kanade, Horn and Schunck [72] at the same time suggested to embed equation (2.2) into a global, variational optimization problem, and to add a *global* regularization term:

$$J(\hat{u}_1, \hat{u}_2) = \inf_{u_1,u_2 \in V} \int_\Omega (I_x u_1 + I_y u_2 + I_t)^2 + \alpha\big(|\nabla u_1|^2 + |\nabla u_2|^2\big)\, dx\, dy\,, \qquad (2.6)$$

with $\alpha > 0$ denoting the regularization strength. Here the additional assumption is not local constancy, but global smoothness of the flow field throughout Ω, which is implemented by penalizing the magnitude changes of u_1 and u_2.

In order to find a miminum, the first variations of $J(u_1, u_2)$ must vanish, which results in

$$\int_\Omega \begin{pmatrix} v_1 & v_2 \end{pmatrix} \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} + \alpha\big(\nabla u_1 \cdot \nabla v_1 + \nabla u_2 \cdot \nabla v_2\big)\, dx\, dy$$
$$= -\int_\Omega \begin{pmatrix} I_t I_x \\ I_t I_y \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}\, dx\, dy, \quad \forall v_1, v_2 \in V(\Omega)\,, \qquad (2.7)$$

for which it is known a unique solution to exist (See [105]).

By partial integration with respect to $\nabla u_1 \cdot \nabla v_1$ and $\nabla u_2 \cdot \nabla v_2$ one reaches the Euler-Lagrange equations

$$I_x^2\, u_1 + I_x I_y u_2 + I_x I_t - \alpha \Delta u_1 = 0 \qquad (2.8)$$
$$I_x I_y, u_1 + I_y^2 u_2 + I_y I_t - \alpha \Delta u_2 = 0\,, \qquad (2.9)$$

with Neumann boundary conditions $\partial_n u_1 = 0$ and $\partial_n u_2 = 0$ at $\partial\Omega$.

In contrast to the previous method, the values of $(u_1(x), u_2(x))$ are spatially coupled throughout Ω, because of the presence of spatial derivatives of u_1 and u_2 in (2.7) and (2.8), respectively, which arise from the global regularization. Hence,

problem (2.7) *cannot* be decomposed spatially and thus provides no direct clues for coarse-grained parallel computing. On the other hand, the regularization term $\left(|\nabla u_1|^2 + |\nabla u_2|^2\right)$ yields a so-called *fill-in* of flow information at locations where $|\nabla I| \approx 0$, by implicitly interpolating from the neighborhood. Thus this global approach results in dense flow fields, advantageous for many applications. However, it is also known to be more sensitive to noise [7] than the local methods.

2.4 The Combined Local-Global Approach

2.4.1 The Approach

Recently, Bruhn et al. [24, 23] proposed a hybrid optical flow approach, which combines the robustness of local approaches with the density of global approaches. This is reached by convoluting the products of the image derivatives in (2.7) with Gaussian kernels:

$$\underbrace{\int_\Omega \begin{pmatrix} u_1 & u_2 \end{pmatrix} \begin{pmatrix} K_\rho * (I_x^2) & K_\rho * (I_x I_y) \\ K_\rho * (I_x I_y) & K_\rho * (I_y^2) \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} + \alpha\left(\nabla u_1^\top \nabla v_1 + \nabla u_2^\top \nabla v_2\right) dx\, dy}_{=:a(u,v)}$$

$$= \underbrace{\int_\Omega \begin{pmatrix} K_\rho * (I_t I_x) \\ K_\rho * (I_t I_y) \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} dx\, dy}_{=:f(v)}, \quad \forall v_1, v_2 \in V(\Omega)\,, \quad (2.10)$$

which is equivalent to embedding (2.5) instead of the optical flow constraint equation (2.3) into the variational energy in (2.6). By using the bilinear form $a(\cdot,\cdot)$ and linear form $f(\cdot)$, (2.10) can be written as

$$a(u, v) = f(v)\,, \quad \forall v \in V^2(\Omega)\,, \quad (2.11)$$

with $u = (u_1, u_2)^\top$ and $v = (v_1, v_2)^\top$. The corresponding Euler-Lagrange equations read

$$\begin{aligned} K_\rho * I_x^2 u_1 + K_\rho * I_x I_y u_2 + K_\rho * I_x I_t - \alpha \Delta u_1 = 0 \\ K_\rho * I_x I_y u_1 + K_\rho * I_y^2 u_2 + K_\rho * I_y I_t - \alpha \Delta u_2 = 0\,, \end{aligned} \quad (2.12)$$

with Neumann boundary conditions $\partial u_n = 0$ on $\partial\Omega$. By taking the constant terms to the right-hand side, one reaches the form

$$K_\rho * I_x^2 u_1 + K_\rho * I_x I_y u_2 - \alpha \Delta u_1 = -K_\rho * I_x I_t \quad (2.13)$$

$$K_\rho * I_x I_y u_1 + K_\rho * I_y^2 u_2 - \alpha \Delta u_2 = -K_\rho * I_y I_t\,, \quad (2.14)$$

which we compactly write as

$$A\,u = f \tag{2.15}$$

by means of the continuous linear operator $A : V(\Omega) \times V(\Omega) \to V'(\Omega) \times V'(\Omega)$, where $V'(\Omega)$ denotes the dual space of $V(\Omega)$, the vector field $u := (u_1, u_2)$ and the right-hand side f.

Although experimental results [24, 23] have approved the superiority of this approach in comparison to that of Horn and Schunck as well as Lucas and Kanade, even further refinements exist. For example, Bruhn et al. suggested to assume flow field smoothness also along the temporal dimension by replacing regularization term in (2.6) by

$$|\nabla_3 u_1|^2 + |\nabla_3 u_2|^2 \tag{2.16}$$

with $\nabla_3 = (\partial_x, \partial_y, \partial_t)^\top$ denoting the spatio-temporal derivation operator. Furthermore, instead of using quadratic penalization both in the data term and the regularization, the utilization nonlinear penalization functions lead to a better preservation of flow discontinuities. Further improvements have been made by the introduction of vectorial penalizers in the regularization term, which also take the spatial orientation into account and thus implement so-called anisotropic regularization. See [124] for a survey for both types of extensions.

In the experimental section of Chapters 3, (2.10) will serve as the model problem, since as a linear PDE-based problem, it is prototypical for many others in image processing.

2.4.2 Discretization by Finite Elements

In order to compute the vector field u, we consider equation (2.11) to be discretized by piecewise linear finite elements over the triangulated section Ω of the image plane. We arrange the vectors of nodal variables u_1, u_2 corresponding to the finite element discretizations of $u_1(x), u_2(x)$ as follows: $u = (u_1^\top, u_2^\top)^\top$. Taking into consideration the symmetry of the bilinear form $a(u, v)$, this induces the following block structure of the *discretized* version of (2.11):

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} \quad \Leftrightarrow \quad A\,u = f\,, \tag{2.17}$$

where $\forall i, j = 1, \ldots, N$:

$$(A_{11})_{ij} = a\big((\phi_i, 0)^\top, (\phi_j, 0)^\top\big) \tag{2.18}$$

$$(A_{12})_{ij} = a\big((\phi_i, 0)^\top, (0, \phi_j)^\top\big) \tag{2.19}$$

$$(A_{21})_{ij} = (A_{12})_{ji} \tag{2.20}$$

$$(A_{22})_{ij} = a\big((0, \phi_i)^\top, (0, \phi_j)^\top\big) \tag{2.21}$$

$$(f_1)_i = f\big((\phi_i, 0)^\top\big) \tag{2.22}$$

$$(f_2)_i = f\big((0, \phi_i)^\top\big)\ . \tag{2.23}$$

and with ϕ_k denoting linear basis function corresponding to the nodal variable $(u_1)_k$ or $(u_2)_k$, respectively. Again, we use the *same* symbols for referring to nodal finite element weights and the continuous functions and operators they are discretizing.

2.4.3 The Solving

Bruhn et al. [22, 19] made extensive studies on multigrid solvers for (2.8) and other, more advanced optic flow approaches, thereby reaching frame rates of 63 frames per second for images sizes of 120×160 pixels on a single processor. Nowaday image sensors, however, can provide sizes of 2000×2000 or even higher at similar frame rates. Therefore, it is sought for speeding up the computations even further by means of coarse-grained parallel computing.

With almost all variational optic flow approaches, however, A_{11} and A_{22} are of banded structure with six off-diagonals when employing conformal linear finite elements, whereas A_{12} and A_{21} are simply diagonal matrices. Thus, the inverse operator matrix, A^{-1}, is *dense* and a straightforward spatial splitting and subsequent inversion of A, according to a spatial decomposition of the image plane, does not yield the right solution.

This motivates the following chapters, where we will present different kinds of methods to split LSEs like (2.17) spatially into independent smaller problems, such that the merged local solutions will be consistent (up to a given tolerance) with the solution of the original problem.

Chapter 3

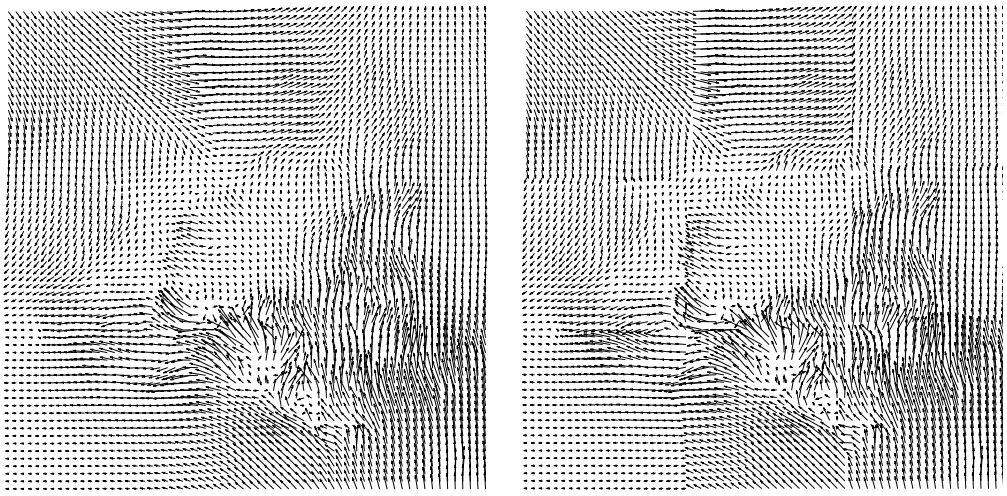
Non-overlapping Domain Decomposition Methods

Obviously the problem requiring solution, given either in variational (2.11) or differential (3.6) form, is global in the sense that all unknowns are coupled. Consequently, a straightforward spatial separation of the problem, i.e. independent solvings followed by assembling the local results to a global one, results in significant errors. See Figure (3.1) for an example. Especially at subdomain boundaries where the regularization term is dominant, i.e. at places with little image structures, significant artifacts appear, since the couplings across boundaries are all neglected.

In contrast to that ad-hoc approach, in the following two chapters we will present approaches to decompose the model problem spatially into independent subproblems while still preserving the global couplings and thus reaching the same solution, depending on the number of iterations being introduced, as with sequential solving.

The motion estimation model problem given either in variational (2.11) or differential (3.6) form, is global, in the sense that all unknowns are more or less strongly coupled. Consequently, a straightforward spatial separation of the problem into local subproblems, in order to distribute the computational effort to several processing nodes, results in strong artifacts across the subdomain boundaries. (See Figure (3.1).) Especially at subdomain boundaries where the spatial regularization term is dominant, i.e. at places with little image structures, significant errors appear, since the couplings across those boundaries are completely neglected. In contrast to this ad-hoc approach, in the following chapter we will present methods to decompose the model problem spatially indirectly. That is by first solving a reformulation on the artificial boundaries and by solving for the remaining unknowns in a subsequent step. Thereby, the spatial couplings will be preserved and one will reach the same solution, depending on the number of iterations being introduced, as by sequentially

solving the original problem.



(a) Original optical flow estimation (b) Separated solving on 3×3 subdomains

Figure 3.1: **An ad-hoc decomposition of a variational motion estimation problem.** Separate solving on subdomains while neglecting the spatial couplings yields strong artifacts at the shared boundaries.

In this chapter, we consider *partitions* $\{\Omega_i \mid i = 1, \ldots, N\}$ of the image section Ω, i.e. subdomains which have no overlap and $\Omega_i \cap \Omega_j = \emptyset,\ j \neq i$, and $\overline{\Omega} = \cup_i \overline{\Omega}_i$. The common boundary between two adjacent subdomains Ω_i and Ω_j we denote by Γ_{ij}, while $\Gamma_{ij} := \overline{\Omega}_i \cap \overline{\Omega}_j \setminus \partial\Omega$. The union of all shared boundaries is referred to by Γ. Furthermore, by Γ_i we denote that subset of a subdomain's boundary $\partial\Omega_i$, which are a part of other local boundaries too, i.e. $\Gamma_i := \partial\Omega_i \setminus \partial\Omega$. Though the methods to be presented below are applicable to arbitrarily shaped partitions, we restrict our considerations to the case to grid-shaped ones in the following.

The central notion behind the so-called *substructuring methods* [112, 101, 120], which are to be explained later, is to restate the original problem on Ω to a variant which defined on the shared subdomain boundaries Γ only. It is then solved for the unknowns on Γ, while in each iteration local problems for the interior, i.e. non-shared, degrees of freedom on each subdomain are to be solved, giving clues for parallel computation. Once a solution for the reduced problem has been determine the remaining unknowns to gain the full solution are calculated in a similar, also highly parallelizable, finalization step.

The chapter is organized as follows. First, we focus on the mathematical foundations for restating the original problem and explicitly show the link between continuous formulations and the algorithmic realization. Second, we present and explain

the various *primal iterative methods* in the order of their appearance in the past two decades. In the third part, we concentrate on the more recently emerged *dual* and *primal-dual methods*. In the concluding experiential part, we present results of experimental studies made for two prominent primal methods, among which are scalability measurements on a parallel machine for up to 144 processing nodes and images sizes of 2000×2000 pixels.

Model Problems. For the time being, we consider the following convex, scalar-valued model problem:

$$J(u) = \inf_{v \in V} \int_\Omega (v - I)^2 + \alpha |\nabla v|^2 \, dx \, dy \tag{3.1}$$

which is also known as the *Definite Helmholtz equation*, with $V = H^1(\Omega)$, $I \in V$ giving image intensities over the image plane section Ω, and $\alpha \in \mathbb{R}$ determining the regularization strength imposed on u. Vanishing of the first variation yields

$$\frac{d}{d\tau} J(u + \tau v)\Big|_{\tau=0} = a(u, v) - (f, v) = 0 \; , \quad \forall v \in V \; , \tag{3.2}$$

with the bilinear form:

$$a(u, v) = \int_\Omega u \, v + \alpha \nabla u \cdot \nabla v \, dx \, dy \; . \tag{3.3}$$

Since

$$a(v, v) \geq \min\{1, \lambda\} \|v\|_V^2 \; , \quad \forall v \in V \; , \tag{3.4}$$

J is strictly convex, and the global minimum is the unique solution to the variational equation:

$$a(u, v) = (f, v) \; , \quad \forall v \in V \; . \tag{3.5}$$

By applying (1.28), partial integration yields the Euler-Lagrange equation along with the *natural* boundary condition [68]:

$$L \, u := -\alpha \Delta u + u = f \quad \text{in } \Omega \; , \quad \partial_n u = 0 \quad \text{on } \partial\Omega \; . \tag{3.6}$$

The solution u to (3.5) is the so-called weak solution to (3.6). Discretization yields a linear system as the algebraic counterpart of (3.5),

$$A \, u = f \; , \tag{3.7}$$

with a symmetric, sparse and positive definite matrix A.

In a subsequent step, the linear system (3.7) is permuted and partitioned in order to group the variables on the shared boundaries Γ as well as the remaining ones on $\overline{\Omega} \setminus \Gamma$:

$$\begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} u_I \\ u_\Gamma \end{pmatrix} = \begin{pmatrix} f_I \\ f_\Gamma \end{pmatrix} , \tag{3.8}$$

with the indices Γ and I referring to each of those two sets, respectively. Furthermore, we need restrictions of (3.8) to each of the subdomains Ω_i, giving

$$\begin{pmatrix} A_{II}^{(i)} & A_{I\Gamma}^{(i)} \\ A_{\Gamma I}^{(i)} & A_{\Gamma\Gamma}^{(i)} \end{pmatrix} \begin{pmatrix} u_I^{(i)} \\ u_\Gamma^{(i)} \end{pmatrix} = \begin{pmatrix} f_I^{(i)} \\ f_\Gamma^{(i)} \end{pmatrix} , \quad i = 1, \ldots, N . \tag{3.9}$$

All relevant concepts and algorithms will be shown for this model problem. The generalization on the algebraic level to the main model problem of optical flow estimation (2.12), which will be assumed in the experimental section, will be straightforward, when partitioning the corresponding LSE (3.7), in the same manner as in (2.17).

Mathematical Preliminaries. The decomposition of problem (3.6) into a set of parallel solvable problems requires boundary conditions different from the natural boundary condition in (3.6). We will collect necessary details in the following paragraphs.

Suppose we wish to have $u|_\Gamma = u_\Gamma$ on Γ, i.e. a Dirichlet boundary condition, with some given function u_Γ:

$$Au = f \quad \text{in } \Omega, \quad \partial_n u = 0 \quad \text{on } \partial\Omega \setminus \Gamma, \quad u = u_\Gamma \text{ on } \Gamma. \tag{3.10}$$

To obtain the corresponding variational formulation as basis of a proper finite element discretization, we define the subspace:

$$V_{0|\Gamma} = \{ v \in V \ : \ v|_\Gamma = 0 \} . \tag{3.11}$$

The variational formulation of (3.10) then reads: Find $u_{0|\Gamma} \in V_{0|\Gamma}$ such that:

$$a(u_{0|\Gamma}, v) = (f, v) - a(Pu_\Gamma, v) , \quad \forall v \in V_{0|\Gamma} . \tag{3.12}$$

where P is an *arbitrary* extension operator $V(\Gamma) \to V(\Omega)$. The desired solution is then given by $u = u_{0|\Gamma} + Pu_\Gamma \in V$.

Alternatively, suppose we wish to have $\partial_n u = u_n$ on Γ, i.e. a Neumann boundary condition, with a given function u_n:

$$Au = f \quad \text{in } \Omega , \quad \partial_n u = 0 \quad \text{on } \partial\Omega \setminus \Gamma , \quad \partial_n u = u_n \quad \text{on } \Gamma . \tag{3.13}$$

The corresponding variational formulation reads: Find $u \in V$ such that:

$$a(u, v) = (f, v) + \langle u_n, v \rangle_\Gamma , \quad \forall v \in V . \tag{3.14}$$

3.1 The Mathematical Basis of Substructuring

3.1.1 The Steklov-Poincaré Operator

3.1.1.1 The Model Problem in Two-Domain Formulation

In order to restate problem (3.6) as a problem on Γ let us start with its so-called multi-domain formulation:

$$L\,u^{(1)} = f^{(1)} \quad \text{in } \Omega_1 \qquad\qquad \partial_{n_1} u^{(1)} = 0 \quad \text{on } \partial\Omega_1 \cap \partial\Omega \tag{3.15}$$

$$L\,u^{(2)} = f^{(2)} \quad \text{in } \Omega_2 \qquad\qquad \partial_{n_2} u^{(2)} = 0 \quad \text{on } \partial\Omega_2 \cap \partial\Omega \tag{3.16}$$

$$u^{(1)} = u^{(2)} \quad \text{on } \Gamma \tag{3.17}$$

$$\partial_n u^{(1)} = \partial_n u^{(2)} \quad \text{on } \Gamma\,. \tag{3.18}$$

which is equivalent to Equation (3.6), i.e.

$$u(x,y) = \begin{cases} u^{(1)}(x,y) & (x,y) \in \Omega_1 \\ u^{(2)}(x,y) & (x,y) \in \Omega_2 \end{cases} \tag{3.19}$$

holds true, for u denoting the solution to the original solution, cf., e.g., [101]. Obviously, this still provides no clues for parallel computation, because of the coupling of both subproblems via the artificial boundary conditions (3.17) and (3.18) on Γ. Instead, let $u^{(1)}$ and $u^{(2)}$ formally denote the solution to either subproblem. Then, by (3.17) we have for the restricted solutions that $u^{(1)}|_\Gamma = u^{(2)}|_\Gamma =: u_\Gamma$. Subsequently, the restricted solution u_Γ is substituted into (3.18) by means of the *Steklov-Poincaré* operator introduced in the following. Once the resulting equation has been solved for u^Γ, the functions $u^{(1)}$ and $u^{(2)}$ then follow from the substitution of u_Γ back into (3.15) and (3.16) with boundary condition (3.17).

Consequently, in order to solve system (3.15)-(3.18), we have to make explicit the dependency between $\partial_n u|_\Gamma$ and $u|_\Gamma$ of the solution u to a boundary value problem.

First, let us decompose u into two functions,

$$u = u_0 + u_f\,, \tag{3.20}$$

which are the unique solutions to the following problems:

$$\begin{cases} L\,u_0 = 0 \text{ in } \Omega, \\ \partial_n u_0 = 0 \text{ on } \partial\Omega \setminus \Gamma, \\ \quad u_0 = u_\Gamma \text{ on } \Gamma \end{cases} \tag{3.21}$$

$$\begin{cases} L\,u_f = f \text{ in } \Omega, \\ \partial_n u_f = 0 \text{ on } \partial\Omega \setminus \Gamma, \\ \quad u_f = 0 \text{ on } \Gamma. \end{cases} \tag{3.22}$$

Obviously, we have that $u_f \in V_{0|\Gamma}$ and:

$$u|_\Gamma = u_0|_\Gamma \tag{3.23}$$
$$\partial_n u = \partial_n u_0 + \partial_n u_f \ . \tag{3.24}$$

Then, the definition of the *Steklov-Poincaré operator* S is as follows(cf. e.g. [101]):

$$S : u_\Gamma \to \partial_n u_0|_\Gamma \tag{3.25}$$

which is known to be symmetric, coercive and continuous. Applying this mapping to the solutions $u^{(1)}, u^{(2)}$ of equations (3.15) and (3.16) in the domains Ω_1 and Ω_2, respectively, equation (3.18) becomes

$$\left(S^{(1)} + S^{(2)}\right) u_\Gamma = -\partial_{n_1} u_f^{(1)}|_\Gamma - \partial_{n_2} u_f^{(2)}|_\Gamma \tag{3.26}$$

with $u_\Gamma = u^{(1)}|_\Gamma = u^{(2)}|_\Gamma$ because of (3.17). (3.26) is denoted as the *Steklov-Poincaré interface equation*.

That is, by (3.26) we have reduced the original problem (3.6) defined on Ω to a problem restricted to the interface Γ, which is also denoted as *interface problem* in the two-subdomain case.

In the following we will show the discretization and solution of (3.26) as well as the calculation of the unknowns not on Γ.

3.1.1.2 The Action of S

First, we will explain the action of the *Steklov-Poincaré operator* by the algebraic representation of its discretized version. For the following two sections, we therefore restrict our considerations to one subdomain with an artificial boundary part Γ.

By equation (3.12), the variational formulation of problem (3.21) reads:

$$\begin{aligned} a(u_{0|\Gamma}, v) &= -a(Pu_\Gamma, v), \quad \forall v \in V_{0|\Gamma}, \\ u_0 &= u_{0|\Gamma} + Pu_\Gamma \in V \ , \end{aligned} \tag{3.27}$$

with P denoting an arbitrary extension operator from $V(\Gamma)$ to $V(\Omega)$, cf. [101].

Discretization yields a linear system with respect to $u_{0|\Gamma}$ of the form

$$A_{II} u_{0|\Gamma} = -A_{I\Gamma} u_\Gamma \ , \tag{3.28}$$

while using the partitioning of A as noted with (3.8). Let us compare the linear systems (3.28) and (3.7), the latter corresponding to the boundary value problem (3.6). Note that the dimension of the latter system is larger because v runs through V in (2.11), whereas in (3.27) v only varies in $V_{0|\Gamma}$.

Now consider $u_0 = u_{0|\Gamma} + Pu_\Gamma$, with $u_{0|\Gamma}$ from (3.27) and (3.28), respectively, and let v vary in V. Note that for $v \in V_{0|\Gamma} \subset V$,

$$a(u_0, v) = a(u_{0|\Gamma} + Pu_\Gamma, v) = a(u_{0|\Gamma}, v) + a(Pu_\Gamma, v) = 0\,, \tag{3.29}$$

due to (3.27). For $v|_\Gamma \neq 0$ and taking into consideration $Lu_0 = 0$ from (3.21), we obtain by (3.14):

$$a(u_0, v) = \left\langle \partial_n u_0, v \right\rangle_\Gamma\,, \quad \forall v \in V\,. \tag{3.30}$$

Since $u_0|_\Gamma = u_\Gamma$ due to (3.21), discretization of this variational equation yields the linear system:

$$\begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} u_0|_I \\ u_\Gamma \end{pmatrix} = \begin{pmatrix} 0 \\ \partial_n u_0|_\Gamma \end{pmatrix}\,, \tag{3.31}$$

from which we conclude by algebraically eliminating $u_0|_I$ (cf. definition (3.25)):

$$S\,u_\Gamma = (A_{\Gamma\Gamma} - A_{\Gamma I} A_{II}^{-1} A_{I\Gamma})\,u_\Gamma = \partial_n u_0|_\Gamma\,, \tag{3.32}$$

which is also known as the *Schur complement* of A.

Hence, the *action of* S on some boundary data u_Γ involves the solution of problem (3.28), exhibiting artificial Dirichlet boundary conditions at Γ and natural ones, i.e. Neumann conditions for our model problem, on $\partial\Omega \setminus \Gamma$. We will refer to those local problems as 'Dirichlet problems' in the sequel.

3.1.1.3 The Action of S^{-1}

In order to make the *action of* S^{-1} explicit as well, we may formally invert equation (3.32). Since S is dense, this is not advisable. Therefore, in practice, one solves the Neumann problem (3.30) for u_0 with $f = 0$ and *given* boundary data $\partial_n u_0$ (compare with (3.14)) and obtains by restriction to the interface Γ: $u_\Gamma = u_0|_\Gamma$.

Alternatively, this can also be algebraically derived from (3.31) by the following factorization:

$$\begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix} = \begin{pmatrix} I & 0 \\ A_{\Gamma I} A_{II}^{-1} & I \end{pmatrix} \begin{pmatrix} A_{II} & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} I & A_{II}^{-1} A_{I\Gamma} \\ 0 & I \end{pmatrix}. \tag{3.33}$$

Inverting this matrix yields

$$\begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix}^{-1} = \begin{pmatrix} I & -A_{II}^{-1} A_{I\Gamma} \\ 0 & I \end{pmatrix} \begin{pmatrix} A_{II}^{-1} & 0 \\ 0 & S^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -A_{\Gamma I} A_{II}^{-1} & I \end{pmatrix}\,, \tag{3.34}$$

by which we conclude

$$S^{-1} \partial_n u_0|_\Gamma = \begin{pmatrix} 0 & I \end{pmatrix} \begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ I \end{pmatrix} \partial_n u_0|_\Gamma = u_0|_\Gamma = u_\Gamma\,, \tag{3.35}$$

see, e.g., [113].

Therefore, the *action of* S^{-1} to some boundary data $\partial_n u_0|_\Gamma$ is computed by solving problem (3.31) with respect to u_Γ, which exhibits artificial Neumann boundary conditions at Γ, for which reason we will denote such problems as 'Neumann problems' in the following.

3.1.2 The Schur Complement System

3.1.2.1 Two Case of Two Subdomains

Using the results of the previous sections, we return now to equation (3.26) in connection with solving the system of equations (3.15)–(3.18).

Suppose the boundary values $u_\Gamma = u^{(1)}|_\Gamma = u^{(2)}|_\Gamma$ on the interface Γ separating Ω_1 and Ω_2 were known. Then $u^{(1)}$ and $u^{(2)}$ within Ω_1 and Ω_2, respectively, can be exactly computed as discussed for problem (3.10). Here, $u_{0|\Gamma} = u_I^{(i)}$ is given by

$$u_I^{(i)} = \left(A_{II}^{(i)}\right)^{-1}(f_I^{(i)} - A_{I\Gamma}^{(i)} u_\Gamma) \;, \quad i = 1, 2 \;. \tag{3.36}$$

Thus, it remains to compute the unknown boundary function $u_\Gamma := u^{(1)}|_\Gamma = u^{(2)}|_\Gamma$. Again, this will be done by solving equation (3.18) stated as the Steklov-Poincaré interface equation (3.26). Since we have shown how to invert the Steklov-Poincaré operator $S^{(i)}$ by means of submatrices of $A^{(i)}$, we are left with the computation of $\partial_{n^i} u_f^{(i)}$, $i = 1, 2$. This can be reached by the same procedure used to compute (3.30). Since $Lu_f^{(i)} = f^{(i)}$, $i = 1, 2$, we obtain

$$a(u_f^{(i)}, v) = (f^{(i)}, v) + \left\langle \partial_{n^i} u_f^{(i)}, v \right\rangle_\Gamma \;, \quad i = 1, 2 \;, \quad \forall\, v \in V \;. \tag{3.37}$$

Discretization yields the linear systems

$$\begin{pmatrix} A_{II}^{(i)} & A_{I\Gamma}^{(i)} \\ A_{\Gamma I}^{(i)} & A_{\Gamma\Gamma}^{(i)} \end{pmatrix} \begin{pmatrix} u_f^{(i)} \\ 0 \end{pmatrix} = \begin{pmatrix} f_I^{(i)} \\ f_\Gamma^{(i)} + \partial_{n_i} u_{f|\Gamma}^{(i)} \end{pmatrix} \;, \quad i = 1, 2 \;. \tag{3.38}$$

Due to the system (3.15)–(3.18), we have to solve these two linear systems simultaneously, along with the system (3.31) applied to either domain $\Omega_i, i = 1, 2$. Since $u_i = u_0^{(i)} + u_f^{(i)}$, summation of the two systems (3.31) and (3.38) for each domain, respectively, gives:

$$\begin{pmatrix} A_{II}^{(i)} & A_{I\Gamma}^{(i)} \\ A_{\Gamma I}^{(i)} & A_{\Gamma\Gamma}^{(i)} \end{pmatrix} \begin{pmatrix} u_I^{(i)} \\ u_\Gamma^{(i)} \end{pmatrix} = \begin{pmatrix} f_I^{(i)} \\ f_\Gamma^{(i)} + \partial_{n_i} u^{(i)}|_\Gamma \end{pmatrix} \;, \quad i = 1, 2 \;. \tag{3.39}$$

We combine these equations into a single system:

$$\begin{pmatrix} A_{II}^{(1)} & 0 & A_{I\Gamma}^{(1)} \\ 0 & A_{II}^{(2)} & A_{I\Gamma}^{(2)} \\ A_{\Gamma I}^{(1)} & A_{\Gamma I}^{(2)} & A_{\Gamma\Gamma}^{(1)} + A_{\Gamma\Gamma}^{(2)} \end{pmatrix} \begin{pmatrix} u_I^{(1)} \\ u_I^{(2)} \\ u_\Gamma \end{pmatrix} = \begin{pmatrix} f_I^{(1)} \\ f_I^{(2)} \\ f_\Gamma + \partial_{n_1} u^{(1)}|_\Gamma + \partial_{n_2} u^{(2)}|_\Gamma \end{pmatrix}, \tag{3.40}$$

where $f_\Gamma = f_\Gamma^{(1)} + f_\Gamma^{(2)}$. By solving the first two equations for $u_I^{(1)}, u_I^{(2)}$ and substitution into the third equation of (3.40), we conclude that (3.18) holds iff:

$$A_{\Gamma I}^{(1)} \big(A_{II}^{(1)}\big)^{-1} (f_I^{(1)} - A_{I\Gamma}^{(1)} u_\Gamma) + A_{\Gamma I}^{(2)} \big(A_{II}^{(2)}\big)^{-1} (f_I^{(2)} - A_{I\Gamma}^{(2)} u_\Gamma) + \big(A_{\Gamma\Gamma}^{(1)} + A_{\Gamma\Gamma}^{(2)}\big) u_\Gamma = f_\Gamma \, .$$

Applying representation (3.32) of the Schur complement we finally obtain

$$\underbrace{\big(S^{(1)} + S^{(2)}\big)}_{=:S} u_\Gamma = \underbrace{f_\Gamma - A_{\Gamma I}^{(1)} \big(A_{II}^{(1)}\big)^{-1} f_I^{(1)} - A_{\Gamma I}^{(2)} \big(A_{II}^{(2)}\big)^{-1} f_I^{(2)}}_{=:\chi} \tag{3.41}$$

being denoted as the *Schur complement system* and corresponding to the discretization of the Steklov-Poincaré interface equation (3.26). Recall that equation (3.41) was derived by substituting (3.15)–(3.17) into (3.18). Accordingly, imposing the solution u_Γ to (3.41) as boundary conditions as required in (3.17), the functions $u^{(1)}$ and $u^{(2)}$ can be computed from (3.15) and (3.16) such that (3.19) holds.

3.1.2.2 The Multiple Subdomain Case

Let us now consider the case of $N > 2$ subdomains $\{\Omega_i | i = 1, \ldots, N\}$. Thereby, we will make use of restriction operators R_i which restrict the nodal variables on Γ those on each local shared boundary $\Gamma_i, i = 1, \ldots, N$, respectively. Its inverses $R_i^\top$ then the extensions by zero are from Γ_i to Γ. Accordingly, let $A^{(i)} u^{(i)} = b^{(i)}$ be the original problem restricted to either subdomain and $A^{(i)} := R_i A R_i^\top$. Furthermore, we consider each local stiffness matrix $A^{(i)}$ to be partitioned as having been introduced with (3.8). Then, the local Schur complements are defined by

$$S^{(i)} = A_{\Gamma\Gamma}^{(i)} - A_{\Gamma I}^{(i)} \big(A_{II}^{(i)}\big)^{-1} A_{I\Gamma}^{(i)}, \quad i = 1, \ldots, N \, . \tag{3.42}$$

As opposed to the two-subdomain case, the local problem associated to $A_{II}^{(i)}$ can exhibit only artificial, Dirichlet boundary conditions, which is the case for interior subdomains Ω_i, i.e. $\partial\Omega_i \cap \partial\Omega = \emptyset$.

Subsequently, the Schur complement equation here reads

$$\underbrace{\Big(\sum_{i=1}^{N} R_i^\top S^{(i)} R_i\Big)}_{=:S} u_\Gamma = \underbrace{\sum_{i=1}^{N} R_i^\top f_\Gamma^{(i)} - \sum_{i=1}^{N} R_i^\top A_{\Gamma I}^{(i)} \big(A_{II}^{(i)}\big)^{-1} f_I^{(i)}}_{=:\chi}, \tag{3.43}$$

which can be derived in the same manner as in the two-subdomain case. With this formulation the direct clues for parallel computation become quite clear: the action of the global Schur complement matrix S mainly consists of those of the local matrices $S^{(i)}$, which themselves require the solving local Dirichlet problems (see above). Since the $R_i^\top S^{(i)} R_i$ can be applied independently, so can be the local solutions, that is at the same time on several processing nodes. This will be the starting point of almost all methods determining a solution for u_Γ to be presented below. Once a such has been determined, the remaining values on $\overline{\Omega} \setminus \Gamma$ are given by

$$u_I^{(i)} = \left(A_{II}^{(i)}\right)^{-1} \left(f_I^{(i)} - A_{I\Gamma}^{(i)} R_i \, u^\Gamma\right) , \quad i = 1, \ldots, N , \tag{3.44}$$

cf. (3.36) also.

An important issue with the case of multiple subdomains is the fact that for many prominent model problems, such as the Poisson problem or that of linear elasticity [98], for interior subdomains, i.e. those where $\partial\Omega_i \cap \partial\Omega = \emptyset$, the corresponding $A^{(i)}$ become singular, and so the corresponding $S^{(i)}$ through (3.35), which is due to purely artificial Neumann boundary conditions in that case. As a consequence, if particular $A^{(i)}$ and $S^{(i)}$ are singular, it also means their null spaces kernel$\left(A^{(i)}\right)$ and kernel$\left(S^{(i)}\right)$, respectively, are non-empty. In the case of the Poisson equation

$$-\Delta u = f \quad \text{on } \Omega_i, \qquad u = 0 \quad \text{on} \quad \partial\Omega \tag{3.45}$$

for example, replacing the original Dirichlet boundary conditions by any Neumann boundary conditions yields a singular problem, since u will then be defined only up to a constant. Consequently, the operator's null space would comprise all constant functions on Ω_i, which would hold true for the Schur complement operator on Γ_i too. Those null spaces will become important with the coarse-grid preconditioners being presented in Section 3.2.3.3 and Section 3.2.4.1 below.

3.2 Iterative Substructuring Methods

The starting point of all substructuring methods is the solving of equation (3.43). Roughly, they can be classified in three groups.

The first and oldest group, denoted by *direct substructuring methods*, aim at assembling and solving (3.43) explicitly. Thereby, the global Schur complement S is not assembled in one step. Instead, the local Schur complements $S^{(i)}$ are assembled themselves by even smaller ones in a recursive manner. That is, the direct substructuring methods process upwards a tree of local Schur complement matrices, at the top of which is the global one S. Once the global Schur complement equation has been solved, typically by factorization, the tree is descended and it is backsolved for

the remaining variables according to (3.44). Parallel computation comes into play both in the assembling and the backsolve phase, and thus very large linear systems can be solved on parallel machines. However, due to the Schur complements being dense matrices, direct methods exhibit a high demand both in computation and in memory.

In contrast, with so-called *iterative substructuring methods*, the second group of methods, the Schur complements in (3.43) are never calculated explicitly. Instead, iterative solving methods are employed, such as Richardson or PCG iteration, where only the actions of the $S^{(i)}$ to some vector, but not the operator matrices themselves are computed. This amounts to solving N local Dirichlet problems at every iteration in parallel, cf. Section 3.1.1.2. Thus, preconditioning is necessary, since S is usually very ill-conditioned. In fact, methods of this group only differ in the type of preconditioner they employ, which, however, arises from quite different approaches and leads to different algorithms in the end.

More recently, dual approaches have arisen, denoted by *Finite Element Tearing and Interconnecting (FETI)*. There, (3.43) is broken up into the local Schur complement equations $S^{(i)}u_\Gamma^{(i)} = \chi^{(i)}, i = 1, \ldots, N$, yielding a duplication of variables in u_Γ, and restating them as quadratic minimization problems. Equality of the duplicated variables then is enforced by additional constraints. By Lagrangian relaxation of the such defined constrained optimization problem, it is then restated as a dual problem with respect to Lagrange multipliers λ, which is solved by a modified PCG iteration.

In the following, we will focus on iterative substructuring methods as well as FETI methods, and elaborate on their mathematical foundation as well as algorithmic implementation. Direct substructuring methods will not be focused on further, because of their disadvantages previously mentioned and since they have been overcome by the more recent iterative methods.

3.2.1 One-level Methods on Two Subdomains

We will start with the two most basic iterative methods on two subdomains and explain their link to the Schur complement equation.

Thereby, we always assume (3.41) for being solved by a Richardson iteration using different kinds of preconditioners P:

$$u^{k+1} = u^k + \theta P(\chi - S\, u^k), \tag{3.46}$$

with $\theta > 0$ being an acceleration parameter and $k \geq 0$ denoting the iteration count. As a matter of fact, all the following methods only differ in the type of preconditioner P, which, however, arises from quite different approaches and also leads to different kind of algorithms in the end. The same holds with the primal methods in the case of multiple subdomains, where PCG iteration will be employed.

Although the following algorithms do not provide clues for parallel computation, they will give the building blocks for those in the multiple-subdomain case, and elucidate the connection between the iterative solving of the interface equation (3.41) and the local problems on each subdomain.

Finally, note that the following preconditioners are commonly referred to as *interface preconditioners*, according to their definition on the common boundary Γ.

3.2.1.1 The Dirichlet-Neumann Method

Let us start with the algorithm first. The *Dirichlet-Neumann* method, cf. e.g. [12, 17], is given in Algorithm 1. It is known to converge to the true solution u_Γ of (3.41) for an acceleration parameter $0 < \theta < \frac{2}{\nu}$, where ν denoting the greatest eigenvalue of $S^{(2)^{-1}}S$. That is, one has to first solve a local problem Ω_1 with artificial Dirichlet boundary conditions at Γ, followed by a problem on Ω_2 with artificial Neumann boundary conditions. The two problems in Algorithm 1 also allow for the following weak formulation, see, e.g., [101]:

$$\begin{cases} a_1(u^{(1),k}, v^{(1)}) = (f^{(1)}, v^{(1)})_{\Omega_1} & \forall v^{(1)} \in V^0(\Omega_1) \\ u^{(1),k+1} = u_\Gamma^{(k)} \text{ on } \Gamma \end{cases} \tag{3.47}$$

$$\begin{cases} a_2(u^{(2),k}, v^{(2)}) = (f^{(2)}, v^{(2)})_{\Omega_2} & \forall v^{(2)} \in V^0(\Omega_2) \\ a_2(u^{(2),k}, P\nu) = (f^{(2)}, P\nu)_{\Omega_2} + (f^{(1)}, P\nu)_{\Omega_1} - a_1(u^{(1),k}, P\nu) & \forall \nu \in V(\Gamma)\ , \end{cases} \tag{3.48}$$

where the last equation corresponds the weak formulation of the Neumann boundary conditions on Γ in in the second problem in Algorithm 1.

Discretization then results in the linear systems of equations

$$A_{II}^{(1)} u_I^{(1),k+1} = f_I^{(1)} - A_{I\Gamma} u_\Gamma^k \tag{3.49}$$

and

$$\begin{pmatrix} A_{II}^{(2)} & A_{I\Gamma}^{(2)} \\ A_{\Gamma I}^{(2)} & A_{\Gamma\Gamma}^{(2)} \end{pmatrix} \begin{pmatrix} u_I^{(2),k+1} \\ u_\Gamma^{k+1/2} \end{pmatrix} = \begin{pmatrix} f_I^{(2)} \\ f_\Gamma - A_{\Gamma I}^{(1)} u^{(1),k+1} - A_{\Gamma\Gamma}^{(1)} u_\Gamma^k \end{pmatrix}\ . \tag{3.50}$$

Moreover, algebraic solving for $u_I^{(1),k+1}$ in (3.49) yields

$$u_I^{(1),k+1} = \left(A_{II}^{(1)}\right)^{-1} f_I^{(1)} - \left(A_{II}^{(1)}\right)^{-1} A_{I\Gamma}^{(1)} u_\Gamma^k\ , \tag{3.51}$$

Algorithm 1: The Dirichlet-Neumann method on two subdomains

initialize u_Γ^1 with an arbitrary value

iterate $k = 1, 2, \ldots$ **until** convergence

$$\text{solve for } u^{(1),k+1}\text{:} \qquad \begin{cases} Lu^{(1),k+1} = f^{(1)} & \text{in } \Omega_1, \\ \partial_n u^{(1),k+1} = 0 & \text{on } \partial\Omega_1 \setminus \Gamma, \\ u^{(1),k+1} = u_\Gamma^k & \text{on } \Gamma \end{cases}$$

$$\text{solve for } u^{(2),k+1}\text{:} \qquad \begin{cases} Lu^{(2),k+1} = f^{(2)} & \text{in } \Omega_2, \\ \partial_n u^{(2),k+1} = 0 & \text{on } \partial\Omega_2 \setminus \Gamma \\ \partial_n u^{(2),k+1} = \partial_n u^{(1),k+1} & \text{on } \Gamma\ , \end{cases}$$

$$\text{update:} \qquad u_\Gamma^{k+1} \leftarrow \theta u^{(2),k+1}|_\Gamma + (1-\theta)\, u_\Gamma^k,$$

whereas eliminating $u_I^{(2),k+1}$ in (3.50) results in

$$\begin{aligned} \big(A_{\Gamma\Gamma}^{(2)} - A_{\Gamma I}^{(2)} \big(A_{II}^{(2)}\big)^{-1} A_{I\Gamma}^{(2)}\big) u_\Gamma^{k+1/2} \\ = f_\Gamma - A_{\Gamma I}^{(1)} u_I^{(1),k+1} - A_{\Gamma\Gamma}^{(1)} u_\Gamma - A_{\Gamma I}^{(2)} \big(A_{II}^{(2)}\big)^{-1} f_I^{(2)}\ , \end{aligned} \tag{3.52}$$

respectively. Substituting the expression for $u_I^{(1),k+1}$ into (3.52) and subsequent application of (3.42) results in

$$S^{(2)} u_\Gamma^{k+1/2} = f_\Gamma - A_{\Gamma I}^{(2)} \big(A_{\Gamma\Gamma}^{(1)}\big)^{-1} f_I^{(1)} - A_{\Gamma I}^{(2)} \big(A_{\Gamma\Gamma}^{(2)}\big)^{-1} f_I^{(2)} - S^{(1)} u_\Gamma^k\ . \tag{3.53}$$

Also, we can identify the remaining terms which contain submatrices or subvectors of the original linear system of equations as the right-hand side of (3.41):

$$S^{(2)} u_\Gamma^{k+1/2} = \chi - S^{(1)} u_\Gamma^k\ . \tag{3.54}$$

Finally, by eliminating for $u_\Gamma^{k+1/2}$ and inserting the result into the update equation of Algorithm 1 gives

$$u_\Gamma^{k+1} = \theta S^{(2)^{-1}} \big(\chi - S^{(1)} u_\Gamma^k\big) + (1-\theta) u_\Gamma^k \tag{3.55}$$

$$\Leftrightarrow \quad u_\Gamma^{k+1} = u_\Gamma^k + \theta S^{(2)^{-1}} \big(\chi - S u_\Gamma^k\big)\ . \tag{3.56}$$

The latter equation reveals that the algorithm indeed is a Richardson iteration on the interface variables u_Γ employing the local Schur complement $S^{(2)}$ as preconditioner.

Furthermore, (3.55) can be also explained by the details given in Section 3.1.1.2 and Section 3.1.1.3: the problem on Ω_1 having Dirichlet conditions at Γ is equivalent to the action of $S^{(1)}$, and the Neumann problem on Ω_2 to the action of $S^{(2)^{-1}}$.

The reason for using $S^{(2)^{-1}}$ as a preconditioner here is because of the fact that its eigenvalue spectrum is equivalent to $S = S^{(1)} + S^{(2)}$ [12]. In particular, it is known for condition number κ, i.e. the ratio of the largest to the smallest eigenvalue, of the preconditioned operator to be independent from the mesh size h:

$$\kappa\left(\left(S^{(2)}\right)^{-1}S\right) \leq \hat{C} \tag{3.57}$$

for a positive constant $\hat{C}$. As a consequence, the error reduction rate, or convergence rate, of the Dirichlet-Neumann algorithm is bound upwards independently of the number of unknowns, i.e. the size of the problem. All preconditioners providing this property are commonly referred to as *optimal*.

Note that, since $S^{(i)}, i = 1, 2$ is a symmetric preconditioner, if $A^{(i)}$ is symmetric also, PCG iteration can be used to accelerate the Richardson procedure.

3.2.1.2 The Neumann-Neumann Method

Another important interface preconditioner is given by the *Neumann-Neumann* preconditioner [14, 42, 85], reading

$$P_{NN} := \left(\rho_1 S^{(1)^{-1}} + \rho_2 S^{(2)^{-1}}\right) \tag{3.58}$$

where ρ_1 and ρ_2 denote two positive weights chosen such that $\rho_1 + \rho_2 = 1$. Thus, the Richardson iteration is of the following form:

$$u_\Gamma^{k+1} = u_\Gamma^k + \theta\Big(\rho_1 S^{(1)^{-1}} + \rho_2 S^{(2)^{-1}}\Big)\Big(\chi - (S^{(1)} + S^{(2)})\, u_\Gamma^k\Big)\,. \tag{3.59}$$

Consequently, the preconditioning step requires to solve a Neumann problem on *both* subdomains, which is given in detail in Algorithm 2.

That is, the amount of computational work doubles in comparison to the previous method. However computation time remains nearly the same if each subdomain problem is solved on two processing nodes in parallel.

Besides the fact that the NN preconditioner is also optimal, its advantages arise from the possibility of compensating for coefficient jumps at Γ through adjusting the weights ρ_1, ρ_2 accordingly. The same holds for crosspoints on Γ in the case of more than two subdomains also.

Algorithm 2: The Neumann-Neumann method on two subdomains

initialize u_Γ^1 with an arbitrary value

iterate $k = 1, 2, \ldots$ **until** convergence

$$\text{solve for } u^{(i),k+1}\text{:} \quad \begin{cases} Lu^{(i),k+1} = f^{(i)} & \text{in } \Omega_i, \\ \partial_n u^{(i),k+1} = 0 & \text{on } \partial\Omega_i \setminus \Gamma, \qquad i = 1, 2 \\ u^{(i),k+1} = u_\Gamma^k & \text{on } \Gamma \end{cases}$$

$$\text{solve for } w^{(i),k+1} \quad \begin{cases} Lw^{(i),k+1} = 0 & \text{in } \Omega_i, \\ \partial_n w^{(i),k+1} = 0 & \text{on } \partial\Omega_i \setminus \Gamma, \quad i = 1, 2 \\ \partial_n w^{(i),k+1} = \partial_n & w^{(1),k+1} - \partial_n w^{(2),k+1} \text{ on } \Gamma \end{cases}$$

$$\text{update:} \quad u_\Gamma^{k+1} \leftarrow u_\Gamma^k - \theta\big(\sigma_1 w^{(1),k+1}|_\Gamma - \sigma_2 w^{(2),k+1}|_\Gamma\big) \, .$$

3.2.1.3 Other Methods

Several other methods have emerged. Besides variants to the Dirichlet-Neumann method, such as the *Robin method* [88], we here mention the *probing preconditioning method* [31], where it is aimed at estimating a preconditioner matrix *explicitly*, which can consists of only a few diagonals, prior to the iteration, which can then be effectively inverted while solving. In order to get a good estimate, the action of S is probed, i.e. calculated for a small set of chosen vectors.

Finally, for the standard model problem, the Poisson problem, an explicit preconditioner based on an algebraic diagonalization of S has been devised, giving the so-called *J-operator* [11].

3.2.2 One-level Methods on Multiple Subdomains

After having explained the role of interface preconditioners as well as their computation for two well-known types on two subdomains, in the following we will concentrate on the case of many subdomains. In this section, we restrict our focus to one-level preconditioners, which provide only a very limited parallel scalability, and proceed in the succeeding sections with two-level extensions which will overcome this drawback.

Typically, a Krylov subspace method is employed, such as PCG or GMRES [63, 68], in order to solve the Schur complement equation (3.43). Since A is symmetric and positive definite for our model problem, so is S, for which reason we only consider

PCG iteration as solving procedure in the following.

As with the Richardson iteration in the two-subdomain case, here too the methods only differ in the type of preconditioner employed as well as the choice of initial value in a few cases. The operator application step

$$u \leftarrow \left(\sum_{i=1}^{N} R_i^\top S^{(i)} R_i\right) r \tag{3.60}$$

always remains the same, and mainly amounts to solving N Dirichlet problems, cf. Section 3.1.1.2, in parallel. Also note that the actions of R_i and $R_i^\top$ correspond to a scattering of the input vector r to the N computing nodes, or a gathering of the results from them, respectively. Both operations also allow for concurrent execution, depending on the used hard- and software platform.

Moreover, upper bounds for condition number $\kappa(P^{-1}S)$ in dependence of H and therefore the number of subdomains, measured by $\frac{1}{H^2}$, play an important role with the analysis of the forthcoming preconditioners P^{-1}. Thereby, it is well known, [69, p. 272], for $\kappa(P^{-1}S)$ to limit the convergence rate of the PCG iteration according to:

$$\frac{||\hat{u} - u^k||}{||\hat{u} - u^0||} \leq 2\,\frac{c^k}{1 + c^{2k}} \qquad \text{with} \qquad c = \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right) \tag{3.61}$$

with $\hat{u}$ denoting the true solution, u^0 the initial value and k the iteration count. We will make use of this relationship in the experiments presented at the end of this section.

3.2.2.1 The Dirichlet-Neumann Preconditioner

Generalizing the two-subdomain case, half of the local Schur complement inverses $S^{(i)^{-1}}$ serve as the building blocks of the *Dirichlet-Neumann (DN) preconditioner* [44, 9], given by

$$P_{DN}^{-1} := \sum_{i \in I_R} R_i^\top S^{(i)^{-1}} R_i \;, \tag{3.62}$$

where I_R denotes a subset of all subdomain indices, such that none of corresponding subdomains are adjacent. Note that choosing I_R is equivalent to applying a red-black coloring to the partition of Ω. See Fig. 3.2(a) for an example, where I_R then refers to the indices of the 'red' subdomains. Consequently, the DN preconditioner can only be applied for partitions allowing for such a coloring.

Note that for certain model problems, the $S^{(i)}$ of interior subdomains can be singular, as already mentioned in Section 3.1.2.2. As a remedy, the singular $\left(A^{(i)}\right)^{-1}$

in (3.35) are replaced by pseudo-inverses $\left(A^{(i)}\right)^{\dagger}$, [42, 46], which can be obtained by adding small constants to the diagonal entries of $\left(A^{(i)}\right)$, for example. Then, through computing the action of $\left(A^{(i)}\right)^{\dagger}$ in (3.35), one determines the result of applying the pseudo-inverses $\left(S^{(i)}\right)^{\dagger}$.

3.2.2.2 The Neumann-Neumann Preconditioner

Fortunately, a coloring of the subdomains is not required with the *Neumann-Neumann (NN) preconditioner* [14, 85, 42, 47, 49]. Here, all Schur complement inverses are involved equally:

$$P_{NN}^{-1} := \left(\sum_i R_i^{\top} D^{(i)} S^{(i)^{-1}} D^{(i)} R_i\right) , \tag{3.63}$$

where $D^{(i)}$ are diagonal weighting matrices to be focused on below. That is, the NN preconditioner requires solving local Neumann problems on each of the subdomain. However, since this can be done in parallel, the run-time is more or less equal to that of the DN preconditioner. In addition, the NN preconditioner is more robust against coefficient jumps of A across subdomain boundaries and applies to partitions where a red-black coloring is not possible.

The $D^{(i)}$ denote weighting matrices, which, in general, are chosen such that they give a partition of unity, that is

$$\sum_i R_i^{\top} D_i R_i = I \, . \tag{3.64}$$

In most cases, their diagonal entries are determined by the pseudo-inverses of the counting functions $\nu_i(x) \in V(\Gamma_i), i = 1, \ldots, N$, the latter giving the number of subdomains a point x on Γ is a boundary point of. To be concrete, the $\nu_i(x)$ can be defined as:

$$\nu_i(x) := \begin{cases} |N_x| & x \in \partial\Omega_i \cap \partial\Omega \\ 1 & x \in (\partial\Omega_i \cap \partial\Omega) \setminus \Gamma \\ 0 & x \in \Gamma \setminus \partial\Omega_i \end{cases} \quad i = 1, \ldots, N \, , \tag{3.65}$$

where N_x denotes the indices of those subdomains, whose boundaries contain x, and $|N_x|$ refers to the number of such. Subsequently, the pseudo-inverses $\nu_i^{\dagger}(x)$, which again give the diagonal entries of the $D^{(i)}$, read

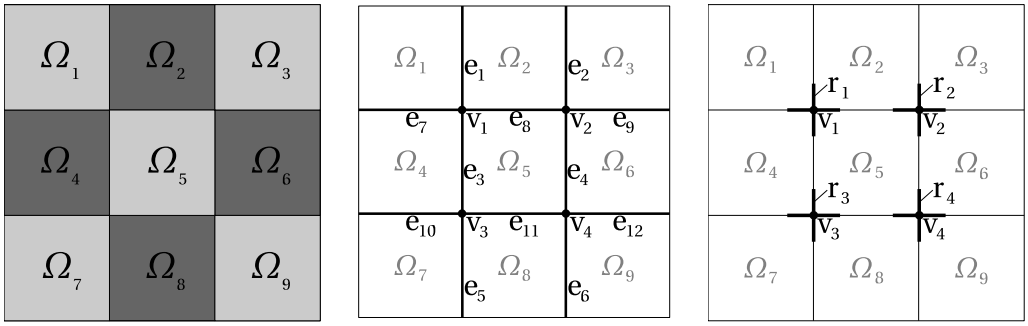
$$\nu_i^{\dagger}(x) = \begin{cases} \nu_i^{-1}(x) & x \in \partial\Omega_i \\ 0 & x \in (\Gamma \cup \partial\Omega) \setminus \partial\Omega_i \end{cases} \, . \tag{3.66}$$

However, in case of heavily varying coefficients in the original operator matrix A across subdomain boundaries, which cause the convergence of the Dirichlet-Neumann method to deteriorate, $|N_x|$ in (3.65) will be replaced by a sum of weightening functions, $\sum_{j \in N_x} \rho_j^\chi$, in order to compensate for that hindrance. See, e.g. [119, 120] for further details.

Both for the Neumann-Neumann as well as Dirichlet-Neumann preconditioned operator matrix S, the condition number is known to be bounded according to

$$\kappa(P^{-1}S) \leq C \frac{1}{H^2} \left(1 + \log \frac{H}{h}\right)^2 \tag{3.67}$$

where $P^{-1} = P_{NN}^{-1}$ or $P^{-1} = P_{DN}^{-1}$, respectively, and C being a positive, scalar constant. This reveals that the conditioning of $P^{-1}S$ deteriorates significantly with decreasing subdomain sizes H, i.e. increasing number of subdomains. An explanation is given by the error propagation argument: Since P_{BJ} couples only the unknowns at adjacent subdomain boundaries, a residual error propagates only with the speed of one subdomain per iteration. Consequently, the necessary number of iterations must be at least $1/H$. Therefore, the parallel scalability of a Krylov subspace method utilizing NN or DN preconditioning is quite limited. This will be overcome by the introduction of a second, coarse-level preconditioner, to be presented later.



(a) Partition with red-black coloring as employed with the Dirichlet-Neumann preconditioner.

(b) Dividing Γ further into edges and vertices as utilized with the Block-Jacobi preconditioner.

(c) Defining local regions around the vertices gives the basis for the Vertex Space preconditioner.

Figure 3.2: **Exemplary partitions used with the different substructuring methods.**

3.2.2.3 The Block-Jacobi Preconditioner

As opposed to the Neumann-Neumann preconditioner, the *Block Jacobi (BJ) preconditioner* is built upon a more geometrical decomposition of S. Here, the Schur

complement equation is permuted according to the edges and vertices of Γ, such that S can be written as

$$S = \begin{pmatrix} S_{E_1E_1} & \dots & S_{E_1E_m} & S_{E_1V} \\ S_{E_2E_1} & \dots & S_{E_2E_m} & S_{E_2V} \\ \vdots & \ddots & \vdots & \vdots \\ S_{E_mE_1} & \dots & S_{E_mE_m} & S_{E_mV} \\ S_{VE_1} & \dots & S_{VE_m} & S_{VV} \end{pmatrix}, \tag{3.68}$$

where the subscript E_iE_j refers to the edge on the boundary between subdomain Ω_i and Ω_j, and V to the set of all vertices. See Figure 3.2(b) for an illustrative example. Note that $S_{E_iE_j}$ is a zero-matrix if $\partial\Omega_i$ and $\partial\Omega_j$ have no point in common.

Neglecting the couplings between the edges and vertices of adjacent subdomains then gives the basis for the preconditioner

$$P_{BJ}^{-1} := \begin{pmatrix} S_{E_1E_1}^{-1} & 0 & & \dots & 0 \\ 0 & S_{E_2E_2}^{-1} & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & S_{E_{N_E}E_{N_E}}^{-1} & 0 \\ 0 & \dots & & 0 & \tilde{S}_{VV}^{-1} \end{pmatrix} \tag{3.69}$$

where $\tilde{S}_{VV}$ not only contains the diagonal of S_{VV}, but also the couplings of adjacent vertices.

By defining $R_{E_i}^\top$ as the extension by zero from the local edge Γ_{E_i} to Γ, and similarly $R_V^\top$ as that from Γ_V to Γ, (3.69) can be compactly written as

$$P_J^{-1} = \sum_{i=1}^{m} R_{E_i}^\top S_{E_iE_i}^{-1} R_{E_i} + R_V^\top \tilde{S}_V^{-1} R_V \ . \tag{3.70}$$

Again, the $S_{E_iE_i}^{-1}$ are never computed explicitly. Since the inverses of the Schur complements can be written as

$$S_{E_iE_i}^{-1} = \begin{pmatrix} 0 & I \end{pmatrix} \begin{pmatrix} A_{I_{ij}I_{ij}} & A_{I_{ij}E_i} \\ A_{I_{ij}E_i}^\top & A_{E_iE_i} \end{pmatrix} \begin{pmatrix} 0 \\ I \end{pmatrix}^{-1} , \tag{3.71}$$

with I_{ij} referring to the interior nodes of Ω_i and Ω_j, the action of $S_{E_iE_i}^{-1}$ corresponds to solving a local Neumann problem on $\Omega_i \cup \Omega_j \cup E_{ij}$.

On the other hand, due to $A_{VV} = R_V A R_V^\top$ being spectrally equivalent to S_{VV}, e.g. [112], A_{VV}^{-1} is commonly used to approximate S_{VV}^{-1}.

As with the previous preconditioners, the condition number of the BJ preconditioner P_{BJ}^{-1} also worsens for increasing number of subdomains, which is due to

neglecting the couplings between subdomains. In particular, the upper bound (3.67) applies also, [16, 44]. In addition, coefficient jumps between subdomains are known to have a much stronger negative influence on the condition number κ.

3.2.3 Two-Level Preconditioners

Similar to the one-level overlapping methods (cf. Sec. 4.1), also iterative substructuring methods exhibit a limited parallel scalability, which is due to the to deterioration of condition number for an increasing number of subdomains. In particular, the latter fact is reflected by the factor $\frac{1}{H^2}$ in the upper bound of the conditioner numbers, (3.67).

As in the overlapping case this is overcome by augmenting the existing preconditioner by a coarse-grid preconditioning, which leads to an crude error propagation throughout all unknowns on Γ for every iteration. However, as opposed to the overlapping two-level methods, the involved coarse problems here are of much smaller size in relation to those on the fine grid. Usually, they consist of only one unknown per subdomain. Thereby, almost optimal preconditioners are constructed, in the sense that the condition number bounds are almost independent of the number of subdomains and thus from the subdomain mesh size H.

We will start with two such approaches which extend the Block Jacobi preconditioner, and then move on to the extension of the Neumann-Neumman preconditioner, which will lead us to the widely used *Balancing Neumann-Neumann (BNN) preconditioner*.

All of the following so-called *two-level preconditioners* yield condition numbers, which do not deteriorate with decreasing H. In particular, the following upper limit will hold:

$$\kappa(P^{-1}S) \leq C \left(1 + \log \frac{H}{h}\right)^2 \tag{3.72}$$

for a positive constant C, which only depends on the relative subdomain size $\frac{H}{h}$. Consequently, the number of PCG iterations in relation to a given error threshold remains almost constant, if the number of subdomains is increased while fixing the subdomain size. In the alternative case that the subdomain sizes shrink, because of keeping the total problem size constant, (3.72) even decreases slowly, due to the finer coarse-grid couplings. On the other hand however, the coarse systems to be solved provide no clues for coarse-grained parallelization, and thus are usually computed sequentially on one central processing node, which increases the computation and communication time by some fixed costs per iteration significantly. Thus, two-level preconditioners provide a much better scalability than their one-level counterparts, although at additional fixed computational costs.

3.2.3.1 The Bramble-Pasciak-Schatz Preconditioner

One of first two-level preconditioners was proposed by Bramble et al. in 1986 [16], whose main idea is to replace the purely vertices-based submatrix $\tilde{S}_{VV}$ in (3.69) by a coarse variant of the original stiffness matrix A, giving the so-called *Bramble-Pasciak-Schatz (BPS) preconditioner*.

Let A_H be the stiffness matrix of a finite element discretization of (2.11) on the coarse triangulation $\mathcal{T}^H$. Furthermore, let $R_H^\top$ be the *weighted* extension matrix interpolating the nodal variables on $\mathcal{T}^H$ to those on $\mathcal{T}^h_{|\Gamma}$. The BPS preconditioner then reads:

$$P_{BPS}^{-1} := \sum_{i=1}^{m} R_{E_i}^\top \tilde{S}_{E_i E_i}^{-1} R_{E_i} + R_H^\top A_H^{-1} \hat{R}_H \ . \tag{3.73}$$

That is, in contrast to the Block Jacobi preconditioner (3.70) the solely vertices-based term $R_V^\top S_{VV}^{-1} R_V$ is replaced by the matrix $R_H^T A_H R_H$, which yields a coarse coupling of all unknowns on Γ. Thereby, a global error propagation is realized for every application of P_{BPS}.

In particular, it is known that (3.72) holds for $P^{-1} = P_{BPS}^{-1} S$ in two dimensions. In the three-dimensional case[1] (3.73) does not hold, for which reason the BPS preconditioner has been further developed to the *Wire Basket preconditioners* [110, 111, 18], where the coarse problem is build upon the wire basket, i.e. the union of edges and vertices, in order to reach the same upper limit (3.72) in three dimensions as well.

3.2.3.2 The Vertex Space Preconditioner

In aiming to make the condition number independent of the relative subdomain sizes also, i.e. abolishing the term $\frac{H}{h}$, couplings in local regions around the vertices at Γ are considered in addition.

For each $i = 1, \ldots, N_V$, let Γ_{δ_i} refer to the union of the vertex node Γ_{V_i} and all those nearby nodes on Γ, which have a maximum distance of δ to Γ_{V_i}. See Fig. 3.2(c) for an example. Furthermore, let $R_{\delta_i}^\top, i = 1, \ldots, N_V$ be matrix which extend by zero a nodal vector from Γ_{δ_i} to Γ. Then, by $S_{\delta_i} := R_{\delta_i}^\top S R_{\delta_i}^\top$ we denote the Schur complement with respect to each such vertex region. In making use of these components, the *Vertex Space (VS) preconditioner* [113, 114, 10, 64] reads as

[1] There, the coarse problem is defined on the vertices also, but is coupled to variables on common faces and edges.

follows:

$$P_{VS} := \sum_{i=1}^{m} R_{E_i}^{\top} \tilde{S}_{E_i E_i}^{-1} R_{E_i} + R_H^{\top} A_H^{-1} R_H + \sum_{i=1}^{N_V} R_{\delta_i}^{\top} S_{\delta_i \delta_i}^{-1} R_{\delta_i} \ , \tag{3.74}$$

which extends P_{BPS} additively. Because of these additional terms, the residual errors can propagated faster across the subdomains, and it is known for the conditioner number to be limited according to, [48, 114]

$$\kappa(P_{VS}^{-1} S) \leq C \left(1 + \log \frac{H}{\delta}\right)^2 \ , \tag{3.75}$$

for relatively small coefficient discontinuities between the subdomains and

$$\kappa(P_{VS}^{-1} S) \leq C \left(1 + \log \frac{H}{h}\right) \ , \tag{3.76}$$

at the presence of significant coefficient jumps. That is, for choosing an overlap extent $\delta > h$, the iteration number can be further diminished in comparison to the BPS preconditioner. On the other hand, the computational costs per iteration are higher.

Typically, the action of $S_{\delta_i \delta_i}^{-1}$ is computed by the same approach as used for $S_{E_i E_i}$ in Section 3.2.2.3, where the submatrix $A^{(i)}$ here is the restriction of A to the union of those subdomains, i.e. edges and vertices, whose closures do intersect with Γ_{δ_i}. Alternatively, the employment of probing preconditioners has also been found satisfactory, [33, 32].

In the three dimensional case [64], which is not in the main focus in this work, edge regions and their corresponding Schur complements are introduced in addition to the vertex ones. Similar to the latter, an edge region comprises the nodes of an edge Γ_{E_i} and all those nodes on adjacent faces, which have a maximum distance of δ from Γ_{E_i}. In doing so, (3.75) and (3.76) also holds for a three-dimensional setting.

3.2.3.3 The Balancing Neumann-Neumann Method

Motivation and Approach. As opposed to the previous approaches which define a coarse grid preconditioner upon a geometrical decomposition of S, in the following approach the coarse space was originally motivated from the joint, low-dimensional null spaces of the local Schur complements $S^{(i)}$. The latter are present for certain model problems, such as the Poisson problem, cf. Section 3.1.2.2. Since for our model problem those local null spaces are all empty, i.e. all $A^{(i)}$ and $S^{(i)}$ are invertible, we will consider the Poisson problem for the time being, in order to better explain the original motivation. Later on, we will generalize the findings to our model problem.

With the Poisson problem (3.45) as the problem to be decomposed, the solution to the corresponding local Schur complement equations

$$S^{(i)}u^{(i)} = r^{(i)} , \qquad i = 1, \ldots, N , \tag{3.77}$$

on *inner* subdomains, at the presence of Neumann boundary conditions only, are only defined up to a constant. Let the indices of those subdomains be referred to by N_π in the following. There, $S^{(i)^{-1}}$ is usually approximated by the *Moore-Penrose pseudo-inverse*, [63], $S^{(i)^\dagger}$.

Moreover, in order to assure *any* solution to (3.77) exist, it is required the right-hand side $r^{(i)}$ to lie within the range of $S^{(i)}$. The other way round, in case that S is symmetric, which is the case here, one can equivalently enforce the projection of $r^{(i)}$ into the null space of $S^{(i)}$ to vanish:

$$\left\langle v, r^{(i)} \right\rangle = 0, \quad \forall v \in \text{kernel}\big(S^{(i)}\big), \quad \forall i \in N_\pi \tag{3.78}$$

i.e. to be L^2-orthogonal to the null space[2] of $S^{(i)}$. This is usually realized by defining matrices $Z^{(i)}$ whose columns span spaces which include the corresponding null spaces, i.e.

$$\text{kernel}\big(S^{(i)}\big) \subset \text{range}\big(Z^{(i)}\big), \qquad \forall i \in N_\pi , \tag{3.79}$$

and to then require

$$Z^{(i)^\top} r^{(i)} = 0 , \qquad \forall i \in N_\pi \tag{3.80}$$

to hold.

Based on that, the (low-dimensional) space of (weighted) non-admissible right-hand sides vectors is defined by

$$V_{r_0} := \left\{ w \in V \ : \ w = \sum_{i \in \pi} R^{(i)^\top} D^{(i)} u_i, \quad u_i \in \text{range}\big(Z^{(i)}\big) \right\} , \tag{3.81}$$

and let

$$R_0 := \begin{pmatrix} Z^{(\pi_1)^\top} D^{(\pi_1)} R^{(\pi_1)} & Z^{(\pi_1 - 1)^\top} D^{(\pi_1 - 1)} R^{(\pi_1 - 1)} & \ldots \end{pmatrix} \tag{3.82}$$

be the corresponding projection from $V(\Gamma)$ into $V_{r_0}(\Gamma)$. For the Poisson problem for example, V_{r_0} consists of vectors which are piecewise constant on the interior

[2] Note that in the case of symmetry $\text{range}\big(S^{(i)^\perp}\big) = \text{kernel}\big(S^{(i)^\top}\big) = \text{kernel}\big(S^{(i)}\big)$.

subdomains (whose corresponding local problems have artificial Neumann boundary conditions only) and zero else.

In order the global right-hand side vector r to lie in the orthogonal space of V_{r_0}, a possible component within V_{r_0} is removed by

$$\tilde{r} = \big(I - S R_0^\top S_0^{-1} R_0\big) r \tag{3.83}$$

prior to applying the NN preconditioner, where S_0 is defined as the projection of S into V_{r_0}, i.e. $S_0 := R_0^\top S R_0$. This so-called *balancing* step is applied to the result of the NN preconditioning,

$$\tilde{u} = \Big(\sum_i R_i^\top D^{(i)} S^{(i)^\dagger} D^{(i)} R_i\Big) \tilde{r} \;, \tag{3.84}$$

again,

$$\tilde{y} \leftarrow \big(I - R_0^\top S_0^{-1} R_0 S\big) \tilde{u} \;, \tag{3.85}$$

thereby making the preconditioner symmetric.

The Algorithm. Integrating (3.83), (3.84) and (3.85) into a PCG iteration gives Algorithm 3, see, e.g. [112]. Note that these steps can also be written as one expression:

$$P_{BNN}^{-1} := \big(I - R_0^\top S_0^{-1} R_0 S\big)\Big(\sum_i R_i^\top D^{(i)} S^{(i)^\dagger} D^{(i)} R_i\Big)\big(I - S R_0^\top S_0^{-1} R_0\big) + R_0^\top S_0^{-1} R_0 \;, \tag{3.86}$$

which is referred to as the *Balancing Neumann-Neumann* preconditioner.

In comparison to NN preconditioner, the two-level variant requires two additional applications of S, as well as three times that of $R_0^\top S_0^{-1} R_0$. Whereas the computation of each the former is exactly the same as with the previous methods, the inversion of S_0 provides no clues for coarse-grained parallelization. Thus it needs to be carried out on a central processing node which requires additional collecting and spreading communication steps (represented here by the operators R_0 and $R_0^\top$, respectively) for each occurrence of $R_0^\top S_0^{-1} R_0$. Computation costs w.r.t. to the inversion itself are practically neglectible, since S_0 is very small. Typically, S_0 is calculated explicitly or factorized prior to the iteration.

Algorithm 3: PCG iteration with Balancing Neumann-Neumann preconditioning

initialize u^0 with an arbitrary value

$r^0 \leftarrow \chi - Su^0$

iterate $k = 1, 2, \ldots$ **until** convergence

$$\begin{aligned}
\tilde{r}^{k-1} &\leftarrow \left(I - SR_0^\top S_0^{-1} R_0\right) r^{k-1} \\
\hat{r}^{k-1} &\leftarrow \left(\sum_{i=1}^{N} R_i^\top D^{(i)} S^{(i)^\dagger} D^{(i)} R_i\right) \tilde{r}^{k-1} \\
y^{k-1} &\leftarrow \left(I - R_0^\top S_0^{-1} R_0 S\right) \hat{r}^{k-1} \\
\beta^k &\leftarrow \frac{y^{k-1^\top} q^{k-1}}{y^{k-2^\top} q^{k-2}} && [\beta^1 = 0] \\
p^k &\leftarrow y^{k-1} + \beta^k p^{k-1} && [p^1 = q^0] \\
\alpha^k &\leftarrow \frac{y^{k-1^\top} q^{k-1}}{p^{k^\top} S p^k} \\
\lambda^k &\leftarrow \lambda^{k-1} + \alpha^k p^k \\
r^k &\leftarrow r^{k-1} - \alpha S p^k
\end{aligned}$$

The second role of S_0^{-1}. Besides guaranteeing well-posedness and uniqueness, the balancing steps lead to a global yet coarse information propagation. In particular, $R_0^\top S_0^{-1} R_0$ plays the role of a coarse global operator there. This becomes clear when considering the coarse space V_{r_0} in case of the Poisson problem as model problem. As explained above, there the local null spaces of interior subproblems consist of constant functions and thus the coarse space is of the form

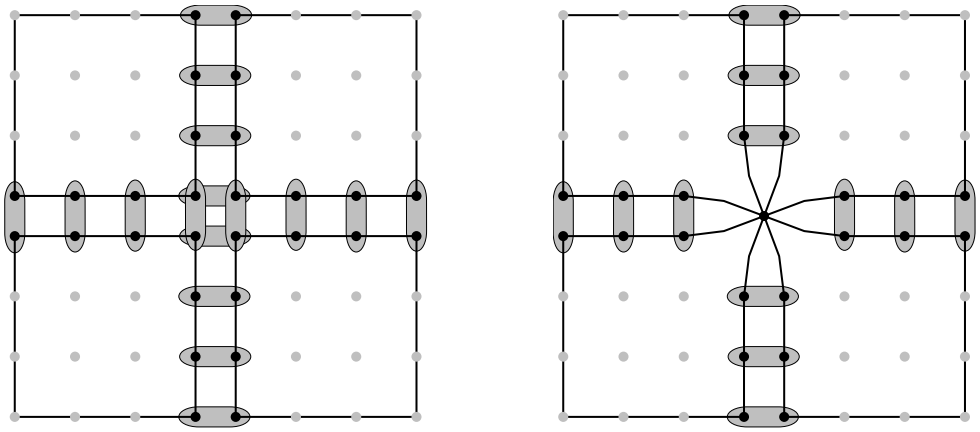
$$V_{r_0} = \operatorname{span}\left\{ R^{(i)^\top} \nu_i^\dagger \ : \ \operatorname{kernel}\left(S^{(i)}\right) \neq \emptyset \right\} , \tag{3.87}$$

with the weights $\nu_i^\dagger$ as defined in (3.66). That is, the action of R_0 is to compute a weighted sum for each inner subdomain and S_0 gives a global coupling of all such sums according to a coarse representation of S. Therefore, the balancing steps lead to a global, yet coarse, error propagation throughout all unknowns on Γ. Thereby, the only local error propagation property of the NN preconditioner is compensated for. In fact, the condition number of $P_{BNN}^{-1} S$ is known to be bounded according to (3.72) also, and in addition to be independent of strong coefficient jumps across

subdomain boundaries. Hence, P_{BNN} is close to an optimal preconditioner and in comparison to the BPS and Vertex Space preconditioners the explicit knowledge about the image partition geometry is much less.

Empty null spaces. Let us return to our model problem given in Section 2.4. There, V_{r_0} is empty, since none of the local Schur complements $S^{(i)}$ are singular, cf. Section 3.1.2.2. However, following a similar suggestion of Toselli et al. [120, 119], own experiments have shown that assuming the coarse space (3.87) and its corresponding operators R_0 and S_0 with the motion estimation problem yields the same advantageous convergence behavior as for the Poisson problem.

3.2.4 Finite Element Tearing and Interconnection Methods



(a) Completely torn mesh as used with the dual FETI method

(b) Partially torn mesh as used with the primal-dual FETI method

Figure 3.3: **Examples of torn meshes as used with the dual and primal-dual FETI methods.** Gray patches indicate necessary equality constraints in the underlying energy minimization problems.

3.2.4.1 The One-Level FETI Method

Approach. Substructuring approaches presented so far aimed at solving for the global vector u_Γ in (3.43) by PCG iteration. In contrast, the following methods consider the local problem $S^{(i)}u^{(i)} = \chi^{(i)},\ i = 1, \ldots, N$ *separately* — yielding a duplication of unknowns from the view of the global problem — and enforce equality across the subdomains boundaries by additional constraints. See Figure 3.3(a) for an illustration. This gives the outline of the so-called *Finite Element Tearing and*

Interconnection (FETI) method [54, 55, 117] which can be seen as a *dual method* of BNN preconditioning and has gained strong interest especially in the area of computational mechanics.

In order to re-establish continuity across the subdomain boundaries, the approach is to restate the local problems $S^{(i)}u^{(i)} = \chi^{(i)}$ as separate optimization problems first, which are then coupled by additional constraints enforcing equality of the duplicated unknowns.

First, we will make use of the following definitions:

$$\hat{S} := \begin{pmatrix} S^{(1)} & 0 & \dots & 0 \\ 0 & S^{(2)} & \ddots & \vdots \\ \dots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & S^{(N)} \end{pmatrix}, \quad \hat{u}_\Gamma := \begin{pmatrix} u_\Gamma^{(1)} \\ \vdots \\ u_\Gamma^{(N)} \end{pmatrix}, \quad \hat{\chi} := \begin{pmatrix} \chi^{(1)} \\ \vdots \\ \chi^{(N)} \end{pmatrix} \tag{3.88}$$

by which the N local problems can be written as

$$\hat{S}\hat{u} = \hat{\chi} \, . \tag{3.89}$$

Note that common variables in the global problem (3.43) have been split up into several local ones. Hence, $\hat{u}$ has more components than does u_Γ.

Subsequently, the inter-connection of the thus duplicated variables is realized by equality constraints

$$B_\Gamma \hat{u}_\Gamma = 0 \, , \tag{3.90}$$

with the matrix B_Γ having a row for each pair of variables to be equal, and each row consisting of exactly one 1 and one -1, respectively, at the columns corresponding to each variable pair and zero else.

Now, the purpose of the FETI method is to formulate (3.89) as an optimization problem which is constrained by the equality conditions stated in (3.90), which reads

$$\begin{cases} J(\hat{u}_\Gamma) := \frac{1}{2}\hat{u}_\Gamma^\top \hat{S}\hat{u}_\Gamma - \hat{\chi}^\top \hat{u}_\Gamma \to \min \\ B_\Gamma \hat{u}_\Gamma = 0 \end{cases} . \tag{3.91}$$

In a next step, the equality constraints are relaxed by a Lagrangian approach. That is, by introducing the multiplier variables $\lambda \in \mathbb{R}^{N_B}$, with N_B denoting the number of equality equations in (3.90), we define the Lagrangian as

$$\mathcal{L}(\hat{u}_\Gamma, \lambda) := \frac{1}{2}\hat{u}_\Gamma^\top \hat{S}\hat{u}_\Gamma - \hat{\chi}^\top \hat{u}_\Gamma + \lambda^\top B_\Gamma \hat{u}_\Gamma \, . \tag{3.92}$$

by which the original problem (3.91) can be rewritten as the saddle-point problem

$$\inf_{\hat{u}_\Gamma} \max_{\lambda} \mathcal{L}(\hat{u}_\Gamma, \lambda) \tag{3.93}$$

a solution $(\hat{u}_\Gamma, \lambda)$ to which amounts to solving the following (static) equilibrium equations

$$\begin{cases} \hat{S}\hat{u}_\Gamma + B_\Gamma^\top \lambda = \hat{\chi} \\ B_\Gamma \hat{u}_\Gamma = 0 \end{cases} . \tag{3.94}$$

Algebraically solving the first equation for $\hat{u}_\Gamma$ then gives

$$\hat{u}_\Gamma = \hat{S}^{(-1)}(\hat{\chi} - B_\Gamma^\top \lambda) \, , \tag{3.95}$$

while $\hat{S}^{(-1)}$ has block structure also and consists of the local inverses ${S^{(i)}}^{-1}$. However, central to FETI methods is the assumption that some of the $S^{(i)}$ are *not* invertible, which we will assume for the time being. Also for the time being, we consider the Poisson problem as model problem, as with the explanation of the Balancing NN preconditioning above. In that case, the non-existing ${S^{(i)}}^{-1}$ in $\hat{S}^{-1}$ are replaced by any suitable pseudo-inverse ${S^{(i)}}^\dagger$ giving the global pseudo-inverse $\hat{S}^+$ and (3.95) can be rewritten as

$$\hat{u}_\Gamma = \hat{S}^+(\hat{\chi} - B_\Gamma^\top \lambda) - Z\alpha \, , \tag{3.96}$$

with $Z\alpha$ giving a null space component of the solution, i.e. piecewise constant vectors on interior subdomains in case of the Poisson model problem. In particular, Z is made of the local null space-spanning matrices $\{Z^{(i)} \mid S^{(i)} \text{ singular}\}$ and we have that $\text{range}\big(Z\big) = \text{kernel}\big(\hat{S}\big)$.

Furthermore, the pseudo-inverse in (3.96) can only be applied if $\hat{u}_\Gamma$ in (3.96) is perpendicular to the null space of $\hat{S}$, i.e.

$$\hat{\chi} - B_\Gamma^\top \lambda \perp \text{kernel}\big(\hat{S}\big) \tag{3.97}$$

with $\perp$ denoting orthogonality with respect to the euclidian 2-norm. Due to the definition of Z, (3.97) can be equivalently written as

$$Z^\top(\hat{\chi} - B_\Gamma^\top \lambda) = 0 \, . \tag{3.98}$$

In a next step, the expression for $\hat{u}$ in (3.96) is inserted into the second equation of (3.94), which, after a minor modification, results in an equation

$$B_\Gamma \hat{S}^+ B_\Gamma^\top \lambda = B_\Gamma \hat{S}^+ \chi_\Gamma - B_\Gamma Z\alpha \, , \tag{3.99}$$

where u_Γ as been eliminated. By utilizing the abbreviating notation $F := B_\Gamma \hat{S}^+ B_\Gamma^\top$, $G^\top := Z^\top B_\Gamma^\top$ and $d := B_\Gamma \hat{S}^+ \hat{\chi}$ this can be compactly written as

$$F\lambda + G\alpha = d \; . \tag{3.100}$$

In combination with the necessary condition (3.98), which is rewritten using aforementioned abbreviations as well as $e := Z^\top \hat{\chi}$, we obtain the system

$$\begin{cases} F\lambda + G\alpha = d \\ \qquad G^\top \lambda = e \end{cases} , \tag{3.101}$$

where the original variables $\hat{u}_\Gamma$ being sought for have been eliminated.

Because problem (3.91) is convex, (3.101) can be derived alternatively by considering the dual problem

$$\max_\lambda \mathcal{C}(\lambda) := \max_\lambda \inf_u \mathcal{L}(u, \lambda) \; , \tag{3.102}$$

see, e.g. [55, 117]. For that reason, $\hat{u}_\Gamma$ and λ are commonly called primal and dual (or multiplier) variables.

The Algorithm. With the FETI method, the dual vector λ in (3.101) first is solved for, which is done in the null space $\text{kernel}\big(G^\top\big)$ in order to be independent from α. To be precise, this is realized by the projection

$$P := I - G(G^\top G)^{-1} G^\top \; , \tag{3.103}$$

which is chosen such that $\text{range}\big(P\big) = \text{kernel}\big(G^\top\big)$. Multiplying the first equation (3.101) by $P^\top$ then simplifies the system to

$$P^\top F\lambda + G\alpha = P^\top d \; . \tag{3.104}$$

Subsequently, the solving procedure is a PCG iteration applied to (3.104), where the preconditioning step involves a projection by P before and by $P^\top$ afterwards, respectively, see Algorithm 4. There, $\tilde{M}^{-1}$ denotes a generic preconditioner, on which we will detail later on, and we have that $Q = I$. Note that because of the specific choice of the initial value, the second equation in (3.101) is fulfilled from the beginning. Parallelization can be employed in the in the computation of $\hat{S}^\dagger$, which is contained in F, i.e. by solving the actions of the local pseudo-inverses $S^{(i)^\dagger}$ and $S^{(i)^{-1}}$, respectively, on different processing nodes.

Furthermore, P and $P^\top$ not only play the role of projections that are Q-orthogonal to $\text{range}\big(G^\top\big)$ and $\text{range}\big(G\big)$, respectively, but also that of a global, coarse error propagating step [93]. In particular P and $P^\top$ can be seen as the dual counterparts of

Algorithm 4: PCG for the FETI problem iteration using an augmented preconditioning step and a generic preconditioner $\tilde{M}^{-1}$

$$\lambda^0 \leftarrow QG\big(G^\top QG\big)^{-1}e$$
$$r^0 \leftarrow d - F\lambda^0$$

iterate $k = 1, 2, \ldots$ **until** convergence

$$\begin{aligned}
\textit{Project:} \quad & q^{k-1} \leftarrow Pr^{k-1} \\
\textit{Precondition:} \quad & z^{k-1} \leftarrow \tilde{M}^{-1}q^{k-1} \\
\textit{Project:} \quad & y^{k-1} \leftarrow P^\top z^{k-1} \\
& \beta^k \leftarrow \frac{{y^{k-1}}^\top q^{k-1}}{{y^{k-2}}^\top q^{k-2}} \qquad [\beta^1 = 0] \\
& p^k \leftarrow y^{k-1} + \beta^k p^{k-1} \quad [p^1 = q^0] \\
& \alpha^k \leftarrow \frac{{y^{k-1}}^\top q^{k-1}}{{p^k}^\top F p^k} \\
& \lambda^k \leftarrow \lambda^{k-1} + \alpha^k p^k \\
& r^k \leftarrow r^{k-1} - \alpha F p^k
\end{aligned}$$

the balancing terms (3.83) and (3.85) of the primal BNN method, respectively. Unlike S_0 however, $(G^\top G)^{-1}$ is not a coarse version of the fine operator F here, but also yields a global coarse coupling during iteration. In particular, the bounds given in (3.72) hold for the preconditioner $P\tilde{M}^{-1}P^\top F$ also, see [94, 117, 77, 78].

The most standard preconditioner is given by

$$\tilde{M}^{-1} := B_\Gamma \hat{S} B_\Gamma^\top = \sum_{i=1}^{N} B_\Gamma^{(i)} \hat{S}^{(i)} {B_\Gamma^{(i)}}^\top \ , \tag{3.105}$$

which is denoted by the *FETI Dirichlet preconditioner* [54, 93]. Note that (3.105) mainly amounts to solving N local Dirichlet problems in parallel, cf. Section 3.1.1.2. See, e.g. [102] for alternative preconditioners.

In addition, (3.103) is the most simple choice too. In general, the projection is of the form

$$P := I - QG(G^\top QG)^{-1}G^\top \ . \tag{3.106}$$

where Q is a symmetric, positive definite scaling matrix accounting for possible coefficient jumps in A across the subdomain boundaries. Thus, $Q = \tilde{M}^{-1}$ is a common choice. In terms of implementing $(G^\top QG)^{-1}$, it is common to factorize $G^\top QG$ in the initialization, e.g. by Cholesky decomposition, in order to keep computation costs low during iteration when it is applied.

Once a solution λ has been computed, with respect to a given error tolerance, the null space component α is determined according to

$$\alpha = (G^\top G)^{-1} G^\top Q(d - F\lambda) \,, \tag{3.107}$$

by which the complete solution to the dual problem (3.101) is obtained. Finally, the primal solution u_Γ is calculated according to equation (3.95). Also here, direct clues for coarse-grained parallelization exist.

Empty Null spaces. Unlike with the BNN method, the algorithms needs to be changed for the of case of non-singular local matrices $A^{(i)}$ and therefore empty null spaces $\text{kernel}\big(S^{(i)}\big)$. Similarly, applying the above approach to model problems exhibiting this property would yield an empty matrix Z and $P = I$. Thus, global error propagation would not be present by the above approach. As a remedy, it is shown [50, 119] that one can define an artificial coarse space $\text{range}\big(Z\big)$, if the columns vectors of Z are chosen such that at least the following assumption[3] holds:

$$\text{range}\big(Z\big) \cap \text{kernel}\big(B_\Gamma\big) = \emptyset \,. \tag{3.108}$$

If so, $G^\top$ is defined as above: $G^\top = Z^\top B_\Gamma^\top$. Subsequently, P is chosen to be

$$P := I - G(G^\top FG)^{-1} G^\top F \,, \tag{3.109}$$

that is, P is built upon the operator F, unlike the original definition in (3.106) where Q is some scaling matrix. Furthermore, the initialization of the iteration needs to be modified to:

Initialize

$$\begin{aligned} \lambda^0 &\leftarrow G\big(G^\top FG\big)^{-1} F^\top d \\ r^0 &\leftarrow d - F\lambda^0 \end{aligned} \,, \tag{3.110}$$

see [50]. Thereby, one obtains a modified one-level FETI method with the same convergence properties as above, and therefore it becomes applicable to our original model problem as well.

[3] for further details we refer to [119]

The Second Function of P and Similarities to the BNN Method. Interestingly, P not only has the function of ensuring every update to lye in the null space of $G^\top$, but can also be seen as a coarse preconditioner.

In order to elucidate this fact, let us take a closer look at its definition (3.106). Note that the columns of the product $G^\top = Z^\top B_\Gamma^\top$ consist of the rows of B_Γ having been projected into the null space of $\hat{S}$. By definition, B_Δ is of size $|\Gamma| \times |\Gamma|$. In contrary, since almost always $\text{kernel}\big(\hat{S}\big)$ is of much smaller dimension than $\text{range}\big(\hat{S}\big)$, $G^\top$ has much fewer rows than columns. Consequently, the product $G^\top QG$ is of much smaller size than the operator matrix F. Moreover, since $\big(G^\top QG\big)^{-1}$ is known to be dense, it realizes a global coupling of all unknowns in a coarser dual subspace when applied. Therefore, P not only has the function of projecting into the subspace $\text{kernel}\big(G\big)^\top$ and thus asserting the constraint (3.97), but also that of a global, coarse preconditioner having the effect of a global error propagation at every iteration [93].

Furthermore, one discovers similarities between Algorithm 4 and the Balancing Neumann-Neumann method presented in Sec. 3.2.3.3, since also with the latter the coarse-grid correction steps can be interpreted as projection steps. Thus, $\big(G^\top QG\big)^{-1}$ could be seen as the dual counterpart to S_C^{-1}, as well as the recurring projections $P = I - G(G^\top G)^{-1}G^\top$ to $\big(I - \hat{R}_C^\top S_C^{-1}\hat{R}_C S\big)$, and the initial projections $QG\big(G^\top QG\big)^{-1}$ in comparison to $\hat{R}_C^\top S_C^{-1}\hat{R}_C$ [120, 55]. However, unlike with the BNN method, the coarse problem as well as the projections here do not depend on the operator F, for which reason it is only a similarity in the algebraic structure. Note that, because of the latter observation, the pre-projection step with the FETI method cannot be left out.

3.2.4.2 The Dual-primal FETI Method

More recently, mixed dual-primal approaches have attracted considerable interest, motived both by the unsatisfying results of dual FETI method applied to problems of higher order, such as fourth-order plate and shell problems [53] in computational mechanics, as well as by the fact that the coarse operator of the original method, i.e. without the adaption explained in the preceding subsection, is dependent on the model problem's null space. The idea behind the so-called *Dual-Primal FETI (FETI-DP) methods* [51, 52, 95, 100, 75] is to partially reverse the process of tearing nodes, and thus the duplication of variables, which was the starting point of the one-level FETI method, such that a few variables in (3.43) remain global. Those degrees of freedom then are continuous across the subdomains by construction, thereby diminishing the number of constraints in comparison to the purely dual method. Finally, one reaches a primal subproblem in addition to the remaining dual one,

for which reason the FETI-DP method can be seen as intermediate between the purely dual one-level or two-level FETI methods and the purely primal Balancing Neumann-Neumann methods.

Approach. One of the standard cases with DP-FETI methods is to have primal variables at the crosspoints of Γ, i.e. at vertex points being elements of more than two subdomain boundaries. Thus, we will denote the duplicated nodes by Γ_E and the crosspoints by Γ_C, according to which we set up the following global-local Schur complement systems:

$$\begin{pmatrix} S_{EE}^{(1)} & 0 & \dots & 0 & S_{EC}^{(1)} \\ 0 & S_{EE}^{(2)} & \ddots & \vdots & S_{EC}^{(2)} \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \dots & 0 & S_{EE}^{(N)} & S_{EC}^{(N)} \\ S_{EC}^{(1)^\top} & S_{EC}^{(2)^\top} & \dots & S_{EC}^{(N)} & S_{CC} \end{pmatrix} \begin{pmatrix} u_E^{(1)} \\ u_E^{(2)} \\ \vdots \\ u_E^{(N)} \\ u_c \end{pmatrix} = \begin{pmatrix} \chi_E^{(1)} \\ \chi_E^{(2)} \\ \vdots \\ \chi_E^{(N)} \\ \chi_c \end{pmatrix}, \tag{3.111}$$

$$\Leftrightarrow \begin{pmatrix} S_{EE} & S_{EC} \\ S_{EC} & S_{CC} \end{pmatrix} \begin{pmatrix} u_E \\ u_C \end{pmatrix} = \begin{pmatrix} \chi_E \\ \chi_C \end{pmatrix} \quad \Leftrightarrow \quad \hat{S}\hat{u} = \hat{\chi} \tag{3.112, 3.113}$$

which, unlike equation (3.88), contains global dependencies in the unknowns at Γ_C. As a consequence, none of the local matrices

$$S^{(i)} = \begin{pmatrix} S_{EE}^{(i)} & S_{EC}^{(i)} \\ S_{EC}^{(i)^\top} & S_{CC}^{(i)} \end{pmatrix} \tag{3.114}$$

is singular. As a consequence, here will be no need for a projection as with the dual FETI method.

Subsequently, by algebraically eliminating for the global variables, in the same manner as for the degrees of freedom on $\bar{\Omega}\setminus\Gamma$ in Section 3.1.2, we obtain the reduced system

$$\underbrace{\left(S_{EE} - S_{EC}S_{CC}^{-1}S_{EC}^\top\right)}_{=:\tilde{S}_{EE}} = \underbrace{\chi_E - S_{CE}^\top S_{CC}^{-1}\chi_C}_{\tilde{\chi}_{EE}} . \tag{3.115}$$

Again, this is embedded into a minimization problem with additional constraints ensuring equality of the duplicated degrees of freedom:

$$\begin{cases} J(u_E) := \dfrac{1}{2}u_E^\top \tilde{S}_{EE}u_E - \tilde{\chi}_E^\top u_E \to \min \\ B_E u_E = 0 \end{cases} . \tag{3.116}$$

Note that unlike $\hat{S}$ in (3.91), here $\tilde{S}_{EE}$ is not purely block-diagonal, due to the term $S_{EC}S_{CC}^{-1}S_{EC}^{\top}$ in (3.115) realizing a global, coarse discretization of S on Γ_C.

As with the one-level FETI approach, problem (3.116) can be rewritten as a saddle-point problem with respect to $\tilde{u}_\Gamma$ and the Lagrange variables λ, which would result in the system of equations

$$\begin{cases} \tilde{S}\hat{u}_E + B_E^{\top}\lambda = \hat{\chi} \\ B_E\hat{u}_E = 0 \end{cases} . \tag{3.117}$$

However, in order to explain the final algorithm better we will take two steps back and reverse both the elimination for the unknowns at Γ_C as well as for those on $\bar{\Omega} \setminus \Gamma$ and consider the fully expanded system

$$\begin{pmatrix} A_{RR}^{(1)} & 0 & \dots & 0 & A_{RC}^{(1)} \\ 0 & A_{RR}^{(2)} & \ddots & \vdots & A_{RC}^{(2)} \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \dots & 0 & A_{RR}^{(N)} & A_{RC}^{(N)} \\ A_{RC}^{(1)^{\top}} & A_{RC}^{(2)^{\top}} & \dots & A_{RC}^{(N)^{\top}} & A_{CC} \end{pmatrix} \begin{pmatrix} u_R^{(1)} \\ u_R^{(2)} \\ \vdots \\ u_R^{(N)} \\ u_C \end{pmatrix} = \begin{pmatrix} f_R^{(1)} \\ f_R^{(2)} \\ \vdots \\ f_R^{(N)} \\ f_C \end{pmatrix} \tag{3.118}$$

$$\Leftrightarrow \quad \begin{pmatrix} A_{RR} & A_{RC} \\ A_{RC} & A_{CC} \end{pmatrix} \begin{pmatrix} u_R \\ u_C \end{pmatrix} = \begin{pmatrix} f_R \\ f_C \end{pmatrix} , \tag{3.119, 3.120}$$

where the subscript R refers to the local nodes, i.e. non-crosspoints here, and C to the global nodes, i.e. crosspoints in our case. The constrained minimization problem then reads

$$\begin{cases} J(u) := \frac{1}{2}\begin{pmatrix} u_R^{\top} & u_C^{\top} \end{pmatrix} \begin{pmatrix} A_{RR} & A_{RC} \\ A_{RC}^{\top} & A_{CC} \end{pmatrix} \begin{pmatrix} u_R \\ u_C \end{pmatrix} - \begin{pmatrix} f_R^{\top} & f_C \end{pmatrix} \begin{pmatrix} u_R \\ u_C \end{pmatrix} \to \min \\ \\ B_R u_R = 0 \end{cases} , \tag{3.121}$$

while B_R only affects duplicated variables on torn edge nodes, i.e.

$$B_R := \begin{pmatrix} 0 & 0 \\ 0 & B_E \end{pmatrix} . \tag{3.122}$$

As with the dual FETI method, (3.121) is then stated as saddle-point problem by introducing Lagrange multipliers λ, a solution to which is obtained by solving

the following equilibrium equations:

$$\begin{pmatrix} A_{RR} & A_{RC} & B_R^\top \\ A_{RC}^\top & A_{CC} & 0 \\ B_R & 0 & 0 \end{pmatrix} \begin{pmatrix} u_R \\ u_C \\ \lambda \end{pmatrix} = \begin{pmatrix} f_R \\ f_C \\ 0 \end{pmatrix} \tag{3.123}$$

which corresponds to (3.117).

Algebraically solving the first equation for u_R yields

$$u_R = A_{RR}^{-1}\big(f_R - B_R^\top \lambda - A_{RC} u_C\big) \,, \tag{3.124}$$

which, inserted into the second and third equation results in

$$\begin{pmatrix} B_R A_{RR}^{-1} B_R^\top \lambda & B_R A_{RR}^{-1} A_{RC} u_C \\ -A_{RC}^\top A_{RR}^{-1} B_R^\top \lambda & (A_{CC} - A_{RC}^\top A_{RR}^{-1} A_{RC}) \end{pmatrix} \begin{pmatrix} \lambda \\ u_C \end{pmatrix} = \begin{pmatrix} B_R A_{RR}^{-1} f_R \\ f_C - A_{RC}^\top A_{RR}^{-1} f_R \end{pmatrix}. \tag{3.125}$$

By utilizing the notation

$$\begin{aligned} F_{RR} &:= B_R A_{RR}^{-1} B_R^\top \\ F_{RC} &:= B_R A_{RR}^{-1} A_{RC} \\ F_{CC} &:= A_{CC} - A_{RC}^\top A_{RR}^{-1} A_{RC} \\ d_R &:= B_R A_{RR}^{-1} f_R \\ d_C &:= f_C - A_{Rc}^\top A_{RR}^{-1} f_R \end{aligned} \tag{3.126}$$

(3.125) can be compactly rewritten as

$$\begin{pmatrix} F_{RR} & F_{RC} \\ F_{RC}^\top & F_{CC} \end{pmatrix} \begin{pmatrix} \lambda \\ u_C \end{pmatrix} = \begin{pmatrix} d_R \\ d_C \end{pmatrix} \,. \tag{3.127}$$

In comparing this system of equations to that in (3.101) two important observations can be made. First, not all primal variables have been eliminated here. Those on non-torn nodes are still present since no Lagrange multipliers are associated to them. Second, because of these global couplings, none of the local problems is singular and consequently no subspace constraint is required as by the second equation in (3.101).

Solving. With respect to solving (3.127) however, a solution is first sought to the dual vector λ in the equation

$$\Big(F_{RR} + F_{RC} F_{CC}^{-1} F_{RC}^\top\Big)\, \lambda = d_R - F_{RC} F_{CC}^{-1} d_C \,, \tag{3.128}$$

where u_C has been eliminated. As with the dual FETI method, PCG iteration is utilized. Note that unlike F in (3.99), here the operator matrix $\left(F_{RR} + F_{RC}F_{CC}^{-1}F_{RC}^{\top}\right)$ consists of two components. The first one, F_{RR} is also block-diagonal and thus will lead to local and independent problems. The second one, $F_{RC}F_{CC}^{-1}F_{RC}^{\top}$, on the other hand, results in a coarse, global problem, which arises *not* from a null space, but the non-torn nodal variables in (3.111) and (3.118), respectively.

Let us replace all elements withing the brackets in (3.128) except for F_{CC} by its definitions given in (3.126) again, in order to detail on the real operations to be carried out while solving (F_{CC} will be represented explicitly). The operator application step within PCG iteration then is implemented by the following substeps [54]:

$$\begin{aligned} y_{R,1} &\leftarrow B_R A_{RR}^{-1} B_R^{\top}\, x \\ z_{C,1} &\leftarrow A_{RC}^{\top} A_{RR}^{-1} B_R^{\top}\, x \\ z_{C,2} &\leftarrow F_{CC}^{-1}\, z_{C,1} \\ y_{R,2} &\leftarrow B_R A_{RR}^{-1} A_{RC}\, z_{C,2} \\ y &\leftarrow y_{R,1} + y_{R,2}\,. \end{aligned} \tag{3.129}$$

while x refers to the vector to which the bracketed term in (3.128) is applied. Because of the block-diagonal structure A_{RR}, its inversion in the first and second step is again implemented by concurrent solving of the corresponding local problems for each subdomain. Its result is then used in the first and second step of (3.129), whereas with the first one it is multiplied by B_R, which mainly corresponds to a restriction onto the edge nodes, while the second step, $A_{RC}^{\top}$ gains a weighted restriction of the local variables to the global ones on the crosspoints (which can also be parallelized in the same manner). In the third step, the global error propagation is realized, whereas the lower-dimensional matrix F_{CC} is usually calculated explicitly or factorized beforehand, in order to minimize the computational effort during iteration. Similarly, $A_{RR}^{-1}A_{RC}$ in the fourth step is also pre-calculated explicitly, since the number of columns in A_{RC} is only that of the number of coarse degrees of freedom. To summarize, one application of the operator mainly amounts to solving local problems with hybrid boundary conditions for every subdomain in parallel (A_{RR}^{-1}), as well as to solving a lower-dimensional problem (F_{CC}^{-1}).

Preconditioning. Standard preconditioners are similar to those of the one-level FETI method. For example, the *FETI-DP Dirichlet preconditioner* here reads

$$\tilde{M}_D := \sum_{i=1}^{N} D^{(i)} B_R^{(i)} \begin{pmatrix} 0 & 0 \\ 0 & S_{EE}^{(i)} \end{pmatrix} B_R^{(i)^{\top}} D^{(i)} \tag{3.130}$$

with the Schur complements

$$S_{EE}^{(i)} = A_{EE}^{(i)} - A_{IE}^{(i)^\top} \left(A_{II}^{(i)}\right)^{-1} A_{IE}^{(i)} \tag{3.131}$$

and $D^{(i)}$ being weighting functions accounting for coefficient jumps across subdomain boundaries. For our model problem, we suggest to use the weights $\gamma_i^\dagger$ having been proposed in [119] and introduced here in the previous section.

Similarly to the one-level method, with the *FETI-DP lumped preconditioner* it is omitted the second term of the Schur complement (3.131), resulting in

$$\tilde{M}_L := \sum_{i=1}^{N} D^{(i)} B_R^{(i)} \begin{pmatrix} 0 & 0 \\ 0 & A_{EE}^{(i)} \end{pmatrix} B_R^{(i)^\top} D^{(i)} \ . \tag{3.132}$$

Extensions to three Three Dimensions. So far, the primal problem was defined only at the subdomains' crosspoints. However, especially for three-dimensional problems, FETI-DP with primal problem so defined gives unsatisfying results in terms of the convergence rate (see, e.g., [52]). As a remedy, recent research activities [52, 76, 75] focus on extending the primal problem. For example, continuity can not only be realized by the identity of the crosspoint variables, but also by that of the per-edge means. In that case, the dual problem is reduced to asserting equality of the deviations from the per-edge means only. In the three-dimensional case this expands to the face-means and/or edge means. In terms of implementating such mean identities, two different approaches have emerged: First, edge and/or face mean equalities of adjacent subdomains are enforced by additional Lagrange multipliers, which are treated as additional components of the coarse, yet primal variables u_C, and hence are also eliminated when going from equation (3.127) to equation (3.128), see [52, 76]. Alternatively, such means can be represented explicitly by appropriate coarse basis functions, see [75].

Scalability. Although the FETI-DP methods provide the same convergence rate bounds as the purely dual ones, their cost per iteration is lower since only one coarse problem needs to be solved, which, in general, is also smaller, see, e.g., [51].

3.3 Experimental Studies

Here we study the numerical properties of primal non-overlapping DD methods experimentally. As two representative algorithms we chose the one-level Neumann-Neumann (NN) as well as the two-level Balancing Neumann-Neumann (BNN) preconditioners in connection with PCG iteration in order to parallelize the CLG optical

flow problem, see Section 2.4. In terms of the latter, we consider the corresponding LSE (2.17) being partitioned in the same manner as for the definite Helmholtz equation in (3.9) and (3.9).

Experiments are conducted on two different data sets: An image pair of the well-known Marble sequence (512 × 512 pixels), and a synthetically generated particle image pair (2000 × 2000 pixels) as it appears in particle image velocimetry (PIV).

In particular, the experimental studies focus at the following issues:

(i) The impact of interface preconditioning on the convergence rate of the Krylov subspace iteration.

(ii) The evolution of the convergence rate while increasing the number of subdomains N having the total number of unknowns n fixed.

(iii) The impact of the subdomain solver's precision on the convergence rate.

(iv) The scalability behavior on a real parallel machine (PC-cluster with up to 144 processing nodes).

In all experiments, we assume the the total number of unknowns n, and hence $1/h^2$, with h denoting the mesh-size of the finite element discretization, which is fixed in our considerations and is only the number of subdomains N, and thus $1/H^2$, with H denoting the maximum subdomain diameter, which varies with our experiments. Obviously, there are other scenarios, e.g. those where it sought to increase the discretization accuracy. In that case, one seeks to decrease mesh-size h while keeping the total computation constant, which is reached by decreasing H also, i.e. increasing the number of subdomains and thus distributing the additional computational effort on more processing nodes. In most image processing applications, however, the primary goal is to lower the total computation time by increasing the number of subdomains, i.e. decreasing H while having h fixed.

3.3.1 Parameter Selection and Input Data

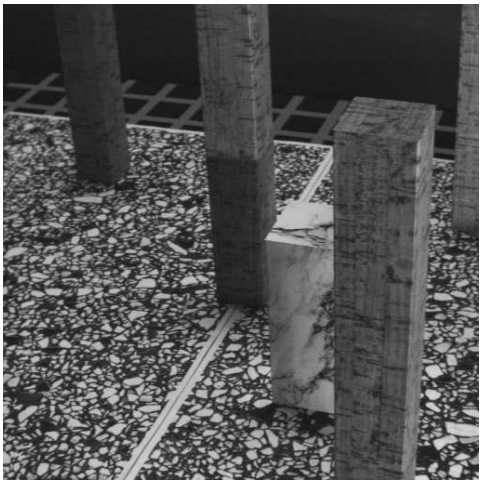
For ease of implementation, in the following we assume the image plane Ω to be partitioned into equally sized, quadratic subdomains Ω^i in all experiments, i.e. the aspect ratio, whose influence on convergence was not in our focus, was one always.

Two distinct datasets served as input data. First, frame 16 and 17 of the well-known *Marble sequence*[4], Fig. 3.4, was chosen as a representative for the class of a moving three-dimensional scene, and served as input image pair for the experiments 1–2. In order to investigate the practicability of the proposed methods in the context

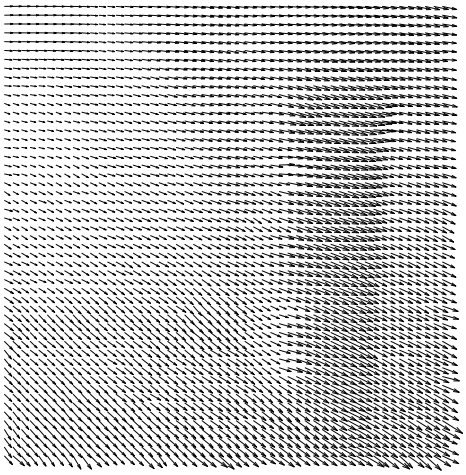
[4]Created 1993 by Michael Otte, Institut für Algorithmen und Kognitive Systeme, University of Karlsruhe, Germany. Available at `http://i21www.ira.uka.de/image_sequences`.

of *particle image velocimetry (PIV)* also, as a second data set we generated an 2000×2000 image pair consisting of small particle movements (see Fig. 3.4), which were used for the experiments 2–4. We will refer to the first dataset by 'Marble 512' and to the second one by 'PIV 2000' in the following.

The parameter values of the CLG motion estimation were: $\alpha = 1000$, $\sigma = 2.6$, $\rho = 1.8$, while the intensity values were from the range $[0, 255]$. The implementation was realized in C/C++ using the Intel compiler 7.0, with O3 optimization option, and MPI-conform[58, 57] inter-process communication libraries on the Linux OS on standard PC hardware. Vectorization operations (SSE(2) or 3dnow-commands) were not used.



(a) Frame #16 of the marble data set



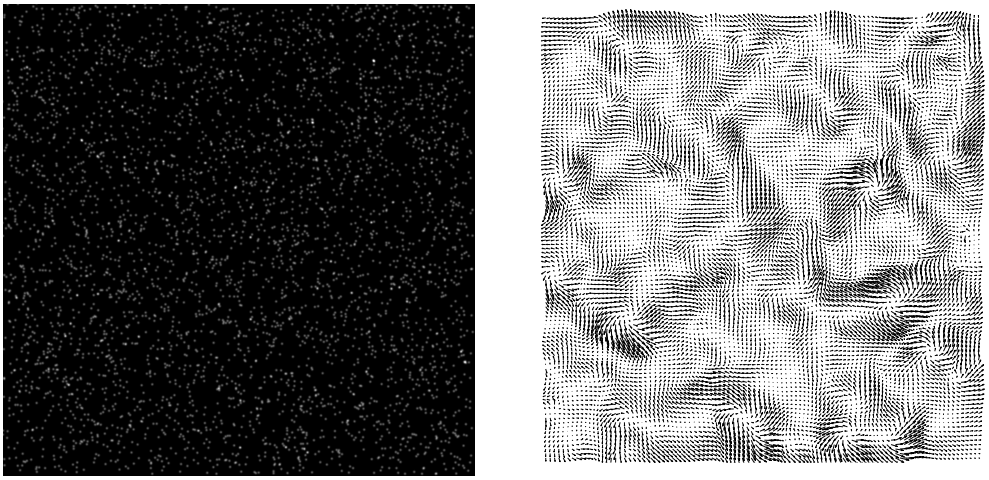
(b) True motion field between frame #16 and #17

Figure 3.4: **Input data for experiments with the marble data set.** (512×512 pixels, maximum velocity: 2.58 pixels/frame)

3.3.2 Algorithms and Implementation Details

We briefly explain the important details in terms of the parallel implementation. In case of the used parallelization methods these are the implementation of the action of the Steklov-Poincaré operator S, cf. Sec. 3.1.2.2, that of the NN preconditioner P_{NN}, see Sec. 3.2.2.2, and, in case of the BNN method only, that of the coarse preconditioner S_C^{-1}, see Sec. 3.2.3.3.

A already explained above, S can be written as a sum of local Schur complements: $\left(\sum_i R_{\Gamma_i}^{\Gamma^\top} S_i R_{\Gamma_i}^{\Gamma^\top}\right)$. That is, its application, e.g. within an PCG iteration in order to solve (3.43), allows for a separation into N local operators S_i, whose actions can



(a) First image of input image pair (b) True motion field

Figure 3.5: **Input data for the particle sequence experiments.** (2000×2000 pixels, maximum velocity: 0.96 pixels/frame)

then be carried in parallel. Again, the local Schur complements are never computed explicitly, but mainly amount to solving local Dirichlet problems, as pointed out in Sec. 3.1.1.2. Similarly, the action of P_{NN} is computed by solving the local Neumann problems associated with the local Schur complement inverses $(S^i)^{-1}$.

Besides the computational aspects, also the question of how to assign optimally — in terms of communication time — the parallelizable computations to the available processing nodes, is of importance. This then results in a specific *communication pattern*. For the NN method however, this is rather obvious: Since the results of applying S and P_{NN} are needed in all remaining operations, such as matrix-vector multiplications, we have chosen to carry them out on one central node. On the other hand, the computation of S_i and $(S^i)^{-1}$, respectively, are carried out on N separate processing nodes in parallel. Consequently, for each occurrence of S and P_{NN} within the iteration, it was necessary to select and distribute the corresponding local interface variables to each of the nodes, and in turn to collect and merge the local results into a global vector on the central node. The aforementioned two operation groups–which are denoted by *scattering* and *gathering*, respectively, in the context of parallel programming exactly correspond to the actions of the restriction matrices $R^{\Gamma}_{\Gamma_i}, i = 1, \ldots, N$ and its transpose $R^{\Gamma^\top}_{\Gamma_i}, i = 1, \ldots, N$, respectively. In particular, a scatter operation amounts to sending to each local node, associated with the subdomain Ω^i, the subset $\partial\Omega^i \cap \Gamma$ of nodal variables. In turn, the transposed operator $R^{\Gamma^\top}_{\Gamma_i}$ amounts to receiving and adding the nodal values of all local shared

boundaries.

In case of the BNN method, additional considerations for implementing the action of the coarse Schur complement inverse $S_0^{-1} = \left(R_0 S R_0\right)^{-1}$, cf. Sec. 3.2.3.3, need to be made. Unlike P_{NN}, a straightforward implementation of its definition would involve the inversion of S, which does not allow for a substitution by the local $S^{(i)}$, as can be seen easily. Instead, we found it efficient to compute S_C explicitly beforehand, and then to invert it during iteration up to an sufficient accuracy. In particular, the entries of S_C are determined column-wise, by carrying out

$$\left(S_C\right)_{\cdot j} \leftarrow R_{\Gamma_C}^{\Gamma}\Big(\sum_i R_{\Gamma_i}^{\Gamma^\top} S_i R_{\Gamma_i}^{\Gamma}\Big) R_{\Gamma_C}^{\Gamma^\top} e_j \,, \tag{3.133}$$

with e_j denoting a vector containing a one-entry at j and zero-entries else. Obviously, this would need to be done N times.

However, a closer look on $R_{\Gamma_C}^{\Gamma^\top}$ reveals that its j-th column affects only the boundaries of subdomain Ω_j, as well as its left, right, upper and lower neighbors, if any, as well as the vertex variables of its diagonal neighbors. Making use of this fact, several columns of S_0 can be computed by (3.133) at one time, by having several one-entries in the e_j-vectors. In addition, we have found the vertex couplings to be negligible, such that in total only eight executions of (3.133) are necessary. See Figure 3.6 for an illustration.

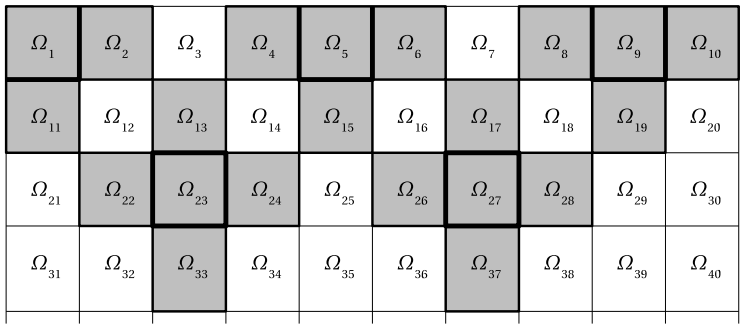


Figure 3.6: **Illustration of the optimized initialization scheme for the coarse Schur complement matrix S_0.** Each gray cross depicts non-zero couplings in S_0, i.e. on the coarse level, except for those at vertices. Also, the black framed subdomains correspond to one-entries in the vector e_0 (see text), and thereby gray indicates a non-zero entry of $R_{\Gamma_C}^{\Gamma} S R_{\Gamma_C}^{\Gamma^\top} e_0$.

Finally, the initial calculation of the right-hand sides of the interface equation (3.43), as well as the concluding calculation of the remaining inner nodal variables by (3.44) allow for parallel execution by concurrent solving of the corresponding local Dirichlet systems also.

3.3.3 The Impact of Interface Preconditioning

In this first experiment, the goal was to observe the influence of interface preconditioning in general. Therefore, we solved equation (3.43) w.r.t. u_Γ once making use of the NN preconditioner and once without using it. As local solver PCG iteration was used too, with a relative residual error tolerance of 10^{-5}. As input data served image 16 and 17 of the Marble sequence, see above.

Table 3.1 depicts the number of necessary *outer* (P)CG iterations to reach an residual error of 10^{-3}, i.e. $||\chi - Su_\Gamma^{(N)}||_2/||\chi||_2 < 10^{-3}$, with χ denoting the right-hand-side of (3.43). It clearly shows that, in agreement with theory [101], the system becomes more and more ill-conditioned if the number of subdomains increases. Using the Neumann-Neumann preconditioner (3.63), however, largely compensates this effect and enables shorter computation times through parallelization.

Partition	Subdomain size	Preconditioner	Necessary outer iterations
2×2	256×256	NN	6
2×2	256×256	none	42
4×4	128×128	NN	7
4×4	128×128	none	42

Table 3.1: **The impact of interface preconditioners while solving the Schur complement equation.** The number of necessary iterations to reach a relative residual error of 10^{-3} is given. Results show the strong improvement in convergence when employing NN-preconditioning within a CG iteration.

3.3.4 Convergence in Dependence on the Number of Subdomains

Here we studied the influence of the number of subdomains, and thereby that of $1/H$, on the convergence rate of the PCG iteration, while using NN or BNN preconditioning, and compared them to the theoretical upper limits given in (3.67) and (3.75), respectively.

In order to measure the convergence rate ρ after k PCG iterations, we first determined the relative residual error by

$$\epsilon^{(k)} := \frac{||\hat{u} - Au^{(k)}||_2}{||\hat{u} - Au^{(0)}||} \tag{3.134}$$

and subsequently estimated ρ by

$$\rho = \sqrt[k]{\epsilon^{(k)}}\,. \tag{3.135}$$

Furthermore, the convergence rate bounds were computed from the theoretical condition number bounds by (3.61). Thereby, the constant parameter C in (3.67) and (3.75) was set to $\frac{1}{6}$.

Results are depicted in Figure 3.7. First, let us consider the results for the Marble sequence (blue curves). As expected by theory, the convergence rates deteriorate for increasing subdomain numbers using the NN preconditioner, since it lacks a global error propagation. In contrast, with the BNN preconditioner, the rate remains relatively constant, because of the additional coarse-grid correction. However, for PIV Sequence (black curves), the deterioration using the NN preconditioner is much less than for the Marble sequence, and the rate is even slightly better for larger number of subdomains. The explanation for these quite differing observations is found in the fact that with the PIV data the spatial couplings play a lesser role than with the Marble sequence, since with the former an image gradient is available almost everywhere throughout the image plane, which is not the case with the latter where the spatial smoothness term yields a fill-in of motion vectors at regions without any or only few image gradients. This phenomenon is clearly illustrated by the per-pixel L^2-error between resulting motion fields and their corresponding groundtruth solutions (obtained without parallelization), see Figure 3.8. Consequently, for the Marble sequence, the lack of global information propagation of the NN preconditioner affects convergence much less when the number of subdomains and therefore locality is increased. In view of the fact that the BNN preconditioner demands more than three times the computational effort than the NN per iteration, the additional coarse-grid preconditioning is reasonable when spatial couplings are dominant across a greater number of subdomains.

3.3.5 Convergence in Dependence on the Precision of the Local Solver

Besides the degree of image plane partition, also the influence of the local Dirichlet and Neumann problems‘ solving precision in connection with the NN preconditioner was studied, in order to reach a rule of thumb for the minimum error tolerance needed to reach a given tolerance level for the outer PCG iteration.

Therefore, we observed the final relative residual error for the Schur complement equation while varying the maximum residual error threshold of the local Dirichlet and Neumann problem solvings. Again, PCG iteration was used as local solver also. The results shown in Table 3.2 for the Marble sequence applying a 4×4 or 8×8 partition, suggest that local solving must be about one order of magnitude more precise than the desired precision for the Schur complement equation.

Residual error of the local solver	Resulting residual error of the Schur complement problem	
	4×4 subdomains	8×8 subdomains
10^{-5}	$2.1 \cdot 10^{-4}$	$3.6 \cdot 10^{-4}$
10^{-4}	$2.7 \cdot 10^{-3}$	$6.1 \cdot 10^{-3}$
10^{-3}	$1.2 \cdot 10^{-2}$	$3.0 \cdot 10^{-2}$
10^{-2}	$1.6 \cdot 10^{-1}$	$3.6 \cdot 10^{-1}$
10^{-1}	$7.9 \cdot 10^{-1}$	$9.3 \cdot 10^{-1}$

Table 3.2: **Influence of the local solver's precision.** The first column shows the error thresholds up to which the local Dirichlet and Neumann problems were solved. The second and third column give the resulting error of the outer PCG iteration employing BNN preconditioning for the Schur complement problem. Results show that the precision of the local solvers need to be about one order of magnitude higher than the desired precision of the Schur complement problem.

3.3.6 Scalability Study on a Parallel Computer

In order to study the advantage of employing substructuring methods compared to non-parallel methods in practice, we conducted scalability experiments on a state-of-the-art parallel machine[5] varying the number of processing nodes from four up to 144, i.e. using partitions of 2×2 to 12×12.

Due to limited access to the PC-cluster, only experiments for the particle image data set could be made. Full multigrid iterations derived from an algorithm for the original problem [20, 21, 22, 19] served as local solvers for the Dirichlet and the Neumann problems[6]. The algorithmic parameters, such as the outer number of PCG iterations as well as the number of V- or W-cycles and number of pre- and post-recursion Gauss-Seidel relaxations in connection with the local solvers, were adjusted manually, in order for the L^2-error[7] to be lower than 10^{-3} in comparison to a reference solution. The reference solution was obtained by solving the original problem up to a residual error of less than 10^{-8}.

For each of the different experiments, both the time spent for computations (either in the master node or the slave nodes) and the communication time were measured. Results for employing each of the two preconditioning techniques are depicted in the diagrams 3.9(a) and (b). At first glance, one observes that the run-times

[5]The dedicated PC-cluster *HELICS*, 512 Dual AMD Athlon MP 1.4 GHz processing nodes, Myrinet2000 network, Interdisciplinary Center for Scientific Computing, University of Heidelberg, Germany

[6]Thanks to Andrés Bruhn for providing us with his implementation.

[7]i.e. $\frac{||w-\hat{w}||_2}{||\hat{w}||_2}$

with BNN preconditioning are about four times larger than for NN preconditioning, which is congruent with fact the former requires two additional applications of the Schur complement operator S as well as three inversions of its coarse variant, S_0, see (3.86). On the other hand, both preconditioners show almost the same scalability characteristics, which, as we found in Section 3.3.4, is a data-dependent fact for the NN preconditioner. However, in this case, NN preconditionering clearly beats BNN preconditioning, since the latter cannot compensate for its higher computational effort per iteration by an increase in scalability.

In order to assess the practical use of substructuring methods, we compared the measured run-times to that of Bruhn's highly-optimized full multigrid implementation for the original problem, which needed 8.25 seconds (in average) on the same dataset when being run on *one* processing node of the aforementioned PC-cluster (Note that the code was optimized for current Intel Pentium machines on smaller images sizes, e.g. 160×120, where run-times around 0.02 seconds per frame were measured, [22]). Comparing this result to those of the two substructuring methods shows that only with NN preconditioning is it possible to reach a speed-up, but at the cost of at least 25 or more processing nodes, which can be most clearly seen with the speed-up diagrams in Fig. 3.10. Further analysis revealed that the reason for the substructuring methods not to perform better lies in the about four to eight times higher number of smoothing cycles for the local multigrid solvers when being employed with the (local) Dirichlet problems (which do not appear in the original problem). In particular, one V-cycle with two pre- and post-Gauss-Seidel relaxations per level have been employed with the Neumann problems (both the global as well as for the local ones), whereas four W-cycles in connection with six to eight Gauss-Seidel smoothing steps per level were necessary to solve the local Dirichlet problems. These findings must be seen in view of the fact that the current multigrid solver had been optimized with respect to Neumann boundary conditions. Because of very limited access to the PC-cluster, further experiments with different local solvers, e.g. PCG iteration, could not be conducted.

Besides the communication time, also the communication volume was recorded, see the diagram in Fig. 3.11. It results from the following phases of the parallel algorithm: (a) initial distribution of input data, i.e. the image pair, (b) exchange of values on shared boundaries nodes, and (c) collection of the final vector field from the slave processes. Whereas the communication volumes for (a) and (c) are nearly independent of the number of subdomains[8], for step (b) they grow linearly with the square root of the number of subdomains, as can be seen in Fig. 3.11, which is

[8] In fact, in our implementation it increases slightly with the number of subdomains for (a), since there the distributed regions must have a small overlap in order to obtain the same convolution results on the shared boundary nodes of neighboring local systems.

consistent with the fact that the overlap between subdomains is one dimension less than that of the problem. This observation clearly shows the advantage of the non-overlapping substructuring methods in terms of higher scalabilities in comparison to overlapping methods being presented in Chapter 4.

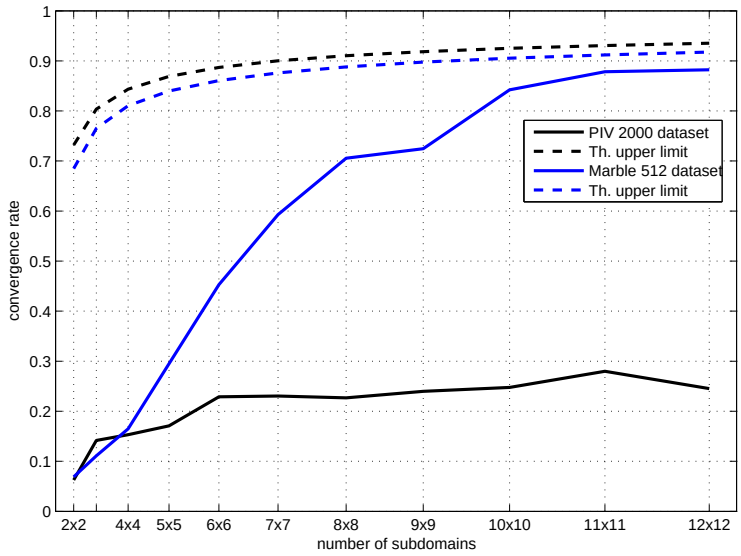
3.4 Conclusion

In this chapter, we explained the mathematical foundation of substructuring methods, in particular the Steklov Poincaré operator and the Schur complement equation, and gradually derived the associated algorithms for the parallelization of a prototypical linear PDE problem. Moreover, we explained the common primal iterative methods and have shown their link to the aforementioned theory. Furthermore, we addressed the numerical problem of local information propagation with one-level algorithms and elucidated its remedy by means of additional coarse-grid preconditioners. Finally, we focused on the more recent dual and primal-dual substructuring methods, which consider local Schur complement problems within the setting of an constrained optimization problem, and compared them to their primal counterparts.

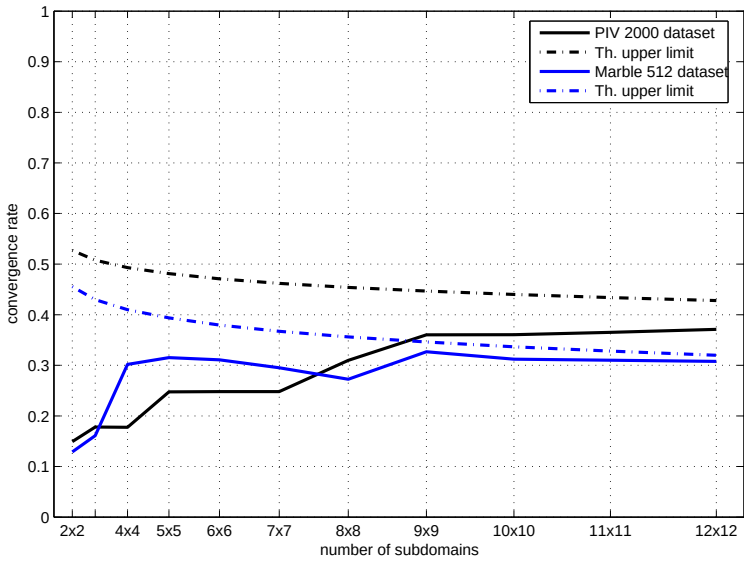
Within the experimental section, we studied two representative primal iterative substructuring methods with respect to their convergence behavior in dependence on the number of subdomains (while having the total number of unknowns fixed), the influence of the local solver's precision, the scalability on a real parallel machine, and the speed-up in comparison to a highly-tuned sequential multigrid implementation. Although the experiments confirm the very good scalability of the two investigated non-overlapping methods, the unequally higher number of iterations and recursions involved with solving the local Dirichlet problems, in comparison to the corresponding Neumann problems, suggest further investigations, e.g. on better prolongation/restriction operators with the Dirichlet solvers or alternative solving methods, like conjugate gradient iteration.

By the derivation of the dual FETI method, we have given an alternative two-level substructuring approach and have shown its similarity to the Balancing NN method. Although empirical studies were not made, convergence rates are most likely to be comparable. The subsequently presented primal-dual FETI approach then gave means to freely chose the coarse couplings, which allows to adjust information propagation during iteration with respect to the given model problem, the partitioning of the image domain and the hardware requirements imposed by the specific parallel computer on which the computations are carried out. Future work is to concentrate on an empirical comparison of the BNN and the dual FETI method on a representative model problem, as well as an empirical studies of primal-dual FETI implementations with different kinds of (primal) coarse dependencies.

To conclude, we have shown that substructuring methods are feasible means to parallelize computational intensive PDE-base problems. Thereby, variational image processing approaches on large two-dimensional data sets or standard-size three-dimensional sets, come into reach of real-time computing.

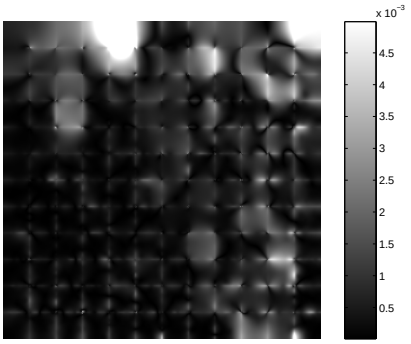


(a) with one-level Neumann-Neumann preconditioning

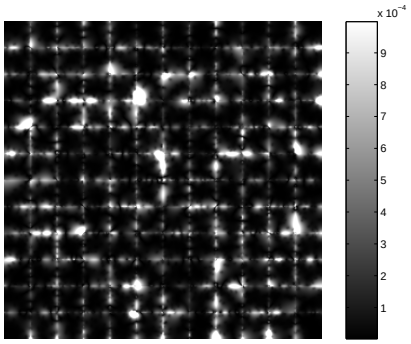


(b) with two-level Balancing Neumann-Neumann preconditioning

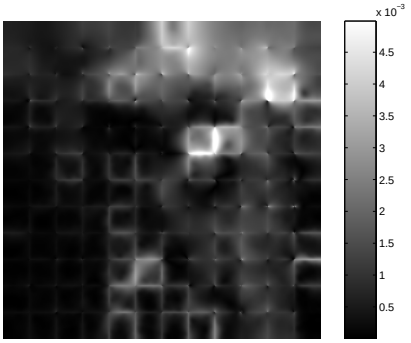
Figure 3.7: **Measured convergence rates and theoretical upper limits.** The measured (solid) convergence rates for each data set and each of the two preconditioners are depicted, as well as their corresponding theoretical upper limits (dashed), respectively. Whereas with BNN preconditioning the rates stay almost constantly low, with the NN preconditioner the rates increase rapidly for problems with dominant spatial couplings across the subdomain borders.



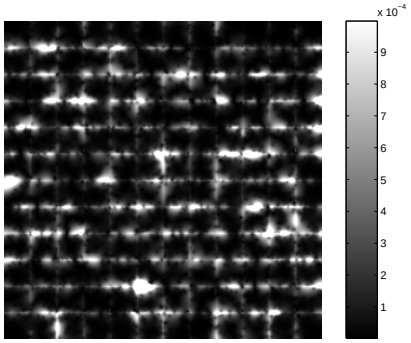
(a) Marble seq. with NN preconditioning



(b) Particle seq. with NN preconditioning

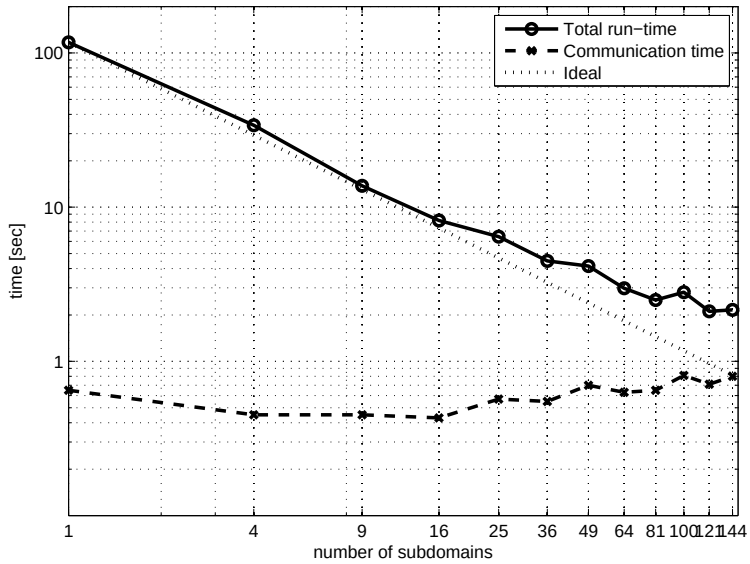


(c) Marble seq. with BNN preconditioning

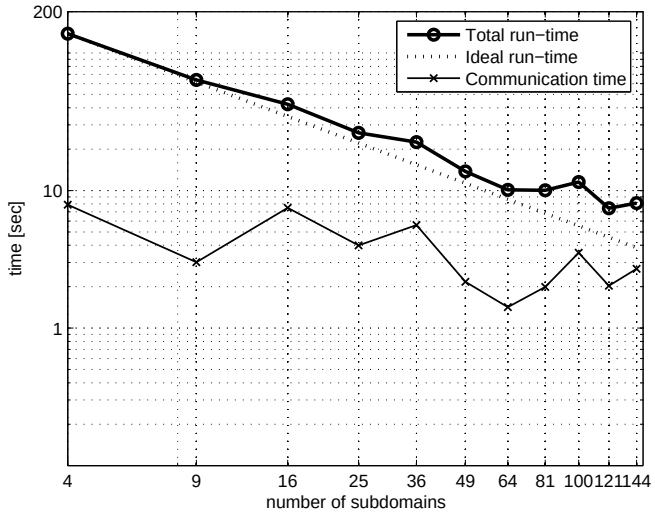


(d) Particle seq. with BNN preconditioning

Figure 3.8: **Per-pixel L^2-errors.** Depicted are the per-pixel L^2-errors for 12×12 decompositions with respect to a highly accurate reference solution of the original model problem. Whereas with the particle sequence the one-level NN preconditioner already yields good results because of the spatial couplings are very local, for the Marble data set, BNN preconditioning helps diminish significant strong errors present at some of the subdomains. Note that local errors are uniformly smaller than 10^{-3} and 10^{-4}, respectively.

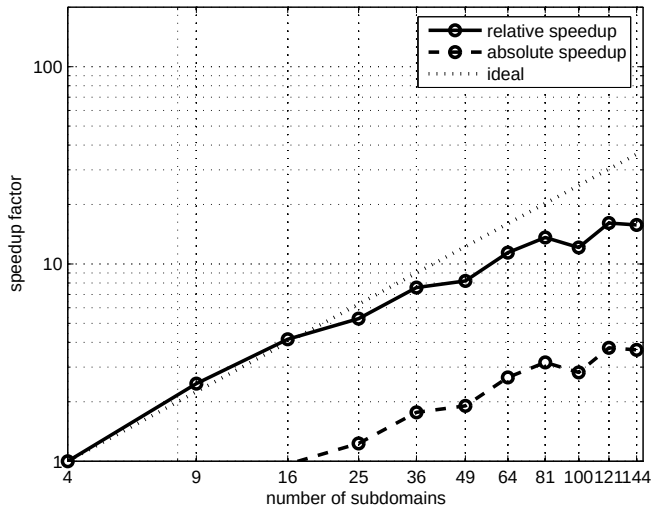


(a) with one-level Neumann-Neumann preconditioning

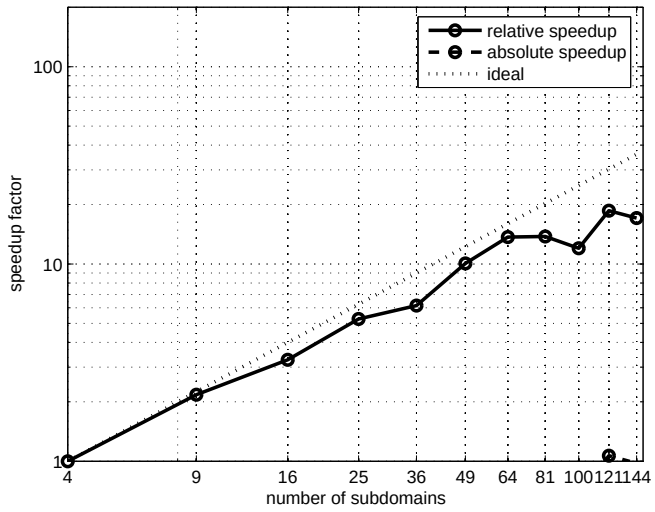


(b) with two-level Balancing Neumann-Neumann preconditioning

Figure 3.9: **Measured run and communication times on a PC-cluster.** The logarithmic plots depict the total run-time for varying numbers of subdomains in comparison to the ideal development (relative to the minimal number of subdomains) for both preconditioner types. In addition, the part of the communication time is shown. Although both NN and BNN preconditioning exhibit a similar very good scalability, the BNN preconditioner cannot compensate for the three times higher cost per iteration.



(a) with one-level Neumann-Neuman preconditioning



(b) with two-level Balancing Neumann-Neuman preconditioning

Figure 3.10: **Measured relative and absolute speed-up factors.** The speed-up factors are shown either relative to the run-time on a 2×2 partition, or to a highly-optimized non-parallel multigrid implementation on the same hardware.

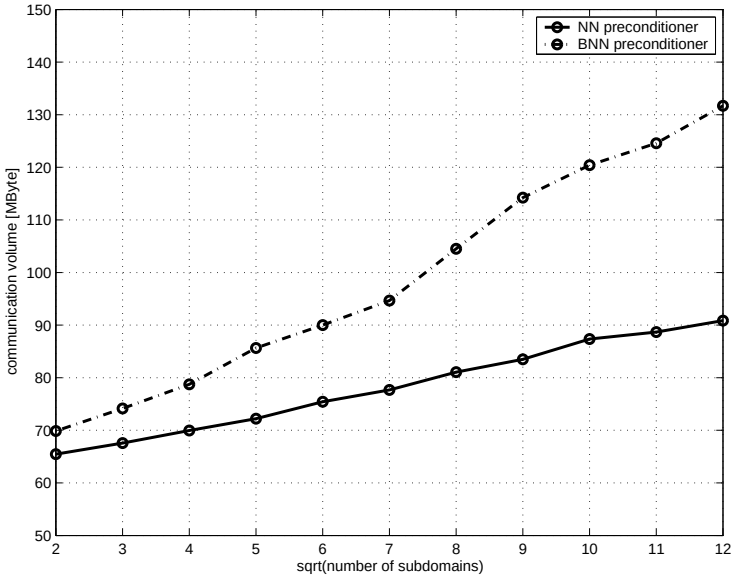


Figure 3.11: **Measured communication volume for a varying number of subdomains.** The measured communication volume is plotted for the two different preconditioner types over the square root of the number of subdomains. This clearly shows that, in accordance with theory, the amount of data to be exchanged between the subdomain problems expands one-dimensionally.

Chapter 4

Overlapping Domain Decomposition Methods

Besides the family of *non-overlapping* methods, in the following chapter we will focus on the second large class of *overlapping* domain decomposition methods. Unlike with the former, here, parallel preconditioners are realized by local inversions of the original problem being restricted to overlapping subdomains.

We will explain the most important approaches and their algorithmic realization in what follows, beginning with earlier basic variants on two subdomains and proceeding to recent, more complex methods on many subdomains and several resolution levels. Also, we will focus on relations to classical multigrid methods, which appear as degenerate cases of methods being presented below. The objective of this chapter is firstly to outline the basic mathematical and algorithmic principles of this class of decomposition approaches, and secondly to point out and discuss its connections to the main topic of this work, the substructuring approach.

Throughout this chapter, we consider *overlapping coverings* $\{\Omega_i \mid i = 1, \ldots, N\}$ of Ω, i.e. $\Omega = \cup_i \Omega_i$ and $\Omega_i \cap \Omega_j \neq \emptyset$, Ω_j is neighbor of $\Omega_i, i = 1, \ldots, N$, while denoting $\{\Omega_i\}$ as *subdomains*. Moreover, let $\Gamma_i := \partial\Omega_i \setminus \partial\Omega$, $i = 1, \ldots, N$ be those parts of the local boundary which are interior of Ω.

Model Problem. As model problem, again, we consider the Definite Helmholtz equation (3.1), as introduced in Chapter 3. Unlike the notation used in the partitioning (3.9), in the following, we refer to the sets Γ_i and $\overline{\Omega}_i \setminus \Gamma_i$, respectively, in the superscript, which yields the form:

$$\begin{pmatrix} A_i^{II} & A_i^{I\Gamma} \\ A_i^{I\Gamma^\top} & A_i^{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} u^I \\ u^\Gamma \end{pmatrix} = \begin{pmatrix} f^I \\ f^\Gamma \end{pmatrix} , \quad i = 1, \ldots, N . \tag{4.1}$$

4.1 One-level Methods

In order to explain the principles of the basic overlapping methods — the alternating, multiplicative and additive Schwarz methods — first we consider a covering of only two subdomains, Ω_1 and Ω_2, either in the continuous, discrete non-matching or discrete matching case, and then move on to the case of multiple subdomains. See Fig. 4.1 for partition examples for these three cases.

4.1.1 The Case of Two Subdomains

4.1.1.1 The Alternating Schwarz Method

The first overlapping domain decomposition method was already proposed over a hundred years ago by H. Schwarz [106]. There a solution to problem (3.6) is computed indirectly by the algorithm:

Algorithm 5: Multiplicative Schwarz iteration in the continuous case

initialize u_1^0, u_2^0 by an arbitrary value

iterate $k = 1, 2, \ldots$ **until** convergence

$$\text{solve for } u_1^{k+1} \text{ in } \begin{cases} Lu_1^{k+1} = f_1 & \text{in } \Omega_1 \\ u_1^{k+1} = u_2^k|_{\Gamma_1} & \text{on } \Gamma_1 \\ \partial_n u_1^{k+1} = 0 & \text{on } \partial\Omega_1 \setminus \Gamma_1 \end{cases}$$

$$\text{solve for } u_2^{k+1} \text{ in } \begin{cases} Lu_2^{k+1} = f_2 & \text{in } \Omega_2 \\ u_2^{k+1} = u_1^{k+1}|_{\Gamma_2} & \text{on } \Gamma_2 \\ \partial_n u_2^{k+1} = 0 & \text{on } \partial\Omega_2 \setminus \Gamma_2 \end{cases}$$

while information exchange takes place at the artificial boundaries Γ_1 and Γ_2, respectively, and with $f_i := f_{|\Omega_i}$ and $f_i, u_i \in V_i := V(\Omega_i)$. (In the following, subscripts always refer to subdomains). In particular, both subproblems are coupled by artificial Dirichlet boundary conditions applied at Γ_i, respectively, whereas the original boundary conditions remain at $\partial\Omega_i \setminus \Gamma$. Finally, once that u_1^k and u_2^k have converged, the global solution is given by

$$u^{k+1}(x,y) := \begin{cases} u_1^{k+1}(x,y) : & (x,y) \in \Omega_1 \\ u_2^{k+1}(x,y) : & (x,y) \in \Omega_2 \end{cases}, \tag{4.2}$$

which is known to be equal to the non-decomposed original solution, i.e. to problem $Lu = f$. Thereby, the convergence speed is linear, i.e.

$$\|\hat{u} - u^k\|_A \leq \rho^k \|\hat{u} - u^0\|_A \ ,$$

while $\rho < 1$ denotes the reduction factor.

In the *discrete case* having non-matching meshes $\mathcal{T}^h(\overline{\Omega}_i), i = 1, 2$ on either subdomain, however, additional interpolation operators which map nodal variables from $\overline{\Omega}_1$ to Γ_2 and from $\overline{\Omega}_2$ to Γ_1, respectively, are required:

Algorithm 6: Alternating Schwarz iteration on non-matching grids

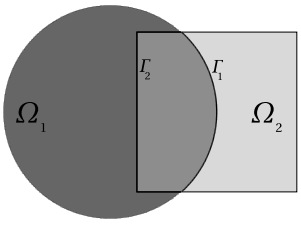
initialize u_1^0, u_2^0 by an arbitrary value

iterate $k = 1, 2, \ldots$ **until** convergence

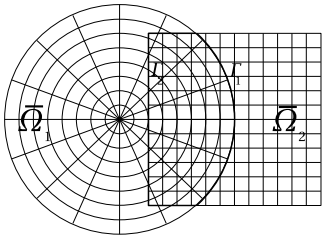
$$\text{solve for } u_1^{k+1} \text{ in } \begin{cases} \left(A_1^{II} \ A_1^{I\Gamma}\right) u_1^{k+1} = f_1^I & \text{on } \overline{\Omega}_1 \setminus \Gamma_1 \\ u_1^{k+1} = I_{\Omega_2 \to \Gamma_1} u_2^k & \text{on } \Gamma_1 \end{cases}$$

$$\text{solve for } u_2^{k+1} \text{ in } \begin{cases} \left(A_2^{II} \ A_2^{I\Gamma}\right) u_2^{k+1} = f_2^I & \text{on } \overline{\Omega}_2 \setminus \Gamma_2 \\ u_2^{k+1} = I_{\Omega_1 \to \Gamma_2} u_1^{k+1} & \text{on } \Gamma_2 \end{cases}$$

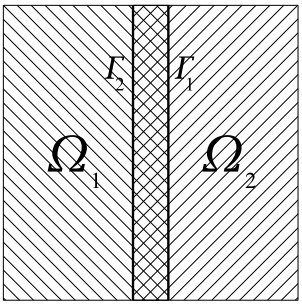
with $A_i u_i = b_i, i = 1, 2$ resulting from separate Finite Element discretization on either sub-mesh and while making use of the matrix partition as explained in beginning of this chapter.



(a) Overlapping continuous subdomains used with the alternating Schwarz method in the continuous case



(b) Overlapping discrete subdomains with non-matching grids as with the the alternating Schwarz method in the discrete case



(c) Overlapping discrete subdomains with matching grids as used with the multiplicative Schwarz method

Figure 4.1: **Examples for overlapping coverings used with the different Schwarz methods.**

4.1.1.2 The Multiplicative Schwarz Method

However, in case the subdomain‘s meshes are conformal, i.e. match, one can dispense with the interpolation mappings, thereby gaining the following procedure:

Algorithm 7: Multiplicative Schwarz iteration on matching grids

initialize u_1^0, u_2^0 by an arbitrary value

iterate $k = 1, 2, \ldots$ **until** convergence

$$\text{solve for } u_1^{k+1} \text{ in } \begin{cases} \left(A_1^{II} A_1^{I\Gamma}\right) u_1^{I,k+1} = f_1^I & \text{on } \overline{\Omega}_1 \setminus \Gamma_1 \\ u_1^{\Gamma,k+1} = u_{2|\Gamma}^k & \text{on } \Gamma_1 \end{cases}$$

$$\text{solve for } u_2^{k+1} \text{ in } \begin{cases} \left(A_2^{II} A_2^{I\Gamma}\right) u_2^{I,k+1} = f_2^I & \text{on } \overline{\Omega}_2 \setminus \Gamma_2 \\ u_2^{\Gamma,k+1} = u_{1|\Gamma}^{k+1} & \text{on } \Gamma_2 \end{cases} .$$

Taking the Dirichlet boundary values to the right-hand side and solving for u_1^I and u_2^I, respectively, results in

$$\begin{aligned} u_1^{I,k+1} &= A_1^{II^{-1}} \left(f_1^I - A_1^{I\Gamma} u_{2|\Gamma_1}^{I,k}\right) \\ u_2^{I,k+1} &= A_2^{II^{-1}} \left(f_2^I - A_2^{I\Gamma} u_{1|\Gamma_2}^{I,k+1}\right) , \end{aligned} \tag{4.3}$$

which can be written as interleaved Richardson iterations by further adding $u_i^{I,k} - u_i^{I,k}, i = 1, 2$, respectively, on either right-hand side, and by drawing $-u_i^{I,k}, i = 1, 2$ within the brackets:

$$\begin{aligned} u_1^{I,k+1} &= u_1^{I,k} + A_1^{II^{-1}} \left(f_1^I - A_1^{II} u_1^{I,k} - A_1^{I\Gamma} u_{2|\Gamma_1}^{I,k}\right) \\ u_2^{I,k+1} &= u_2^{I,k} + A_2^{II^{-1}} \left(f_2^I - A_2^{II} u_2^{I,k} - A_2^{I\Gamma} u_{1|\Gamma_2}^{I,k+1}\right) . \end{aligned} \tag{4.4}$$

Still, we have a formulation based on local vectors u_i, b_i. In aiming at a representation by means of global vectors u and b, which will be necessary for subsequent analysis, we modify (4.4) to

$$\begin{aligned} u_{1|\Omega_1 \cap \Omega_2}^{I,k} &\leftarrow u_{2|\Omega_1 \cap \Omega_2}^{I,k} \\ u_1^{I,k+1} &= u_1^{I,k} + A_1^{II^{-1}} \left(f_1^I - A_1^{II} u_1^{I,k} - A_1^{I\Gamma} u_{2|\Gamma_1}^I\right) \\ u_{2|\Omega_1 \cap \Omega_2}^{I,k} &\leftarrow u_{1|\Omega_1 \cap \Omega_2}^{I,k+1} \\ u_2^{I,k+1} &= u_2^{I,k} + A_2^{II^{-1}} \left(f_2^I - A_2^{II} u_2^{I,k} - A_2^{I\Gamma} u_{1|\Gamma_2}^{I,k+1}\right) , \end{aligned} \tag{4.5}$$

which means that $u_1^{I,k}$ and $u_2^{I,k}$ are updated by the most recent values of $u_2^{I,k}$ and $u_1^{I,k+1}$, respectively, in the overlap region. Note that we do not change the

scheme, because $u_i^{I,k} - A_i^{II^{-1}} A_i^{II} u_i^{I,k}$ cancels out at the right-hand sides. Most important, by this formulation, $u_1^{I,k}$ and $u_2^{I,k}$ no longer differ in their common variables in the overlap region, and hence can be replaced by restrictions of the global vector u:

$$\begin{aligned} u^{k+1/2} &= u^k + R_{I_1}^\top A_1^{II^{-1}} \big(f_1^I - A_1^{II} u_{|I_1}^k - A_1^{I\Gamma} u_{|\Gamma_1}^k\big) \\ u^{k+1} &= u^{k+1/2} + R_{I_2}^\top A_2^{II^{-1}} \big(f_2^I - A_2^{II} u_{|I_2}^{k+1/2} - A_2^{I\Gamma} u_{|\Gamma_2}^{k+1/2}\big)\,, \end{aligned} \tag{4.6}$$

with $R_{I_1}^\top$ and $R_{I_2}^\top$ denoting extensions by zero from $\overline{\Omega}_1$ and $\overline{\Omega}_2$ to $\overline{\Omega}$, respectively. Finally, due to $A_i^{II} u_{|I_i} + A_i^{I\Gamma} u_{|\Gamma_i} = R_{I_i} A u$ and $f_i^I = R_{I_i} f$, $i = 1, 2$, we find the expressions in brackets to be restrictions of the global residuals $(f - Au^k)$ and $(f - Au^{k+1/2})$, respectively, and end up with

$$\begin{aligned} u^{k+1/2} &= u^k + R_{I_1}^\top A_1^{II^{-1}} R_{I_1} \big(f - Au^k\big) \\ u^{k+1} &= u^{k+1/2} + R_{I_2}^\top A_2^{II^{-1}} R_{I_2} \big(f - Au^{k+1/2}\big). \end{aligned} \tag{4.7}$$

This reveals that one iteration of the multiplicative Schwarz method corresponds to two Richardson relaxation steps, which apply the preconditioners $B_1 := R_{I_1}^\top A_1^{II^{-1}} R_{I_1}$ and $B_2 := R_{I_2}^\top A_2^{II^{-1}} R_{I_2}$ alternatingly.

Moreover, the effect on the iteration error $e^k = \hat{u} - u^k$, with $\hat{u}$ denoting the true solution vector, can be shown straightforwardly. Since $A\hat{u} = f$, we can introduce the error vector to (4.7) as follows:

$$\begin{aligned} u^{k+1/2} &= u^k + B_1 A\, e^k \\ u^{k+1} &= u^{k+1/2} + B_2 A\, e^{k+1/2}\,. \end{aligned} \tag{4.8}$$

Multiplying by -1 and adding $\hat{u}$ on both sides then yields

$$\begin{aligned} e^{k+1/2} &= (I - B_1 A)\, e^k \\ e^{k+1} &= (I - B_2 A)\, e^{k+1/2}\,. \end{aligned} \tag{4.9}$$

Finally, substituting for $e^{k+1/2}$ in the second equation results in

$$e^{k+1} = (I - B_2 A)(I - B_1 A)\, e^k\,, \tag{4.10}$$

which not only shows the sequential manner of this method, but also gives grounds for denoting it as a multiplicative procedure. Because of the relaxations are applied sequentially, direct clues for parallel computation are not present with this method, although it will serve as a building block in a parallel algorithm in the case of multiple grid levels. (See below).

4.1.1.3 The Additive Schwarz Method

Besides multiplicative methods, an important variant is given by the *Additive Schwarz method*:

Algorithm 8: Additive Schwarz iteration

initialize u_1^0, u_2^0 by an arbitrary value

iterate $k = 1, 2, \ldots$ **until** convergence

$$\text{solve for } u_1^{k+1} \text{ in } \begin{cases} \left(A_1^{II} A_1^{I\Gamma}\right) u_1^{I,k+1} = f_1^I & \text{on } \overline{\Omega}_1 \setminus \Gamma_1 \\ u_1^{k+1}|_{\Gamma_1} = u_2^k|_{\Gamma_1} & \text{on } \Gamma_1 \end{cases}$$
$$\text{solve for } u_2^{k+1} \text{ in } \begin{cases} \left(A_2^{II} A_2^{I\Gamma}\right) u_2^{I,k+1} = f_2^I & \text{on } \overline{\Omega}_2 \setminus \Gamma_2 \\ u_2^{k+1}|_{\Gamma_2} = u_1^k|_{\Gamma_2} & \text{on } \Gamma_2 \end{cases}.$$

The main difference from the former methods that solving the local problems can be carried out *concurrently* and not sequentially since the second one does not depend on the result of the first one (note that the second problem relies on the old iterate u_1^k).

In form of a Richardson iteration this reads

$$u^{k+1} = u^k + R_{I_1}^\top A_1^{II^{-1}} R_{I_1}\left(f - Au^k\right) + R_{I_2}^\top A_2^{II^{-1}} R_{I_2}\left(f - Au^k\right), \tag{4.11}$$

which can be compactly written as

$$u^{k+1} = u^k + (B_1 + B_2)\left(f - Au^k\right) \ , \tag{4.12}$$

and, after some minor restatements, results in the error propagation rule

$$e^{k+1} = \left(I - (B_1 + B_2)\right) e^k \, , \tag{4.13}$$

both making the additive character more obvious.

Unlike the multiplicative or alternating methods, here the local subdomain problems can be solved in parallel. However, the iteration count to reach the same final error as using with the multiplicative method is about twice (see, e.g. [101]), that is, theoretically, a speed-up by parallelization cannot be reached here, in general.

4.1.2 The Case of Multiple Subdomains

Before combining the idea of domain decomposition with that of multiple grid solving, let us first focus on the extension of the previous methods to the case of more than two subdomains and its representation as preconditioners. Thus, we are now

looking at $N > 2$ overlapping subdomains $\{\Omega_i \,|\, \Omega_i \cap \Omega_{i+1} \neq \emptyset,\ i = 1, \ldots, N-1\}$ forming a covering of Ω, which we denote as the *multiple-subdomain case* in the following. See Fig. 4.2 for an example.

4.1.2.1 The Multiplicative Schwarz Method

Skipping the explicit formulation of N coupled subproblems similar to that of the two-subdomain case in (7), with the multiplicative Schwarz method, one implicitly carries out the following relaxation steps:

$$\begin{aligned} u^{k+1/N} &= u^k + B_1\big(f - Au^k\big) \\ u^{k+2/N} &= u^{k+2/N} + B_2\big(f - Au^{k+2/N}\big) \\ &\vdots \\ u^{k+1} &= u^{k+(N-1)/N} + B_N\big(f - Au^{k+(N-1)/N}\big)\,. \end{aligned} \tag{4.14}$$

Again, coarse-grained parallel computation cannot be employed here, since every step depends on the results of the previous one.

However, with a variant to (4.14), relaxations on mutually non-overlapping subdomains are carried out at the same time, In particular, it is considered a coloring of the subdomains with a minimum number of K colors such that overlapping subdomains always have differing ones (see, e.g., Fig. 4.2) in case of a rectangular decomposition (hatches there). Then, relaxations which correspond to subdomains of the same color are executed in parallel, while those of different color are carried out consecutively. Let $\{P_1, \ldots, P_K\}$ be the set of subdomain indices for each of the K colors. Then the iteration rule reads

$$\begin{aligned} u^{k+1/K} &= u^k + \sum_{i \in \mathrm{P}_1} B_i\big(f - Au^k\big) \\ u^{k+2/K} &= u^{k+2/K} + \sum_{i \in \mathrm{P}_2} B_i\big(f - Au^{k+2/K}\big) \\ &\cdots \\ u^{k+1} &= u^{k+(M-1)/K} + \sum_{i \in \mathrm{P}_K} B_i\big(f - Au^{k+(K-1)/K}\big)\,. \end{aligned} \tag{4.15}$$

Obviously, this scheme is different from (4.14) in the way update information is propagated, namely not sequentially from the unknowns on Ω_1 to Ω_N, but between the differently colored groups of subdomains. However, this is known to have no significant difference in the overall convergence speed.

4.1.2.2 The Additive Schwarz Method

In the extreme case of having as many colors as subdomains one obtains the additive Schwarz method on multiple subdomains:

$$u^{k+1} = u^k + \sum_{i=1}^{N} B_i\big(f - Au^k\big) \;, \tag{4.16}$$

which provides the highest degree of parallelism, but the lowest degree of information propagation per iteration. Even worse, it is known, see e.g., [112], that convergence is not guaranteed while using it as Richardson preconditioner. Therefore, it is typically used a as a building block in multi-level methods. (See below).

4.1.2.3 Schwarz Methods as Parallel Preconditioners

As with case of two subdomains, both the multiplicative and the additive method can be stated as single preconditioners B being applied to the residual $r^k = f - Au^k$ in a Richardson iteration, which we will do in the following.

Whereas for the additive method we just have

$$B_{AS} := \sum_{i=1}^{N} B_i = \sum_{i=1}^{N} R_i^{I^\top} A_i^{-1} R_i^I \;, \tag{4.17}$$

for the multiplicative method, let us recall the error propagation rule in (4.10) for the case of two subdomains, which, for the purely sequential algorithm, is of the form

$$e^{k+1} = (I - B_N A)\cdots(I - B_2 A)(I - B_1 A)\, e^k \tag{4.18}$$

$$(\hat{u} - u^{k+1}) = (I - B_N A)\cdots(I - B_2 A)(I - B_1 A)(\hat{u} - u^k) \;. \tag{4.19}$$

Subsequent subtraction of $\hat{u}$ on both sides, substitution of $\hat{u}$ by $A^{-1}f$ and some minor modifications then yield the iteration rule

$$u^{k+1} = u^k + \big(I - (I - B_N A)\cdots(I - B_2 A)(I - B_1 A)\big) A^{-1}(f - Au^k) \;. \tag{4.20}$$

Hence, the multiplicative method is equivalent to applying the preconditioner

$$B_{MS} := \Big(I - \prod_{i}^{N}(I - B_i A)\Big) A^{-1} = \Big(I - \prod_{i}^{N}(I - R_i^{I^\top} A_i^{-1} R_i^I A)\Big) A^{-1} \;. \tag{4.21}$$

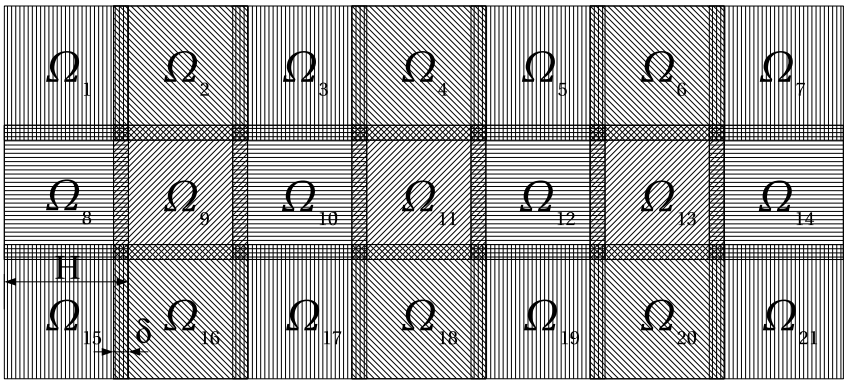


Figure 4.2: **Example of an overlapping decomposition with coloring as employed with the parallelized multiplicative Schwarz iteration.**

4.1.2.4 Links to Gauss-Seidel and Jacobi Iteration

Interestingly, the well-known Gauss-Seidel and Jacobi iterations can be interpreted as degenerate cases of multiplicative and additive Schwarz iterations. For both cases, a degenerate decomposition is assumed such that the subdomains have no variables in common, i.e. $\overline{\Omega}_i \cap \overline{\Omega}_j = \emptyset, \forall j \neq i$, and each subdomain consists of only one node. That is, the number of subdomains is equal to the total number of unknowns which is equal to the number of nodes on Ω. Consequently, each of the matrices A_i^I, $i = 1, \ldots, N$ is is dimension 1×1 only, and contains the diagonal element of the original operator matrix A_{ii}. Hence, in that case the iteration rule for the additive Schwarz method can be written as

$$u^{k+1} = u^k + \sum_{i=1}^{N} R_{I_i}^\top D_{ii}^{-1} R_{I_i} \big(f - Au^k\big) = u^k + D^{-1}\big(f - Au^k\big)\,, \tag{4.22}$$

with $D = \mathrm{diag}(A)$, which describes the Jacobi iteration.

On the other hand, the multiplicative iteration can be stated as

$$\begin{aligned} u^{k+1/N} &= u^k + R_{I_1}^\top D_{11}^{-1} R_{I_1} \big(f - Au^k\big) \\ u^{k+2/N} &= u^k + R_{I_2}^\top D_{22}^{-1} R_{I_2} \big(f - Au^k\big) \\ &\cdots \\ u^{k+N/N} &= u^{k+(N-1)/N} + R_{I_N}^\top D_{NN}^{-1} R_{I_N} \big(f - Au^k\big)\,, \end{aligned} \tag{4.23}$$

which is equivalent to the steps

$$\text{for } i = 1, \ldots, N \text{ do} \quad \big(u\big)_i \leftarrow \big(u\big)_i + \frac{1}{D_{ii}} \big(f - Au\big)_i\,, \tag{4.24}$$

describing Gauss-Seidel iteration applied to $A\,u = f$.

Furthermore, when assuming each subdomain to contain more than one node, but still have no overlap among each other, both Schwarz methods correspond to *block* Jacobi and block Gauss-Seidel iterations, respectively, being applied to the system

$$\begin{pmatrix} A_1^{II} & 0 & 0 & \dots \\ 0 & A_2^{II} & 0 & \dots \\ \vdots & \vdots & \ddots & \\ 0 & 0 & \dots & A_N^{II} \end{pmatrix} u = f, \tag{4.25}$$

which becomes clear if one replaces D_{ii} by A_i^{II} in (4.22) and (4.23) above.

Moreover, in the case of overlapping subdomains each containing at least one non-overlapped node, the procedures described in (4.14) and (4.16) can be interpreted as generalized block Gauss-Seidel or generalized block Jacobi iterations, respectively, i.e. those with an overlap, being applied to the LSE

$$\begin{pmatrix} A_1^{II} & A_1^{I\Gamma} R_{\Gamma_1} R_{I_2}^\top & A_1^{I\Gamma} R_{\Gamma_1} R_{I_3}^\top & \dots & A_1^{I\Gamma} R_{\Gamma_1} R_{I_N}^\top \\ A_2^{I\Gamma} R_{\Gamma_2} R_{I_1}^\top & A_2^{II} & A_2^{I\Gamma} R_{\Gamma_2} R_{I_3}^\top & \dots & A_2^{I\Gamma} R_{\Gamma_2} R_{I_N}^\top \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_N^{I\Gamma} R_{\Gamma_N} R_{I_1}^\top & A_N^{I\Gamma} R_{\Gamma_N} R_{I_2}^\top & \dots & \dots & A_N^{II} \end{pmatrix} \begin{pmatrix} u_1^I \\ u_2^I \\ \vdots \\ u_N^I \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{pmatrix}. \tag{4.26}$$

There, the block-matrices A_i^{II} do overlap, unlike in (4.25), i.e. have those coefficients in common which lie in the overlap regions.

4.1.2.5 Scalability Characteristics and Comparison to Substructuring

For the case of elliptic problems, as with our model problems, the condition number of the multiplicative and additive one-level preconditioners exhibit a strongly dependence on the number of subdomains by which Ω is covered. In particular, the number of iterations grows proportionally with $1/H$ when employing them within PCG iteration, i.e. the with square root of the number of subdomains in case of quadratic subdomains. As with the one-level iterative substructuring methods, the reason for this effect becomes clear in view of the information propagation argument: within each iteration, update information is transported further for only one subdomain. Again, in terms of parallel scalability, this is a major drawback, since with increasing number of subdomains, and processing nodes respectively, the speed gain by parallelization is quickly taken away by the additional number of iterations necessary.

As opposed to the non-overlapping methods, the overlap width δ has a significant influence on the convergence rate as well. Typical values for δ are 10 to 20 percent of H, while the convergence rate deteriorates significantly when there is no overlap, i.e. information exchange takes place via the shared boundaries only. In addition, it is known, see e.g. [113], that the convergence speed is independent from the total number of unknowns, if the overlap δ is kept proportional to H. That is, then it is independent of the discretization mesh size h.

On the other hand, having to exchange more than the boundary variables between adjacent subdomain processes increases the communication volume involved with each iteration by several times. Consequently, non-overlapping methods running on distributed parallel machines exhibit, in general, a better scalability, since the communication volumes and therefore the communication costs grow with a smaller magnitude than the number of subdomains. However, the implementation of the overlapping algorithms is much less complex since only restrictions of the original operator matrix A are involved. Whereas with the iterative substructuring approaches, the local matrices need to be disassembled in various manners, depending on the specific substructuring algorithm, which renders the implementation of the overall solving scheme more complex.

4.2 Multi-level Algorithms

In order to make convergence rates independent from the number of subdomains, addititional coarse-level relaxations are introduced, which yield a global, though coarse, information propagation in each iteration. Unlike the two-level preconditioners presented in Chapter 3, several levels of relaxations based on different mesh sizes are employed, similar to multigrid approaches. On each level, either the additive or multiplicative one-level overlapping method is applied. In addition, the relaxations on each resolution level can be carried out consecutively or in parallel, which is also referred to by the adjectives multiplicative or additive, respectively.

First we will consider the case of only one coarse relaxation and explain the employment of either Schwarz method. Then we focus on the case of an arbitrary number of resolution levels, describing each reasonable combination of additive and multiplicative methods and address their use for parallel computing.

4.2.1 Two-Level Methods

Let $\mathcal{T}^C$ be a grid on Ω with a coarser mesh-size than $\mathcal{T}^h$, and let $A_C u_C = f_C$ be the corresponding system of nodal variables resulting from a finite element discretization of the model problem. Moreover, let the operator R_C appropriately restrict a nodal

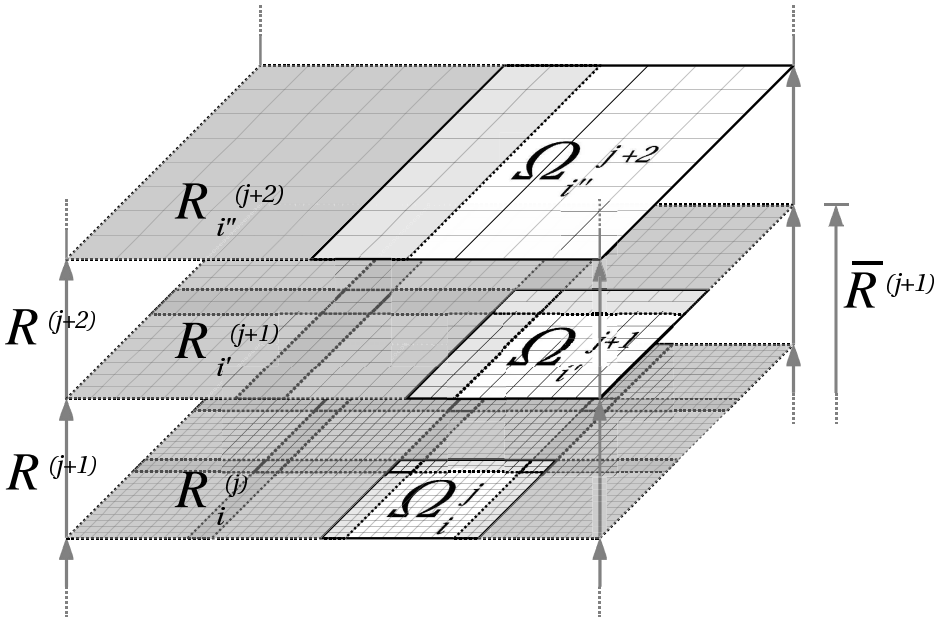


Figure 4.3: **Illustration of the different restriction operators employed with the multi-level methods.**

vector u of the fine discretization $A\,u = f$ on Ω to u_C. Vice-versa, $R_C^\top$ denotes an interpolation.

Making use of these definitions, the multiplicative two-level method is described by the following iteration rule:

$$\begin{aligned} u^{k+1/2} &= u^k + R_C^\top A_C^{-1} R_C \big(f - A u^k\big) \\ u^{k+1} &= u^{k+1/2} + B\big(f - A u^{k+1/2}\big)\,, \end{aligned} \tag{4.27}$$

while B stands for one of the previously presented one-level Schwarz preconditioners. That is, additive or multiplicative Schwarz relaxation steps are carried out alternating with the coarse ones, while the latter involve the non-decomposed solving of a global, yet smaller LSE involving A_C. Due to the latter step, update information, though aggregated, is exchanged between all unknowns mutually for each iteration. With the additive variant,

$$u^{k+1} = u^k + \big(B + R_C^\top A_C^{-1} R_C\big)\big(f - A u^k\big)\,, \tag{4.28}$$

the coarse-grid relaxation is done in parallel with the fine ones, thereby increasing parallelism but worsening the convergence rate.

By having introduced a global updating step, the iteration speed no longer deteriorates when increasing the number of subdomains, i.e. it is independent of $1/H$. However, with this two-level methods, there is still a rather low limit in terms of parallel scalability. This is because of the computation time for sequentially inverting A_C fully compensating the gain of solving smaller local problems associated with B in parallel.

4.2.2 Multiplicative Multi-level Methods

In aiming to exploit parallelism with the coarse relaxation steps too, so-called *multi-level methods* are constructed by recursively applying the two-level method to the coarse step. That is, the action of A_C^{-1} in (4.27) is again approximated by method (4.27) this time considering $\mathcal{T}^H(\Omega)$ as the fine level.

In order to formalize this notion, we first need to define the following notations. We consider M different grids $\mathcal{T}^{H_j}(\Omega), j = 1, \ldots, M$ on Ω, whose mesh sizes H_j increase with j and which will be referred to as *levels* in the following. Besides the corresponding finite element discretizations $A^{(j)}u^{(j)} = f^{(j)}$ on each level, let $R^{(j)}$ be the restriction of a nodal vector defined on grid $\mathcal{T}^{H_j}$ to the next coarser grid $\mathcal{T}^{H_{j+1}}$, and $R^{(j)^\top}$ the corresponding interpolation operator. Moreover, we will make use of cumulative variants $\bar{R}^{(j)} := \prod_{l=1}^{j} R^{(l)}$ restricting directly from the finest to the j-th level.

On each level j, we consider an overlapping covering $\{\Omega_i^j, j = 1, \ldots, N_j\}$, which is compatible with $\mathcal{T}^{H_j}$, with the same properties as in the one-level case, and N_j denoting the number of subdomains. Furthermore, on each level, we will make use of operators $R_i^{(j)}$, which restrict a nodal vector corresponding to the whole mesh $\mathcal{T}^{H_j}$ to one corresponding to the subdomain mesh $\mathcal{T}^{H_j}_{|\overline{\Omega}_i^j}$ only. See Figure 4.3 for an illustration.

By means of this notation, we can formalize the idea which was outlined above, and state the following iteration rule:

$$\begin{aligned}
u^{(k+\frac{1}{M})} &= u + \bar{R}^{(M)^\top} A^{(M)^{-1}} \bar{R}^{(M)} \Big(f - A u^{(k)} \Big) \\
u^{(k+\frac{2}{M})} &= u + \bar{R}^{(M-1)^\top} B^{(M-1)} \bar{R}^{(M-1)} \Big(f - A u^{(k+\frac{1}{M})} \Big) \\
&\vdots \\
u^{(k+\frac{M}{M})} &= u + B^{(1)} \Big(f - A u^{(\frac{M-1}{M})} \Big),
\end{aligned}$$

with $B^{(j)}$ representing either the additive or the multiplicative Schwarz preconditioner, as given in Section 4.1.2.3, applied to level $\mathcal{T}^{H_j}$.

The number of levels M is usually adapted to the decomposition size, i.e. an optimal trade-off between the computation time for inverting $A^{(M)}$ at level M and the computation and communication time for additional Schwarz relaxations on intermediate levels (including all additional restriction and interpolation operations) can be sought, while keeping the overlap on each level low.

Since the relaxations are carried out consecutively, these algorithms are denoted as *multiplicative multi-level* (independent from the type of Schwarz preconditioners used on each level).

Although the steps in (4.2.2) give a good description of the main idea, it is in general not an effective formulation for implementation. The reason for this is that all assignments are formulated on the finest grid, i.e. each residual would be calculated on the finest grid, then restricted across several levels to the level on which the preconditioner is applied, followed by an interpolation of update back to the finest grid. Instead, in cases where the $\{A^{(j)}\}$ are Galerkin, i.e. either $A^{(j)}$ or the $R^{(j)}$ have been constructed such that $A^{(j+1)} = R^{(j+1)}A^{(j)}R^{(j+1)^\top}$, for $j = 1, \ldots, M-1$ holds, the updating steps in (4.2.2) can be done only on the current grid level j and the result is interpolated only to the next finer one. As an algorithm, this reads:

Algorithm 9: Multiplicative multi-level Richardson iteration

initialize u^0 by an arbitrary value

iterate $k = 1, 2, \ldots$ **until** convergence

for $j = 1, 2, \ldots, M-1$ **do** $u^{(j+1)} \leftarrow R^{(j+1)}u^{(j)}$

$$
\begin{aligned}
r^{(M)} &\leftarrow (f^{(M)} - A^{(M)}u^{(M)}) \\
u^{(M)} &\leftarrow u^{(M)} + A^{(M)^{-1}}r^{(M)} \\
r^{(M-1)} &\leftarrow R^{(M)^\top}(f^{(M)} - A^{(M)}u^{(M)}) \\
u^{(M-1)} &\leftarrow u^{(M-1)} + B^{(M-1)}r^{(M-1)} \\
&\vdots \\
r^{(j)} &\leftarrow R^{(j+1)^\top}(f^{(j+1)} - A^{(j+1)}u^{(j+1)}) \\
u^{(j)} &\leftarrow u^{(j)} + B^{(j)}R^{(j)}r^{(j)} \\
&\vdots \\
r^{(1)} &\leftarrow R^{(2)^\top}(f^{(2)} - A^{(2)}u^{(2)}) \\
u^{(1)} &\leftarrow u^{(1)} + B^{(1)}R^{(1)}r^{(1)}
\end{aligned}
\tag{4.29}
$$

where the for-loop is necessary for the initialization of the $u^{(j)}$ at each iteration.

Obviously, this variant requires less sequential instructions in comparison to Alg. (4.2.2), since up to the last relaxation, all correction steps are executed on the coarser grids having less nodes, i.e. fewer unknowns. In addition, the operators $\{R^{(j)}\}$ as well as their inverses need fewer instructions than does their cumulative versions $\{\bar{R}^{(j)}\}$ for $j > 1$, since they operate on smaller vectors. The only additional costs, in terms of execution time, are due to the initial calculation of $A^{(j)}$ and $f^{(j)}$ for $j > 1$ before the iteration as well as the computation of the $u^{(j)}$.

In comparison to the multiplicative two-level method, the advantage of the multi-level extension is the much higher degree of parallel scalability, since the non-parallelizable inversion on the coarsest grid here requires only an negligible computational effort since A^M is much smaller than A_C.

4.2.3 Additive Multi-level Methods

In order to gain parallelism across the levels, the per-level steps in Algorithm 9 can also be carried out consecutively (while using u^k on all right-hand and u^{k+1} on all left-hand sides), which gives the *additive multi-level methods*, similar to those in the two-level case. Here too, the convergence rate deteriorates in comparison to the multiplicative algorithms, though the total parallelization gain can — depending on the problem and the computing hardware — be higher.

Furthermore, in the case of the additive multi-level algorithm with additive Schwarz relaxation steps on each level, the highest degree of parallelization is achievable. Typically, that is implemented by combining the level restriction operators $\bar{R}^{(j)}$ and the subdomain restriction operators $R_i^{(j)}$ on each level to a single operator:

$$\bar{R}_i^{(j)} := R_i^{(j)} \bar{R}^{(j)}, \quad i = 1, \ldots, N^{(j)}, \quad j = 1, \ldots, M \ , \tag{4.30}$$

which restricts a vector defined on whole $\overline{\Omega}_i$ to the subdomain $\overline{\Omega}_i^{(j)}$ on level j directly. Thereby, the $\sum_{j=1}^{K} M_j$ relaxations

$$u \leftarrow u + \left(\sum_{j=1}^{K} \sum_{i=1}^{M_j} \bar{R}_i^{(j)^\top} A_i^{(j),II^{-1}} \bar{R}_i^{(j)} \right) \left(f - Au \right) \ , \tag{4.31}$$

can be calculated independently.

4.2.4 Multi-level Methods as Parallel Preconditioners

As with the one-level methods, overlapping multi-level methods are typically used as preconditioners in a Krylov subspace iteration, such as PCG or GMRES. There,

besides the operator application step $A \cdot x$, for some vector x, one has to implement the application of the preconditioner B to a vector r.

With the additive multi-level method this is straightforward, namely by replacing the term $\big(f - Au\big)$ by a residual function r in (4.21). For the multiplicative method however, as given by Algorithm 9, one has to replace every occurrence of f by r as well as initializing u by zero, which therefore gives a slightly different procedure:

Algorithm 10: The multiplicative multi-level method as preconditioner

$$
\begin{aligned}
& r^{(1)} \leftarrow r \\
& \textbf{iterate } k = 1, 2, \ldots \textbf{ until } \text{convergence} \\
& \quad \textbf{for } j = 1, 2, \ldots, M-1 \textbf{ do } r^{(j+1)} \leftarrow R^{(j+1)} r^{(j)} \\
& \qquad u^{(M)} \leftarrow {A^{(M)}}^{-1} r^{(M)} \\
& \qquad r^{(M-1)} \leftarrow {R^{(M)}}^{\top} (r^{(M)} - A^{(M)} u^{(M)}) \\
& \qquad u^{(M-1)} \leftarrow B^{(M-1)} r^{(M-1)} \\
& \qquad\quad \vdots \\
& \qquad r^{(j)} \leftarrow {R^{(j+1)}}^{\top} (r^{(j+1)} - A^{(j+1)} u^{(j+1)}) \\
& \qquad u^{(j)} \leftarrow B^{(j)} R^{(j)} r^{(j)}, \quad j = M-3, \ldots, 1
\end{aligned}
\tag{4.32}
$$

This gives a non-symmetric preconditioner, suitable for GMRES or BiCG-Stab iteration, for example. A symmetrized version is reached by carrying out the pairwise steps in Alg. (10) in reverse order, i.e. starting from the finest level up to the coarsest one, see, e.g., [113].

4.2.5 Links to Multigrid Methods

Algorithm (10) can be written in a recursive manner too. In case of the symmetric multiplicative preconditioner with N_1 pre-recursive and N_2 post-recursive Schwarz relaxation steps, for example, one obtains Algorithm 11, with $r \leftarrow \textbf{v-cycle}(r, 0, 1)$.

Interestingly, in the extreme case of having only one node per subdomain and no overlap in connection with additive Schwarz relaxation steps, i.e. $B^{(j)} = {D^{(j)}}^{-1}$, with $D^{(j)}$ being the diagonal of $A^{(j)}$, Algorithm 11 is equivalent to a multigrid v-cycle with Jacobi smoothing, which is also denoted as *diagonal scaling* in the domain decomposition context. In the case of multiplicative Schwarz relaxation steps on each level, this corresponds to Gauss-Seidel smoothing. In those cases, Algorithm 11 is utilized as independent solver and not as preconditioner, while $u \leftarrow \text{v-cycle}(f, 0, 1)$ gives the solution. Furthermore, when reverting back to the

Algorithm 11: V/W-cycle multigrid with Schwarz smoothing

v-cycle $(r^{(j)}, u^{(j)}, j)$
  if $j == M$ **then**
    $u^{(M)} \leftarrow u^{(M)} + A^{(M)^{-1}} r^{(M)}$
  else
    for $i = 1, \ldots, N_1$ **do** $u^{(j)} \leftarrow u^{(j)} + B^{(j)}\big(r^{(j)} - A^{(j)} u^{(j)}\big)$
    $u^{(j)} \leftarrow u^{(j)} + R^{(j+1)^\top}$ **v-cycle**$\Big(R^{(l+1)}\big(r^{(j)} - A^{(j)} u^{(j)}\big), 0, j+1\Big)$
    for $i = 1, \ldots, N_2$ **do** $u^{(j)} \leftarrow u^{(j)} + B^{(j)}\big(r^{(j)} - A^{(j)} u^{(j)}\big)$
  end
  return $u^{(j)}$

standard case of overlapping subdomains, the additive and multiplicative Schwarz methods are denoted *Schwarz smoothers*.

Finally, the *full multigrid* algorithm is reached, if the v-cycles are started not only one time from the finest level, but used to correct a coarse intermediate solution $u^{(j)}$ on every grid level (except for the coarsest one), which gives Algorithm 12.

Although one can interpret the multiplicative multi-level methods as a generalized multigrid methods, note that the motivation to incorporate relaxations steps on coarser levels for the latter is of a slightly different nature. Multigrid methods rely on the fact that with the standard iterative LSE solving methods lower spatial frequency components in the error decline much slower that those with high spatial frequencies. Hence, the idea is to do certain iterations on coarser grids where a part of the persisting error appears as having high spatial frequencies. Recursion is used to exploit the fast attenuation of the iterative solver for almost all frequency components, with respect to the original resolution of the error. In terms of domain decomposition methods on the other hand, coarse-level iterations have been introduced to overcome locality of the error correction propagation arising from the spatial decomposition. Recursion there is used as a mean to exploit parallelism for the coarse iterations too.

Algorithm 12: Full multigrid with Schwarz smoothing

$$
\begin{aligned}
r^{(1)} &\leftarrow f^{(1)} \\
r^{(2)} &\leftarrow R^{(2)} r^{(1)} \\
&\vdots \\
r^{(M)} &\leftarrow R^{(M)} r^{(M-1)} \\
u^{(M)} &\leftarrow {A^{(M)}}^{-1} r^{(M)} \\
u^{(M-1)} &\leftarrow {R^{(M)}}^{\top} u^{(M)} \\
u^{(M-1)} &\leftarrow u^{(M-1)} + \textbf{v-cycle}(r^{(M-1)}, u^{(M-1)}, M-1) \\
u^{(M-2)} &\leftarrow {R^{(M-1)}}^{\top} u^{(M-1)} \\
u^{(M-2)} &\leftarrow u^{(M-2)} + \textbf{v-cycle}(r^{(M-2)}, u^{(M-2)}, M-2) \\
&\vdots \\
u^{(1)} &\leftarrow u^{(1)} + \textbf{v-cycle}(r^{(1)}, u^{(1)}, 1).
\end{aligned}
$$

4.2.6 Scalability and Comparison to Iterative Substructuring

Given a sufficient number of coarse-level updates, the convergence rates of the above-explained multi-level preconditioners are independent of the number of subdomains, since thereby update information is propagated between all subdomains on the finest level within each iteration. Naturally, the influence of the overlap widths δ on each level on the convergence speed still remains, similar to the single-level case. Again, for a fix ratio δ/H, the number of iterations becomes independent of the total number of unknowns.

In comparison to the two-level substructuring methods, obviously, multi-level overlapping methods involve much more total costs in comparison to their single-level level counterparts. Whereas with the former a small, non-decomposed system with a few unknowns per subdomain has to be inverted on the coarse level, with the latter, the one-level Schwarz updates need to conducted on several, incrementally coarser levels, resulting in significant additional computation and communication demands.

On the other hand, and especially with the two-level approaches, the non-overlapping procedures require more complex implementations again, in particular since for the overlapping ones same or very similar one-level schemes are to be carried

out on each of the different levels.

4.3 Summary

In this chapter, we presented the common overlapping domain decomposition methods after having explained their foundation on classical two-subdomain Schwarz methods. Besides relations to standard multigrid methods, we focused on the different factors influencing the convergence rates and discussed the disadvantages of overlapping methods versus non-overlapping ones, depending on the computational environment.

Chapter 5

Motion Estimation with High-order Regularization

Though the main focus of the first half of this work is on studying the parallelization of linear PDE-based problems at the example of CLG motion estimation, this chapter is devoted to the model problem itself and in particular with respect to computational fluid mechanics.

In a number of domains affecting our everyday life, the analysis of image sequences involving fluid phenomena is of importance. This includes, for instance, domains such as visualization in experimental fluid mechanics [2, 96], environmental sciences (meteorology [6, 39, 84, 99, 131], oceanography [41]) or medical imaging [3, 56, 115]. For all these domains it is of primary interest to extract reliable velocity fields, though this is far from being the ultimate goal of the analysis. Differential or integrated information from the velocity field is indeed far more valuable for concerned experts. For example, it is essential to characterize fluid flows to extract the vorticity fields, the streamlines, or the singular points of the flows. All these features may be estimated *indirectly* from the velocity field by differentiation or by integration. Among all these information, the two potential functions called the *velocity potential* and the *stream function* are of great interest: (*i*) their gradients provide a description of the *irrotational* and the *solenoidal* components of the velocity fields; (*ii*) their Laplacians give access to the *vorticity* and the *divergence* of the velocity fields; (*iii*) their level lines allow us to extract directly the streamlines and the equipotential curves of the velocity potentials; (*iv*) their extrema provide the location of the singular points of major interest [40] (namely *sources, sinks and vortexes*).

Knowing the curl and the divergence of the flow, the extraction of such potential functions can be done by solving two Dirichlet problems. Such an estimation is

particularly difficult for sparse velocity fields such as those obtained by the usual correlation methods [96] since an additional interpolation step is needed [108, 109]. Dense motion estimation, on the other hand, allows us to recover these potential functions more accurately. However, such an estimation as proposed in [40] is not "direct". It requires a process of three steps: First, the motion field has to be extracted, next the motion field is separated into its irrotational and solenoidal components in Fourier space, and finally the potential functions are estimated from these two vector fields by integration. Obviously, this not only involves a rather complex implementation, but also the problem of numerical inaccuracies and image artifacts. By contrast, we propose to estimate *directly* the velocity potential and the stream function in one joint energy minimization directly. Thereby, one problem is the high degree of derivatives involved in the regularization term, which we tackle by means of auxiliary functions, similarly to the approach in [40].

The organization of this chapter is as follows: First, we will give mathematical definitions of the aforementioned flow field components and their representation by potential functions. Second, we will gradually develop our new direct estimation approach while especially detailing on the regularization involving third-order derivatives. Third, we will present thorough experimental studies on synthetic and real data.

5.1 The Helmholtz Decomposition

Let us consider the smooth optical flow field $u = (u_1(x, y), u_2(x, y))^\top$ defined over the image plane section Ω, as defined in Chapter 2. Without loss of generality we can extend u to the whole plane and assume that it vanishes at infinity. Then it allows for the decomposition into a divergence free component (denoted *solenoidal*) and a curl free component (denoted as *irrotational*), which is known as the *Helmholtz Decomposition* of a vector field:

$$u = u_{so} + u_{ir} \,, \tag{5.1}$$

with

$$\text{div } u_{so} = \frac{\partial u_1}{\partial x} + \frac{\partial u_2}{\partial y} = 0 \quad \text{and} \quad \text{curl } u_{ir} = -\frac{\partial u_1}{\partial x} + \frac{\partial u_2}{\partial y} = 0 \,. \tag{5.2}$$

In case of a non-zero border condition, the decomposition also includes a *laminar* component which is neither irrotational nor solenoidal:

$$u = u_{so} + u_{ir} + u_{lam} \,. \tag{5.3}$$

Furthermore, it is well known that both u_{so} and u_{ir} can be derived from potential functions ϕ and $\psi : \Omega \times [1, T] \to \mathbb{R}$ denoted as *stream potential* and *velocity potential*, respectively:

$$u_{ir} = \nabla\phi = \left(\frac{\partial\phi}{\partial x}, \frac{\partial\phi}{\partial y}\right)^\top \tag{5.4}$$

$$u_{so} = \nabla\psi^\perp = \left(-\frac{\partial\psi}{\partial y}, \frac{\partial\psi}{\partial x}\right)^\top . \tag{5.5}$$

Thus, an optical flow field can be represented uniquely by

$$u = \nabla\phi + \nabla\psi^\perp + u_{lam} . \tag{5.6}$$

Following the proposition by Corpetti et al. [39, 40], we assume u_{lam} to be estimated separately by the (multi-resolution) Horn-Schunck method with a very strong regularization and to be removed from the image pair in advance.

5.2 Direct Estimation of the Potential Functions

Let us start with the brightness constancy equation, (2.1), using the such represented vector field:

$$I(\mathbf{x} + \nabla\phi(\mathbf{x}, t) + \nabla\psi^\perp(x, t), t + 1) = I(\mathbf{x}, t) , \tag{5.7}$$

and $\mathbf{x} := (x, y)$. Embedding into an energy framework yields the energy functional

$$J(\phi, \psi) := \int_\Omega \Big(I(\mathbf{x} + \nabla\phi(\mathbf{x}, t) + \nabla\psi^\perp(x, t),\ t + \Delta t) - I(\mathbf{x}, t)\Big)^2 d\mathbf{x} . \tag{5.8}$$

Similar to the Horn-Schunck approach in Chapter 2, the idea is linearize with respect to the displacement functions. Here, this means to apply a first-order Taylor series expansions for $\nabla\psi^\perp$ and $\nabla\phi$ separately, giving the two energy functionals

$$\begin{aligned} J_1(\phi, \psi) &:= \int_\Omega \Big(\nabla I(\mathbf{x} + \nabla\phi,\ t + \Delta t)^\top \nabla\psi^\perp + I(\mathbf{x} + \nabla\phi,\ t + \Delta t) - I(\mathbf{x}, t)\Big)^2 d\mathbf{x} \\ J_2(\phi, \psi) &:= \int_\Omega \Big(\nabla I(\mathbf{x} + \nabla\psi^\perp, t + \Delta t)^\top \nabla\phi + I(\mathbf{x} + \nabla\psi^\perp,\ t + \Delta t) - I(\mathbf{x}, t)\Big)^2 d\mathbf{x} . \end{aligned}$$

By means of the notation

$$\nabla I_\xi(\mathbf{x}) := \nabla I(\mathbf{x} + \nabla\xi, t + \Delta t) \tag{5.9}$$

$$\partial I_\xi := I(\mathbf{x} + \nabla\xi,\ t + \Delta t) - I(\mathbf{x}, t), \qquad \xi \in \{\phi, \psi\} , \tag{5.10}$$

we write those compactly as

$$J_1(\phi,\psi) = \int_\Omega \left(\nabla I_\phi^\top \nabla\psi^\perp + \partial I_\phi\right)^2 d\mathbf{x} \tag{5.11}$$

$$J_2(\phi,\psi) = \int_\Omega \left(\nabla I_\psi^\top \nabla\phi + \partial I_\psi\right)^2 d\mathbf{x}\,. \tag{5.12}$$

Thereby, the idea is to minimize in alternation each energy with respect to the function for which it was linearized, respectively, while keeping the other function fixed. Specifically, the first variation of J_1 w.r.t. ϕ is set equal to zero, as well as the first variation of J_2 w.r.t. ψ is set to zero, yielding the following coupled equations:

$$\begin{cases} \displaystyle\int_\Omega \left(\left(\nabla I_\phi \nabla I_\phi^\top\right)\nabla\psi^\perp + \partial I_\phi \nabla I_\phi^\top\right)\nabla\tilde{\psi}^\perp\, d\mathbf{x} = 0 \\ \displaystyle\int_\Omega \left(\left(\nabla I_\psi \nabla I_\psi^\top\right)\nabla\phi + \partial I_\psi \nabla I_\psi^\top\right)\nabla\tilde{\phi}\, d\mathbf{x} = 0\,, \end{cases} \tag{5.13}$$

with $\tilde{\phi}$ and $\tilde{\psi}$ denoting the arbitrary test functions. Note that we still have an under-determined problem, which is reflected by the fact the matrices $\nabla I_\phi \nabla I_\phi^\top$ and $\nabla I_\psi \nabla I_\psi^\top$ in (5.13) are singular.

As an ad-hoc regularization, we added small scalars ϵ to the diagonal entries of those to two matrices, i.e. the innermost brackets in (5.13) are replaced by $\nabla I_\phi \nabla I_\phi^\top + \epsilon I$ and $\nabla I_\psi \nabla I_\psi^\top + \epsilon I$, with I referring to the identity matrix, respectively. However, while carrying out several experiments using a first-order finite elements discretization, this has emerged to be an inappropriate approach, since it leads to an systematic underestimation of ψ and ϕ. As explanation for this observation, it became clear that additional terms of the form $\epsilon\nabla\psi^\perp\nabla\tilde{\psi}^\perp$ and $\epsilon\nabla\phi\nabla\tilde{\phi}$, to the integrands in (5.13), respectively, correspond to additional terms $\epsilon\|\nabla\psi^\perp\|^2$ and $\epsilon\|\nabla\phi\|^2$ in (5.11) and (5.12), respectively. Thereby, any magnitudes of the solenoidal field, $\nabla\psi^\perp$, and the irrotational field, $\nabla\phi$, are penalized while seeking for the minima of J_1 and J_2, respectively, which explains the systematic underestimation.

5.3 A Structure-preserving Regularization

5.3.1 The Approach

This gives raise for better a regularization. In particular, an important property besides non-biasing is the preservation of sink-, source- and vortex-like patterns, which argues against standard regularizations as employed with the Horn-Schunck approach. In other works, we aim at extending the general smoothness assumption on the optical flow field towards a notion of smoothness by means of those structures.

The starting point thereby was the work of Suter [116] where it was proposed to use a so-called second-order div-curl regularization:

$$\int_\Omega \|\nabla \text{div } u\|^2 + \|\nabla \text{curl } u\|^2 \, d\mathbf{x} \,, \tag{5.14}$$

i.e. not discontinuities in the flow field itself, but in the divergence and curl are penalized. However, since we are interested in directly estimating the potential functions ϕ and ψ instead of u, we modify it by replacing u by the representation introduced in (5.6) (having $u_{lam} \equiv 0$), by which we obtain

$$\int_\Omega \|\nabla \text{div } \nabla\phi\|^2 + \|\nabla \text{curl } \nabla\psi^\perp\|^2 \, d\mathbf{x} \,, \tag{5.15}$$

since div $\nabla\psi^\perp = 0$ and curl $\nabla\phi = 0$ by definition (see (5.2), (5.4-5.5)).

Unfortunately, the third-order derivatives in (5.15) make a direct numerical approach quite complicated. To be concrete, the corresponding Euler-Lagrange equations would contain sixth-order derivatives of ϕ and ψ, respectively. To remedy this numerical problem, we subsequently follow the approach of Corpetti et al. [39] and introduce *auxiliary variables* ξ_1 and ξ_2, enforce them to approximate curl $\nabla\psi^\perp$ and div $\nabla\phi$ by additional (soft) constraints and impose the original discontinuity penalizing constraint on them, i.e.

$$\int_\Omega \gamma \left(\big(\text{div } \nabla\phi - \xi_2\big)^2 + \big(\text{curl } \nabla\psi^\perp - \xi_1\big)^2 \right) + \lambda \left(\|\nabla\xi_2\|^2 + \|\nabla\xi_1\|^2 \right) \, d\mathbf{x} \,, \tag{5.16}$$

with the regularization strength parameters $\gamma, \lambda \in \mathbb{R}^+$. By this, the degree of derivation can be lowered to the order of four in the Euler-Lagrange equations, at the cost of slightly weakening the regularization constraints. Merging the regularization and the data-driven energies (5.8) into one functional finally gives

$$\begin{aligned} J(\phi, \psi, \xi_1, \xi_2) := \int_\Omega & \Big(I(\mathbf{x}, t) - I(\mathbf{x} + \nabla\phi + \nabla\psi^\perp, \; t + \Delta t) \Big)^2 \\ & + \gamma \left(\big(\text{div } \nabla\phi - \xi_2\big)^2 + \big(\text{curl } \nabla\psi^\perp - \xi_1\big)^2 \right) + \lambda \left(\|\nabla\xi_2\|^2 + \|\nabla\xi_1\|^2 \right) d\mathbf{x} \,, \end{aligned} \tag{5.17}$$

which we again linearize with respect to $\nabla\psi^\perp$ and $\nabla\phi$ separately:

$$J_1(\phi, \psi, \xi_1) := \int_\Omega \Big(\nabla I_\phi^\top \nabla\psi^\perp + \partial I_\phi \Big)^2 + \gamma \big(\text{curl } (\nabla\psi^\perp) - \xi_1\big)^2 + \lambda \|\nabla\xi_1\|^2 \, d\mathbf{x} \,, \tag{5.18}$$

$$J_2(\phi, \psi, \xi_2) := \int_\Omega \Big(\nabla I_\psi^\top \nabla\phi + \partial f_\psi \Big)^2 + \gamma \big(\text{div } (\nabla\phi) - \xi_2\big)^2 + \lambda \|\nabla\xi_2\|^2 \, d\mathbf{x} \,, \tag{5.19}$$

while keeping only the involved regularization terms. Again, the idea is to minimize iteratively and by alternation both energies with respect to either ϕ or ψ, respectively, while, in addition, the minima w.r.t. the auxiliary variables have to be calculated.

5.3.2 Discretization and Solving

In order to avoid the implementation efforts involved with finite elements of second order, here a discretization is realized by finite differences. Hence, the Euler-Lagrange equations for J_1 w.r.t. ψ, ξ_1 and for J_2 w.r.t.ϕ and ξ_2, respectively, are determined:

$$\begin{cases} \operatorname{curl}\left(\left(\nabla I_\phi^\top \nabla \psi^\perp + \partial I_\phi\right)^\top \nabla I_\phi\right) + \gamma\left(\Delta^2\psi + \Delta\xi_1\right) = 0 \\ \gamma\left(\Delta\psi + \xi_1\right) - \lambda\Delta\xi_1 = 0 \\ \operatorname{div}\left(\left(\nabla I_\psi^\top \nabla \phi + \partial I_\psi\right)^\top \nabla I_\psi\right) + \gamma\left(\Delta^2\phi - \Delta\xi_2\right) = 0 \\ \gamma\left(-\Delta\phi + \xi_2\right) - \lambda\Delta\xi_2 = 0\,, \end{cases} \tag{5.20}$$

with

$$\Delta^2 = \partial_{x^4} + 2\partial_{x^2y^2} + \partial_{y^4}\,, \tag{5.21}$$

denoting the so-called *biharmonic operator*, which was approximated by a standard 13-points-stencil [68], and require mixed Dirichlet and Neumann boundary conditions

$$\begin{aligned} \phi(\mathbf{x}) &= \sigma_\phi(\mathbf{x}) \quad \wedge \quad \nabla_n\phi(\mathbf{x}) = \rho_\phi(x), \quad \forall \mathbf{x} \in \partial\Omega \\ \psi(\mathbf{x}) &= \sigma_\psi(\mathbf{x}) \quad \wedge \quad \nabla_n\psi(\mathbf{x}) = \rho_\psi(x), \quad \forall \mathbf{x} \in \partial\Omega\,, \end{aligned} \tag{5.22}$$

with boundary functions $\sigma_\phi, \sigma_\psi, \rho_\phi$ and ρ_ψ (see the second but last paragraph in Section 5.4 for details on their concrete selection in our case).

5.3.3 Embedding into a Multi-resolution Framework

Since first experiments with small velocities (below one pixel per image pair) show promising results, we integrate the current approach into a *multiresolution framework* in order to estimate ϕ and ψ also for larger displacements. Therefore, we consider a pyramid of M discretization meshes with the original, the finest one at the bottom being labeled by index 1. Moreover, we consider approximations of every function in (5.18) and (5.19) on each resolution level l – indicating the level by upper indices – while operators P^l provide a prolongation (P^l) or restriction ($P^{l\top}$) from level l to $l-1$ or $l+1$, respectively.

Given the solutions ϕ^{l-1} and ψ^{l-1} from resolution level $l-1$, while at level M with setting $\phi^M = 0$ and $\psi^M = 0$, ϕ^l and ψ^l are calculated by iteratively carrying out

$$(\psi^l, \xi_1^l) = (P\psi^{l-1}, 0) + \arg\min_{\psi,\xi_1} J_1(\phi, P\psi^{l-1}, \psi, \xi_1, \gamma_l, \lambda_l) \tag{5.23}$$

$$(\phi^l, \xi_2^l) = (P\phi^{l-1}, 0) + \arg\min_{\phi,\xi_2} J_2(P\phi^{l-1}, \phi, \psi, \xi_2, \gamma_l, \lambda_l)\,, \tag{5.24}$$

with

$$J_1(\phi, \psi_0, \psi, \xi_1, \gamma, \lambda) := \tag{5.25}$$
$$\int_\Omega \Big(\nabla \bar{I}_\phi^\top \nabla \psi^\perp + \partial \bar{I}_\phi\Big)^2 + \gamma\big(\text{curl}\,\nabla(\psi_0+\psi)^\perp - \xi_1\big)^2 + \lambda\|\nabla\xi_1\|^2\, d\mathbf{x}$$

$$J_2(\phi_0, \phi, \psi, \xi_2, \gamma, \lambda) := \tag{5.26}$$
$$\int_\Omega \Big(\nabla \bar{I}_\psi^\top \nabla \phi + \partial \bar{I}_\psi\Big)^2 + \gamma\big(\text{div}\,\nabla(\phi_0+\phi) - \xi_2\big)^2 + \lambda\|\nabla\xi_2\|^2\, d\mathbf{x}\,,$$

and

$$\begin{aligned} \bar{I}(\mathbf{x}, t) &:= I(\mathbf{x}, t) \\ \bar{I}(\mathbf{x}, t+\Delta t) &:= I(\mathbf{x} + \nabla\phi_0 + \nabla\psi_0^\perp, t+\Delta t). \end{aligned} \tag{5.27}$$

Note that except for the coarsest resolution level M, ϕ and ψ are incremental refinements of the whole solution, thereby describing only small velocities and thus keeping the linearization error low. By contrast, the regularization terms apply to the complete potential functions ($P\phi^{l-1} + \phi$ and $P\psi^{l-1} + \psi$), not only to the increments. For the auxiliary variables, on the other hand, the multi-resolution is not applied, since they are not involved in the linearization. Hence, ξ_1 and ξ_2 are *not* increments, but calculated independently at each resolution.

5.4 Experimental Studies

This section is organized into three parts. First, the quantitative and qualitative influence of the parameters γ and λ is investigated in Section 5.4.1. Second, the proposed method is compared with the method of Corpetti et al. [40] on artificial motion fields, i.e. with ground truth, in Section 5.4.2. Finally, results for a real image sequence will be presented.

In all experiments, the Euler-Lagrange equations (5.20) were solved sequentially in 3000 iterations using an incomplete CLG solver iterating 50 times in each (outer)

iteration. Two resolution levels (including the original one) have been used for the experiments in Sections 5.4.1 and 5.4.2 and three for those in Section 5.4.3.

In terms of the boundary functions in (5.22), because of lack of data, it has been assumed $\sigma_\phi = 0$, $\sigma_\psi = 0$, $\rho_\phi = 0$ and $\rho_\psi = 0$ at distance of 30% of the image height and width, respectively. That is, the image plane has been artificially enlarged by 30% of its width and height, respectively, on all sides, in order to have sufficient space for ϕ and ψ to decay[1]. The image intensity has been set to zero on this artificial frame. Consequently, there, only the regularization term in (5.17) determined values for ϕ and ψ.

As error measures the *average squared L_2 norm error* and the *average angular error* (mean and first standard derivation), see [7], applied to the flow fields resulting from ϕ and/or ψ were chosen. An error measure on ϕ and ψ directly was avoided, thus the potentials are only defined up to a constant.

The intensities of all input images have been normalized to the range $[0, 1]$.

5.4.1 Parameter Studies

In early experimental studies the influence of the parameters γ and λ were investigated on synthetic potentials in order to have a ground truth (cf. Figure 5.1). The associated synthetic flow field was applied to a real image, i.e. the real image was mapped using the velocity field to obtain a second image resulting in the input image pair for the current experiments. Note that the vector field consist of an exact spatial overlap of the true components we wish to determine and distinguish.

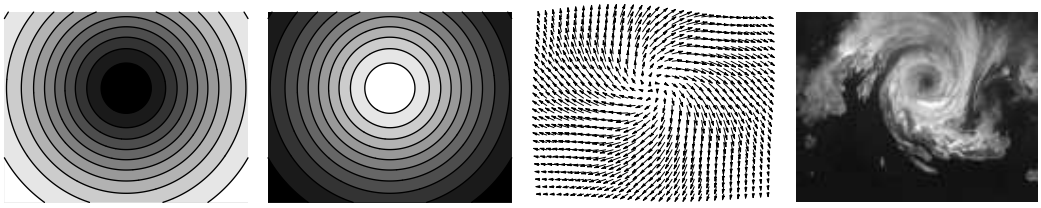
Figure 5.3 shows the results for estimating the potential functions with varying values for $\gamma \in \{0.1, 0.25, 0.5, 1, 2, 3, 4\}$ and a fixed $\lambda = 10^3$. These results clearly illustrate the positive regularizing effect of the high-order smoothness terms in (5.18) and (5.19) which were made computationally tractable by means of auxiliary functions. It's remarkable that both vector field components can be distinguished — despite a complete spatial overlap and only partially given image structures — by subsequent linearizations of a single data term (cf. Section 5.1).

Whereas Figs. 5.4(a,c) and 5.3(d,k) show that the potential fields have been reconstructed well, Figs. 5.4(b,d) reveal a relatively large angular error of $10° - 20°$ of the respective velocity fields. This is plausible since the velocity fields are related to the *derivatives* of the quantities we estimate directly (potential functions). As a consequence, inaccuracies appear amplified. However, it should be noted again that the potential functions are the quantities of primary interest for flow pattern analysis. A comparison with an approach which uses *indirect* computation of the

[1]Note that this is (approximately) consistent with the initial assumption (w.r.t. the Helmholtz Decomposition) that the velocity field vanishes at infinity.

potential functions by integrating velocity fields along stream lines is the object of the next section.

Figure 5.4 shows that for low values of the regularizing parameter γ the corresponding auxiliary function must not be smoothed too much (too large values of λ). This finding reveals a dependency between γ and λ, the closer investigation of which is left for future work.



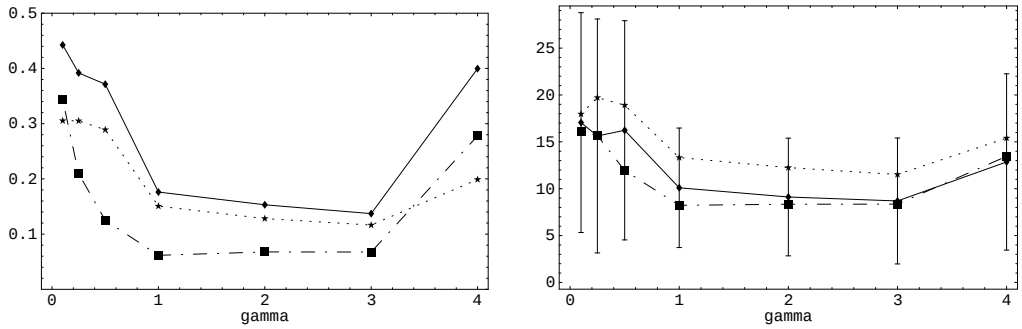
(a) Synthetic velocity potential ϕ (b) Synthetic stream function ψ (c) Corresponding velocity field $\nabla\phi + \nabla\psi^{\perp}$ (d) Image to which (c) was applied to

Figure 5.1: **Input data for the parameter studies.** Based on the synthetic potential functions depicted in (a) and (b), the image in (d) was mapped by the corresponding velocity field (c) to obtain a synthetic image pair with ground truth data. (128×100 pixels)

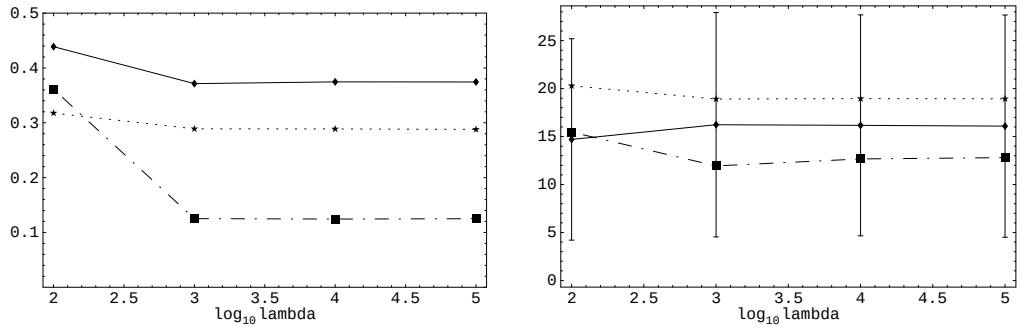
5.4.2 Comparison with Existing Approaches

In [40] an approach was presented in which the potential functions were approximated *indirectly* by first estimating a motion field using a regularization similar to (5.17) and a subsequent integration along the stream lines in order to obtain the velocity potential and the stream function. Both approaches were compared in two experiments here based on given synthetic potential functions (cf. Figure 5.5). In order to have another reference solution, a Horn and Schunck motion estimation [72] has been carried out with both data sets in addition. The parameters of all methods have been optimized manually. Note that the setting of comparison experiment 1 is the same as for the parameters studies.

The results for experiment 1 (Figure 5.6 and Table 5.1) and experiment 2 (Figure 5.7 and Table 5.2) both show that the new approach yield good results similar to those obtained by the the approach of Corpetti et al., despite the higher order of differential equations involved in the minimization. Furthermore, they show that the standard regularization of Horn and Schunck is insufficient to preserve the desired image structures, since this regularization penalizes strong discontinuities like those in the center of the velocity field in experiment 1 resulting from a vortex and a source, which we want to preserve. Even for weak regularizations the results of the Horn and Schunck estimator are less accurate in both experiments.



(a) squared L^2-error for varying γ (b) angular error for varying γ



(c) squared L^2-error for varying λ (d) angular error for varying λ

Figure 5.2: **Quantitative parameter studies.** (a,c) Average squared L_2-error of $\hat{u}_{ir}+\hat{u}_{so}$ (solid), $\hat{u}_{ir}$ (pointed) and $\hat{u}_{so}$ (dashed) depending on (a) γ and (c) λ. (b,d) Average angular error of $\hat{u}_{ir} + \hat{u}_{so}$ (solid), $\hat{u}_{ir}$ (pointed) and $\hat{u}_{so}$ (dashed) depending on (b) γ and (d) λ. Direct estimation of potential functions as the objects of primary interest leads to a small global error of the corresponding gradient velocity fields (a,c) but to local angular errors from $10° - 20°$.

5.4.3 Reconstructing the Vortexes of a Landing Air Plane

Finally, the new approach has been applied on an image pair coming from a real image sequence. The sequence is a recording of the motion of smoke behind a landing passenger air plane. It contains a strong vortex in the center and a weaker but larger one in the other direction with its center laying outside the image plane, approximately 50% from the right border. In addition a weak source is present in the right half, centered vertically (cf. Figure 5.8(a,b)).

In order to eliminate the laminar component of the velocity field, the laminar flow has been approximated roughly by a Horn and Schunck estimator with a strong regularization $\alpha^2 = 10^6$ and used to map the second image of the input image pair back (cf. Section 5.1).

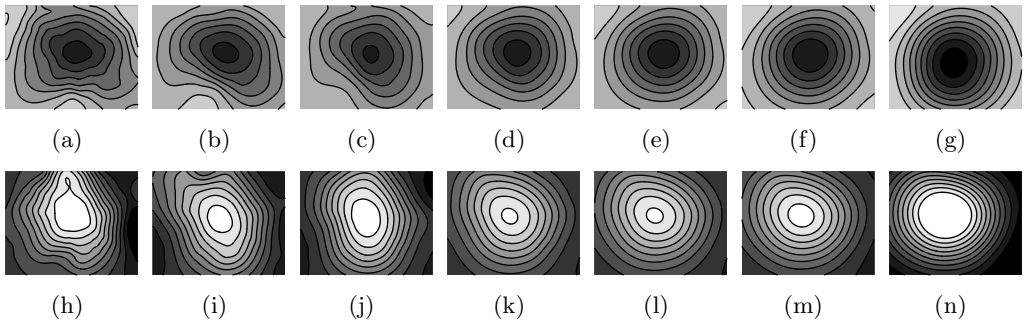


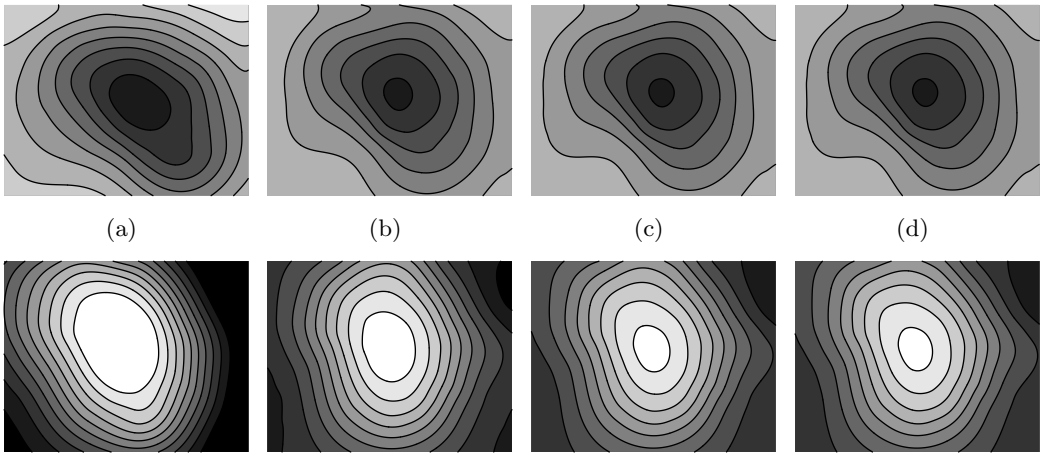
Figure 5.3: **Qualitative parameter study for γ.** Estimated velocity potentials ϕ ((a)-(d)) and stream functions ψ ((e)-(h)) for γ taking values of $\{0.1, 0.25, 0.5, 1, 2, 3, 4\}$ and $\lambda = 10^3$. Both vector field components can be distinguished, despite a complete spatial overlap. The positive effect of the high–order regularization implemented by means of auxiliary functions is clearly visible.

The results in Figure 5.8 show that both the main vortex and the weak source are well reconstructed, despite the lack of image structures in the lower half of the sequence. But the weaker counter-vortex is detected only in outlines. This is plausible since the latter one has its maximum outside the image plane and the stream function ψ is set to be zero on the large border (see beginning of this section).

5.5 Conclusion

The problem of computing highly non-rigid fluid flow from image sequences involves important application issues. In contrast to traditional variational approaches for optical flow computation which have disadvantages in this context, we dealt with this problem by using higher-order regularization terms which merely penalize changes of the principal flow constituents. The approach was made computationally tractable by the use of auxiliary functions. A significant feature of the approach is that the associated potential functions are directly computed. This is a favorable property regarding the recognition and analysis of flow patterns. Numerical experiments confirmed that both components can be estimated separately by subsequent linearizations of a single data term. A comparison with an indirect approach revealed no loss in performance, despite the higher order of differential equations to be solved.

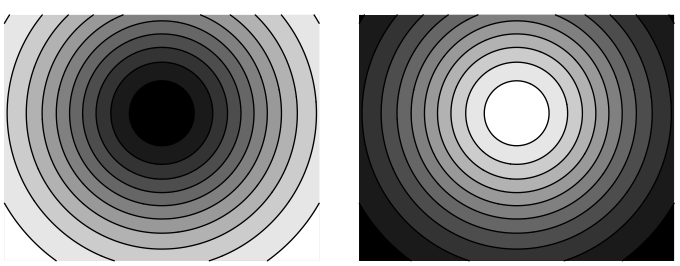
The research area of variational fluid flow estimation from image sequences is rapidly evolving. We refer in Section 9.2 to the most recent developments since the completion of this thesis.



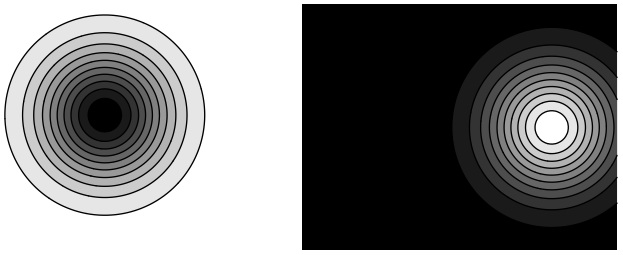
(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 5.4: **Qualitative parameters study for λ.** Estimated velocity potentials ϕ ((a)-(d)) and stream functions ψ ((e)-(h)) for λ taking values of $\{10^2, 10^3, 10^4, 10^5\}$ and $\gamma = 0.5$. Parameters for enforcing regularization γ and smoothing the auxiliary functions λ are not independent. For low values of γ, the auxiliary functions must not be smoothed too much.



(a) Synth. velocity potential ϕ (b) Synth. stream potential ψ (c) Corresponding velocity field (max. 1.86 pixels/frame)



(d) Synth. velocity potential ϕ (e) Synth. velocity potential ψ (f) Corresponding velocity field (max. 1.89 pixels/frame)

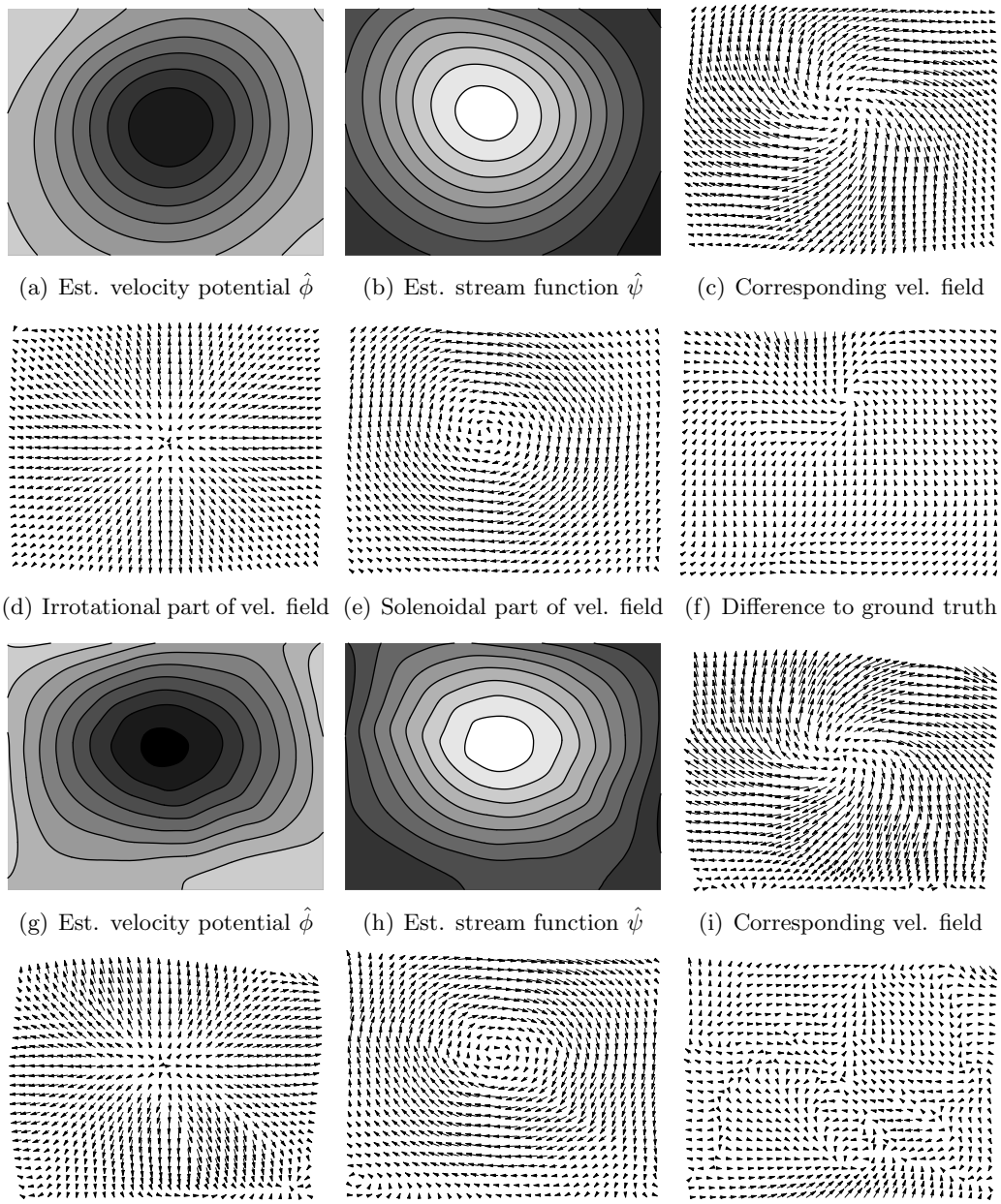
Figure 5.5: **Input data for comparison experiment 1 and 2**. Either synthetic velocity field has been used to map the image in Fig. 5.1(d) in order to generate an image pair.

Approach/ Error measure		**Direct approach** $\gamma = 3.0, \lambda = 1000$	**Corpetti et al.** $\alpha = 300, \lambda = 250$	**Horn&Schunck** $\alpha^2 = 0.17$
Av. squared L^2-error	$u_{ir} + u_{so}$	137.11	175.81	208.979
$\times 10^3$	u_{ir}	116.38	152.89	
	u_{so}	67.51	102.24	
Av. angular error	$u_{ir} + u_{so}$	$8.69° \pm 6.72°$	$10.12° \pm 7.95°$	$8.93° \pm 6.39°$
(mean/1. std. dev.)	u_{ir}	$11.52° \pm 5.58°$	$12.89° \pm 7.71°$	
	u_{so}	$8.35° \pm 5.39°$	$10.36° \pm 6.12°$	

Table 5.1: **Measured errors of comparison experiment 1.** Both the *average squared L^2-error* and the *average angular error* of the resulting velocity field ($u_{ir} + u_{so}$) and separately for the irrotational and solenoidal components (u_{ir}, u_{so}) are given. Approach-dependent parameters have been optimized manually. The errors of the Horn and Schunck approach show clearly that, despite the flow directions are estimated with similar accuracy, there is a significant higher error in the magnitudes which results from the penalization of the discontinuity in the velocity field (cf. Fig. 5.5(c)).

Approach/ Error measure		**Direct approach** $\gamma = 0.5, \lambda = 100$	**Corpetti et al.** $\alpha = 300, \lambda = 250$	**Horn&Schunck** $\alpha^2 = 0.07$
Av. squared L^2-error	$u_{ir} + u_{so}$	162.18	168.7	170.514
$\times 10^3$	u_{ir}	60.69	70.59	
	u_{so}	129.56	65.76	
Av. angular error	$u_{ir} + u_{so}$	$14.68° \pm 9.59°$	$14.65° \pm 11.51°$	$16.19° \pm 10.26°$
(mean/1. std. dev.)	u_{ir}	$10.56° \pm 5.93°$	$11.15° \pm 7.52°$	
	u_{so}	$14.11° \pm 9.72°$	$10.29° \pm 7.36°$	

Table 5.2: **Measured errors of comparison experiment 2.** While the approximation quality compared to the indirect approach of Corpetti et al. is more equal here, the difference in approximation quality to the approach of Horn and Schunck is not as distinct as in experiment 1 since the discontinuities of the velocity field are smaller in this case see Fig. 5.5(f).



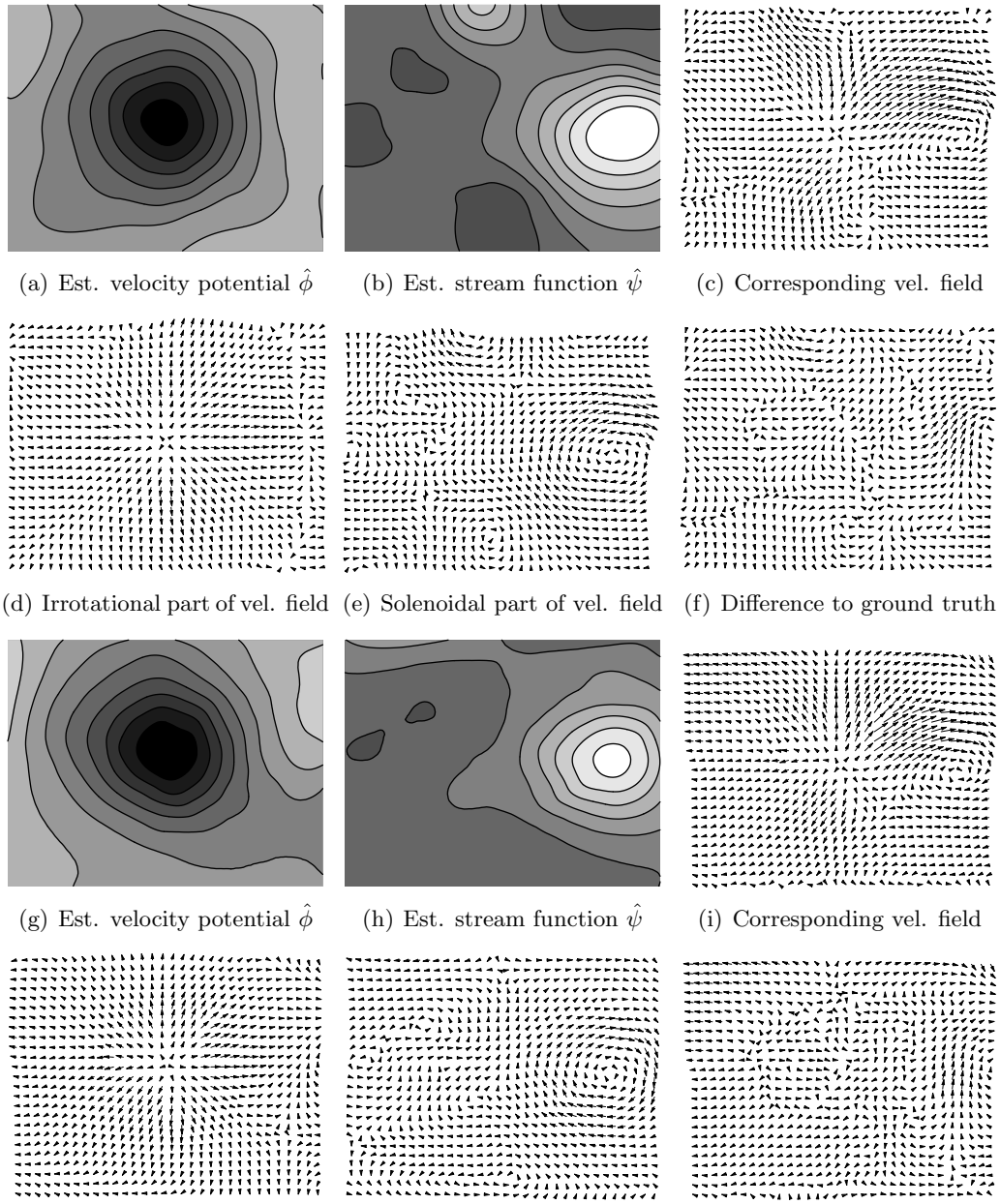
(a) Est. velocity potential $\hat{\phi}$ (b) Est. stream function $\hat{\psi}$ (c) Corresponding vel. field

(d) Irrotational part of vel. field (e) Solenoidal part of vel. field (f) Difference to ground truth

(g) Est. velocity potential $\hat{\phi}$ (h) Est. stream function $\hat{\psi}$ (i) Corresponding vel. field

(j) Irrotational part of vel. field (k) Solenoidal part of vel. field (l) Difference to ground truth

Figure 5.6: **Direct versus indirect approach on data set 1.** (a)–(f) Results of the proposed direct approach ($\gamma = 3.0, \lambda = 1000$). (g)–(l) Results of the indirect approach by Corpetti et al. ($\alpha = 300, \lambda = 250$). Both methods lead to very similar results in this case, despite significant differences in the approach.



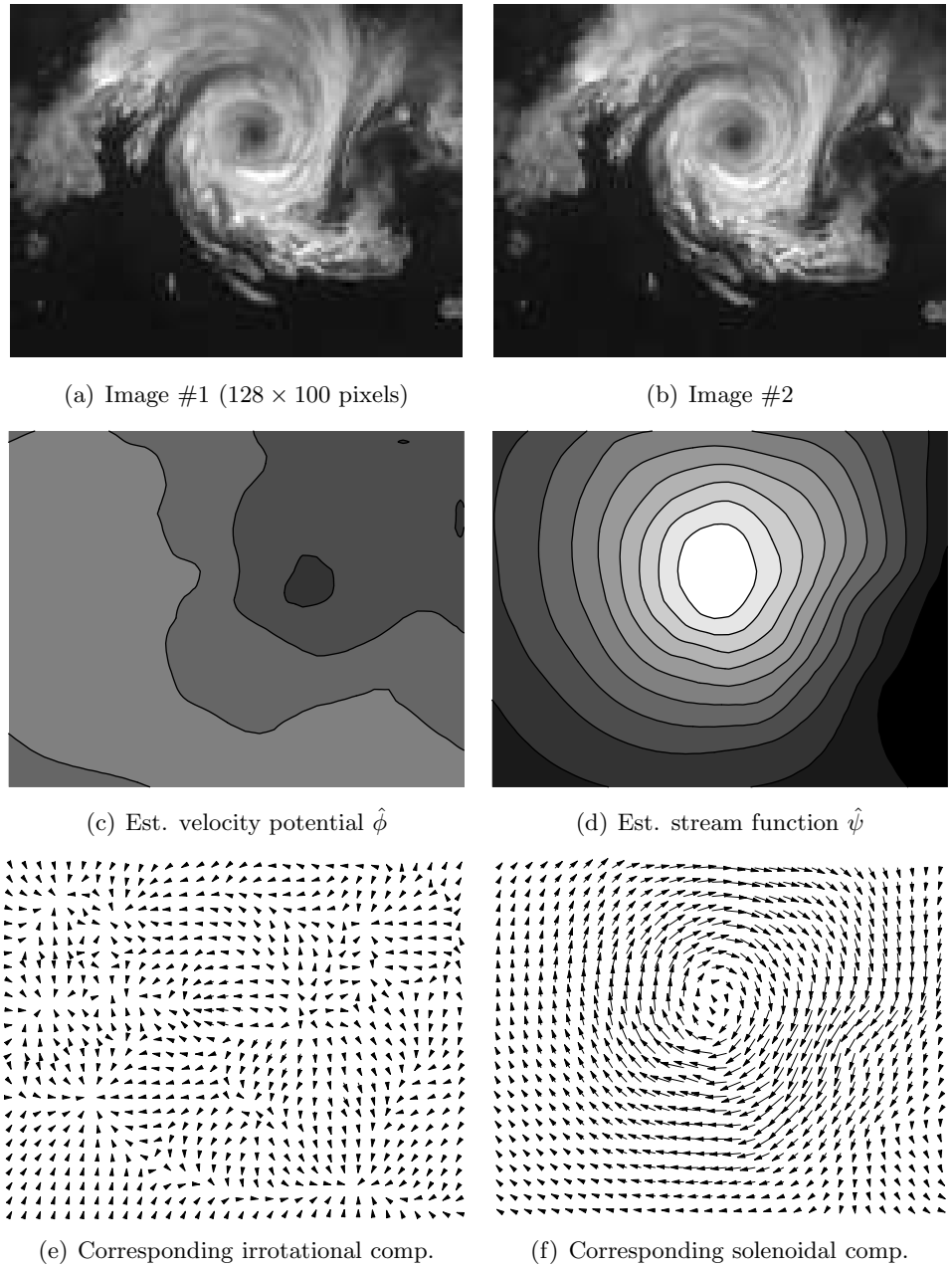
(a) Est. velocity potential $\hat{\phi}$ (b) Est. stream function $\hat{\psi}$ (c) Corresponding vel. field

(d) Irrotational part of vel. field (e) Solenoidal part of vel. field (f) Difference to ground truth

(g) Est. velocity potential $\hat{\phi}$ (h) Est. stream function $\hat{\psi}$ (i) Corresponding vel. field

(j) Irrotational part of vel. field (k) Solenoidal part of vel. field (l) Difference to ground truth

Figure 5.7: **Direct versus indirect approach on data set 2.** (a)–(f) Results of the proposed direct approach ($\gamma = 0.5, \lambda = 100$). (g)–(l) Results of the indirect approach by Corpetti et al. ($\alpha = 300, \lambda = 250$). Also here the results are quite similar, although the false second maximum of the stream function is more distinct with the direct approach.



(a) Image #1 (128 × 100 pixels)

(b) Image #2

(c) Est. velocity potential $\hat{\phi}$

(d) Est. stream function $\hat{\psi}$

(e) Corresponding irrotational comp.

(f) Corresponding solenoidal comp.

Figure 5.8: **Results of the real-world experiment.** The images in (a) and (b) are part of sequence showing a wake vortex of a landing air plane. Both the main vortex and the weak source are well reconstructed by the new approach, despite the lack of image structures in the lower half of the sequence. However, the weaker counter-vortex is reconstructed only in outlines. This is plausible since the latter one has its maximum outside the image plane and the velocity potential ψ is set to be zero on the enlarged border. ($\gamma = 0.5$, $\lambda = 10^3$, maximum velocity: 2.75 pixels/frame)

Chapter 6

TV-based Variational Image Restoration

Variational image restoration employing the Total Variation (TV) norm regularization has been recognized for giving very good results in comparison to those based on L^2-norm regularization, because of its edge-preserving property. On the other hand, the underlying partial differential equations are nonlinear and demand a significantly higher computational effort for solution by dedicated methods. Furthermore, because of the variational nature of the approach, the solution methods do not provide direct clues for coarse-grained parallelism, neither with respect to a spatial nor to a functional decomposition. Due to the nonlinearity, standard non-overlapping domain decomposition methods, e.g. substructuring methods as proposed in Chapter 3, cannot be applied directly, but only within iterations based on local linearizations. In contrast, in the succeeding two chapters different approaches for the direct domain decomposition of nonlinear problems will be discussed. Since TV-based image restoration will serve as the model problem there, details on the approach and common solving methods will be discussed in the following chapter first.

The plan of this chapter is as follows. First the underlying optimization problem in comparison to the L^2-based restoration problem is explained, followed by addressing the difficulties in solving the corresponding nonlinear equations systems. Second, an outline of the four most known dedicated solving techniques is given, followed by presentation and discussion of experimental results for two meaningful examples.

6.1 Regularization Based on the TV-norm

6.1.1 Problem Statement

TV-based image restoration is founded on the following minimization problem [104, 103, 27]:

$$\min_u \int_\Omega \frac{1}{2}(Ku - f)^2 \, d\mathbf{x} + \alpha \, J_{TV}(u) =: J(u) \, , \tag{6.1}$$

where $f \in H^1(\Omega) = V(\Omega)$ denotes the degenerated input image, u the reconstructed image being sought for, $K : V(\Omega) \to V(\Omega)$ a linear compact integral operator describing the degradation process. J_{TV} refers to the so-called *Total Variation norm*, given in weak formulation by

$$J_{TV}(\xi) := \sup_{w \in (C_0^\infty(\Omega))^2} \left\{ \int_\Omega \xi \, \mathrm{div} \, w \, d\mathbf{x} \; : \; \|w\|_\infty \leq 1 \right\}, \tag{6.2}$$

with test functions w and $\xi \in L^1(\Omega)$, see, e.g., [121]. Furthermore, by J_{TV} one can construct the *space of Bounded Variations*, denoted by $BV(\Omega)$, which consists of all functions $\xi \in L^1(\Omega)$ for which

$$\|\xi\|_{BV} := \|\xi\|_{L^1(\Omega)} + J_{TV}(\xi) \leq \infty \, , \tag{6.3}$$

and it is $u \in BV(\Omega)$ in (6.1).

Note that for all $\xi \in W^{1,1}(\Omega)$ the explicit formulation

$$J_{TV}(\xi) := \int_\Omega |\nabla \xi| \, d\mathbf{x} \tag{6.4}$$

can be given[1]. However, for $\xi \in H^1(\Omega) = W^{1,2}(\Omega) = V(\Omega)$, as we assume for u in the following, (6.4) is commonly approximated by

$$J_{TV,\beta}(\xi) := \int_\Omega \sqrt{(\nabla \xi)^2 + \beta} \, d\mathbf{x}, \tag{6.5}$$

with β being small scalar constant in $\mathbb{R}$. The reason for using this approximation is to obtain a differentiable functional that allows for using standard tools in a common Hilbert-space setting. In the following we will always approximate[2] J_{TV} by $J_{TV,\beta}$.

[1] $W^{1,1}(\Omega) := \{v \in L^1(\Omega) \; : \; \partial^\alpha v \in L^1(\Omega) \, , \; 0 \leq |\alpha| \leq 1\}$

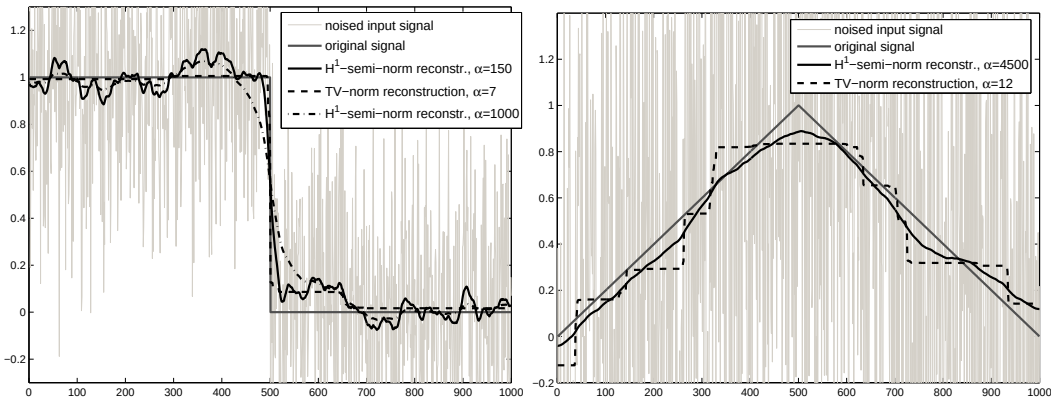
[2] However, we point out that discrete implementations of (6.1) exist, that do not rely on the approximation (6.5), but use the exact definition (6.2) of the TV-measure [26]. Convergence of the corresponding iteration is slow, however, and any acceleration will result in a primal-dual iteration similar to that investigated in Section 6.2.4.

Furthermore, note that problem (6.1) can be regarded as a nonlinear instance of the class of *Tikhonov regularizations* [118], which is applied to the inverse problem

$$f = K\,u + \eta\,, \tag{6.6}$$

with η being a Gaussian-distributed random function modeling the measurements noise. Due the compactness of K, problem (6.6) is ill-posed, which gives rise to the regularization in connection with the least-squares minimization as stated in (6.1), see [43] and the references therein. Note that (6.6) is also denoted as *image degradation model* in the restoration context.

Additionally, in comparison to approaches applying the squared L^2-norm, i.e. $\int_\Omega (\nabla u)^2\, d\mathbf{x}$, as regularizer, TV regularization has the superior property of not penalizing strong magnitudes of ∇u, and thus strong discontinuities of u much less, leading to considerably better results, which is illustrated for a 1D-example in Figure 6.1(a). However, at the presence of weak gray value gradients, staircase-like artifacts are known to appear, emerging from the preference of piecewise constant gray levels, see Figure 6.1(b).



(a) Reconstructions of a ramp function (b) Reconstruction of a triangular function

Figure 6.1: **L^2-norm versus TV-norm regularization at the example of 1D denoising.** Depicted are reconstructions of (a) a step function signal perturbed and (b) a triangle function signal both having being perturbed by additive Gaussian noise (signal-to-noise ratios: 1.7 dB and -11 dB, respectively, $K \equiv I$, $\beta = 10^{-14}$). In case of the step function, TV-regularization leads to a much better reconstruction, since it allows for strong discontinuities. On the other hand, staircase-like artifacts do occur with smooth gradients.

6.1.2 Euler-Lagrange Equation

Computing formally the first variation of the functional of (6.1), while using the TV-norm approximation (6.5), we obtain

$$\int_{\Omega} K^*(Ku-f)\, v \; + \alpha \frac{\nabla u^\top \nabla v}{\sqrt{(\nabla u)^2+\beta}}\, d\mathbf{x} \;=\; 0, \qquad \forall v \in V(\Omega), \tag{6.7}$$

which is the first variation of J being set equal to zero, and with K^* denoting the adjoint operator to K. Here we see that $\beta > 0$ also has the function to prevent (6.7) from becoming singular at locations where $|\nabla u| = 0$. It is known, [1], that a solution to the approximation converges to that of the original problem for β approaching 0. Equation (6.7) will serve as our model problem in the sequel.

The Euler-Lagrange equations being derivable from (6.7) by partial integration read:

$$\big(Ku-f\big)\, K^* - \alpha\, \nabla \cdot \frac{\nabla u}{|\nabla u|_\beta} \;= 0\,, \tag{6.8}$$

where $|\cdot|_\beta := \sqrt{|\cdot|^2+\beta}$, and with the Neumann boundary condition

$$\frac{\partial u}{\partial n} = 0 \quad \text{on} \quad \partial\Omega\,, \tag{6.9}$$

where n denotes the outer unit normal on $\partial\Omega$.

We rewrite (6.8) to

$$\Leftrightarrow K\; K^*u - \alpha\, \nabla \cdot \frac{\nabla u}{|\nabla u|_\beta} \;= K^*\, f\,, \tag{6.10}$$

and define $\phi(u) := K\; K^*u - \alpha\, \nabla \cdot \frac{\nabla u}{|\nabla u|_\beta}$, of which we will make use of later on.

6.2 Solving Methods

Four common iterative solving approaches will be presented in the following, which also reflects chronological order of their appearance.

Concerning the perturbation parameter β, note in advance that it is either set to its final value, e.g. 10^{-8} for all iterations, or initialized by some large value and then gradually diminished within while solving, which is referred to as *'continuation'* in TV restoration literature. However, the denotation 'continuation' in optimization theory is commonly utilized for non-convex objective functionals whose number of local minima change with the continuation parameter. This is not the case here, since $J(u)$ is convex even for $\beta \to 0$. Therefore, we will use the denotation *β-decrement procedure* in the following.

6.2.1 Steepest Descent

In their first publication on TV-based regularization [104], Rudin, Osher and Fatemi also proposed the following *time-marching scheme*:

$$\frac{\partial u}{\partial t} = -\phi(u) = \alpha\nabla\cdot\left(\frac{\nabla u}{|\nabla u|_\beta}\right) - K^*(Ku - f), \quad t > 0, \tag{6.11}$$

with initial condition $u = z$ at $t = 0$. Hereby, one obtains as the steady state a solution to (6.10). After time-discretization, an explicit (forward Euler) marching scheme leads to the iteration rule

$$u^{k+1} = u^k - \tau\ \phi(u^k) \tag{6.12}$$

for some fixed time step size τ, $0 < \tau \leq 1$, and the iteration count $k \in \mathbb{N}$; which is equivalent to the steepest descent method. Unfortunately, this method has poor asymptotic convergent rates. Furthermore, it almost always leads to oscillation at the minimum, and has little robustness for small values of β, such as $\beta = 10^{-8}$.

6.2.2 Fixed Point Iteration

In order to overcome the disadvantages of the steepest descent method, Vogel and Oman [122, 123, 43], proposed the fixed point iteration

$$\alpha\nabla\cdot\left(\frac{\nabla u^{k+1}}{|\nabla u^k|_\beta}\right) - K^*(Ku^{k+1} - f) = 0, \tag{6.13}$$

with initial value $u^0 = z$, leading to solve the linear equation system

$$\left(K^*K - \alpha\nabla\cdot\left(\frac{\nabla}{|\nabla u^k|_\beta}\right)\right)u^{k+1} = K^*f\,, \tag{6.14}$$

for each iteration k. The weak formulation reads:

$$\int_\Omega K^*(Ku^{k+1} - f)v + \alpha\left(\frac{\nabla u^{k+1}}{|\nabla u^k|_\beta}\right)^\top \nabla v\ d\mathbf{x} = \int_\Omega K^*f\ v\ d\mathbf{x}, \quad \forall v \in V(\Omega). \tag{6.15}$$

The main idea of this approach is to fix the the nonlinear term $L(u^k) := \nabla\cdot\left(\nabla/|\nabla u^k|_\beta\right)$ in (6.14) and to consider it as a linear equation

$$\left(K^*K - \alpha L(u^k)\right)u^{k+1} = K^*f \tag{6.16}$$

instead. In doing so, $L(u^k)$ can be interpreted as a diffusion operator being applied to u^{k+1}, cf. [122]. Since L depends on the solution u^k of the previous iteration, this method is denoted as *lagged diffusivity fixed point iteration* [122]. However, later Heers et al. [70] found, that this procedure had been already introduced as the *Kačanov method* over 35 years ago [73, 59]. After discretization, standard methods from numerical linear algebra, e.g., conjugate gradient iteration in connection with multigrid or FFT-based preconditioning can be applied.

Although the designated method turned out to be robust and convergence is ensured, [43], its convergence rate is only linear [35].

6.2.3 Newton's Method

In aiming to improve the convergence rate, Vogel and Oman [122], as well as Chan, Chan and Zhou [36] suggested to apply Newton's method, leading to the iteration rule

$$u^{k+1} = u^k - H_\phi^{-1}(u^k)\,\phi(u^k)\,, \tag{6.17}$$

with $\phi(u^k)$ denoting the gradient of J at u^k, as defined in (6.10), as well as its Hessian $H_\phi(u^k)$, given by

$$H_\phi(u) = K^*K - \alpha\nabla\cdot\left(\frac{1}{|\nabla u|_\beta}\left(I - \frac{\nabla u \nabla u^\top}{|\nabla u|_\beta^2}\right)\nabla\right). \tag{6.18}$$

That is, at each iteration one has to solve the equation

$$\left(K^*K - \alpha\nabla\cdot\left(\frac{1}{|\nabla u|_\beta}\left(I - \frac{\nabla u \nabla u^\top}{|\nabla u|_\beta^2}\right)\nabla\right)\right)\delta u = -\phi(u^{(k)})\,, \tag{6.19}$$

with respect to an update δu, followed by setting $u^{k+1} \leftarrow u^k + \delta u$. The variational formulation of 6.19 reads:

$$\int_\Omega K^*K\ \delta u\ v + \alpha\ \frac{1}{|\nabla u|_\beta}\nabla v\left(I - \frac{\nabla u \nabla u^\top}{|\nabla u|_\beta^2}\right)\nabla\delta u\ d\mathbf{x} = -\left\langle\frac{\partial J}{\partial u}, \delta u\right\rangle,\ \forall v \in V(\Omega). \tag{6.20}$$

Although the convergence of this procedure is quadratical, its domain of convergence has turned out to be very small, which is especially the case for small values of β since the nonlinear characteristics becomes most apparent then. I.e. the Newton Method practically always diverges for reasonable small values of β, such as 10^{-6}, even though u is initialized with values close to the solution, e.g. by the non-degradated ground truth image.

As a remedy, Chan, Chan, and Zhou [36, 29] proposed a decrement procedure, where Newton's Method is carried out several times for different, decreasing values of β, i.e. β is fixed within the Newton iteration. In detail, β is initially set to a relatively high value of e.g. 1 and the decreased by a heuristic procedure — e.g. multiplication by a fixed factor in the simplest variant — until the the final value is attained. However, our experimental studies have revealed that the decrement step and the Newton iteration can be interleaved, i.e. β can be decreased *within* the Newton procedure without loss of convergence. In doing so, a significantly faster convergence could be observed, see Figure 6.4(b)[3] (see Sec. 6.2.5 for details on input data and parameters values).

6.2.4 Primal-dual Newton's Method

6.2.4.1 Mitigating the Nonlinearity

Although the Newton method, in connection with an appropriate decrement procedure on β, is appropriate for solving the TV-based reconstruction problem; much heuristics and parameter tuning is involved there. In order to overcome the main cause for the numerical instability, which is the presence of the term $\nabla u/|\nabla u|_\beta$ in (6.7), Chan et al. [29, 35, 13, 30] suggested to substitute this term by the auxiliary function

$$w := \frac{\nabla u}{|\nabla u|_\beta}, \qquad w \in (V(\Omega))^2, \tag{6.21}$$

which can also be interpreted as the unit normal vector of the level sets of u, cf. [29]. Applying this substitution to the Euler-Lagrange equations (6.10) then yields the coupled system

$$\begin{cases} K^*(Ku - f) - \alpha \nabla \cdot w := \varphi(u, w) = 0 \\ w|\nabla u|_\beta - \nabla u := \psi(u, w) = 0. \end{cases} \tag{6.22}$$

The advantage of this is substitution is the first equation now to be linear in u, in contrast to the original E.-L. equation.

In order to solve the coupled problems (6.22), Newton's method is applied, leading to the equation system

$$\begin{pmatrix} K^*K & -\alpha\nabla\cdot \\ -\left(I - \frac{w\nabla u^\top}{|\nabla u|_\beta}\right)\nabla & |\nabla u|_\beta \end{pmatrix} \begin{pmatrix} \delta u \\ \delta w \end{pmatrix} = -\begin{pmatrix} \varphi(u, w) \\ \psi(u, w) \end{pmatrix} \tag{6.23}$$

[3] Also here, the reduction factor was chosen manually to be the smallest one from a discrete set of values leading to convergence

for the updates $\delta u,\ \delta w$, which is obtained by linearizing $\big(\phi(u,w),\psi(u,w)\big)$ around $(u,w)^\top$ (see [29] for further details). In a next step, the second equation in (6.23) is algebraically solved for δw and $\psi(u,w)$ is replaced by its definition, giving

$$\delta w = \frac{1}{|\nabla u|_\beta}\left(I - \frac{w\nabla u^\top}{|\nabla u|_\beta}\right)\nabla\delta u - w + \frac{\nabla u}{|\nabla u|_\beta}\,. \tag{6.24}$$

Subsequently, replacing by this δw in the first equation of (6.23) results in

$$\left(K^*K - \alpha\nabla\cdot\left(\frac{1}{|\nabla u|_\beta}\left(I - \frac{w\nabla u^\top}{|\nabla u|_\beta}\right)\nabla\right)\right)\delta u = -\phi(u). \tag{6.25}$$

By this, one solves (6.25) for the update δu only, followed by evaluating (6.24) in order to obtain the corresponding update for w.

Weak formulations of (6.24) and (6.25) are given by

$$\int_\Omega \delta w\ \tilde{w}\ d\mathbf{x} = \int_\Omega \frac{1}{|\nabla u|_\beta}\ \tilde{w}\left(I - \frac{w\nabla u^\top}{|\nabla u|_\beta}\right)\nabla\delta u - w^\top\tilde{w} + \frac{\nabla u^\top\tilde{w}}{|\nabla u|_\beta}\ d\mathbf{x},\ \forall\tilde{w}\in(V(\Omega))^2, \tag{6.26}$$

and

$$\int_\Omega K^*K\ \delta u\ v + \alpha\ \frac{1}{|\nabla u|_\beta}\nabla v\left(I - \frac{w\nabla u^\top}{|\nabla u|_\beta}\right)\nabla\delta u\ d\mathbf{x} = - \int_\Omega K^*(Ku-f)\ v + \alpha\left(\frac{\nabla u}{|\nabla u|_\beta}\right)^\top\nabla v\ d\mathbf{x}, \qquad \forall v\in V(\Omega), \tag{6.27}$$

respectively.

6.2.4.2 The Algorithm

After applying an appropriate discretization, the inner loop of the iteration consists of the following steps:

(i) solve (6.25) for the update δu^k

(ii) evaluate (6.24) for the update δw^k

(iii) determine the step size τ_u for δu^k

(iv) determine the step size τ_w for δw^k such that $\|w^k + \delta w^k\|_\infty \leq 1$

(v) set $u^{k+1} \leftarrow u^k + \tau_u \delta_u$

(vi) set $w^{k+1} \leftarrow w^k + \tau_w \delta_w$

(vii) set $k \leftarrow k + 1$

where u^k and w^k are initialized by arbitrary feasible values. These steps are iterated until the L^2 norm of the gradient, $\|\frac{dJ}{du}\|_{L^2}$, has fallen below a given threshold.

Let us take a closer look at the first step. Discretization of (6.25) (or (6.27) when using finite elements, respectively), yields a linear equation system of the form $A\delta u = b$. A typically is a sparse and banded matrix (assuming a regular discretization grid), whose inverse A^{-1} is dense, which suggests the employment of an iterative solving technique not requiring A^{-1} to be calculated explicitly. Furthermore, it is known [29], A to be regular and positive definite, if $\alpha > 0$, K is invertible and $|w_i| < 1$, $\forall i$. Though, A is also known to be non-symmetric, which prevents from applying appropriate methods like conjugated gradient iteration. As a remedy Chan et al. propose, [29], to instead solve the symmetrized equation $\frac{1}{2}(A + A^\top)\,\delta u = b$, which corresponds to modifying the left-hand sides of (6.25) to

$$\left(K^*K - \alpha\nabla \cdot \left(\frac{1}{|\nabla u|_\beta} \left(I - \frac{1}{2} \frac{w\nabla u^\top + \nabla u w^\top}{|\nabla u|_\beta} \right) \nabla \right) \right) \delta u = \dots \,, \tag{6.28}$$

or in the weak formulation (6.27) to

$$\int_\Omega K^*K\ \delta u\ v + \alpha \frac{1}{|\nabla u|_\beta} \nabla v \left(I - \frac{1}{2} \frac{w\ \nabla u^\top + \nabla u^\top w}{|\nabla u|_\beta} \right) \nabla \delta u\ d\mathbf{x} = \dots \,. \tag{6.29}$$

Empirically, it has turned out, [29], that the symmetrized operator $\frac{1}{2}(A + A^\top)$ converges to the original operator A while applying Newton's method, since w converges to $\frac{\nabla u}{|\nabla u|_\beta}$, without changing the convergence rate significantly. Hence, conjugated gradient iteration can now be applied to the symmetrized problem, which was also done with our experiments.

Whereas the implementation of step (ii) is obvious, different line search methods for step (iii) can be employed in order to guarantee global convergence with respect to u. Chan et al. suggested the commonly utilized *Armijo's rule*, cf., e.g., [8], which amounts to determining the first m in ascending order, starting with $m = 0$, for which the following inequality holds true:

$$J(u) - J(u + \tau_0 \nu^m \delta u) \geq -\kappa \tau_0 \nu^m \left\langle \frac{dJ(u)}{du}, \delta u \right\rangle \,, \tag{6.30}$$

where ν denotes a given reduction factor, τ_0 an initial step size guess, κ a scalar parameter, and with J and $\langle \frac{dJ}{du}, \delta u \rangle$ as defined in (6.1) and (6.7), respectively. Once

such an m has been found, the final step size is given by $\tau_0 \nu^m$. Our experiments have shown, that setting $\tau_0 = 1$, $\nu = \frac{1}{2}$, and κ to a value within $[10^{-6}, 10^{-4}]$ lead to satisfactory step size selections. However, in most experiments, using a constant step size of $\tau_u = 1$ led to convergence also, while the convergence rate of the primal-dual Newton Method could not be improved in utilizing Armijo's rule. Since J has to be evaluated several times for each step size selection, we skipped choosing a fixed step size in almost all experiments.

Moreover, the determination of proper step sizes for w in order to fulfill $|w_i| < 1$, $\forall i$, see step (iv), also by own experiments has shown to be essential with respect to convergence. Therefore, at each iteration k, it is necessary to solve a problem of the form

$$\tau_w = \rho \sup \{ \tau : | w_i^k + \tau \delta w_i^k | < 1, \ \forall i \} , \tag{6.31}$$

where ρ is a given scalar parameter being in the range $[0, 1]$. A simple but robust procedure to determine τ_w is to test step sizes $\tau = (\frac{1}{2})^k$, for k being iteratively incremented by 1, beginning with zero, and to then set $\tau_w = \tau$ for the first $\hat{k}$, for which the inequality (6.31) holds. Furthermore, manually choosing $\rho = 0.9$ has been found most appropriate in terms of convergence.

Besides the step size selections, the decrement procedure for β, as suggested with Newton's method in Section 6.2.3, can be applied here also, which has led to small convergence speed improvements with our experimental studies. On the other hand, convergence was reached in all cases also without this. The latter is a distinct property in comparison to the (solely primal) Newton method, where the decrement w.r.t. β is essential for convergence.

6.2.5 Experimental Studies

The main goals of the experimental studies have been first to approve the results presented in the designated publications and second to compare the four presented methods with respect to their convergence characteristics as well as computational effort. The studies were restricted to the denoising problem only, i.e. K was the identity in all experiments, since only general convergence characteristics were to be studied.

6.2.5.1 Input Data, Parameter Values and Error Measures

Standard conforming piecewise linear finite elements were used for discretization. As ground truth images, an artificial, Fig. 6.2(a), as well as an (artificially generated) real world example[4], Fig. 6.3(a), were utilized. Both data sets did contain intensity

[4]being provided to us by Prof. R. Plemmons of Wake Forest University, Winston-Salem, U.S.A.

values of the range $[0, 1]$ and were perturbed by adding (artificially generated) white, mean-free Gaussian noise of variance $\sigma^2 = \frac{1}{4}$, in order to obtain the input images, depicted in Fig. 6.2(b) and Fig. 6.3(b).

With respect to choosing a value for the perturbation parameter β, a common value is 10^{-2} in literature, given a range of $[0, 255]$ for the intensity values. This corresponds to choosing 10^{-8} for intensities being normalized to $[0, 1]$, as it was the case also here. Using smaller values than 10^{-8}, which decreases the convergence rate further, has shown to change the result on 2D-data only slightly; e.g. it leads to an improvement of only 0.2% in the rel. L^2 error for the example depicted in Fig. 6.2.

As the solving method for the linear equation system occurring in the presented methods, preconditioned conjugate gradient iteration while using the inverse of the diagonal of the operator matrix as preconditioner was used. As error threshold, the final relative residual L^2 error must have been dropped below 10^{-4}.

Finally, as error measures, both the final (nonlinear) relative residual error

$$\frac{\|u - \alpha\nabla \cdot \left(\frac{\nabla u}{|\nabla u|_\beta}\right) - f\|_{L^2}}{\|f\|_{L^2}} \tag{6.32}$$

for $\beta = 0$, as well as the relative L^2 error to a reference solution $\hat{u}$ were utilized. The reference solution $\hat{u}$ was calculated in applying the primal Newton method with a final L^2 norm of the gradient of 10^{-12}.

6.2.5.2 Results

First the qualitative results for a synthetic example, Figs. 6.2(g)–(j), and a real world example, Figs. 6.3(a)–(e), in utilizing the Newton method were studied for different regularization strengths α, see Figs. 6.2(g)–(j) and Figs. 6.2(g)–(j), respectively. The outcomes show the superior properties of TV-based denoising especially at the presence of heavy noise.

Second, the convergence behavior for each of the four described solving methods was observed in terms of the relative residual L^2 error and the relative L^2 error to the corresponding reference solution (see Figs. 6.4(b), (d), (e) and (f)). In comparing the residual error of the four methods as in Fig. 6.4(f) the superior quadratic convergence of the Newton methods becomes clear.

In addition, differing β-decrement procedures were studied with the two Newton methods, Figs. 6.4(a)–(d). With the primal Newton method, cf. Section 6.2.3, we compared the decrement method proposed by Chan et al., which diminishes β outside the Newton procedure, with our new approach of integrating the decrement into the Newton procedure, see Figs. 6.4(a)[5] and (b), respectively. By this we observed,

[5]The error threshold for each nested application of Newton's method was 10^{-5} w.r.t. to the rel.

that our decrement methods leads to significant better convergence rates while convergence was always reached, if only the reduction rate with respect to β was high enough.

In addition, the convergence behavior of the primal-dual Newton method with and without β-decrement was explored (see Figs. 6.4(c) and (d), respectively) showing that the decrement improves the convergence speed only slightly. This was also observed for $\alpha = 0.5$ and the same input data, as well as for the real world example depicted in Fig. 6.3. Besides the step size for w, depicted in both diagrams also, the two components of the dual variable w are shown in Fig. 6.5. In order to obtain a meaningful depiction, the latter were scaled by the Euclidean length of the gradient of the corresponding solution to u. The density plots clearly show, that the components of $\nabla u/|\nabla u|_\beta$ are correctly approximated by w.

Moreover, comparing the best results of the (solely) primal Newton method and the primal-dual Newton method, Figs. 6.4(b) and (d), shows that the latter only takes approx. 25% more iterations than the former. In consideration of the fact that one iteration of the non-dual method comprises fewer instructions than for the primal-dual method, employing the former method can be even faster, depending on the implementation. E.g. for a Matlab v7.0 implementation on a 3 GHz Intel Pentium machine, the non-dual method took 461 s in comparison to 592 s for the primal-dual method to converge. This proportion was nearly the same for a choosing weaker regularization strength $\alpha = 0.5$, as well as with for different input data, e.e. the real world example in Fig. 6.3(a) for $\alpha = 1$.

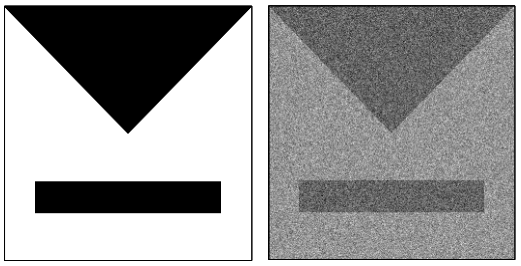
Finally, with respect to the additional step size selection algorithms, the convergence speeds of both Newton-based approaches could not be improved by using Armijo's rule w.r.t. δu. In contrast to that, the latter was necessary to obtain convergence of the steepest descent method. Concerning a step size selection for δw in view of the constraint given in Section 6.2.4, which are shown in Figs. 6.4(c) and (d) also, the reported strong necessity in terms of convergence of the primal-dual method was proved in addition.

6.3 Conclusion

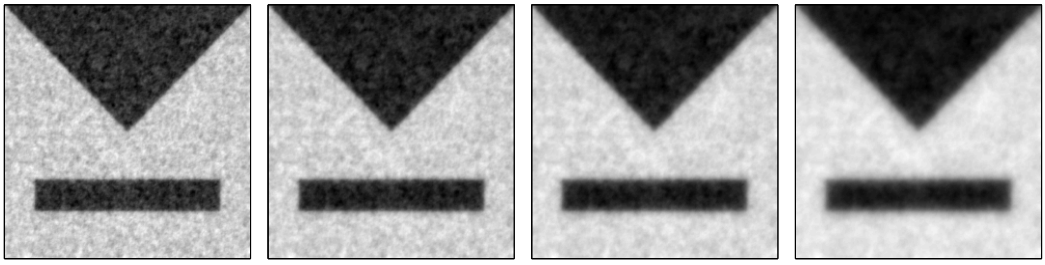
We presented the TV-based image restoration approach and explained its advantages in comparison to L^2-based approaches. The ill-posedness of the associated Euler-Lagrange equations was explained and the standard remedy of introducing a

nonlinear residual, which corresponds to the threshold of 10^{-4} on $\|\phi\|_{L^2}$ used in the experiments by Chan et al. Furthermore, a reduction factor of 0.25 for β, which was applied at each outer iteration, was found to be the smallest one not leading to divergence, while manually trying values at steps of 0.05.

perturbation parameter β was presented. Furthermore, four of the most common iterative solving methods were described and their different approaches to deal with the nonlinearity were discussed. Finally, numerical results for a synthetic as well as for a real world example, showing the convergence behavior for each method, as well as the difference between TV-based and L^2-based denoising for varying regularization strengths were given and discussed. In terms of computation time, the experiments revealed that although the primal-dual Newton Method needs less iterations to converge, the primal Newton method turned out to be faster, at least for the implementation employed here.

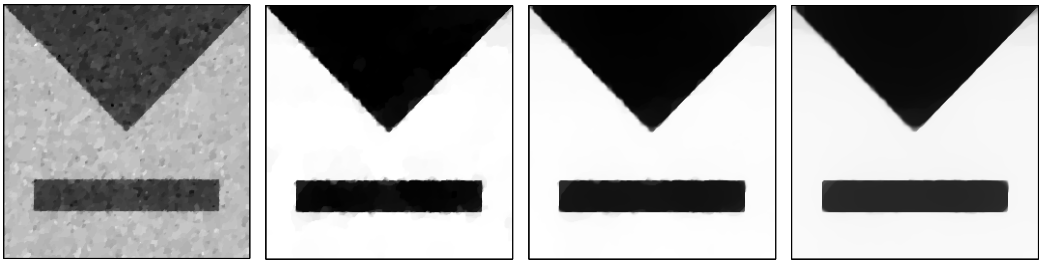


(a) Ground truth (b) with noise added



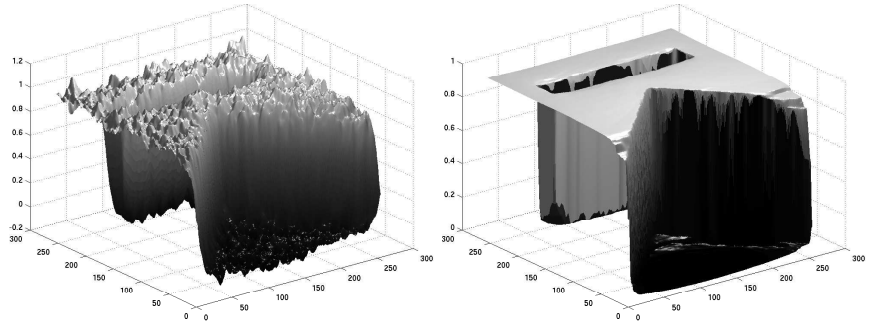
(c) $\alpha = 1$, $e = 0.14$ (d) $\alpha = 5$, $e = 0.127$ (e) $\alpha = 10$, $e = 0.127$ (f) $\alpha = 20$, $e = 0.15$

Denoising results for L^2-norm regularization



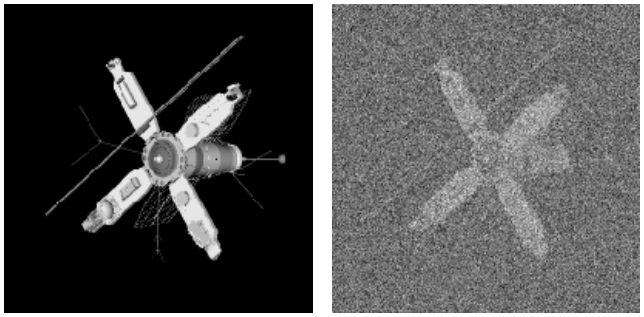
(g) $\alpha = \frac{1}{4}$, $e = 0.1$ (h) $\alpha = \frac{1}{2}$, $e = 0.057$ (i) $\alpha = 1, e = 0.061$ (j) $\alpha = 2, e = 0.089$

Denoising results for TV-norm regularization

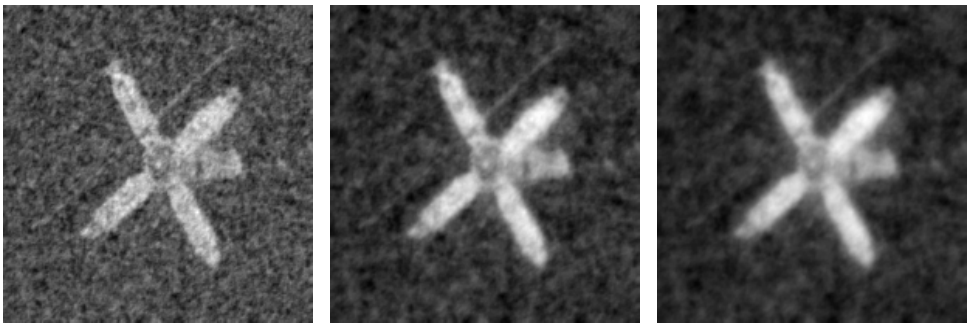


(k) 3D-plot of (d) (l) 3D-plot of (h)

Figure 6.2: **L^2-norm and TV-norm regularization in comparison for a synthetic data set.** (a) Synthetic ground truth image (256×256 pixels, intensity range $[0, 1]$). (b) Input data generated by adding mean-free white Gaussian noise (signal-to-noise ratio: -0.5 dB). (c)–(f) Denoising results and its rel. L^2 error to ground truth in applying L^2 norm-based restoration for varying regularization strengths α. (f)–(h) Results and relative L^2 error in applying TV-based restoration for varying regularization strengths α ($\beta = 10^{-8}$ in all experiments). (k) and (l) 3D-plots of the best results for each regularization type.



(a) Ground truth (b) with noise added



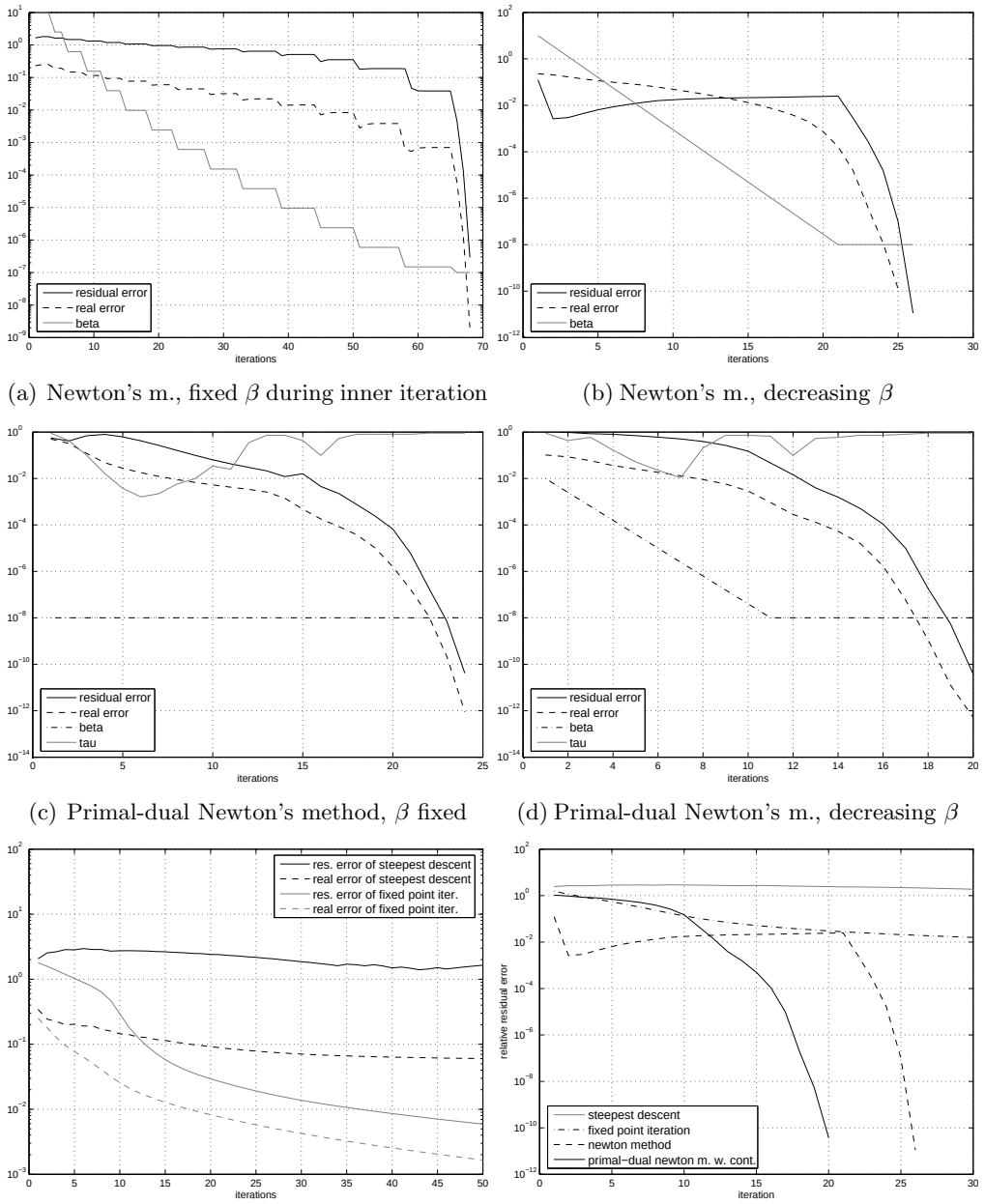
(c) $\alpha = 1$, $e = 0.54$ (d) $\alpha = 5$, $e = 0.4$ (e) $\alpha = 10$, $e = 0.41$

Denoising results for L^2-norm regularization

(f) $\alpha = \frac{1}{4}$, $e = 0.37$ (g) $\alpha = \frac{1}{2}$, $e = 0.31$ (h) $\alpha = 1$, $e = 0.36$

Denoising results for TV-norm regularization

Figure 6.3: **L^2-norm and TV-norm regularization in comparison for real-world data set.** (a) Ground truth image (200×200 pixels, intensity range $[0, 1]$). (b) Generated input image by adding mean-free white Gaussian noise (signal-to-noise ratio -7 dB). (f)–(h) Denoising results and relative L^2 error to the ground truth in applying L^2 norm-based restoration for varying regularization strengths α. (f)–(h) Results and relative L^2 error in applying TV-based restoration for varying regularization strengths α ($\beta = 10^{-8}$ in all experiments) .



(a) Newton's m., fixed β during inner iteration

(b) Newton's m., decreasing β

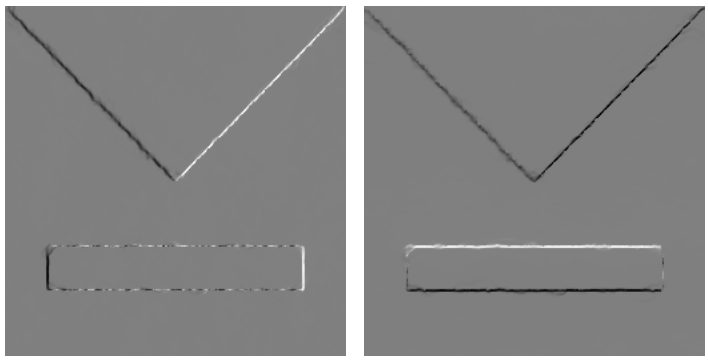
(c) Primal-dual Newton's method, β fixed

(d) Primal-dual Newton's m., decreasing β

(e) Steepest descent and fixed point iteration

(f) Evolvement of L^2-error for each method

Figure 6.4: **Convergence diagrams for the different solving methods.** Input data: synthetic example, as depicted in Fig. 6.2(b). Error measure: relative L^2 error. Parameter values: $\alpha = 1$, $\beta = 10^{-8}$ in all experiments. (a) $\beta_0 = 10$, red. factor for β: 0.25, rel. res. error threshold for Newton's method: 10^{-5}. (b) $\beta_0 = 10$, $N_\beta = 21$. (c) $\rho = 0.9$. (d) $\beta_0 = 10^{-2}$, $N_\beta = 11$, $\rho = 0.9$. See Sec. 6.2.5 for further descriptions.



(a) $w_1|\nabla u| = \partial_x u$ (b) $w_2|\nabla u| = \partial_y u$

Figure 6.5: **Depiction of the dual variable w when carrying out the primal-dual Newton's method for the synthetic data set.** It is depicted the x- and y-component of w, multiplied by the Euclidean norm of ∇u at each node, which are the result of applying the primal-dual Newton Method to the artificial example depicted in Fig. 6.2(a) ($\alpha = 1$, $\beta = 10^{-8}$).

Chapter 7

A Control Approach to Nonlinear Domain Decomposition

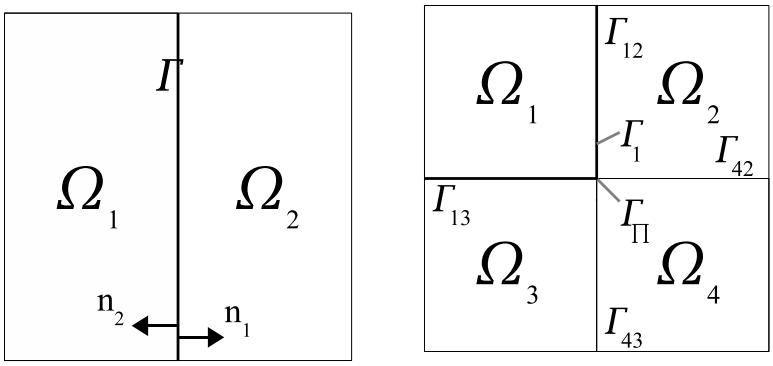
Whereas the previous chapters dealt with the domain decomposition of *linear* PDE problems, in the remainder of this work we focus on the *non-overlapping* decomposition of *nonlinear* PDE-based and energy minimization-based problems. In the case of nonlinear PDE problems, overlapping Schwarz methods can be directly applied as well (see, e.g. [91, 92] and the references therein). With non-overlapping methods however, a local linearization via a Newton-like algorithm is usually employed and an iterative substructuring technique is used to parallelize the resulting linear system at each nonlinear iteration [25, 45].

More recently, there a different approach has been proposed for nonlinear non-overlapping DD, namely by means of optimal control theory [86, 65, 67, 66]. In the scant literature available concerning that approach, the nonlinear model problem is almost always that of Navier-Stokes fluid estimation. In this chapter, we give a step-by-step explanation of the control-based decomposition in generic variational formulation and show its application to the TV-based denoising problem explicitly.

The organization of the chapter is as follows. We start by introducing the central energy minimization problem with nonlinear equality constraints on two subdomains, and explain its link to the multi-domain formulation (see Chapter 3). Thereafter, we apply a Lagrangian relaxation, derive the corresponding optimality system, and focus on the numerical disadvantages in solving the latter directly. Instead, we focus on the two different ways for calculating the gradient, then being used in a gradient descent iteration. We extend the findings to the case of many subdomains by considering a 2×2 partition and elucidate the special treatment

of corner points. For both partition cases, we present results of numerical experiments based on our model problem, thereby studying the convergence behavior and showing the empirical feasibility of the approach.

7.1 The Case of Two Subdomains



(a) Partitioning of Ω into two subdomains.

(b) Partitioning into four subdomains with one corner point Γ_Π.

Figure 7.1: **Illustration of the two model partitions used.**

7.1.1 Problem Statement

As in Chapter 3, we assume a partition of Ω into two subdomains Ω_1, Ω_2, such that $\Omega = \Omega_1 \cup \Omega_2, \Omega_1 \cap \Omega_2 = \emptyset$, and let Γ denote the common boundary, i.e. $\Gamma := \overline{\Omega}_1 \cap \overline{\Omega}_2$. See Fig. 7.1(a) for an example.

As a representative for the class of nonlinear elliptic PDE problems we consider that of TV denoising, as introduced in the previous chapter. In particular, here we have $K = I$, such that the nonlinear PDE reads

$$u - \alpha \nabla \cdot \left(\frac{\nabla u}{|\nabla u|_\beta} \right) = f \tag{7.1}$$

$$\Leftrightarrow \qquad A(u) = f \ , \tag{7.2}$$

where $\beta > 0$, and with the homogeneous Neumann boundary conditions

$$\frac{\partial u}{\partial n} = 0 \text{ on } \partial\Omega \ . \tag{7.3}$$

Let us consider the restriction of (7.1) to either of the subdomains, while modifying the Neumann boundary conditions as follows:

$$\begin{aligned} A(u_1) = f_1 \quad \text{and} \quad \frac{\partial u_1}{\partial n_1} = g \text{ on } \Gamma, \quad \frac{\partial u_1}{\partial n} = 0 \text{ on } \partial\Omega_1 \setminus \Gamma\,, \\ A(u_2) = f_2 \quad \text{and} \quad \frac{\partial u_2}{\partial n_2} = -g \text{ on } \Gamma, \quad \frac{\partial u_2}{\partial n} = 0 \text{ on } \partial\Omega_2 \setminus \Gamma\,, \end{aligned} \tag{7.4}$$

where g is a given function in $L^2(\Gamma)$, and

$$u = \begin{cases} u_1(\mathbf{x}) & \mathbf{x} \in \Omega_1 \\ u_2(\mathbf{x}) & \mathbf{x} \in \Omega \setminus \Omega_1\,. \end{cases} \tag{7.5}$$

That is, by (7.4) the original problem $A(u) = f$ is split into two subproblems, whose Neumann boundary conditions have been modified so that the normal derivatives of the local solutions u_1 and u_2 are equal to g and $-g$, respectively. Since g is given, we can thus write $u_1(g)$ and $u_2(g)$.

Furthermore, (7.4) shall now serve as the constraints of the constrained optimization problem

$$\min_{u_1,u_2,g} \frac{1}{2}\int_\Gamma (u_1 - u_2)^2 \, d\chi + \frac{\gamma}{2}\int_\Gamma g^2 \, d\chi := J_\Gamma\big(u_1, u_2, g\big) \tag{7.6}$$
$$\text{subject to (7.4)},$$

cf. [86], whereas for the time being we assume $\gamma = 0$, i.e. neglect the second integral. Then, for a global minimum $(\hat{u}_1, \hat{u}_2, \hat{g})$ of J_Γ it obviously holds that $\hat{u}_1(g) = \hat{u}_2(g)$ on Γ, because of the definition of J_Γ; as well as $\partial_n \hat{u}_1 = g = -\partial_n u_2$, because of the construction of the constraint equations. That is, a solution to problem (7.6) does, in negligence of the second integral, also satisfy

$$\begin{cases} A_1(u_1) = f_1 \quad \text{and} \quad \dfrac{\partial u_1}{\partial n_1} = 0 \text{ on } \partial\Omega_1 \setminus \Gamma \\ \quad \dfrac{\partial u_1}{\partial n_1} = g = -\dfrac{\partial u_2}{\partial n_2} \quad \text{on} \quad \Gamma \\ \quad u_1 = u_2 \quad \text{on} \quad \Gamma \\ A_2(u_2) = f_2 \quad \text{and} \quad \dfrac{\partial u_2}{\partial n_2} = 0 \text{ on } \partial\Omega_2 \setminus \Gamma \end{cases} \tag{7.7}$$

which is the so-called *multi-domain formulation* of the original problem $A(u) = f$. Interestingly, if A would be a linear operator, the optimization problem (7.6) could therefore be associated with the well-known class of non-overlapping domain

decomposition methods, *substructuring methods*, since the multi-domain formulation serves as basis of the Steklov-Poincaré interface equation there, cf. [101]. Although, since one assumption for the latter is linearity of the operator, it is not applicable here and hence classical substructuring methods are not feasible in this case. On the other hand, by following the constrained optimization-based approach (7.6), we are also able to exploit parallelization for nonlinear problems.

Concerning the second integral in (7.6), its purpose is to prevent getting arbitrarily large solutions for g, since its magnitude is not involved in the first integral. However, experiments with the chosen model problem on two and four subdomains showed convergence for $\gamma = 0$, see details below.

Independent of that, the problem in (7.6) belongs to the well-known class of *optimal control problems*, see, e.g., [89], where g here is a *boundary control*; J_Γ is the *objective functional*; (7.4) are the *state equations* and $\big(u_1(g), u_2(g)\big)$ are denoted as the *state*.

7.1.2 Lagrange Relaxation and the Optimality System

A direct approach to the optimal control problem (7.6) could be to solve the optimality system which belongs to the Lagrange relaxation of the constrained optimization problem, see, e.g., [66].

In order to simplify the Lagrange multiplier function to be defined below, we will make use of the weak formulation of the constraint equation system (7.4):

$$\begin{aligned} a_1(u_1, v_1) &= b_1(v_1) + \big(u_1, v_1\big)_\Gamma, \qquad \forall v_1 \in V_1, u_1 \in V_1 \\ a_2(u_2, v_2) &= b_2(v_2) - \big(u_2, v_2\big)_\Gamma, \qquad \forall v_2 \in V_2, u_2 \in V_2, \end{aligned} \tag{7.8}$$

cf., e.g., [4], which can be written in more compact form

$$\big(F_1(u_1, g), v_1\big)_{\Omega_1} = 0, \qquad \forall v_1 \in V_1,\ u_1 \in V_1 \tag{7.9}$$

$$\big(F_2(u_2, g), v_2\big)_{\Omega_2} = 0, \qquad \forall v_2 \in V_2,\ u_2 \in V_2\,, \tag{7.10}$$

by making use of the nonlinear operators $F_i(\cdot, \cdot)$:

$$\big(F_i(u_i, g), v_i\big)_{\Omega_i} := a_i(u_i, v_i) - b_i(v_i) + (-1)^{i-1}(g, v_i)_\Gamma, \quad \forall v_i \in V_i,\ i = 1, 2. \tag{7.11}$$

Now, we are ready to give a compact definition of the Lagrange functional by

$$L(u_1, u_2, g, \lambda_1, \lambda_2) := J_\Gamma(u_1, u_2, g) - \big(F_1(u_1, g), \lambda_1\big)_{\Omega_1} - \big(F_2(u_2, g), \lambda_2\big)_{\Omega_2}, \tag{7.12}$$

where $\lambda_1 \in V_1, \lambda_2 \in V_2$ are the Lagrange multiplier functions.

First-order necessary conditions for finding a solution $(\hat{u}_1, \hat{u}_2, \hat{g})$ to the original problem (7.6) are to find a stationary point $(\hat{u}_1, \hat{u}_2, \hat{g}, \hat{\lambda}_1, \hat{\lambda}_2)$ of the Lagrange functional L. That is, one has to solve the system

$$\nabla^5 L(u_1, u_2, g, \lambda_1, \lambda_2) = 0,$$

where $\nabla^5 := (\partial/\partial u_1, \partial/\partial u_2, \partial/g, \partial/\partial\lambda_1, \partial/\partial\lambda_2)$, which gives the *optimality system* to the Lagrange relaxation and is derived in detail in the following.

Partially differentiating $L(u_1, u_2, g, \lambda_1, \lambda_2)$ with respect to the Lagrange functions λ_1 and λ_2, respectively, yield the so-called *state equations*

$$\left\langle \frac{\partial L}{\partial \lambda_i}, v_i \right\rangle_{\Omega_i} = 0, \qquad \forall v_i \in V_i,\ i = 1, 2 \tag{7.13}$$

$$\Leftrightarrow \qquad \big(F_i(u_i, g), v_i\big)_{\Omega_i} = 0, \qquad \forall v_i \tag{7.14}$$

$$\Leftrightarrow \qquad a_i(u_i, v_i) = b_i(v_i) + (-1)^{i-1}(g, v_i)_\Gamma, \qquad \forall v_i \tag{7.15}$$

which are just the constraint equations of the original problem. On the other hand, deriving with respect to u_1 and u_2, respectively, results in the *adjoint* or *co-state equations*

$$\left\langle \frac{\partial L}{\partial u_i}, v_i \right\rangle_{\Omega_i} = 0, \qquad \forall v_i \in V_i,\ i = 1, 2 \tag{7.16}$$

$$\Leftrightarrow \qquad \left\langle \frac{\partial J_\Gamma}{\partial u_i}, v_i \right\rangle_{\Omega_i} - \left(\left\langle \frac{\partial F_i}{\partial u_i}, v_i \right\rangle_{\Omega_i}, \lambda_i \right)_{\Omega_i} = 0, \qquad \forall v_i \tag{7.17}$$

$$\Leftrightarrow \qquad a_i'(u_i;\, v_i, \lambda_i) = \big(u_{1|\Gamma} - u_{2|\Gamma}, (-1)^{i-1} v_{i|\Gamma}\big)_\Gamma, \qquad \forall v_i \tag{7.18}$$

where $a_i'(u_i;\, v_i, \lambda_i)$ is defined as follows:

$$a_i'(u_i;\, v_i, \lambda_i) := \left(\left\langle \frac{\partial F_i(u_i, g)}{\partial u_i}, v_i \right\rangle_{\Omega_i}, \lambda_i \right)_{\Omega_i} \tag{7.19}$$

$$= \int_{\Omega_i} v_i\ \lambda_i + \alpha \left(\frac{1}{(\nabla u_i^\top \nabla u_i + \beta)^{1/2}} \nabla v_i^\top - \frac{1}{(\nabla u_i^\top \nabla u_i + \beta)^{3/2}} \nabla u_i^\top \nabla v_i \nabla u_i^\top \right) \nabla \lambda_i \, d\mathbf{x} \tag{7.20}$$

$$= \int_{\Omega_i} v_i \lambda_i + \frac{\alpha}{|\nabla u_i|_\beta} \nabla v_i^\top \left(I - \frac{\nabla u_i \nabla u_i^\top}{|\nabla u_i|_\beta^2} \right) \nabla \lambda_i \, d\mathbf{x}. \tag{7.21}$$

Additionally, solutions to u_1 and u_2 to (7.18) are denoted as *co-states*. Finally, by partially deriving for the control g we obtain the *optimality condition*

$$\left\langle \frac{\partial L}{\partial g}, \tilde{g} \right\rangle_\Gamma = 0, \qquad \forall \tilde{g} \in L^2(\Gamma),\ i = 1,2 \tag{7.22}$$

$$\Leftrightarrow \qquad \left\langle \frac{\partial J_\Gamma}{\partial g}, \tilde{g} \right\rangle_\Gamma - \left(\left\langle \frac{\partial F_1}{\partial g}, \tilde{g} \right\rangle_\Gamma, \lambda_1 \right)_\Gamma - \left(\left\langle \frac{\partial F_2}{\partial g}, \tilde{g} \right\rangle_\Gamma, \lambda_2 \right)_\Gamma = 0, \quad \forall \tilde{g} \tag{7.23}$$

$$\Leftrightarrow \qquad \gamma\big(g, \tilde{g}\big)_\Gamma + \big(\lambda_{1|\Gamma} - \lambda_{2|\Gamma}, \tilde{g}\big)_\Gamma = 0, \quad \forall \tilde{g}. \tag{7.24}$$

To summarize, in order to find a solution for the Lagrange relaxation with the model problem one has to solve the following coupled system of five equations

$$\begin{cases} \displaystyle\int_{\Omega_i} u_i\, v_i \; + \alpha \frac{\nabla u_i^\top \nabla v_i}{|\nabla u_i|_\beta}\, d\mathbf{x} = \int_{\Omega_i} f_i\, v_i\, d\mathbf{x} + (-1)^{i-1}\big(g, v_i\big)_\Gamma\,, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \forall v_i \in V_i\ ,\ i = 1,2\,, \\ \displaystyle\int_{\Omega_i} v_i \lambda_i + \frac{\alpha}{|\nabla u_i|_\beta} \nabla v_i^\top \left(I - \frac{\nabla u_i \nabla u_i^\top}{|\nabla u_i|_\beta^2} \right) \nabla \lambda_i\, d\mathbf{x} = \big(u_{1|\Gamma} - u_{2|\Gamma}, v_{i\Gamma}\big)_\Gamma\,, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \forall v_i \in V_i\ ,\ i = 1,2\,, \\ \qquad\qquad\qquad\qquad \gamma\big(g, \tilde{g}\big)_\Gamma = -\big(\lambda_{1|\Gamma} - \lambda_{2|\Gamma}, \tilde{g}\big)_\Gamma\,, \quad \forall \tilde{g} \in L^2(\Gamma). \end{cases} \tag{7.25}$$

A direct approach would be to discretize and solve the whole system in a fixed point iteration[1], similar to that in Chapter 6, which is usually denoted by *one-shop method* in literature, see, e.g., [66]. However, because of the large number of unknowns involved, the convergence rate is known to be relatively slow. Furthermore, and more important with respect to parallel computing, the solving (7.25) cannot be broken apart into independent iterations, hence no possibilities for coarse-grained parallel computation would be given.

7.1.3 Gradient-based Solving

As opposed to the one-shot method, other methods minimize J_Γ by carrying out a gradient descent with respect to g, while determining $u_1(g)$ and $u_2(g)$ according to (7.4). In particular, for each iteration k, given the current value g^k, one first determines the states $u_1^k(g^k)$ and $u_2^k(g^k)$ by solving the state equations, secondly

[1] That is, one would solve the first two equations w.r.t. u_1 and u_2, the second two equations w.r.t. λ_1 and λ_2 and the last w.r.t. g, respectively.

calculates $\lambda_1^k(u_1^k)$ and $\lambda_2^k(u_2^k)$ via the co-state equations, and thirdly determines the gradient $\frac{d}{dg}J_\Gamma\big(u_1(g^k), u_2(g^k), g^k\big)$ from which an update δg for g is deduced (See Algorithm 13 for an example.). The advantage of this iterative method, in terms of parallelization, lies in the fact each of the state and co-state equations to be solvable independently from each other. That is, u_1 and u_2 can be calculated concurrently, as well as λ_1 and λ_2, respectively. Thereby, values only on the common boundary Γ need to be exchanged.

7.1.3.1 Calculating the Gradient

In the following we will focus on the computation of the total derivative $\nabla J_\Gamma = \frac{d}{dg}J_\Gamma\big(u_1(g^k), u_2(g^k), g^k\big)$ in general, as well as for our model problem in particular.

We start with the formal structure of the total derivate of J_Γ with respect to g at (u_1, u_2, g) in an arbitrary direction $\tilde{g} \in L^2(\Gamma)$:

$$\left\langle \frac{dJ_\Gamma}{dg}, \tilde{g} \right\rangle_\Gamma = \left\langle \frac{\partial J_\Gamma}{\partial u_1}, \left\langle \frac{\partial u_1}{\partial g}, \tilde{g} \right\rangle_\Gamma \right\rangle_\Gamma + \left\langle \frac{\partial J_\Gamma}{\partial u_2}, \left\langle \frac{\partial u_2}{\partial g}, \tilde{g} \right\rangle_\Gamma \right\rangle_\Gamma + \left\langle \frac{\partial J_\Gamma}{\partial g}, \tilde{g} \right\rangle_\Gamma . \quad (7.26)$$

Defining $\tilde{u}_i(g) := \left\langle \frac{\partial u_i}{\partial g}, \tilde{g} \right\rangle_\Gamma$, $i = 1, 2$, this can be rewritten as

$$\left\langle \frac{dJ_\Gamma}{dg}, \tilde{g} \right\rangle_\Gamma = \left\langle \frac{\partial J_\Gamma}{\partial u_1}, \tilde{u}_1 \right\rangle_\Gamma + \left\langle \frac{\partial J_\Gamma}{\partial u_2}, \tilde{u}_2 \right\rangle_\Gamma + \left\langle \frac{\partial J_\Gamma}{\partial g}, \tilde{g} \right\rangle_\Gamma . \quad (7.27)$$

Here, $\tilde{u}_1(g)$ and $\tilde{u}_2(g)$ are the directions of infinitesimal change of the state functions u_1 and u_2, respectively, in dependence of the direction of infinitesimal change $\tilde{g}$. Or, in other words, $\tilde{u}_1$ and $\tilde{u}_2$ represent the variation directions in the state depending on of a variation direction in the control g. Therefore, $\tilde{u}_1(g)$ and $\tilde{u}_2(g)$ are commonly referred to as *sensitivities* in literature [66].

In general, two ways of calculating $\tilde{u}_1(g)$ and $\tilde{u}_2(g)$ exist. The first one follows from considering the total derivative of the state equations with respect to g, which is given by

$$\left(P_i \left\langle \frac{d}{dg} F_i(u_i, g), \tilde{g} \right\rangle_\Gamma , v_i \right)_{\Omega_i} = 0, \quad \forall v_i \in V_i,\ i = 1, 2 , \quad (7.28)$$

with $P_i : V(\Gamma_i) \to V(\Omega_i)$ denoting an extension by zero. Due to u_1 and u_2 depending on the control g, the chain rule applies again, yielding

$$\Leftrightarrow \qquad \left(\left\langle \frac{\partial F_i}{\partial u_i}, P_i \left\langle \frac{\partial u_i}{\partial g}, \tilde{g} \right\rangle_\Gamma \right\rangle_{\Omega_i} + P_i \left\langle \frac{\partial F_i}{\partial g}, \tilde{g} \right\rangle_\Gamma , v_i \right)_{\Omega_i} = 0, \quad \forall v_i. \quad (7.29)$$

By substitution of $\left\langle \frac{\partial u_i}{\partial g}, \tilde{g} \right\rangle_\Gamma$, $i = 1, 2$ through $\tilde{u}_1$ and $\tilde{u}_2$, respectively, we obtain

$$\Leftrightarrow \qquad \left(\left\langle \frac{\partial F_i}{\partial u_i}, \tilde{u}_i \right\rangle_{\Omega_i} + P_i \left\langle \frac{\partial F_i}{\partial g}, \tilde{g} \right\rangle_\Gamma, v_i \right)_{\Omega_i} = 0, \quad \forall v_i \tag{7.30}$$

$$\Leftrightarrow \qquad \left(\left\langle \frac{\partial F_i}{\partial u_i}, \tilde{u}_i \right\rangle_{\Omega_i}, v_i \right)_{\Omega_i} = - \left(P_i \left\langle \frac{\partial F_i}{\partial g}, \tilde{g} \right\rangle_\Gamma, v_i \right)_{\Omega_i}, \quad \forall v_i \, . \tag{7.31}$$

Hence, with equation (7.31), denoted as *sensitivity equation*, we have given the dependency between $\tilde{u}_1$, $\tilde{u}_2$ and $\tilde{g}$. Unfortunately, it is clear that, since equation (7.31) is not formally inverted, $\tilde{u}_1$, $\tilde{u}_2$ can be determined only for particular $\tilde{g}$, through solving (7.31). In recapitulation of the gradient formulation in (7.27), it becomes obvious that by (7.31) we are only able to compute the action $\frac{dJ_\Gamma}{dg}$ onto a *given* function $\tilde{g}$, but not the gradient itself.

However, in most continuous cases $\tilde{g}$ need to be arbitrary, therefore we are interested in a formulation without the incorporation of the sensitivities $\tilde{u}_1$ and $\tilde{u}_2$, which is referred to as 'calculation of the gradient through adjoint equations' in literature, [66]. Such a formulation can be reached by considering the adjoint equations (7.18),

$$\left(\left\langle \frac{\partial F_i}{\partial u_i}, v_i \right\rangle_{\Omega_i}, \lambda_i \right)_{\Omega_i} = \left\langle \frac{\partial J_\Gamma}{\partial u_i}, v_i \right\rangle_\Gamma, \qquad \forall v_i \in V_i, \; i = 1, 2, \tag{7.32}$$

of the previous section again. Since this holds for any v_i, it is in particular true for setting $v_i = \tilde{u}_i$. Analogical, (7.31) is true for an arbitrary v_i and thus for an particular $v_i = \lambda_i$. In applying these substitutions to (7.32) and (7.31), respectively, and comparing the outcomes, one can deduce that

$$\left\langle \frac{\partial J_\Gamma}{\partial u_i}, \tilde{u}_i \right\rangle_\Gamma = - \left(P_i \left\langle \frac{\partial F_i}{\partial g}, \tilde{g} \right\rangle_\Gamma, \lambda_i \right)_{\Omega_i} . \tag{7.33}$$

Thus, terms in the gradient formulation (7.27) involving the sensitivities $\tilde{u}_1$ and $\tilde{u}_2$ can now be substituted due to (7.33), which yields the gradient formulation

$$\left\langle \frac{dJ_\Gamma}{dg}, \tilde{g} \right\rangle_\Gamma = - \left(P_1 \left\langle \frac{\partial F_1}{\partial g}, \tilde{g} \right\rangle_\Gamma, \lambda_1 \right)_{\Omega_1} - \left(P_2 \left\langle \frac{\partial F_2}{\partial g}, \tilde{g} \right\rangle_\Gamma, \lambda_2 \right)_{\Omega_2} + \left\langle \frac{\partial J_\Gamma}{\partial g}, \tilde{g} \right\rangle_\Gamma \tag{7.34}$$

$$\Leftrightarrow \quad \left\langle \frac{dJ_\Gamma}{dg}, \tilde{g} \right\rangle_\Gamma = \left\langle \frac{\partial J_\Gamma}{\partial g}, \tilde{g} \right\rangle_\Gamma - \left(P_1 \left\langle \frac{\partial F_1}{\partial g}, \tilde{g} \right\rangle_\Gamma, \lambda_1 \right)_{\Omega_1} - \left(P_2 \left\langle \frac{\partial F_2}{\partial g}, \tilde{g} \right\rangle_\Gamma, \lambda_2 \right)_{\Omega_2} . \tag{7.35}$$

Obviously, we have now given a closed-form of the gradient for arbitrary $\tilde{g}$ and not for particular ones only, since there is no need to solve the sensitivity equations in an intermediate step any longer. Instead, one has to solve the adjoint equations (7.32) for λ_1 and λ_2, respectively. Since the latter only involves the state functions u_1, u_2 and the control g, but not the change direction $\tilde{g}$, this solving has to be done only *once* for any $\tilde{g}$.

7.1.3.2 Application to the Model Problem

Now that we have general formulation of the gradient, the concrete ones for our model problem are straightforward. Obviously, in considering the definition of J_Γ as given in (7.6), the gradient formulation in (7.27) for the model problems here reads

$$\left\langle \frac{dJ_\Gamma}{dg}, \tilde{g} \right\rangle_\Gamma = \big(u_{1|\Gamma} - u_{2|\Gamma}, \tilde{u}_{2|\Gamma} - \tilde{u}_{2|\Gamma}\big)_\Gamma + \gamma\big(g, \tilde{g}\big)_\Gamma. \tag{7.36}$$

Furthermore, the sensitivity equations corresponding to (7.31) is of the form

$$\begin{aligned} a_1'(u_1;\, \tilde{u}_1, \xi_1) &= (\tilde{g}, \xi_1)_\Gamma, \quad \forall \xi_1 \in V_1 \\ a_2'(u_2;\, \tilde{u}_2, \xi_2) &= -(\tilde{g}, \xi_2)_\Gamma, \quad \forall \xi_2 \in V_2, \end{aligned} \tag{7.37}$$

with $a_i'(\cdot;\, \cdot, \cdot)$ as defined in (7.19) in connection with the adjoint equations.

Alternatively, in following the adjoint equations approach, the gradient formulation in (7.35) here reads

$$\left\langle \frac{dJ_\Gamma}{dg}, \tilde{g} \right\rangle_\Gamma = \big(\tilde{g}, \lambda_1\big)_\Gamma - \big(\tilde{g}, \lambda_2\big)_\Gamma + \gamma(g, \tilde{g})_\Gamma \tag{7.38}$$

$$= \big(\tilde{g}, \lambda_{1|\Gamma} - \lambda_{2|\Gamma}\big)_\Gamma + \gamma\big(g, \tilde{g}\big)_\Gamma \tag{7.39}$$

or, in explicit formulation

$$\frac{dJ_\Gamma}{dg} = \big(\lambda_{1|\Gamma} - \lambda_{2|\Gamma}\big) + \gamma g, \tag{7.40}$$

where the co-states λ_1, λ_2 are to be determined by solving the adjoint equations

$$\begin{aligned} a_1'(u_1;\, v_1, \lambda_1) &= \big(u_{1|\Gamma} - u_{2|\Gamma}, v_{1|\Gamma}\big)_\Gamma, \qquad \forall v_1 \in V_1 \\ a_2'(u_2;\, v_2, \lambda_2) &= \big(u_{1|\Gamma} - u_{2|\Gamma}, -v_{2|\Gamma}\big)_\Gamma, \qquad \forall v_2 \in V_2 \end{aligned} \tag{7.41}$$

which have been already introduced with the Lagrange multiplier approach, see (7.18).

The relation between the sensitivity and the co-state functions, as shown generally in (7.33), can be reproduced here by setting $\xi_1 = \lambda_1$, $\xi_2 = \lambda_2$ in (7.37), and $v_1 = \tilde{u}_1$, $v_2 = \tilde{u}_2$ in (7.41), resulting in

$$\begin{aligned}(\tilde{g}, \lambda_1)_\Gamma &= \big(u_{1|\Gamma} - u_{2|\Gamma}, v_{1|\Gamma}\big)_\Gamma \\ (\tilde{g}, \lambda_2)_\Gamma &= \big(u_{1|\Gamma} - u_{2|\Gamma}, -v_{2|\Gamma}\big)_\Gamma\end{aligned} \tag{7.42}$$

and thus

$$\big(\tilde{g}, \lambda_{1|\Gamma} - \lambda_{2|\Gamma}\big)_\Gamma = \big(u_{1|\Gamma} - u_{2|\Gamma}, \tilde{u}_{2|\Gamma} - \tilde{u}_{2|\Gamma}\big)_\Gamma, \tag{7.43}$$

which, utilized to substitute the first term at the right-hand side of the first gradient formulation (7.36), just leads to the formulation in (7.38).

7.1.3.3 The Solving Algorithm

After having explained the calculation of $\frac{d}{dg} J_\Gamma\big(u_1(g^k), u_2(g^k), g^k\big)$ in detail, we will give an example of a simple gradient method in Algorithm (13), which was implemented for the experimental studies to be presented in the succeeding section.

As outlined in the beginning of this section, in step 1 of the algorithm, it is solved for the states u_1^k and u_2^k, which can be done on different processing nodes, since the corresponding problems are independent from each other. The same holds true for the adjoint equations in step 2, where the values of u_1^k and u_2^k only on the common boundary Γ have to be determined sequentially beforehand. In step 3, the current derivative is computed, from which an update is deduced in step 4. Furthermore, a line search along the update direction δg is employed (step 5) which has turned out to necessary to guarantee global convergence[2]. This step requires additional solvings of the state equations, in order to determine $u_1(g^k + \tau\delta g^k)$ and $u_2(g^k + \tau\delta g^k)$, respectively, which can be done in parallel again. The remaining steps are self-explanatory.

7.1.3.4 Experimental Studies

After discretization by first-order conforming finite elements, a LU decomposition was chosen as solver for the adjoint equations, which are linear w.r.t. λ_1 and λ_2, respectively. For the state equations, which are nonlinear w.r.t. u_1 and u_2, the primal-dual Newton method, cf. Section 6.2.4, with LU decomposition as inner solver, was employed. The nonlinear problems were solved up to a relative residual error of less then 10^{-10} for all experiments. The regularization strength α was set to the relatively high value 1.0, in order to demonstrate that the algorithm returns

[2] In our implementation we used ten nested iterations starting with an interval of $[0, 5]$

Algorithm 13: Gradient descent w.r.t. to the control g on two subdomains

$g^0 \leftarrow 0$, $k \leftarrow 0$

Choose a feasible g^0.

do

1. *Solve the state equations for* u_1^k, u_2^k *(in parallel):*

$$\begin{aligned} a_1(u_1^k, v_1) &= b_1(v_1) + (g^k, v_1)_\Gamma, \qquad \forall v_1 \in V(\Omega_1) \\ a_2(u_2^k, v_2) &= b_2(v_2) - (g^k, v_2)_\Gamma, \qquad \forall v_2 \in V(\Omega_2) \end{aligned} \tag{7.44}$$

2. *Solve the adjoint equations for* λ_1^k, λ_2^k *(in parallel):*

$$\begin{aligned} a_1'(u_1^k;\, v_1, \lambda_1^k) &= (u_1^k - u_2^k, v_1)_\Gamma, \qquad \forall v_1 \in V(\Omega_1) \\ a_2'(u_2^k;\, v_2, \lambda_2^k) &= (u_1^k - u_2^k, v_2)_\Gamma, \qquad \forall v_2 \in V(\Omega_2) \end{aligned} \tag{7.45}$$

3. *Calculate the gradient:*

$$\nabla J_\Gamma^k \leftarrow (\lambda_{1|\Gamma}^k - \lambda_{2|\Gamma}^k) + \gamma g^k \tag{7.46}$$

4. *Calculate a new update direction* δg^k*:*

$$\delta g^k \leftarrow -\nabla J_\Gamma^k + \rho^k \delta g^{k-1}, \quad \text{where} \quad \rho^k \leftarrow \frac{\nabla J_\Gamma^k \cdot \left(\nabla J_\Gamma^k - \nabla J_\Gamma^{k-1}\right)}{\nabla J_\Gamma^{k-1} \cdot \nabla J_\Gamma^{k-1}} \tag{7.47}$$

5. *Do a line search along* δd^k *to determine a new step size* τ^k*:*

$$\tau^k \leftarrow \min_{0<\tau\le\tau_{\max}} J_\Gamma\big(u_1(g^k + \tau\delta g^k),\, u_2(g^k + \tau\delta g^k),\, g^k + \tau\delta g^k\big) \tag{7.48}$$

6. *Update the control:*

$$g^{k+1} \leftarrow g^k + \tau^k \delta g^k \tag{7.49}$$

7. $k \leftarrow k + 1$

until $\|\nabla J_\Gamma^{k-1}\| / \|\nabla J_\Gamma^0\| \le \epsilon$

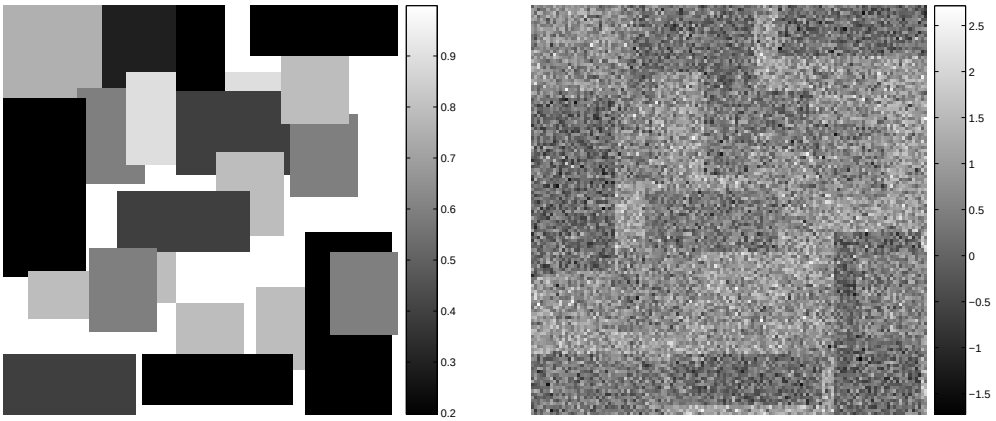
Merge the local solutions:

$$\begin{aligned} u_{|\Omega_1} &\leftarrow u_1^{k-1} \\ u_{|\Omega_2} &\leftarrow u_2^{k-1} \end{aligned} \tag{7.50}$$

the correct global solution also under substantial regularization. Finally, we set $\beta = 10^{-6}$ in all experiments. Figure 7.2(b) served as input image, which was generated by adding Gaussian noise to the synthetic image depicted in Fig. 7.2(a). All error measurements refer to the solution of the original problem calculated without domain decomposition for the same set of parameter values.

The goal of the experiments, whose results are shown in Figure 7.3, was to study the general feasibility of the proposed domain decomposition approach, i.e. convergence, convergence rate, as well as the spatial distribution of the error in comparison of a sequential reference solution with the same parameters and the same local solver's accuracy (in terms of error threshold).

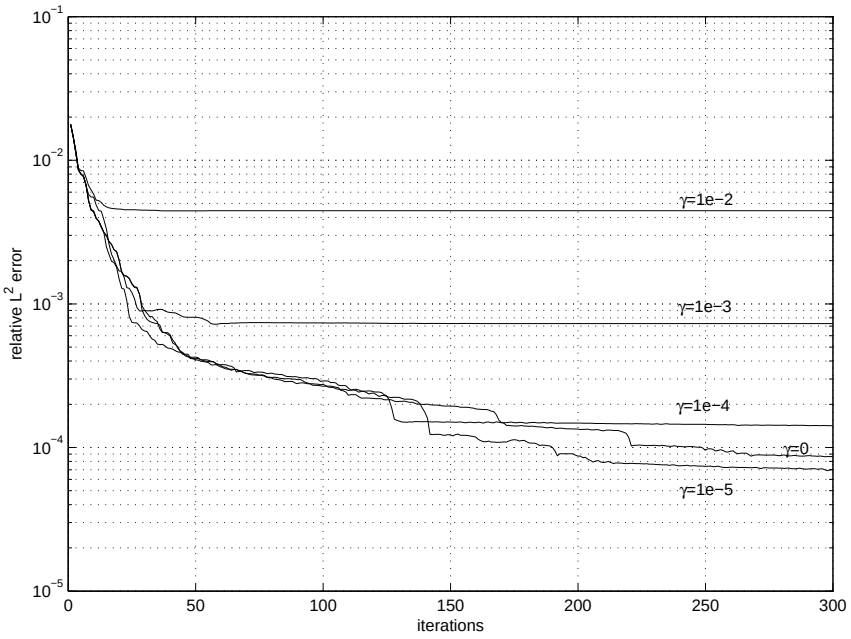
The results in Fig. 7.3(a) reveal that the relative L^2-error linearly drops to 10^{-4} with a relatively good rate of ≈ 0.89 until iteration 50, but then deteriorates to ≈ 0.99. Furthermore, setting $\gamma = 0$, i.e. switching off the regularization of g, does not lead to divergence here. In contrast, experiments show that the convergence behavior does not change significantly for setting γ to values smaller than 10^{-4} or equal to zero. Despite the worsening convergence rate, the density plots of the resulting image in Fig. 7.3(b) show that the remaining error after 50 iterations is acceptable for most image processing applications.



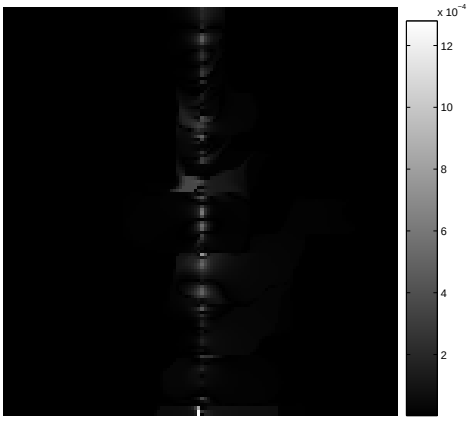
(a) Ground truth

(b) With added Gaussian noise

Figure 7.2: **Ground truth and noised input image** (128 × 128 pixels, signal-to-noise ratio -4.2 dB).



(a) Development of the L^2-error for different control regularization strengths γ



(b) Per-pixel L^2-error to reference solution after 300 iterations

Figure 7.3: **Total and per-pixel L^2-error for a two-subdomain decomposition** ($\alpha = 1.0, \beta = 10^{-6}, \gamma = 10^{-5}$).

7.2 The Case of Many Subdomains

Subsequently, we extend our approach to the case of multiple subdomains. However, it suffices to consider a 2×2 partition, since the special treatment for the one corner point there extends naturally to the case of many corner points. Thus, let us define four open subdomains $\{\Omega_i | i = 1, \ldots, 4\}$, such that $\Omega = \cup_{i=1}^4 \Omega_i$, $\cap_{i=1}^4 \Omega_i = \emptyset$, an having the common boundary $\Gamma := \cup_{\forall j \in N_>(i),\, \forall i} \Gamma_{ij} \cup \Pi$, which is further divided into the corner point $\Gamma_\Pi := \cap_{i=1}^4 \overline{\Omega}_i$, and the sections $\Gamma_{ij} := (\overline{\Omega}_i \cap \overline{\Omega}_j) \setminus \Pi$, $\forall j \in N_>(i)$, $\forall i$, whereas $N_>(i) := \{\Omega_j | \Omega_j$ is cityblock-neighbor of $\Omega_i,\ j < i\}$. See Fig. 7.1(b) for an illustrating example.

7.2.1 Problem Statement

Induced by partitioning of Ω, we decompose the original problem $A(u) = f$ on Ω into the subproblems

$$\begin{aligned}
A(u_1) = f_1 \quad &\text{and} \quad \frac{\partial u_1}{\partial n_1} = g \text{ on } \Gamma, \quad \frac{\partial u_1}{\partial n} = 0 \text{ on } \partial\Omega_1 \setminus \Gamma \\
A(u_2) = f_2 \quad &\text{and} \quad \frac{\partial u_2}{\partial n_2} = -g \text{ on } \Gamma, \quad \frac{\partial u_2}{\partial n} = 0 \text{ on } \partial\Omega_2 \setminus \Gamma \\
A(u_3) = f_3 \quad &\text{and} \quad \frac{\partial u_3}{\partial n_3} = -g \text{ on } \Gamma, \quad \frac{\partial u_3}{\partial n} = 0 \text{ on } \partial\Omega_3 \setminus \Gamma \\
A(u_4) = f_4 \quad &\text{and} \quad \frac{\partial u_4}{\partial n_4} = g \text{ on } \Gamma, \quad \frac{\partial u_4}{\partial n} = 0 \text{ on } \partial\Omega_4 \setminus \Gamma\,,
\end{aligned} \tag{7.51}$$

for $g \in L^2(\Gamma)$, where $\Gamma = \Gamma_{12} \cup \Gamma_{13} \cup \Gamma_{42} \cup \Gamma_{43} \cup \Gamma_\Pi$ here. As with the two-subdomains case, the natural Neumann boundary conditions have been modified. In order to reach a more compact formulation of the Lagrange functional later on, we will make again use of the weak formulation of (7.51),

$$\begin{aligned}
a_1(u_1, v_1) &= b_1(v_1) + \big(g, v_1\big)_\Gamma, \qquad \forall v_1 \in V_1,\ u_1 \in V(\Omega_1) \\
a_2(u_2, v_2) &= b_2(v_2) - \big(g, v_2\big)_\Gamma, \qquad \forall v_2 \in V_2,\ u_2 \in V(\Omega_2) \\
a_3(u_3, v_3) &= b_3(v_3) - \big(g, v_3\big)_\Gamma, \qquad \forall v_3 \in V_3,\ u_3 \in V(\Omega_3) \\
a_4(u_4, v_4) &= b_4(v_4) + \big(g, v_4\big)_\Gamma, \qquad \forall v_4 \in V_4,\ u_4 \in V(\Omega_4)\,,
\end{aligned} \tag{7.52}$$

in the sequel. Furthermore, the objective functional reads:

$$J_\Gamma(u_1,u_2,u_3,u_4,g) := \frac{1}{2}\Bigg\{\sum_{i=2,3}\int_{\Gamma_{1i}}(u_{1|\Gamma_{1i}}-u_{i|\Gamma_{1i}})^2\,d\mathbf{x}+(u_{1|\Gamma_\Pi}-u_{i|\Gamma_\Pi})^2+ \sum_{i=2,3}\int_{\Gamma_{4i}}(u_{4|\Gamma_{4i}}-u_{4|\Gamma_{4i}})^2\,d\mathbf{x}+(u_{4|\Gamma_\Pi}-u_{i|\Gamma_\Pi})^2+\gamma\int_\Gamma g^2\,d\mathbf{x}\Bigg\}. \quad (7.53)$$

In contrast to the two-subdomains case, four constraints for the unknowns at Γ_{Γ_Π} are implicitly applied here: $u_{1|\Gamma}=u_{2|\Gamma}, u_{1|\Gamma}=u_{3|\Gamma}, u_{4|\Gamma}=u_{2|\Gamma}$, and $u_{4|\Gamma}=u_{3|\Gamma}$, which turned out to have an impact on the step size selection of gradient-based methods in experiments, as will be described later on.

Finally, the problem of optimal control is stated as:

$$\min_{u_1,\ldots,u_4,g} J_\Gamma(u_1,u_2,u_3,u_4,g) \quad (7.54)$$
$$\text{subject to (7.51).}$$

7.2.2 The Optimality System

Although an iterative, gradient-based method shall be used for solving (7.54), the Lagrange functional

$$L(u_1,\ldots,u_4,g,\lambda_1,\ldots,\lambda_4) := J_\Gamma(u_1,u_2,u_3,u_4,g) - \quad (7.55)$$
$$\sum_{i=1}^{4} a_i(u_i,\lambda_i)+b_i(\lambda_i)+(-1)^{i-1}\big(g,\lambda_i\big)_\Gamma\,, \quad (7.56)$$

and its optimality system are first elaborated here, since the involved adjoint equations do also appear with the gradient-based algorithm. Again, partially deriving for the Lagrange multiplier functions $\lambda_1,\ldots,\lambda_4$ yields the state equations

$$\left\langle\frac{\partial L}{\partial\lambda_i},v_i\right\rangle=0 \quad\Leftrightarrow\quad a_i(u_i,v_i)=b_i(v_i)+(-1)^{i-1}\big(g,v_i\big)_\Gamma,\quad i=1,\ldots,4\,, \quad (7.57)$$

whereas deriving for the control function g results in the optimality condition

$$\left\langle\frac{\partial L}{\partial g},\tilde{g}\right\rangle=0,\quad\forall\tilde{g}\quad\Rightarrow\quad\gamma(g,\tilde{g})_\Gamma+\sum_{i=1}^{4}(-1)^{i-1}\big(\tilde{g},\lambda_i\big)_\Gamma=0\,. \quad (7.58)$$

The adjoint equations here read as follows:

$$\begin{aligned}
a_1'(u_1;\, v_1, \lambda_1) &= \sum_{i=2,3} \big(u_{1|\Gamma_{1i}} - u_{2|\Gamma_{1i}}, v_{1|\Gamma_{1i}}\big)_{\Gamma_{1i}} + \big(u_{1|\Gamma_\Pi} - u_{i|\Gamma_\Pi}\big) v_{1|\Gamma_\Pi} \\
a_2'(u_2;\, v_2, \lambda_2) &= \sum_{i=1,4} \big(u_{i|\Gamma_{i2}} - u_{2|\Gamma_{i2}}, -v_{2|\Gamma_{i2}}\big)_{\Gamma_{i2}} - \big(u_{i|\Gamma_\Pi} - u_{2|\Gamma_\Pi}\big) v_{2|\Gamma_\Pi} \\
a_3'(u_3;\, v_3, \lambda_3) &= \sum_{i=1,4} \big(u_{i|\Gamma_{i3}} - u_{3|\Gamma_{i3}}, -v_{3|\Gamma_{i3}}\big)_{\Gamma_{i3}} - \big(u_{i|\Gamma_\Pi} - u_{3|\Gamma_\Pi}\big) v_{3|\Gamma_\Pi} \\
a_4'(u_4;\, v_4, \lambda_4) &= \sum_{i=2,3} \big(u_{4|\Gamma_{4i}} - u_{i|\Gamma_{4i}}, v_{4|\Gamma_{4i}}\big)_{\Gamma_{4i}} + \big(u_{4|\Gamma_\Pi} - u_{i|\Gamma_\Pi}\big) v_{4|\Gamma_\Pi}
\end{aligned} \tag{7.59}$$

with $a'(\cdot;\, \cdot, \cdot)$ as defined in (7.19).

7.2.3 Calculation of the Gradient

Since the general mathematical structure for computation of the gradient $\left\langle \frac{dJ_\Gamma}{dg}, \tilde{g} \right\rangle$, is described in Section 7.1.3.1, we immediately proceed to the explicit calculation for the model problem.

The gradient formulation involving the sensitivities, i.e. variation directions, $\tilde{u}_1, \ldots, \tilde{u}_4$, here reads

$$\begin{aligned}
\left\langle \frac{dJ_\Gamma}{dg}, \tilde{g} \right\rangle = &\sum_{i=2,3} \big(u_{1|\Gamma_{1i}} - u_{i|\Gamma_{1i}}, \tilde{u}_{1|\Gamma_{1i}} - \tilde{u}_{i|\Gamma_{1i}}\big)_{\Gamma_{1i}} + \sum_{i=2,3} \big(u_{1|\Gamma_\Pi} - u_{i|\Gamma_\Pi}, \tilde{u}_{1|\Gamma_\Pi} - \tilde{u}_{i|\Gamma_\Pi}\big)_{\Gamma_\Pi} \\
&+ \sum_{i=2,3} \big(u_{4|\Gamma_{4i}} - u_{i|\Gamma_{4i}}, \tilde{u}_{4|\Gamma_{4i}} - \tilde{u}_{i|\Gamma_{4i}}\big)_{\Gamma_{4i}} + \sum_{i=2,3} \big(u_{4|\Gamma_\Pi} - u_{i|\Gamma_\Pi}, \tilde{u}_{4|\Gamma_\Pi} - \tilde{u}_{i|\Gamma_\Pi}\big)_{\Gamma_\Pi} \\
&+ \gamma(g, \tilde{g})_\Gamma .
\end{aligned} \tag{7.60}$$

The associated sensitivity equations are of the form

$$\begin{aligned}
a_1'(u_1;\, \tilde{u}_1, v_1) &= \sum_{i=2,3} (\tilde{g}_{1|\Gamma_{1i}}, v_{1|\Gamma_{1i}})_{\Gamma_{1i}} + \tilde{g}_{|\Gamma_\Pi} v_{1|\Gamma_\Pi} \\
a_2'(u_2;\, \tilde{u}_2, v_2) &= -\sum_{i=1,i4} (\tilde{g}_{|\Gamma_{i2}}, v_{2|\Gamma_{i2}})_{\Gamma_{i2}} - \tilde{g}_{|\Gamma_\Pi} v_{2|\Gamma_\Pi} \\
a_3'(u_3;\, \tilde{u}_3, v_3) &= -\sum_{i=1,4} (\tilde{g}_{|\Gamma_{i3}}, v_{3|\Gamma_{i3}})_{\Gamma_{i3}} - \tilde{g}_{|\Gamma_\Pi} v_{3|\Gamma_\Pi} \\
a_4'(u_4;\, \tilde{u}_4, v_4) &= \sum_{i=2,3} (\tilde{g}_{|\Gamma_{4i}}, v_{4|\Gamma_{4i}})_{\Gamma_{4i}} + \tilde{g}_{|\Gamma_\Pi} v_{4|\Gamma_\Pi}
\end{aligned} \tag{7.61}$$

corresponding to equations (7.31), for $i = 1, \ldots, 4$, in the generic formulation.

Again, by setting $v_i = \tilde{u}_i$ in (7.59), and $v_i = \lambda_i$ in (7.61) we obtain the following equations:

$$\begin{aligned}
\sum_{i=2,3} \left(u_{1|\Gamma_{1i}} - u_{2|\Gamma_{1i}}, \tilde{u}_{1|\Gamma_{1i}}\right)_{\Gamma_{1i}} + \left(u_{1|\Gamma_\Pi} - u_{i|\Gamma_\Pi}\right) \tilde{u}_{1|\Gamma_\Pi} = \\
\sum_{i=2,3} (\tilde{g}_{1|\Gamma_{1i}}, \lambda_{1|\Gamma_{1i}})_{\Gamma_{1i}} + \tilde{g}_{|\Gamma_\Pi} \lambda_{1|\Gamma_\Pi} \,, \\
\sum_{i=1,4} \left(u_{i|\Gamma_{i2}} - u_{2|\Gamma_{i2}}, -\tilde{u}_{2|\Gamma_{i2}}\right)_{\Gamma_{i2}} - \left(u_{i|\Gamma_\Pi} - u_{2|\Gamma_\Pi}\right) \tilde{u}_{2|\Gamma_\Pi} = \\
- \sum_{i=1,4} (\tilde{g}_{|\Gamma_{i2}}, \lambda_{2|\Gamma_{i2}})_{\Gamma_{i2}} - \tilde{g}_{|\Gamma_\Pi} \lambda_{2|\Gamma_\Pi} \,, \\
\sum_{i=1,4} \left(u_{i|\Gamma_{i3}} - u_{3|\Gamma_{i3}}, -\tilde{u}_{3|\Gamma_{i3}}\right)_{\Gamma_{i3}} - \left(u_{i|\Gamma_\Pi} - u_{3|\Gamma_\Pi}\right) \tilde{u}_{3|\Gamma_\Pi} = \\
- \sum_{i=1,4} (\tilde{g}_{|\Gamma_{i3}}, \lambda_{3|\Gamma_{i3}})_{\Gamma_{i3}} - \tilde{g}_{|\Gamma_\Pi} \lambda_{3|\Gamma_\Pi} \,, \\
\sum_{i=2,3} \left(u_{4|\Gamma_{4i}} - u_{i|\Gamma_{4i}}, \tilde{u}_{4|\Gamma_{4i}}\right)_{\Gamma_{4i}} + \left(u_{4|\Gamma_\Pi} - u_{i|\Gamma_\Pi}\right) \tilde{u}_{4|\Gamma_\Pi} = \\
\sum_{i=2,3} (\tilde{g}_{|\Gamma_{4i}}, \lambda_{4|\Gamma_{4i}})_{\Gamma_{4i}} + \tilde{g}_{|\Gamma_\Pi} \lambda_{4|\Gamma_\Pi} \,,
\end{aligned} \tag{7.62}$$

between the sensitivities $\tilde{u}_i$ and the co-states λ_i. Applying these relations to replace the sensitivities in (7.60) by the co-states, yields

$$\begin{aligned}
\left\langle \frac{dJ}{dg}, \tilde{g} \right\rangle = \sum_{i=2,3} \left(\tilde{g}_{|\Gamma_{1i}}, \lambda_{1|\Gamma_{1i}} - \lambda_{i|\Gamma_{1i}}\right)_{\Gamma_{1i}} + \tilde{g}_{|\Gamma_\Pi} \left(\lambda_{1|\Gamma_\Pi} - \lambda_{i|\Gamma_\Pi}\right) \\
+ \sum_{i=2,3} \left(\tilde{g}_{|\Gamma_{i4}}, \lambda_{4|\Gamma_{i4}} - \lambda_{i|\Gamma_{i4}}\right)_{\Gamma_{i4}} + \tilde{g}_{|\Gamma_\Pi} \left(\lambda_{4|\Gamma_\Pi} - \lambda_{i|\Gamma_\Pi}\right) + \gamma(g, \tilde{g})_\Gamma,
\end{aligned} \tag{7.63}$$

i.e. the gradient formulation independent of $\tilde{g}$, which is analogical to (7.38) for the two-subdomains case. The explicit formulation reads:

$$\begin{aligned}
\frac{dJ}{dg} = \sum_{i=2,3} P_{\Gamma_{1i}} \left(\lambda_{1|\Gamma_{1i}} - \lambda_{i|\Gamma_{1i}}\right) + P_{\Gamma_\Pi} \left(\lambda_{1|\Gamma_\Pi} - \lambda_{i|\Gamma_\Pi}\right) \\
+ \sum_{i=2,3} P_{\Gamma_{4i}} \left(\lambda_{4|\Gamma_{i4}} - \lambda_{i|\Gamma_{i4}}\right)_{\Gamma_{i4}} + P_{\Gamma_\Pi} \left(\lambda_{4|\Gamma_\Pi} - \lambda_{i|\Gamma_\Pi}\right) + \gamma g \,,
\end{aligned} \tag{7.64}$$

with $P_{ij} : V(\Gamma_{ij}) \to V(\Gamma)$ and $P_\Pi : V(\Gamma_\Pi) \to V(\Gamma)$ denoting extensions by zero.

Note that the gradient at locations Γ_Π is the sum of four differences, in contrast to the points on $\Gamma\backslash\Gamma_\Pi$ where it is only of one difference. That is, the magnitude of $\frac{dJ}{dg}$ at this corner point is in average four times larger as it is for the remaining locations. Since the step size in step 5 of Algorithm 1 is selected for the whole gradient, experiments have shown that it is systematically chosen too large at Γ_Π, which leads to divergence even when employing line search. As a remedy, an additional scaling factor ν for $\frac{dJ}{dg}$ at Γ_Π is introduced, where a value of $\frac{1}{4}$ has shown to be appropriate.

7.2.4 The Solving Algorithm

As a consequence of the findings in the previous section, with the four-subdomains case, steps 1–3 of Algorithm 1 have to be replaced by the following actions:

1. Solve the state equations for $u_1^k, \ldots, u_4^k$ (in parallel):

$$\begin{aligned} a_1(u_1, v_1) &= b_1(v_1) + \big(g, v_1\big)_\Gamma, \qquad && \forall v_1 \in V(\Omega_1), \\ a_2(u_2, v_2) &= b_2(v_2) - \big(g, v_2\big)_\Gamma, \qquad && \forall v_2 \in V(\Omega_2), \\ a_3(u_3, v_3) &= b_3(v_3) - \big(g, v_3\big)_\Gamma, \qquad && \forall v_3 \in V(\Omega_3), \\ a_4(u_4, v_4) &= b_4(v_4) + \big(g, v_4\big)_\Gamma, \qquad && \forall v_4 \in V(\Omega_4), \end{aligned} \tag{7.65}$$

2. Solve the adjoint equations for $\lambda_1^k, \ldots, \lambda_4^k$ (in parallel):

$$\begin{aligned} a_1'(u_1;\, v_1, \lambda_1) &= \sum_{i=2,3} \big(u_{1|\Gamma_{1i}} - u_{2|\Gamma_{1i}}, v_{1|\Gamma_{1i}}\big)_{\Gamma_{1i}} + \big(u_{1|\Gamma_\Pi} - u_{i|\Gamma_\Pi}\big) v_{1|\Gamma_\Pi}, \\ a_2'(u_2;\, v_2, \lambda_2) &= \sum_{i=1,4} \big(u_{i|\Gamma_{i2}} - u_{2|\Gamma_{i2}}, -v_{2|\Gamma_{i2}}\big)_{\Gamma_{i2}} - \big(u_{i|\Gamma_\Pi} - u_{2|\Gamma_\Pi}\big) v_{2|\Gamma_\Pi}, \\ a_3'(u_3;\, v_3, \lambda_3) &= \sum_{i=1,4} \big(u_{i|\Gamma_{i3}} - u_{3|\Gamma_{i3}}, -v_{3|\Gamma_{i3}}\big)_{\Gamma_{i3}} - \big(u_{i|\Gamma_\Pi} - u_{3|\Gamma_\Pi}\big) v_{3|\Gamma_\Pi}, \\ a_4'(u_4;\, v_4, \lambda_4) &= \sum_{i=2,3} \big(u_{4|\Gamma_{4i}} - u_{i|\Gamma_{4i}}, v_{4|\Gamma_{4i}}\big)_{\Gamma_{4i}} + \big(u_{i|\Gamma_\Pi} - u_{4|\Gamma_\Pi}\big) v_{4|\Gamma_\Pi} \end{aligned} \tag{7.66}$$

3. Calculate the gradient:

$$\begin{aligned} \frac{dJ_\Gamma^k}{dg} \leftarrow & \sum_{i=2,3} P_{\Gamma_{1i}}\big(\lambda_{1|\Gamma_{1i}} - \lambda_{i|\Gamma_{1i}}\big) + \nu P_{\Gamma_\Pi}\big(\lambda_{1|\Gamma_\Pi} - \lambda_{i|\Gamma_\Pi}\big) \\ & + \sum_{i=2,3} P_{\Gamma_{4i}}\big(\lambda_{4|\Gamma_{i4}} - \lambda_{i|\Gamma_{i4}}\big)_{\Gamma_{i4}} + \nu P_{\Gamma_\Pi}\big(\lambda_{4|\Gamma_\Pi} - \lambda_{i|\Gamma_\Pi}\big) + \gamma g \end{aligned} \tag{7.67}$$

7.2.5 Experimental Studies

Although a theoretical approximation of the scalability characteristics of the proposed method has been discussed, the focus of the experimental studies was restricted to the feasibility for the 2×2 case and its influence of the parameters ν and γ.

The algorithm was run on the same input image, Fig. 7.2(b), as with the two-subdomains case, where the common boundary Γ here included the 64th row and 64th column of the discretized image plane. The discretization, the local solving method as well as all the parameter values, except the starting interval of the step size selection which was $[0, 10]$, have been the same as with the previous experiments. Results are depicted in Fig. 7.4.

Besides the influence of the control regularization strength γ, also the impact of the step size reduction ν at Γ_Π were studied. (See the diagrams in Fig. 7.4(a) and (b), respectively). Figure 7.4(a) shows that the convergence behavior for the 2×2 case has not changed significantly in comparison to the two-subdomains case, despite the fact that the rate has deteriorated to ≈ 0.93 after 50 iterations. Leaving out the control regularization has no significant impact on the L^2 error in comparison to setting γ to values small than 10^{-4}. Moreover, studies for the step size reduction factor ν revealed that only values less or equal to $\frac{1}{2}$ led to convergence, whereas greater values always led to divergence at Γ_Π.

Again, the result after 50 iterations in Fig. 7.4(c), as well as the per-pixel relative L^2 error shown in Fig. 7.4(d), are satisfactory for denoising purposes.

7.2.6 Complexity Considerations

As in the two-subdomains case, step (a) and (b) can be obviously carried out on different processing nodes, which holds also for the case of $M_x \times M_y$ subdomains $(M_x > 2, M_y > 2)$.

With respect to inter-process communication for the $M_x \times M_y$ case, note that only variables lying on the interface Γ have to be exchanged within the main loop. To be explicit, the control vector g^k has to be distributed by a central process, carrying out the steps (c)–(g), to the subdomain processes before step (a), which amounts to sending approximately $\frac{4\sqrt{N}}{\sqrt{M}}$ unknowns per subdomain, where $M := M_x \cdot M_y$ and N is the total number of unknowns, i.e. $M\frac{4\sqrt{N}}{\sqrt{M}} = 4\sqrt{MN}$ in total. Before step (b), again only variables on Γ have to be communicated in order to set-up the right-hand sides, but this time only mutually among processes which have adjacent subdomains; i.e. a distributed communication step can be applied, which amounts to interchanging only approx. $\frac{4\sqrt{N}}{\sqrt{M}}$ variables sequentially. Furthermore, before step (c), the local co-

states vectors λ_i^k, being restricted to their local interfaces Γ_i, are gathered by the central process, again resulting in a sequential communication volume of $4\sqrt{MN}$ variables. Finally, the total amount of bytes to be communicated sequentially is given by the expression:

$$V(M,k) = k\left(8\sqrt{MN} + 4\frac{\sqrt{N}}{\sqrt{M}}\right) v_v + V_c\,,$$

where k denotes the total number of iterations, v_v denotes the size of one variable in bytes and V_c the constant amount of bytes for the initial distribution of the input vector f as well as the final collection of the local solutions u_i. Obviously, as it is with standard non-overlapping domain decomposition methods, the communication volume scales only with the square root of the number of subdomains here, which is due to exchanging only the interface variables and indicates very good scalability properties.

With respect to the total computation time, we have the approximation:

$$T(M) = T_{NL}(N/M) + T_L(N/M) + t_{byte}V(M,k(M)) + T_c\,,$$

where $T_{NL}(n)$ and $T_L(n)$ denote the average computation time for solving the nonlinear and linear state systems, respectively, in parallel, T_c the computation time for the steps (c)–(g), which are independent from M, and with t_{byte} denoting the communication time per byte. Note that communication latency times as well as synchronization times have been neglected here. Furthermore, although the outer iteration number $k(M)$ naturally increases with the number of subdomains M, the processing time of the computationally demanding steps (a) and (b) is supposed to decrease strongly, depending on the complexity of the inner solving methods.

Finally, note that depending on the applied line search procedure, coarse-grained parallelization can be employed also there. In the case of nested iteration for example, the two necessary evaluations of J_Γ within each iteration can be carried out simultaneously.

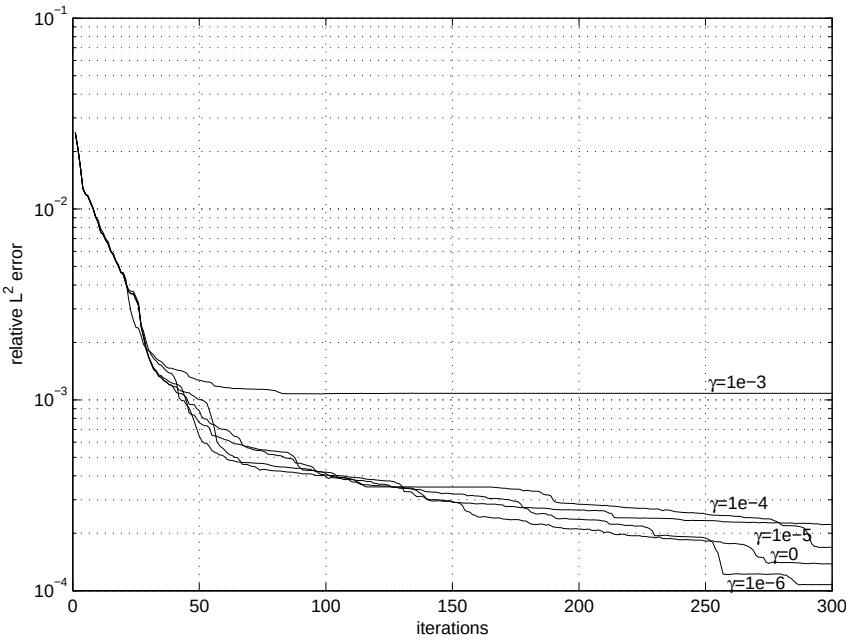
7.3 Conclusion

We investigated a new control-based approach for decomposing nonlinear PDE problems using the example of variational TV-denoising. Beginning with the constrained energy minimization problem on two-subdomain case, we gradually developed the involved local nonlinear equations — both in a generic and in an explicit formulation with respect to our model problem. After having addressed the issue involved with a direct solving of the nonlinear equation systems, we showed an alternative solving

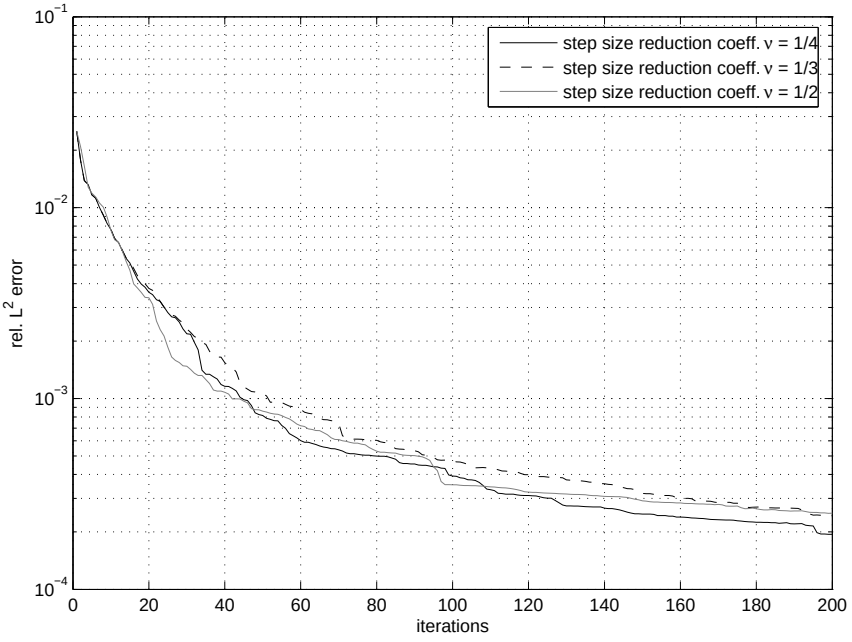
approach by means of a CG iteration. In particular, we explained in detail the two different ways to calculate the gradient of the constrained energy.

When addressing the many-subdomain case, we restrict our considerations to a rectangular partition of only 2×2 subdomains, since the special treatment involved with one corner point directly extends to an arbitrary number of corner points. Again, we show the two alternatives of calculating the gradient and note the involved calculation steps explicitly.

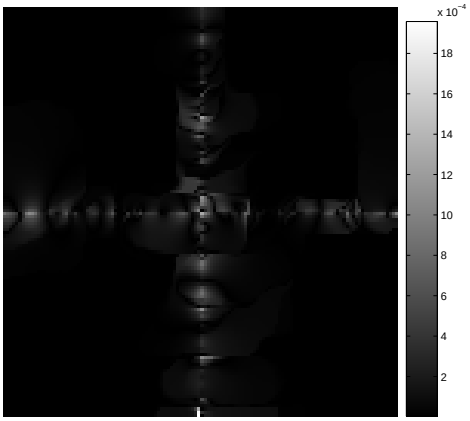
The experimental results for both partition cases suggest that the approach is practically feasible. However, despite an already employed step size selection heuristics, the convergence rate needs to be further improved, which can most likely be reached by the utilization of appropriate preconditioners, the subject of future work.



(a) Development of the L^2-error for different control regularization strengths γ



(b) Development of the L^2-error with step size reduction ($\gamma = 0$)



(c) Per-pixel L^2 error after 300 iterations ($\gamma = 10^{-5}$)

Figure 7.4: **Total and per-pixel L^2-error for a 2×2 decomposition** ($\alpha = 1.0, \beta = 10^{-6}$).

Chapter 8

A Convex Programming Approach to Nonlinear Domain Decomposition

Besides the control approach introduced in the previous chapter, in this chapter we present a different mathematical programming approach to decompose nonlinear problems. Interestingly, in [62] the authors give another derivation, instead of by the multi-domain formulation, of the non-overlapping Robin method [88] — a variant to the Neumann-Neumann method on two subdomains — by stating the decomposed problem as a constrained energy minimization being subsequently relaxed and solved by an augmented Lagrangian relaxation.

In the following we will apply this energy-based decomposition to non-quadratic convex problems, at the example of TV-denoising, as it was proposed in general in [15] for the two-subdomain case. As in the previous chapter, we first consider the two-subdomain case, and then detail on the special treatment of corner points, apparent in the case of many subdomains, at the example of a 2×2 partition.

Again starting with the two-subdomain case, we present the central idea of decomposing the nonlinear problem on the energy level. We proceed with a Lagrangian relaxation of the involved constrained minimization problem, and propose to employ a subgradient iteration to solve the resulting primal-dual problem. As with the control approach in the previous chapter, we use a generic formulation for the derivation of the algorithm and in the end give an explicit formulation for our model problem. Similarly for the case of many subdomains, we focus on the special treatment of one corner point in a 2×2 partition as an example to illustrate the case of many corner points. For both partition cases, experimental results for small images are given, showing the empirical feasibility (for those data sets) and providing an initial study

of the convergence behavior.

8.1 Primal-dual Domain Decomposition

8.1.1 The Approach

The approach is applied to energy minimization formulations of problem in terms of convex functionals:

$$\min_u J(u). \tag{8.1}$$

This encompasses a wide range of problems in image processing and computer vision. In the following, we focus on the nonlinear model problem (6.1).

Given a partition by two open subdomains, Ω_1, Ω_2 (see, e.g., Fig. 7.1(a)), we separate $J(\cdot)$ into two convex functionals: $J_{\Omega_1}(\cdot) := J_{|\Omega_1}(\cdot)$ and $J_{\Omega_2}(\cdot) := J_{|\Omega_2}(\cdot)$. Next, following [15], we rewrite (8.1) as

$$\min_{u_1,u_2} J_{\Omega_1}(u_1) + J_{\Omega_2}(u_2) \tag{8.2}$$

$$\text{subject to:} \quad u_{1|\Gamma} = u_{2|\Gamma}, \tag{8.3}$$

with $u_1 := u_{|\Omega_1} \in V(\Omega_1)$ and $u_2 := u_{|\Omega_2} \in V(\Omega_2)$. That is, by (8.2) we have split the original minimization (8.1) into the minimization of two independent energies while preserving the spatial couplings through (linear) equality constraints.

In order to obtain a solving algorithm for (8.2) which exhibits significant clues for coarse-grained parallelization, we, in a next step, relax the constraints in (8.2) by introducing a Lagrangian multiplier function λ, which leaves us with the primal-dual optimization problem

$$\sup_\lambda \underbrace{\min_{u_1,u_2} \left\{ J_{\Omega_1}(u_1) + J_{\Omega_2}(u_2) + \langle \lambda, u_{1|\Gamma} - u_{2|\Gamma} \rangle_\Gamma \right\}}_{=:J^*(\lambda)}, \tag{8.4}$$

with $\lambda \in L^2(\Gamma)$ being the Lagrangian multiplier function and J^* the dual to J. Interestingly, the two minimizations can be separated:

$$\sup_\lambda \left\{ \min_{u_1} \{ J_{\Omega_1}(u_1) + \langle \lambda, u_1 \rangle_\Gamma \} + \min_{u_2} \{ J_{\Omega_2}(u_2) - \langle \lambda, u_2 \rangle_\Gamma \} \right\}. \tag{8.5}$$

That is, the direct coupling between u_1 and u_2 can be replaced by an indirect coupling through the dual function λ. If we think of the dual λ as being fixed, the two minimization problems, which yield the local solutions $\hat{u}_1(\lambda)$ and $\hat{u}_2(\lambda)$ respectively, are independent from each other and can thus be solved in parallel.

8.1.2 Solving

In terms of solving (8.5) with respect to λ we chose the *subgradient iteration* method (See [8, 15]).

First, we consider

$$u_{1,\lambda} = \arg\min_{u_1}\{J_{\Omega_1}(u_1) - \langle\lambda, u_1\rangle_\Gamma\} \quad \text{and} \quad u_{2,\lambda} = \arg\min_{u_2}\{J_{\Omega_2}(u_2) - \langle\lambda, u_2\rangle_\Gamma\}\,, \tag{8.6}$$

and we define $u_\lambda := \{u_{1,\lambda}, u_{2,\lambda}\}$. Then, the subgradient method generates a sequence of dual feasible points according to the iteration

$$\lambda^{k+1} = \lambda^k + \tau_k\, g^k \tag{8.7}$$

with k denoting the iteration count, $\tau_k \in \mathbb{R}$ appropriate step sizes, and $g^k = g(u_{\lambda^k}) = \frac{dJ^*}{d\lambda}(\lambda^k)$ as subgradients of the subdifferential $\partial J^*(\lambda^k)$. Thereby, it is made use of the fact that

$$J^*(\tilde\lambda) \le J^*(\lambda) + (\tilde\lambda - \lambda)\, g(u_\lambda)\,, \qquad \forall\tilde\lambda \in L^2(\Gamma)\,. \tag{8.8}$$

In particular, we have that

$$g(u_\lambda) = u_{1,\lambda}|_\Gamma - u_{2,\lambda}|_\Gamma\,. \tag{8.9}$$

That is, within one iteration, firstly, u_{1,λ^k} and u_{2,λ^k} have to be determined, according to (8.6), which require independent minimizations and can therefore be done in parallel. Secondly, g^k is calculated according to (8.9), and thirdly, an appropriate step size τ^k is selected and the update is carried out following (8.7). See Algorithm 14 for a formalization.

Concerning the selection of step sizes τ_k, different methods exist (See, e.g., [97]). Typical choices are a constant step size: $\tau_k = \text{const}$, or exponentially decreasing step sizes: $\tau_{k+1} = \rho^k\tau_k,\ \rho \in (0,1),\ \tau_0 > 0$, both of which were used in the experiments to be presented below.

8.1.3 Application to the Model Problem

In a next step, we apply Algorithm 14 to the model problem of TV-based denoising, see Chapter 6. Thereby, the local energies $J_{\Omega_i}(u_i),\ i = 1, 2$ read

$$J_{\Omega_i} := \frac{1}{2}\int_{\Omega_i}(u-f)^2 + \alpha|\nabla u|_\beta\, d\mathbf{x}, \qquad i = 1, 2, \tag{8.11}$$

Algorithm 14: Nonlinear DD by convex programming on a two-subdomain partition

$k \leftarrow 0$. *Choose a* $\tau_0 > 0$ *and a* $\rho \in (0,1)$.

Choose a dual feasible λ^0.

do

(a) *Determine a primal solution:*

$$u_{1,\lambda^k} \leftarrow \arg\min_{u_1} \left\{ J_{\Omega_1}(u_1) + \langle \lambda^k, u_1 \rangle_\Gamma \right\}$$
$$u_{2,\lambda^k} \leftarrow \arg\min_{u_2} \left\{ J_{\Omega_2}(u_2) - \langle \lambda^k, u_2 \rangle_\Gamma \right\}$$

(b) *Calculate the subgradient:*

$$g^k \leftarrow u_{1,\lambda^k}|_\Gamma - u_{2,\lambda^k}|_\Gamma \, .$$

(c) *Update the dual solution:*

$$\lambda^{k+1} \leftarrow \lambda^k + \tau_k \, g^k$$

(d) *Calculate a new step size:*

$$\tau_{k+1} \leftarrow \rho^{k+1} \tau_k$$

(e) *Increment*: $k \leftarrow k+1$

until $\|u_{1,\lambda^k}|_\Gamma - u_{2,\lambda^k}|_\Gamma\|_{L^2} < \epsilon$

Merge the local solutions:

$$\begin{aligned} u_{|\Omega_1} &\leftarrow u_1^{k-1} \\ u_{|\Omega_2} &\leftarrow u_2^{k-1} \end{aligned} \tag{8.10}$$

with $f \in V(\Omega)$ denoting the input image, $\alpha \in \mathbb{R}$ the regularization strength, and $\beta \in \mathbb{R}$ the regularization parameter of the TV norm, and we have $K = I$.

Including the Lagrange multipliers, minima of the subproblems in the problem (8.5) are given by solving the first their variations being equal to zero, which

yields:

$$\int_{\Omega_i} (u_i - f_i)\ \tilde{u}_i + \alpha \left(\frac{\nabla u_i}{|\nabla u_i|_\beta} \right) \nabla \tilde{u}_i \ d\mathbf{x} \ \pm \int_\Gamma \lambda \tilde{u}_i \ d\mathbf{x} \ = \ 0 \qquad \forall \tilde{u} \in H_0^1,\ i = 1, 2, \tag{8.12}$$

or, in PDE-formulation:

$$(u_i - f_i) + \alpha \nabla \cdot \left(\frac{\nabla u_i}{|\nabla u_i|_\beta} \right) \pm R_{\Gamma\Omega_i} \lambda = 0, \qquad i = 1, 2, \tag{8.13}$$

where $R_{\Gamma\Omega_i}$ denotes an extension by zero from Γ to $\Omega_i,\ i = 1, 2$. In comparison with the formulations in Chapter 6, the additional Lagrangian multipliers λ only amount to replacing each occurrence of the original first variation, which is the left-hand side of (6.7), by the left-hand side of (8.12), or, respectively, every occurrence of the left-hand side of (6.10), by the left-hand side of (8.13), in PDE-formulation. Finally, when using the Primal-Dual Newton's method $\nabla u / |\nabla u|_\beta$ need to be replaced by w, see Section 6.2.4.

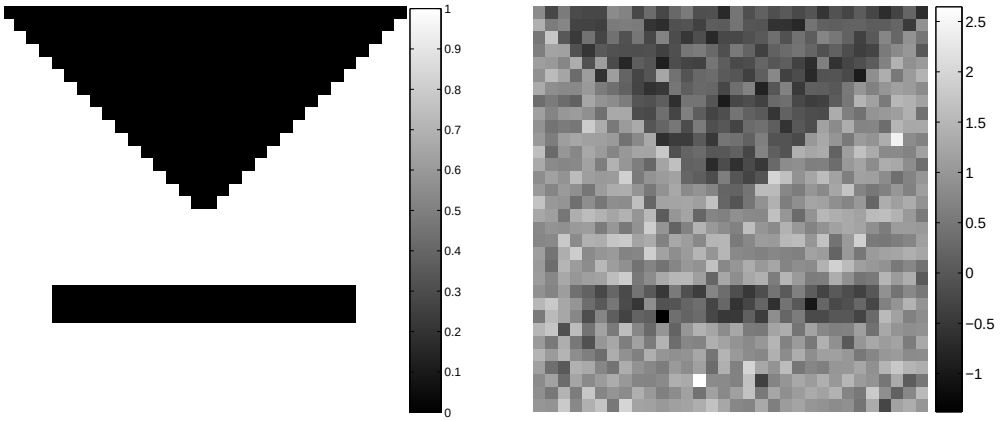
8.1.4 Experimental Studies

Numerical experiments have been made on a 32×32 grid, which was partitioned into two subdomains of size 17×32 and 16×32 (height $\times$ width). Newton's Method with continuation ($\alpha = 0.5,\ \beta = 10^{-6},\ \beta_0 = 10,\ n_\beta = 15,\ e_{L^2} = 10^{-12}$), in connection with PCG iteration with a maximum residual L^2-error of 10^{-3} has been employed as local solver. Results and input data are depicted in Figures 8.2[1] and 8.1, respectively.

The convergence behavior w.r.t. to the relative L^2-error ($\|u^k - u_{ref}\|_2 / \|u_{ref}\|_2$) is depicted in Fig. 8.3, while u_{ref} is a reference solution generated by a sequential run of the local solver with a final residual error of 10^{-12}. Several constant step sizes have been tried, as well as some exponentially diminishing step sizes $\tau^{k+1} = \rho_\tau \tau^k,\ 0 < \rho_\tau < 1$.

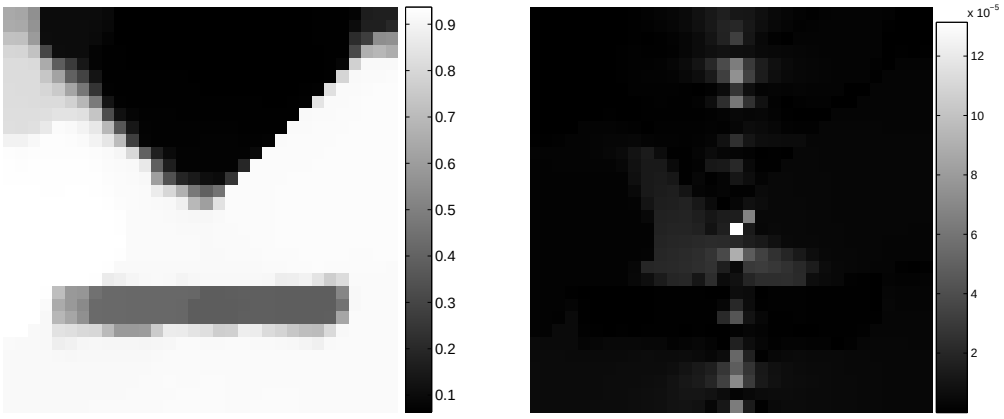
The per-pixel error depicted in Figure 8.2 shows that the remaining error is mainly located at the interface Γ, which is to be expected, since the natural spatial couplings are broken up right there and are relaxed through the Lagrangian functions. Furthermore, the diagram in Fig. 8.3 clearly points out an exponential convergence rate for constant step sizes, while for this small data set, exponentially decreasing step sizes do not help to improve the convergence rate, but instead are most likely to prevent further convergence below a relative error of less than 10^{-4}.

[1] Note that asymmetric artifacts especially occurring in the upper corners of Fig. 8.2(a) are due to the asymmetry of the finite element basis functions used in the discretization.



(a) Ground truth (b) With added Gaussian noise

Figure 8.1: **Ground truth and input image with added Gaussian noise** (32×32 pixels, signal-to-noise ratio 1.4 dB).



(a) Result after 300 subgradient iterations (b) Per-pixel L^2-error to reference solution

Figure 8.2: **Result and per-pixel L^2-error for a two-subdomain decomposition[1].**

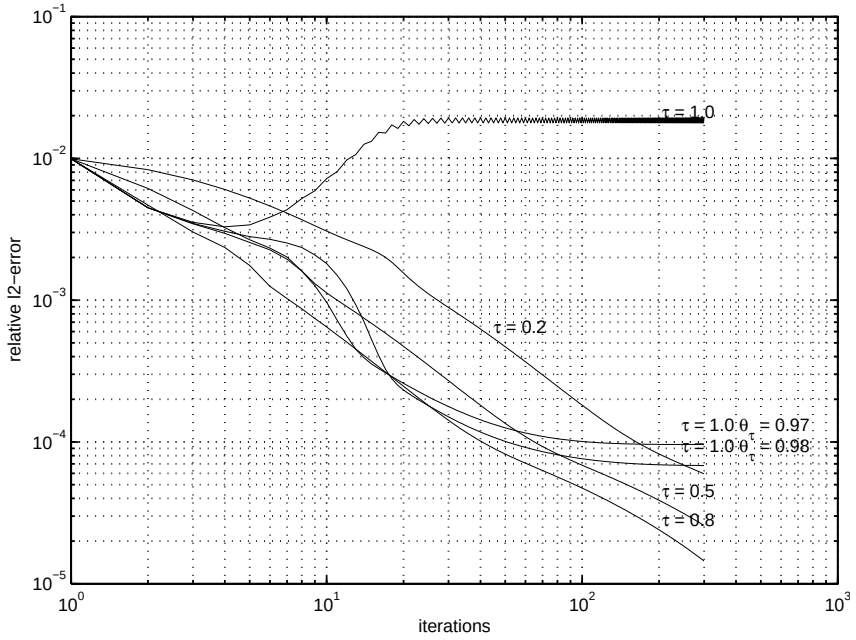


Figure 8.3: **Development of the L^2-error for different step sizes in the two-subdomain case.** It is plotted the relative L^2-error w.r.t. the reference solution for either fixed step sizes $\tau \in \{0.2, 0.5, 0.8, 1.0\}$, or exponentially decreasing step sizes $\tau = \rho_\tau^k,\ \rho_\tau \in \{0.97, 0.98\},\ k$: iteration count, respectively.

8.2 The Case of Many Subdomains

Extending the proposed decomposition approach to more than two subdomains brings up the need for special equality constraints at inner corner points, i.e. points on Γ which are shared by more than two subdomains. Since these special conditions are the same (up to a multiplication by -1) for all other inner corner points, we, as in the previous chapter, restrict our considerations to the case of 2×2 partition with only one corner point Π, Figure 7.1(b) for an example and a mathematical definition.

Then, a straightforward extension of (8.2) in this setting would be

$$\min_{u_1,\ldots,u_4} \sum_{i=1}^{4} J_{\Omega_i}(u_i) \tag{8.14}$$

$$\begin{aligned} \text{subject to:} \quad & u_{1|\Gamma_{12}\cup\Pi} = u_{2|\Gamma_{12}\cup\Pi} \\ & u_{1|\Gamma_{13}\cup\Pi} = u_{3|\Gamma_{13}\cup\Pi} \\ & u_{2|\Gamma_{24}\cup\Pi} = u_{4|\Gamma_{24}\cup\Pi} \\ & u_{3|\Gamma_{34}\cup\Pi} = u_{4|\Gamma_{34}\cup\Pi} \,. \end{aligned} \tag{8.15}$$

Although mathematically correct, numerical experiments based Algorithms 14 extended to (8.14) revealed a highly insufficient convergence behavior in terms of persistent oscillations, even for relatively small step sizes, while the error mainly did concentrate at Π.

Therefore, we propose to have separate constraints for the unknowns on Π, such that

$$\min_{u_1,\ldots,u_4} \sum_{i=1}^{4} J_{\Omega_i}(u_i) \tag{8.16}$$

$$\begin{aligned} \text{subject to:} \quad & u_{1|\Gamma_{12}} - u_{2|\Gamma_{12}} = 0 \\ & u_{1|\Gamma_{13}} - u_{3|\Gamma_{12}} = 0 \\ & u_{2|\Gamma_{24}} - u_{4|\Gamma_{24}} = 0 \\ & u_{3|\Gamma_{34}} - u_{4|\Gamma_{34}} = 0 \\ & u_{1|\Pi} - u_{2|\Pi} = 0 \\ & u_{1|\Pi} - u_{3|\Pi} = 0 \\ & u_{1|\Pi} - u_{4|\Pi} = 0 \,. \end{aligned}$$

After applying Lagrangian relaxation as in the two-subdomain case, the correspond-

ing primal-dual optimization reads:

$$\sup_{\substack{\lambda_{12},\lambda_{13}\\ \lambda_{24},\lambda_{34}\\ \lambda_{\Pi,1},\ldots,\lambda_{\Pi,3}}} \min_{u_1,\ldots,u_4} \sum_{i=1}^{4} J_{\Omega_i}(u_i) + \sum_{i=1}^{3} \sum_{j\in N_>(i)} \langle \lambda_{ij}, u_{i|\Gamma_{ij}} - u_{j|\Gamma_{ij}} \rangle_{\Gamma_{ij}} + \sum_{i=1}^{3} \lambda_{\Pi,i}(u_{1|\Pi} - u_{i|\Pi}) \,. \tag{8.17}$$

Subsequent splitting into local minimization problem yields

$$\begin{aligned} \sup_{\substack{\lambda_{12},\lambda_{13}\\ \lambda_{24},\lambda_{34}\\ \lambda_{\Pi,1},\ldots,\lambda_{\Pi,3}}} \Big\{ & \min_{u_1} \{ J_{\Omega_1}(u_1) + \langle \lambda_{12}, u_1 \rangle_{\Gamma_{12}} + \langle \lambda_{13}, u_1 \rangle_{\Gamma_{13}} + (\lambda_{\Pi,1} + \lambda_{\Pi,2} + \lambda_{\Pi,3})\, u_{1|\Pi} \} \\ & + \min_{u_2} \left\{ J_{\Omega_2}(u_2) - \langle \lambda_{12}, u_1 \rangle_{\Gamma_{12}} + \langle \lambda_{13}, u_1 \rangle_{\Gamma_{13}} - \lambda_{\Pi,1} u_{2|\Pi} \right\} \\ & + \min_{u_3} \left\{ J_{\Omega_3}(u_3) - \langle \lambda_{13}, u_3 \rangle_{\Gamma_{13}} + \langle \lambda_{34}, u_3 \rangle_{\Gamma_{34}} - \lambda_{\Pi,2} u_{3|\Pi} \right\} \\ & + \min_{u_4} \left\{ J_{\Omega_4}(u_4) - \langle \lambda_{24}, u_4 \rangle_{\Gamma_{24}} - \langle \lambda_{34}, u_4 \rangle_{\Gamma_{34}} - \lambda_{\Pi,3} u_{4|\Pi} \right\} \Big\}. \end{aligned} \tag{8.18}$$

Finally, by applying the subgradient method in an analogous manner as above, we obtain Algorithm 15.

8.2.1 Experimental Results

Experiments were conducted on the same grid and input data as in the two-subdomain case, with the difference that the interface Γ did divide the discrete image plane at the 17. column and 17. row, respectively. Again, Newton's method served as inner solving method, using the same parameters as in the previous experiments.

Figure 8.4(b) shows that the error is distributed mainly equally along Γ and does not concentrate at Π as encountered with the non-special constraints on Π. In addition, while conducting the experiments, it has been found to improve convergence at Π if the step size for u_Π is one fourth the step size for the other unknowns.

The error plots in Fig. 8.4 reveal an almost similar convergence rate for step sizes of 0.8 and 0.5 as in the two-subdomain case, although the number of subdomains has doubled. In addition, according to the error plots in Fig. 8.6 the regularization strength α, controlling the strength of the spatial couplings, influences the convergence rate only moderately. Only for $\alpha = 0.2$ the step size had to be reduced in comparison to the other cases in order to prevent divergence at Π.

Algorithm 15: Nonlinear DD by convex programming on a 2×2 partition

$k \leftarrow 0$. *Choose a* $\tau_0 > 0$, *and a* $\rho \in (0,1)$.

Choose a dual feasible λ_{ij}^0 *and* $\lambda_{\Pi,i}^0 = 0$, *e.g.* 0.

$I \leftarrow \{(i,j) \mid 1 \le i \le 4,\ j \in N_>(i)\}$

do

(a) *Solve the local minimization problems (in parallel):*

$$u_{1,\lambda^k} \leftarrow \arg\min_{u_1} \big\{ J_{\Omega_1}(u_1) + \langle \lambda_{12}^k, u_{1|\Gamma_{12}} \rangle_{\Gamma_{12}} + \langle \lambda_{13}^k, u_{1|\Gamma_{13}} \rangle_{\Gamma_{13}} + (\lambda_{\Pi,1}^k + \lambda_{\Pi,2}^k + \lambda_{\Pi,3}^k)\, u_{1|\Pi} \big\}$$

$$u_{2,\lambda^k} \leftarrow \arg\min_{u_2} \Big\{ J_{\Omega_2}(u_2) - \langle \lambda_{12}^k, u_{2|\Gamma_{12}} \rangle_{\Gamma_{12}} - \langle \lambda_{24}^k, u_{2|\Gamma_{24}} \rangle_{\Gamma_{24}} - \lambda_{\Pi,1}^k u_{2|\Pi} \Big\}$$

$$u_{3,\lambda^k} \leftarrow \arg\min_{u_3} \Big\{ J_{\Omega_3}(u_3) + \langle \lambda_{13}^k, u_{3|\Gamma_{13}} \rangle_{\Gamma_{13}} + \langle \lambda_{34}^k, u_{3|\Gamma_{34}} \rangle_{\Gamma_{34}} - \lambda_{\Pi,2}^k u_{3|\Pi} \Big\}$$

$$u_{4,\lambda^k} \leftarrow \arg\min_{u_4} \Big\{ J_{\Omega_4}(u_4) - \langle \lambda_{24}^k, u_{4|\Gamma_{24}} \rangle_{\Gamma_{24}} - \langle \lambda_{34}^k, u_{4|\Gamma_{34}} \rangle_{\Gamma_{34}} - \lambda_{\Pi,3}^k u_{4|\Pi} \Big\}$$

(b) *Update the dual functions (sequentially):*

foreach $(i,j) \in I$ **do** $\lambda_{ij}^{k+1} \leftarrow \lambda_{ij}^k + \tau_k \big(u_{i,\lambda^k}|_{\Gamma_{ij}} - u_{j,\lambda^k}|_{\Gamma_{ij}} \big)$

for $i = 1, \ldots, 3$ **do** $\lambda_{\Pi,i}^{k+1} \leftarrow \lambda_{\Pi,i}^k + \tau_k \big(u_{1,\lambda^k}|_{\Pi} - u_{i+1,\lambda^k}|_{\Pi} \big)$

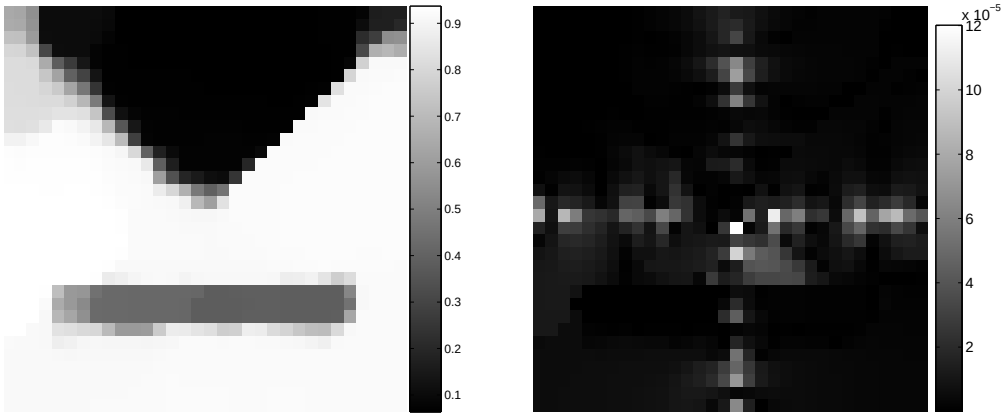
(c) *Calculate a new step size:*

$$\tau_{k+1} \leftarrow \rho^{k+1} \tau_k$$

(d) *Increment:* $k \leftarrow k+1$

until $\left(\sum_{(i,j)\in I} \|u_{i|\Gamma} - u_{j|\Gamma}\|_{L^2} + \sum_{i=1}^{3} \|u_{1|\Pi} - u_{i+1|\Pi}\|_{L^2} \right) < \epsilon$

for $i = 1, \ldots, 4$ **do** $u_{|\Omega_i} \leftarrow u_i$



(a) Result after 300 subgradient iterations (b) Per-pixel L^2-error to reference solution

Figure 8.4: **Result and per-pixel L^2-error for a 2×2 decomposition.**

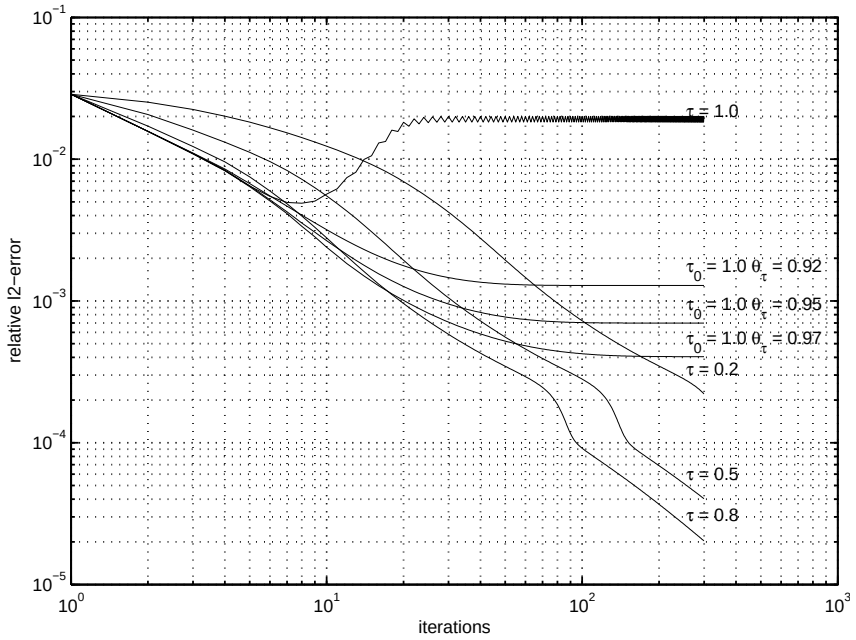


Figure 8.5: **Development of the L^2-error for different step sizes on a 2×2 decomposition.** It is plotted the relative L^2-error w.r.t. the reference solution for either fixed step sizes $\tau \in \{0.2, 0.5, 0.8, 1.0\}$ and exponentially decreasing step sizes $\tau = \rho_\tau^k,\ \rho_\tau \in \{0.92, 0.95, 0.97\},\ k$: iteration count, respectively.

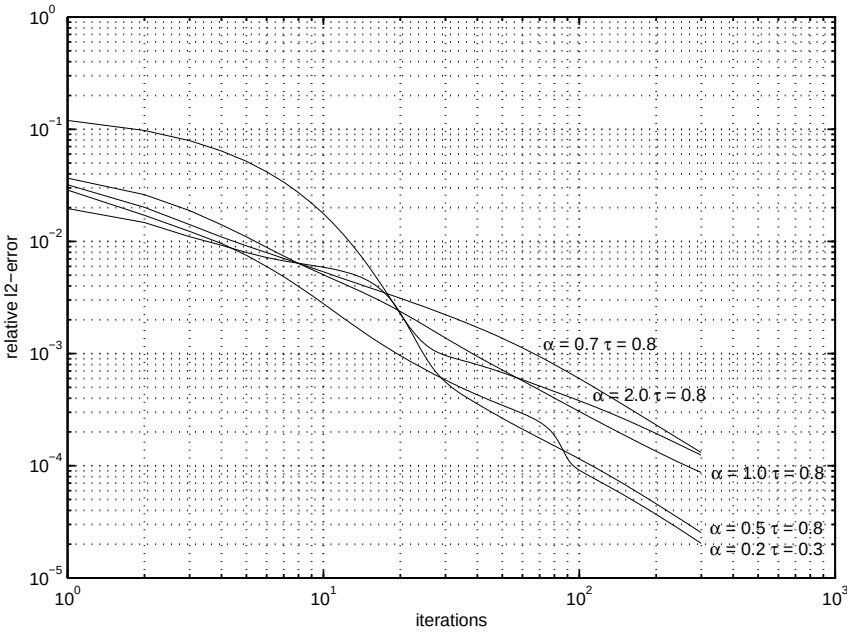


Figure 8.6: **Development of the L^2-error for different regularization strengths on a 2×2 decomposition.** It is plotted the relative L^2-error for different regularization strengths $\alpha \in \{0.2, 0.5, 0.7, 1.0, 2.0\}$.

8.2.2 Comparison to Control-based Decomposition

In comparison to the solving algorithms (both in the two- and the multi-subdomain case) of the control approach in Chapter 7, it is obvious the computational effort per iteration to be less here, since with the control-based algorithms, two (linear or nonlinear) partial differential equations are to be solved per subdomain, whereas with the convex decomposition-based methods it is only one minimization problem per iteration and subdomain. Furthermore, as experiments are suggesting, the convex methods seem to be less sensitive to the step size selection, thus allowing simpler and much less computationally expensive selection rules.

From a theoretical perspective, the parallelization approach here is founded on mathematical programming. By contrast, the control-theoretic decomposition approach, as well as all the linear domain decomposition approaches in Chapter 3 and 4, are based on a splitting of partial differential equations. As mentioned in the introduction of this chapter, in terms of the classical domain decomposition approaches, a link between the mathematical programming and PDE-based formulations can be found in [62].

8.3 Conclusion

In this chapter we proposed a new convex programming approach for the decomposition of nonlinear energy minimization problems, using the example of TV-based denoising. Starting with the two-subdomain case, we split the original minimization problem into separate minimizations of local energies while preserving the global couplings through Lagrangian multiplier functions. Thus, the main computational burden can be distributed to several processing nodes at the cost of an outer sequential iteration, where the Lagrangians updated. Analogue to the previous chapter, we restrict our considerations to only one corner point when addressing the case of arbitrary many subdomains. For both cases, we propose solving by subgradient iterations and give concrete formulations of the resulting computational steps in a generic manner.

As in the previous chapter, experimental results show the practical feasibility of the approach on small data sets. In comparison to the control approach. However, the involved systems of nonlinear equations are less complex and require a smaller computational effort. In addition, the few experimental results show a better and more constant convergence rate.

Chapter 9

Conclusion

9.1 Summary

The main subject of this work is to provide methods for data parallelization through the domain decomposition of linear and nonlinear PDE problems in image processing. Whereas the first three chapters deal with linear domain decomposition methods at the example of a well-known variational motion estimation problem, the last three chapters focus on nonlinear DD techniques based on the model problem of variational image denoising employing TV-regularization. Besides the main subject, in Chapter 5 we propose an extension of the linear model problem in order to improve optic flow reconstructions in the context of computational fluid mechanics.

9.1.1 Linear Domain Decomposition

9.1.1.1 Non-overlapping Methods

After giving a short introduction into variational motion estimation and definition of the model problem, the second chapter starts by elaborating the theoretical basis for non-overlapping DD techniques, first by introducing the Steklov-Poincaré operator and following the Schur complement equation, which gives the basis of all primal substructuring methods. Based on the Schur complement equation, we then develop the first one-level algorithms based on two subdomains and show their equivalence to preconditioning techniques with a Richardson iteration. Subsequently, in the multiple-subdomain case, we first explain the typical algorithmic framework of iterative substructuring methods, namely that of a Krylov-subspace solving applied to the Schur complement equation by means of a parallelized solving step and various different types of parallel preconditioning techniques. We then focus on common one-level preconditioners and address their impact on the condition num-

ber of the preconditioned system. In addition, we address and explain the problem of limited information propagation and show how it is overcome by different two-level preconditioning approaches. Thereby, we put a strong focus on the Balancing Neumann-Neumann preconditioner, because of its prominent role as a current primal substructuring method, and by elaborating also on the algebraic foundation and its application to our model problem. The remaining sections of the theoretical part are dedicated to dual and primal-dual substructuring methods, notably the solely dual FETI and the primal-dual FETI method. In either case, we elucidate the mathematical background by drawing a line from the constrained optimization problem involving *local* Schur complement equations, over its Lagrangian relaxation, up to the dual and primal-dual problem, respectively, solved by an extended PCG iteration. In particular, with the dual approach, we address the intrinsic two-level character of the involved projection, compare it to its primal counterpart in Balancing NN preconditioning, and show the necessary modifications in order to apply the method to our model problem.

In the experimental section we study different numerical aspects by the example of two well-known primal preconditioners, Neumann-Neumann and Balancing Neumann-Neumann, in connection with our model problem using a multi-grid Gauss-Seidel local solver. Thereby, the following empirical findings are made:

- Preconditioning the Schur complement system has a strong impact on the convergence rate. Specifically, theoretical upper limits for either the NN or BNN preconditioner hold. While convergence deteriorates with increasing number of subdomains for NN preconditioning in case the spatial couplings are dominant, it remains nearly constant for BNN preconditioning.

- With respect to the final relative residual error, that of the outer Schur complement solver is about one order of magnitude larger than that of the local subdomains solvers.

- Both the NN and BNN preconditioning method provide a very good *relative* scalability on a relatively cheap PC-cluster even for larger numbers of subdomains. However, the BNN preconditioner cannot compensate for its three times higher computational effort, if spatial couplings are not dominant over larger image regions. Hence, for model problems with primarily local spatial couplings, NN preconditioning is probably preferable over BNN preconditioning.

- In comparison to an highly-tuned multi-grid implementation, only little speed gains can be reached for the NN method at very high costs of 25 processing nodes or more.

- In accordance with theory, the communication volume increases only with the square root of the number of subdomains.

9.1.1.2 Overlapping Methods

Starting with the very first DD method by H. Schwarz we first present the basic one-level algorithms on two and on multiple subdomains, respectively, and identify each of them with additive and multiplicative, respectively, preconditioners. In particular we show that Gauss-Seidel and Jacobi iteration are special cases within this framework. With same numerical argument as with one-level non-overlapping methods — limited spatial information propagation — we move on to two-level and multilevel methods presenting the different combinations of additive (sequential) or multiplicative (parallel) algorithms with respect to either the order of level updates or the order of subdomain updates within each level, respectively. Thereby, we always address the aspect of scalability in terms of parallel computing. Finally, we show that standard multigrid methods appear as special cases in the multilevel framework and, in particular, that level-multiplicative methods can be seen as w-cycle multigrid algorithms while employing Schwarz smoothers.

9.1.2 Nonlinear Domain Decomposition

Instead of considering the extensions of classical non-overlapping DD methods for their application to nonlinear problems, we pursue two distinct alternative approaches to decompose nonlinear PDE problems. By giving a thorough introduction into TV-based image denoising, including all main algorithms as well as several experiments, we chose a prototypical nonlinear model problem.

9.1.2.1 Control Approach

Based on the work of Lee, Gunzburger et al. [86, 65, 67, 66], we give a step-by-step presentation of the control-theoretic approach for the denoising model problem. Both for the two and the multiple-subdomain case, we first formulate the optimal control problem as constrained energy minimization, proceed with its Lagrangian relaxation and derive the corresponding optimality system for our model problem, respectively. Subsequently, we explain the two different iterative solving approaches in general, and show their formulation for the denoising problem. In the experimental sections we show the empirical feasibility on two and 2×2 subdomains and provide explicit formulations for the communication volume and the total run-time in dependence on the number of unknowns and the number of subdomains.

9.1.2.2 Convex Programming Approach

With the convex programming approach we provide another method for a nonlinear non-overlapping decomposition. Given that the model problem can be written as energy minimization, we propose a decomposition on the energy level, while enforcing the inter-subdomain couplings by equality constraints. Similar to the control approach, the constraints are relaxed by the Lagrangian multiplier method resulting in a primal-dual problem, which is then solved by subgradient iteration. Here too, numerical experiments on two and 2×2 subdomains based on the nonlinear denoising problem show the empirical feasibility of the approach.

9.1.2.3 Motion Estimation with High-order Regularization

As an off-topic, we investigate the improvement of the linear model problem, variational motion estimation, with respect to its application to fluid mechanical problems. In particular, after introducing the notion of potential fields as well as regularizations based on high-order derivatives, we present a novel approach for the *direct* estimation of the potential fields from a given image sequence. As with the indirect, three-step approach by Corpetti et al. [39, 40], we too decrease the degree of derivatives by their relaxation through auxiliary functions. Thereby, the maximum order of derivatives in the corresponding partial differential equations can be lowered from six to four. Furthermore, we embed the approach into a multi-resolution framework in order to make larger flow speeds computationally feasible. In the succeeding experimental section, we provide qualitative and quantitative comparisons of our approach to the indirect method as well as the well-known Horn and Schunck motion estimation for several synthetic data sets, which shows the superiority of the new method.

9.2 Future Work

Although answering of the scientific questions posed in the beginning of this work were pursued as posed, new questions or issues appeared in the course of this study, which are to be subject of future investigations.

- Chapter 3:
  - Adaption of the multigrid scheme, i.e. the discretization on lower resolutions as well as the restriction and prolongation operators, for solving the local Dirichlet problems, or investigation of alternative local solvers in connection with NN and BNN preconditioning, in order to increase the total convergence rate of the parallel algorithm.

  - Experimental studies of the dual and the primal-dual FETI method applied to optical flow estimation on a parallel computer, and comparison to the quantitative and qualitative results for NN and BNN preconditioning.

- Chapter 5:

  - Meanwhile, after completion of this thesis and spurred by both positive and negative (cf. Section 5.3.2) results, and in-depth study of the approach (5.8) together with higher-order regularization (5.14) has been conducted in our group [129, 130]. An independent benchmark evaluation as part of work package 5 of the european FLUID project (http://fluid.irisa.fr/) has shown recently that by adding a boundary penalizer to (5.14), and by using a more sophisticated discretization scheme, a well-posed and stable variational approach can be implemented that not only makes the use of auxiliary variables obsolete but also computed significantly more accurate fluid flow estimates than the approach by Corpetti et al. [38].
  - Decomposition of the energies (5.18) and (5.19) by applying the approach of Chapter 8, in order to parallelize the computation of the higher-order regularization problems.

- Chapter 8:

  - Theoretical proof of equivalence of the problem decomposition on the energy level and the original minimization problem.
  - Derivation of the energy-based decomposition from the Steklov-Poincaré equation.
  - Further experimental studies using more advanced solvers, such as the cutting plane method, on larger data sets.

Bibliography

[1] R. Acar and C.R. Vogel. Analysis of total variational penalty methods for ill-posed problems. *Inverse Problems*, 10:1217–1229, 1994.

[2] R. Adrian. Particle imaging techniques for experimental fluid mechanics. *Annal Rev. Fluid Mech.*, 23:261–304, 1991.

[3] A. Amini. A scalar function formulation for optical flow. In *European Conference on Computer Vision, ECCV '94*, pages 125–131, 1994.

[4] K. Atkinson and W. Han. *Theoretical Numerical Analysis*. Springer, Heidelberg, Germany, 2001.

[5] J.P. Aubin. *Approximation of Elliptic Boundary-Value Problem*. John Wiley & Sons, New York, NY, 1972.

[6] L. Bannehr, R. Rohn, and G. Warnecke. A functionnal analytic method to derive displacement vector fields from satellite image sequences. *Int. Journ. of Remote Sensing*, 17(2):383–392, 1996.

[7] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Perfomance of optical flow techniques. *Int. Journal of Computer Vision*, 1(12):43–77, Feb. 1994.

[8] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 2 edition, 1999.

[9] P. Bjørstad and A. Hvidsten. Iterative methods for substructured elasticity problems in structural analysis. In R. Glowinski, G. Golub, G. Meurant, and J. Périaux, editors, *Domain Decomposition Methods for Partial Diff. Eqs.*, pages 301–312, Philadelphia, PA, 1988. SIAM.

[10] P. Bjørstad and M. Skogen. Domain decomposition algorithms of schwarz type, designed for massively parallel computers. In D. Keyes, T.F. Chan, and G. Meurant, editors, *Fifth. Int. Symp. on Domain Decomposition Meth. for Part. Diff. Eqs.*, pages 362–375, Philadelphia, PA, 1992.

[11] P. Bjørstad and O.B. Widlund. *Elliptic Problem Solvers II.* Academic Press, New York, NY, 1984.

[12] P. Bjørstad and O.B. Widlund. Iterative methods for the solution of elliptic problems on regions partitioned into substructures. *SIAM J. of Numer. Anal.*, 23(6):1097–1120, 1986.

[13] P. Blomgren, T. Chan, P. Mulet, and C.K. Wong. Total Variation image restoration: numerical methods and extensions. In *IEEE Proc. Int. Conf. Image Proc.*, volume 3, pages 384–387. IEEE, 1997.

[14] J.-F. Bourgat, Glowinski R., P. Le Tallec, and Vidrascu M. Variational formulation and algorithm for trace operator in domain decomposition calculations. In T. Chan, Glowinski R., J. Périaux, and O. Widlund, editors, *Second Int. Symp. on Domain Decomposition Meth.*, pages 3–16, Philadelphia, PA, 1989. SIAM.

[15] S. Boyd, L. Xiao, and Almir Mutapcic. Notes on decomposition methods, 2003. Notes for course EE3920, http://www.stanford.edu/class/ee392o/#lectures.

[16] J. Bramble, J. Pasciak, and A. Schatz. The construction of preconditioners for elliptic problems by substructuring I. *Math. Comp.*, 47:103–134, 1986.

[17] J. Bramble, J. Pasciak, and A. Schatz. An iterative method for elliptic problems on region partitioned into substructures. *Math. Comp.*, 46:361–369, 1986.

[18] J. Bramble, J. Pasciak, and A. Schatz. The construction of preconditioners for elliptic problems by substructuring IV. *Math. Comp.*, 53(187):1–24, 1989.

[19] A. Bruhn. *Variational Optic Flow Computation: Accurate Modelling and Efficient Numerics.* PhD thesis, Department of Mathematics and Computer Science, Saarland University, Saarbrücken, Germany, 2006.

[20] A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, and C. Schnörr. Real-time optic flow computation with variational methods. In N. Petkov and M. Westenberg, editors, *Proc. Computer Analysis of Images and Patterns (CAIP'03)*, volume 2756 of *LNCS*, pages 222–229. Springer, 2003.

[21] A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, and C. Schnörr. Variational optical flow computation in real-time. *IEEE Trans. Image Processing*, 14(5):608–615, 2005.

[22] A. Bruhn, J. Weickert, T. Kohlberger, and C. Schnörr. A multigrid platform for real-time motion computation with discontinuity-preserving variational methods. *Int. Journal of Computer Vision*, 70(3):257–277, 2006.

[23] A. Bruhn, J. Weickert, and C. Schnörr. Combining the advantages of local and global optic flow methods. In L. van Gool, editor, *Pattern Recognition, Proc. 24th DAGM Symposium*, volume 2449 of *LNCS*, pages 454–462, Zürich, Switzerland, 2002. Springer.

[24] A. Bruhn, J. Weickert, and C. Schnörr. Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *Int. Journal of Computer Vision*, 61(3):211–231, 2004.

[25] X.C. Cai and M. Dryja. Domain decomposition methods for monotone nonlinear elliptic problems. In D. Keyes and J. Xu, editors, *Domain Decomposition Methods in Scientific and Engineering Computing*, pages 335–360, Providence, RI, 1994. AMS.

[26] A. Chambolle. An algorithm for total variation minimization and applications. *J. of Math. Imag. Vision*, 20:89–97, 2004.

[27] A. Chambolle and P.-L. Lions. Image recovery via total variation minimization and related problems. *Numer. Math.*, 76(2):167–188, 1997.

[28] T. Chan, R. Glowinski, J. Périaux, and O.B. Widlund eds. *Third Int. Symposium on Domain Decomposition Methods for Partial Differential Equations*. SIAM, Philadelphia, PA, 1990.

[29] T. Chan, G.H. Golub, and P. Mulet. A nonlinear primal-dual method for Total Variation-based image restoration. In M. Berger, R. Deriche, I. Herlin, J. Jaffre, and J. Morel, editors, *ICAOS'96, 12th Int. Conf. on Analysis and Optimization of sytems: Images, wavelets, and PDEs*, volume 219 of *Lecture Notes in Control and Information Sciences*, pages 241–252, Heidelberg, Germany, 1996. Springer.

[30] T. Chan, G.H. Golub, and P. Mulet. A nonlinear primal-dual method for tv-based image restoration. *SIAM J. of Scientifc Computing*, 20(6):1964–1977, 1999.

[31] T. Chan and D. Keyes. Interface preconditioning for domain-decomposed convection-diffusion operators. In T. Chan, R. Glowinski, J. P'eriaux, and O. Widlund, editors, *Third Int. Symp. on Domain Decomposition Meth. f. Part. Diff. Eqs.*, pages 245–262, Philadelphia, PA, 1990. SIAM.

[32] T. Chan and T. Mathew. An application of the probing technique to the vertex space method in domain decomposition. In R. Glowinski, Y. Kuznetsov G. Meurant, and O. Widlund, editors, *Fourth Int. Symp. on Domain Decomposition Meth. f. Part. Diff. Eqs.*, pages 101–111, Philadelphia, PA, 1991. SIAM.

[33] T. Chan and T. Mathew. Efficient variants of the vertex space domain decomposition algorithms. *SIAM J. of Scientifc Computing*, 15(6):1349–1374, 1994.

[34] T. Chan and T.P. Mathew. Domain decomposition algorithms. In *Acta Numerica 1994*, pages 61–143. Cambridge University Press, 1994.

[35] T. Chan and P. Mulet. Iterative methods for total variation image restoration. Technical Report 96-38, Dept. of Mathematics, Univ. of California at Los Angeles, Los Angeles, CA, 1996.

[36] T. Chan, H. Zhou, and R. Chan. A continuation method for total variation denoising problems. In F.T. Luk, editor, *Proc. of the SPIE Conference on Advanced Signal Processing Algorithms*, pages 314–325. SPIE, 1995.

[37] P.G. Ciarlet. *The Finite Element Method for Elliptic Problem*. North-Holland, Amsterdam, The Netherlands, 1978.

[38] T. Corpetti, D. Heitz, G. Arroyo, E. Mémin, and A. Santa-Cruz. Fluid experimental flow estimation based on an optical-flow scheme. *Experiments in Fluids*, 40:80–97, 2006.

[39] T. Corpetti, E. Mémin, and P. Pérez. Dense estimation of fluid flows. *IEEE Trans. Pattern Anal. Machine Intell.*, 24(3):365–380, 2002.

[40] T. Corpetti, E. Mémin, and P. Pérez. Dense motion analysis in fluid imagery. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *European Conference on Computer Vision, ECCV'02*, volume 2350 of *LNCS*, pages 676–691. Springer, 2002.

[41] S. Das Peddada and R. McDevitt. Least average residual algorithm (LARA) for tracking the motion of arctic sea ice. *IEEE trans. on Geoscience and Remote sensing*, 34(4):915–926, 1996.

[42] Y.-H. De Roeck and P. Le Tallec. Analysis and test of a local domain decomposition preconditioner. In R. Glowinsiki, Y. Kuznetsov, G. Meurant, J. Périaux, and O. Widlund, editors, *Fourth Int. Symp. on Domain Decomposition Methods for Part. Diff. Equations*, pages 112–128, Philadelphia, PA, 1991. SIAM.

[43] D.C. Dobson and C.R. Vogel. Convergence of an iterative method for total variation denoising. *SIAM J. of Numer. Anal.*, 34(5):1779–1791, 1997.

[44] M. Dryja. A method of domain decomposition for three-dimensional finite element elliptic problems. In R. Glowinski et al., editor, *First Int. Symp. on Domain Decomposition Methods for Partial Differential Equations*, pages 43–61, Philadelphia, 1988. SIAM.

[45] M. Dryja and W. Hackbusch. On the nonlinear domain decomposition method. *BIT*, 37(2):296–311, 1997.

[46] M. Dryja and O. Widlund. Additive schwarz methods for elliptic finite element problems in three dimensions. Technical Report 580, Courant Institute of Mathematical Sciences, 1991. Computer Science Technical Report 570.

[47] M. Dryja and O.B. Widlund. Towards a unified theory of domain decomposition algorithms for elliptic problems. In T. Chan, R. Glowinski, J. P'eriaux, and O. Widlund, editors, *Third Int. Symp. on Domain Decomposition Meth. for Elliptic Problems*, pages 3–12, Philadelphia, PA, 1990. SIAM.

[48] M. Dryja and O.B. Widlund. Domain decomposition algorithms with small overlap. *SIAM J. of Scientifc Computing*, 15(3):604–620, 1994.

[49] M. Dryja and O.B. Widlund. Schwarz methods of Neumann-Neumann type for three-dimensional elliptic finite element problems. *Comm. Pure Appl. Math.*, 48(2):121–155, 1995.

[50] C. Farhat, P.S. Chen, and J. Mandel. A scalable lagrange multiplier based domain decomposition method for time-dependent problems. *Int. J. Numer. Meth. Engng.*, 38:3831–3853, 1995.

[51] C. Farhat, M. Lesoinne, P. LeTallec, K. Pierson, and D. Rixen. FETI-DP: A dual-primal unified FETI method—part i: A faster alternative to the two-level FETI method. *Int. J. Numer. Meth. Engng.*, 50:1523–1544, 2001.

[52] C. Farhat, M. Lesoinne, and K. Pierson. A scalable dual-primal domain decomposition method. *SIAM J. of Numer. Anal.*, 7:687–714, 2000.

[53] C. Farhat and J. Mandel. Scalable substructuring by lagrange multipliers in theory and in practice. In P. Bjørstadt, M. Espedal, and D. Keyes, editors, *Proc. Ninth Int. Conf. on Domain Decomposition Methods*, Bergen, Norway, 1996.

[54] C. Farhat and F.X. Roux. A method of finite element tearing and interconnecting and its parallel solution algorithm. *Int. J. Numer. Meth. Engrg.*, 32:1205–1227, 1991.

[55] C. Farhat and F.X. Roux. Implicit parallel processing in structural mechanics. *Compuational Mechanics Advances*, 2(1):1–124, 1994.

[56] J.M. Fitzpatrick and C.A. Pederson. A method for calculating fluid flow in time dependant density images. *Electronic Imaging*, 1:347–352, March 1988. Institute for Graphic Communication.

[57] Message Passing Interface Forum. *MPI-2: Extensions to the Message-Passing Interface*. University of Tennesse, Knoxville, TN, 1995.

[58] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard*. University of Tennesse, Knoxville, TN, 1995.

[59] S. Fučik, A. Kratochvil, and J. Nečas. Kačanov-Galerkin method. *Comment. Math. Univ. Carolinae*, 14(4):651–659, 1973.

[60] R. Glowinski, G.H. Golup, G.A. Meurant, and J. Périeux (Eds.). *First Int. Symposium on Domain Decomposition Methods For Partial Differential Equations*. SIAM, Philadelphia, PA, 1988.

[61] R. Glowinski, G.H. Golup, G.A. Meurant, J. Périeux, and O.B. Widlund (Eds.). *Fourth Int. Symposium on Domain Decomposition Methods For Partial Differential Equations*. SIAM, Philadelphia, PA, 1991.

[62] R. Glowinski and P. Le Tallec. Augmented Lagrangian interpretation of the nonoverlapping Schwarz alternating method. In T. Chan, R. Glowinski, J. Periaux, and O. B. Widlund, editors, *Third Int. Symp. on Domain Decomposition Methods for Partial Differential Equations*, Philadelphia, PA, 1999. SIAM.

[63] G. Golub and C.V. Loan. *Matrix Computations*. The Johns Hopkins Univ. Press, Baltimore, MD, 2nd edition, 1989.

[64] W. Gropp and B. Smith. Experience with domain decomposition in three dimension: Overlapping schwarz methods. In J. Mandel, C. Farhat, and X.-C. Cai, editors, *Sixth Int. Conf. of Domain Decomposition*, volume 157 of *Contemporary Mathematics*, pages 323–334. AMS, 1994.

[65] M. Gunzburger, H. Lee, and J. Peterson. An optimization based domain decomposition method for partial differential equations. *Comput. Math. Appl.*, 37(10):77–93, 1999.

[66] Max D. Gunzburger. *Perspectives in Flow Control and Optimization*. SIAM, Philadelphia, PA, 2003.

[67] M.D. Gunzburger and H.K. Lee. An optimization-based domain decomposition method for the navier-stokes equations. *SIAM J. of Numer. Anal.*, 37(5):1455–1480, 2000.

[68] W. Hackbusch. *Theorie und Numerik elliptischer Differentialgleichungen.* B.G. Teubner, Stuttgart, Germany, 1986.

[69] W. Hackbusch. *Iterative Solution of Large Sparse Systems of Equations.* Springer, Heidelberg, Germany, 1993.

[70] J. Heers, C. Schnörr, and H.S. Stiehl. Globally convergent iterative numerical schemes for nonlinear variational image smoothing and segmentation on a multiprocessor machine. *IEEE Trans. Image Processing*, 10(6):852–864, June 2001.

[71] I. Herrera, D. Keyes, O. Widlund, and R. Yates (Eds.). *Domain Decomposition Methods in Science and Engineering.* National Autnomous University of Mexico, Mexico City, Mexico, 2003.

[72] B.K.P. Horn and B.G. Schunck. Determining optical flow. *Art. Int.*, 17:185–203, 1981.

[73] J. Kačúr, J. Nečas, J. Polák, and J. Souček. Convergence of a method for solving the magnetostatic filed in nonlinear media. *Aplikace Mathematiky*, 13:456–465, 1968.

[74] D.E. Keyes, T.F. Chan, G. Meurant, J.S. Scroggs, and R.G. Voigt eds. *Fifth Int. Symposium on Domain Decomposition Methods for Partial Differential Equations.* SIAM, Philadelphia, PA, 1990.

[75] A. Klawonn and O. Widlund. Dual-primal FETI methods for linear elasticity. Technical Report TR2004-855, Fachbereich Mathematik, Campus Essen, Universität Duisburg-Essen, Germany, 2004.

[76] A. Klawonn, O. Widlund, and M. Dryja. Dual-primal FETI methods for three-dimensional elliptic problems with heterogeneous coefficients. *SIAM J. of Numer. Anal.*, 40(1):159–179, 2002.

[77] A. Klawonn and O.B. Widlund. A domain decomposition method with lagrnage multipliers and inexact solvers for linear elasticity. *SIAM J. of Scientifc Computing*, 22:1199–1219, 2000.

[78] A. Klawonn and O.B. Widlund. FETI and neumann-neumann iterative substructuring methods: Connections and new results. *Comm. Pure Appl. Math.*, 54:57–90, 2001.

[79] T. Kohlberger, E. Mémin, and C. Schnörr. Variational dense motion estimation using the helmholtz decomposition. In L.D. Griffin and M. Lillholm, editors, *Proc. 4th Int. Conf. on Scale-Space Theories in Computer Vision, ScaleSpace'03*, volume 2695 of *LNCS*, pages 432–448. Springer, 2003.

[80] T. Kohlberger, C. Schnörr, A. Bruhn, and J. Weickert. Domain decomposition for parallel variational optical flow compuation. In B. Michaelis and G. Krell, editors, *Pattern Recognition, Proc. 25th DAGM Symposium*, volume 2781 of *LNCS*, pages 196–202. Springer, 2003.

[81] T. Kohlberger, C. Schnörr, A. Bruhn, and J. Weickert. Parallel variational motion estimation by domain decomposition and cluster computing. In T. Pajdla and J. Matas, editors, *Eighth European Conf. on computer Vision, ECCV '04*, volume 3024 of *Springer LNCS*, pages 205–216, Prague, Czech Republic, 2004.

[82] T. Kohlberger, C. Schnörr, A. Bruhn, and J. Weickert. Domain decomposition for variational optical flow computation. *IEEE Trans. Image Processing*, 14(8):1109–1124, 2005.

[83] R. Kornhuber, R. Hoppe, J. Périaux, O. Pironneau, O. Widlund, and J. (Eds.) Xu. *Domain Decomposition Methods in Science and Engineering*. LNCS. Springer, Berlin, Germany, 2004.

[84] R. Larsen, K. Conradsen, and B.K. Ersboll. Estimation of dense image flow fields in fluids. *IEEE trans. on Geoscience and Remote sensing*, 36(1):256–264, Jan. 1998.

[85] P. Le Tallec, De Roeck Y.-H., and Vidrascu M. Domain decomposition methods for large linearly elliptic three dimensional problems. *J. Comp. Appl. Math.*, 34:93–117, 1991.

[86] H.K. Lee. *Optimization Based Domain Decomposition Methods for Linear and Nonlinear Problems*. PhD thesis, Faculty of the Virginia Polytechnic Inst. and State Univ., Blacksburg, VA, 1997.

[87] P. LeTallec. Domain decomposition methods in computational mechanics. *Comp. Meth. Applied to Mech. Engrg.*, 1(2):121–220, 1994.

[88] P.-L. Lion. On the Schwarz alternating method III: A variant for nonoverlapping subdomains. In T. Chan, R. Glowinski, J. P'eriaux, and O. Widlund, editors, *Third Int. Symp. on Domain Decomposition Meth. for Part. Diff. Eqs.*, pages pp. 202–231, Philadelphia, PA, 1990. SIAM.

[89] J.L. Lions. *Optimal Control of Systems Governed by Partial Differential Equations*, volume 170 of *Die Grundlagen der math. Wissenschaften in Einzeldarstellungen*. Springer, Heidelberg, Germany, 1971.

[90] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of the 7th Int. Joint Conference on Artificial Intelligence*, 1981.

[91] S.-H. Lui. On linear monotone iteration and Schwarz methods for nonlinear elliptic pdes. *Numer. Math.*, 93(1):109–129, 2002.

[92] S.-H. Lui. On monotone iteration and Schwarz methods for nonlinear parabolic pdes. *J. Comput. Appl. Math.*, 161:449–468, 2003.

[93] F. Mandel and F.X. Roux. Optimal convergence properties of the FETI domain decomposition method. *Comp. Meth. Applied to Mech. Engrg.*, 115(367–388), 1994.

[94] J. Mandel and R. Tezaur. Convergence of a substructuring method with lagrange multipliers. *NM*, 73:473–487, 1996.

[95] J. Mandel and R. Tezaur. On the convergence of a dual-primal substructuring method. *Numer. Math.*, 88:543–558, 2001.

[96] S.P. McKenna and W.R. McGillis. Performance of digital image velocimetry processing techniques. *Experiments in Fluids*, 32:106–115, 2002.

[97] A. Nédic and D. P. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM J. of Optimization*, 12(1):109–138, 2001.

[98] J. Nečas and I. Hlavácek. *Mathematical theory of elastic and elostaplastic bodies: an introduction*. Elsevier, Amsterdam, 1981.

[99] A. Ottenbacher, M. Tomasini, K. Holmlund, and J. Schmetz. Low-level cloud motion winds from Meteosat high-resolution visible imagery. *Weather and Forecasting*, 12(1):175–184, March 1997.

[100] K.H. Pierson. *A Familiy of Domain Decomposition Methods for the Massively Parallel Solution of Computational Mechanics Problems*. PhD thesis, Aerospace Engineering, University of Colorado at Boulder, Boulder, CO, 2000.

[101] A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Oxford University Press, Oxford, UK, 1999.

[102] D.J. Rixen and C. Farhat. A simple and efficient extension of a class of substructure based preconditioners to heterogeneous structural mechanics problems. *Int. J. Numer. Meth. Engng.*, 44:489–516, 1999.

[103] L.I. Rudin and S. Osher. Total variation based image restoration with free local constraints. In *IEEE Proc. Int. Conf. Image Proc.*, volume 1, pages 31–35, 1994.

[104] L.I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[105] C. Schnörr. Determining optical flow for irregular domains by minimizing quadratic functionals of a certain class. *Int. Journal of Computer Vision*, 6(1):25–38, 1991.

[106] H.A. Schwarz. *Gesammelte Mathematische Abhandlungen*, Vol. 2:133–143, Springer, Berlin, 1890. First published in *Vierteljahrsschrift der naturforschenden Gesellschaft* in Zürich, 15:272–286, Springer, Berlin, 1870.

[107] H.R. Schwarz. *Methode der finiten Elemente*. B.G. Teubner, Stuttgart, Germany, 1980.

[108] J. Shukla and R. Saha. Computation of non-divergent streamfunction and irrotational velocity potential from the observed winds. *Monthly weather review*, 102:419–425, 1974.

[109] J. Simpson and J. Gobat. Robust velocity estimates, stream functions, and simulated Lagrangian drifters from sequential spacecraft data. *IEEE trans. on Geosciences and Remote sensing*, 32(3):479–492, 1994.

[110] B. Smith. A domain decomposition algorithm for elliptic problems in three dimensions. *Numer. Math.*, 60(2):219–234, 1991.

[111] B. Smith. A parallel implementation of an iterative substructuring algorithm for problems in three dimensions. *SIAM J. of Scientifc Computing*, 13(1):364–378, 1992.

[112] B. Smith, P. Bjørstad, and W. Gropp. *Domain Decomposition: Parallel Multilevel Methods for the Solution of Elliptic Partial Differential Equations*. Cambridge Univ. Press, Cambridge, UK, 1996.

[113] B.F. Smith. *Domain Decomposition Algorithms for Partial Differential Equations of Linear Elasticity*. PhD thesis, Courant Insitute of Mathematical Sciences, New York University, 1990.

[114] B.F. Smith. An optimal domain decomposition preconditioner for the finite element solution of linear elasticity problems. *SIAM J. of Scientifc Computing*, 13(1):364–378, 1992.

[115] S.M. Song and R.M. Leahy. Computation of 3D velocity fields from 3D cine and CT images of human heart. *IEEE trans. on medical imaging*, 10(3):295–306, 1991.

[116] D. Suter. Motion estimation and vector splines. In *Proc. Conf. Comp. Vision Pattern Rec.*, pages 939–942, Seattle, WA, June 1994.

[117] R. Tezaur. *Analysis of Langrange Multiplier Based Domain Decomposition*. PhD thesis, Univ. of Colorado at Denver, CO, 1998. URL:http://www-math.cudenver.edu/graduate/thesis/rtezaur.ps.gz.

[118] A. N. Tikhonov. Regularization of incorrectly posed problems. *Soviet Mathematics Doklady*, 4:1624–1627, 1963.

[119] A. Toselli and A. Klawonn. A FETI domain decomposition method for edge element approximations in two dimension with discontinuous coefficients. *SIAM J. of Numer. Anal.*, 39(3):932–956, 2001.

[120] A. Toselli and O. Widlund. *Domain Decomposition Methods - Algorithms and Theory*. Springer, Berlin, Germany, 2004.

[121] C.R. Vogel. *Computational methods for inverse problems*. SIAM, Philadelphia, PA, 2002.

[122] C.R. Vogel and M.E. Oman. Iterative methods for total variation denoising. *SIAM J. of Scientifc Computing*, 17(1):227–238, 1996.

[123] C.R. Vogel and M.E. Oman. Fast, robust total variation-based reconstruction of noisy, blurred images. *IEEE Trans. Image Processing*, 7(6):813–824, 1998.

[124] J. Weickert and C. Schnörr. A theoretical framework for convex regularizers in PDE–based computation of image motion. *Int. J. Computer Vision*, 45(3):245–264, 2001.

[125] B. Wohlmuth. *Discretization Methods and Iterative Solvers Based on Domain Decomposition*, volume 17 of *LNCS*. Springer, Berlin, 2001.

[126] B. Wohlmuth, A. Toselli, and O. Widlund. An iterative substructuring method for Raviart-Thomas vector fields in three dimensions. *JNA*, 37(5):1657–1676, 2000.

[127] J. Xu. Iterative methods by space decomposition and subspace correction. *SIAM Review*, 34(4):581–613, 1992.

[128] J. Xu and J. Zou. Some nonoverlapping domain decomposition methods. *SIAM Review*, 40(34):857–914, 1998.

[129] J. Yuan, P. Ruhnau, E. Mémin, and C. Schnörr. Discrete orthogonal decomposition and variational fluid flow estimation. In *Scale-Space 2005*, volume 3459 of *LNCS*, pages 267–278. Springer, 2005.

[130] J. Yuan, C. Schnörr, and E. Mémin. Discrete orthogonal decomposition and variational fluid flow estimation. *J. Math. Imag. Vision*. in press.

[131] L. Zhou, C. Kambhamettu, and D. Goldgof. Fluid structure and motion analysis from multi-spectrum 2D cloud images sequences. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR '00*, volume 2, pages 744–751, Hilton Head Island, SC, 2000.